Хмельницький національний університет
Факультет програмування та комп'ютерних і телекомунікаційних систем
Кафедра комп'ютерних наук та інформаційних технологій

ДИПЛОМНА РОБОТА
МАГІСТРА

на тему *Інформаційна технологія адаптивного тестування рівня знань*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Виконала: студентка 2 курсу, група *КНм-19-1* _____ *Г.А. Білоус*
Підпис Ініціали, прізвище

Керівник: *ст. викладач кафедри КНІТ* _____ *О.В. Мазурець*
Підпис Ініціали, прізвище

Нормоконтроль: *к.т.н., доцент кафедри КНІТ* _____ *Р.О. Багрій*
Підпис Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КНІТ, д.т.н., професор _____ *О.В. Бармак*
Підпис Ініціали, прізвище

7 12 2020 р.

Хмельницький 2020

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *програмування та комп'ютерних і телекомунікаційних систем*

Кафедра *комп'ютерних наук та інформаційних технологій*

Освітній ступінь *магістр*

Галузь знань *12 – Інформаційні технології*

Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ

Завідувач кафедри *комп'ютерних наук та інформаційних технологій*

_____

(підпис)

д.т.н., професор О.В. Бармак

«7» 09 *2020* року

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ МАГІСТРА

1. Тема дипломної роботи магістра: *«Інформаційна технологія адаптивного тестування рівня знань»*

2. Завдання видано студентці *Білоус Ганні Анатоліївні*
(прізвище, ім'я, по батькові)

3. Керівник роботи *ст. викладач Мазурець Олександр Вікторович*
(прізвище, ім'я, по батькові)

4. Затверджені наказом університету від «9» 09 *2020* р. № 22

5. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Мета роботи – розробка інформаційної технології для адаптивного тестування знань, що використовує показники семантичної важливості ключових термінів інформаційного навчального матеріалу для адаптивного вибору тестових завдань у процесі тестування. Дослідити практичну ефективність інформаційної технології адаптивного тестування рівня знань шляхом розробки й аналізу функціональності та ефективності використання відповідної інформаційної системи.

Реферат

Дипломна робота магістра присвячена розробці інформаційної технології адаптивного тестування рівня знань та відповідної інформаційної системи для дослідження практичної ефективності алгоритмів адаптивного вибору тестових завдань під час контролю рівня знань.

Актуальність роботи. Контроль рівня знань студентів має опиратися на зміст та матеріал навчального курсу в рамках якого він проводиться. Основою курсу є навчальний матеріал, що розкриває його проблематику та є базисом при формуванні знань. Одним із найважливіших показників високоякісної освітньої моделі є контроль рівня знань студентів на основі навчального матеріалу, що вивчається. Найбільш об'єктивним засобом оцінювання рівня знань в даний час вважають тестування, що дозволяє неупереджено оцінити навчальні досягнення студентів.

На сучасному етапі виділено наступні проблеми комп’ютерного тестування за класичною формою: одержана студентом вибірка тестових завдань може не повністю покривати семантичну структуру навчального матеріалу; для повного покриття семантичної структури навчального матеріалу потрібна велика кількість тестових завдань; при оцінюванні не враховується зростання рівня складності тестових завдань через використання в контенті різних за рівнем семантичної значущості термінів; використання фіксованого обсягу тестових завдань незалежно від успішності процесу тестування.

Таким чином, у процесі тестування слід виконувати перевірку розуміння студентом складових семантичної структури навчального матеріалу у вигляді системи заголовків та відповідних множин ключових термінів. Для цього є перспективним використання можливостей інформаційних технологій.

Мета і задачі роботи. Мета роботи полягає у розробці інформаційної технології для адаптивного тестування знань, що використовує показники семантичної важливості ключових термінів інформаційного навчального матеріалу для адаптивного вибору тестових завдань у процесі тестування. Для досягнення мети було розв’язано наступні *задачі*:

– досліджено відомі підходи до адаптивного тестування рівня знань;

– вдосконалено інформаційну модель адаптивного тесту для формального подання елементів терміноорієнтованого адаптивного тестування рівня знань;

– вдосконалено метод адаптивного вибору тестових запитань для динамічного вибору тестових завдань у процесі тестування;

– розроблено інформаційну технологію адаптивного тестування рівня знань, що використовує вдосконалені модель і метод;

– розроблено інформаційну систему адаптивного тестування рівня знань на основі розробленої інформаційної технології;

– досліджено практичну ефективність інформаційної технології адаптивного тестування рівня знань шляхом аналізу функціональності й ефективності застосування відповідної інформаційної системи.

Об’єкт дослідження – процес тестування рівня знань з використанням інформаційних технологій.

Предмет дослідження – інформаційні моделі, методи та технології для автоматизації адаптивної перевірки рівня знань.

Методи дослідження. Під час дослідження були використані загальнонаукові теоретичні та емпіричні методи, які дали змогу дослідити предметну область у питанні адаптивного тестування знань. Для реалізації інформаційної технології були використанні методології проектування інформаційних систем та об’єктно-орієнтований підхід. Під час розробки даталогічної моделі даних було використано реляційний підхід. Для дослідження ефективності інформаційної технології було застосовано метод тестування інформаційних систем та метод статистичної обробки й порівняльного аналізу.

Наукова новизна одержаних результатів. В результаті проведеної роботи були отримані такі результати:

– Вдосконалено інформаційну модель адаптивного тесту, яка відрізняється тим, що містить формальне подання складових предметної області терміноорієнтованого адаптивного тестування рівня знань.

– Вдосконалено метод адаптивного вибору тестових запитань, який відрізняється тим, що використовує показники семантичної важливості ключових термінів навчального матеріалу для динамічного вибору тестових завдань у процесі тестування, застосовуючи один із розроблених алгоритмів адаптивного вибору тестових завдань: регресивний, прогресивний та медіанний.

– Розроблено нову інформаційну технологію адаптивного тестування рівня знань, що використовує вдосконалені інформаційну модель і метод адаптивного вибору тестових запитань й дозволяє за вхідними даними у вигляді пов’язаних із семантичною структурою навчального курсу тестових завдань одержувати в результаті проходження тесту вихідні дані у вигляді оцінки результату тестування, що вираховується залежно від відповідей опитуваного впродовж тестування.

– Розроблено нову інформаційну систему адаптивного тестування рівня знань за розробленою інформаційною технологією, що дозволяє працювати з тестовим матеріалом, створювати цільові вибірки тестових завдань, формувати індивідуальні запити на рівень знань, проводити адаптивне тестування за одним із доступних алгоритмів та обраховувати результати проходження запитів на рівень знань.

Практичне значення одержаних результатів. На основі розробленої інформаційної технології адаптивного тестування рівня знань було створено відповідну автоматизовану систему адаптивного контролю знань. Проведення прикладного дослідження практичної ефективності інформаційної технології адаптивного тестування рівня знань шляхом аналізу функціональності й ефективності застосування відповідної інформаційної системи дозволило встановити, що запропоновані й реалізовані алгоритми адаптивного тестування забезпечують в середньому на 22,06 % більш швидке проведення процесу тестування, а для визначення рівня знань студентів потрібно в середньому на 29,26 % менше тестових завдань.

Апробація результатів дипломної роботи магістра та публікації. За темою дипломної роботи магістра автором виконано 5 наукових публікацій [40,

41, 42, 43, 44]. Основні наукові та практичні результати доповідалися на конференціях:

– доповідь на тему «Параметризація моделі тестового завдання при автоматизованому формуванні тестів» на VII Міжнародної науково-практичної конференції «Інформаційні управляючі системи та технології ICST-ODESSA-2018» (Одеса, 17-18 вересня 2018 року, Одеський національний політехнічний університет);

– доповідь на тему «Метод формального опису елементів моделей автоматизованого формування тестових завдань» на Всеукраїнській науково-практичній конференції «Інтелектуальний потенціал – 2018» (Хмельницький, 14-16 листопада 2018 року, Хмельницький національний університет);

– доповідь на тему «Використання програмного розширення spire.doc для автоматизації роботи з цифровими документами» на XI Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук АПКН-2019» (Хмельницький, 14-15 листопада 2019 року, Хмельницький національний університет);

– доповідь на тему «Інформаційна технологія адаптивного тестування рівня знань» на XII Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук АПКН-2020» (Хмельницький, 9-10 листопада 2020 року, Хмельницький національний університет).

Структура та обсяг роботи. Дипломна робота магістра складається з завдання, реферату, змісту, переліку скорочень, вступу, 4 розділів, висновків, переліку посилань із 44 найменувань та 9 додатків. Загальний обсяг дипломної роботи магістра становить 204 сторінки, з них 104 сторінки основного тексту та 95 сторінок додатків. У роботі наведено 32 рисунка та 9 таблиць.

Ключові слова: тестові завдання, тести, тестування, адаптивне тестування, інформаційна технологія, інформаційний навчальний матеріал, тестовий навчальний матеріал, ключові терміни, інформаційна система.

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
АТ	Адаптивний тест
БД	База даних
ДРМ	Дипломна робота магістра
ІКТ	Інформаційно-комунікаційні технології
ІНМ	Інформаційний навчальний матеріал
КН	Комп’ютерні науки
НК	Навчальний курс
ПЗ	Програмне забезпечення
ТНМ	Тестовий навчальний матеріал
ALEKS	Assessment and LEarning in Knowledge Spaces
CLR	Common Language Runtime
JVM	Java Virtual Machine

Вступ

Дипломна робота магістра присвячена розробці інформаційної технології адаптивного тестування рівня знань та відповідної інформаційної системи для дослідження практичної ефективності алгоритмів адаптивного вибору тестових завдань під час контролю рівня знань.

Актуальність роботи. Контроль рівня знань студентів має опиратися на зміст та матеріал навчального курсу в рамках якого він проводиться. Основою курсу є навчальний матеріал, що розкриває його проблематику та є базисом при формуванні знань. Одним із найважливіших показників високоякісної освітньої моделі є контроль рівня знань студентів на основі навчального матеріалу, що вивчається. Найбільш об'єктивним засобом оцінювання рівня знань в даний час вважають тестування, що дозволяє неупереджено оцінити навчальні досягнення студентів.

На сучасному етапі виділено наступні проблеми комп’ютерного тестування за класичною формою: одержана студентом вибірка тестових завдань може не повністю покривати семантичну структуру навчального матеріалу; для повного покриття семантичної структури навчального матеріалу потрібна велика кількість тестових завдань; при оцінюванні не враховується зростання рівня складності тестових завдань через використання в контенті різних за рівнем семантичної значущості термінів; використання фіксованого обсягу тестових завдань незалежно від успішності процесу тестування.

Таким чином, у процесі тестування слід виконувати перевірку розуміння студентом складових семантичної структури навчального матеріалу у вигляді системи заголовків та відповідних множин ключових термінів. Для цього є перспективним використання можливостей інформаційних технологій.

Мета і задачі роботи. Мета роботи полягає у розробці інформаційної технології для адаптивного тестування знань, що використовує показники семантичної важливості ключових термінів інформаційного навчального

матеріалу для адаптивного вибору тестових завдань у процесі тестування. Для досягнення мети було розв’язано наступні *задачі*:

– досліджено відомі підходи до адаптивного тестування рівня знань;

– вдосконалено інформаційну модель адаптивного тесту для формального подання елементів терміноорієнтованого адаптивного тестування рівня знань;

– вдосконалено метод адаптивного вибору тестових запитань для динамічного вибору тестових завдань у процесі тестування;

– розроблено інформаційну технологію адаптивного тестування рівня знань, що використовує вдосконалені модель і метод;

– розроблено інформаційну систему адаптивного тестування рівня знань на основі розробленої інформаційної технології;

– досліджено практичну ефективність інформаційної технології адаптивного тестування рівня знань шляхом аналізу функціональності й ефективності застосування відповідної інформаційної системи.

Об’єкт дослідження – процес тестування рівня знань з використанням інформаційних технологій.

Предмет дослідження – інформаційні моделі, методи та технології для автоматизації адаптивної перевірки рівня знань.

Методи дослідження. Під час дослідження були використані загальнонаукові теоретичні та емпіричні методи, які дали змогу дослідити предметну область у питанні адаптивного тестування знань. Для реалізації інформаційної технології були використанні методології проектування інформаційних систем та об’єктно-орієнтований підхід. Під час розробки даталогічної моделі даних було використано реляційний підхід. Для дослідження ефективності інформаційної технології було застосовано метод тестування інформаційних систем та метод статистичної обробки й порівняльного аналізу.

Наукова новизна одержаних результатів. В результаті проведеної роботи були отримані такі результати:

– Вдосконалено інформаційну модель адаптивного тесту, яка відрізняється тим, що містить формальне подання складових предметної області термінооріентованого адаптивного тестування рівня знань.

– Вдосконалено метод адаптивного вибору тестових запитань, який відрізняється тим, що використовує показники семантичної важливості ключових термінів навчального матеріалу для динамічного вибору тестових завдань у процесі тестування, застосовуючи один із розроблених алгоритмів адаптивного вибору тестових завдань: регресивний, прогресивний та медіанний.

– Розроблено нову інформаційну технологію адаптивного тестування рівня знань, що використовує вдосконалені інформаційну модель і метод адаптивного вибору тестових запитань й дозволяє за вхідними даними у вигляді пов'язаних із семантичною структурою навчального курсу тестових завдань одержувати в результаті проходження тесту вихідні дані у вигляді оцінки результату тестування, що вираховується залежно від відповідей опитуваного впродовж тестування.

– Розроблено нову інформаційну систему адаптивного тестування рівня знань за розробленою інформаційною технологією, що дозволяє працювати з тестовим матеріалом, створювати цільові вибірки тестових завдань, формувати індивідуальні запити на рівень знань, проводити адаптивне тестування за одним із доступних алгоритмів та обраховувати результати проходження запитів на рівень знань.

Практичне значення одержаних результатів. На основі розробленої інформаційної технології адаптивного тестування рівня знань було створено відповідну автоматизовану систему адаптивного контролю знань. Проведення прикладного дослідження практичної ефективності інформаційної технології адаптивного тестування рівня знань шляхом аналізу функціональності й ефективності застосування відповідної інформаційної системи дозволило встановити, що запропоновані й реалізовані алгоритми адаптивного тестування забезпечують в середньому на 22,06 %. більш швидке проведення процесу

тестування, а для визначення рівня знань студентів потрібно в середньому на 29,26 % менше тестових завдань.

Апробація результатів дипломної роботи магістра та публікації. За темою дипломної роботи магістра автором виконано 5 наукових публікацій [40, 41, 42, 43, 44]. Основні наукові та практичні результати доповідалися на конференціях:

– доповідь на тему «Параметризація моделі тестового завдання при автоматизованому формуванні тестів» на VII Міжнародної науково-практичної конференції «Інформаційні управляючі системи та технології ICST-ODESSA-2018» (Одеса, 17-18 вересня 2018 року, Одеський національний політехнічний університет);

– доповідь на тему «Метод формального опису елементів моделей автоматизованого формування тестових завдань» на Всеукраїнській науково-практичній конференції «Інтелектуальний потенціал – 2018» (Хмельницький, 14-16 листопада 2018 року, Хмельницький національний університет);

– доповідь на тему «Використання програмного розширення spire.doc для автоматизації роботи з цифровими документами» на XI Всеукраїнській науково-практичній конференції «Актуальні проблеми комп’ютерних наук АПКН-2019» (Хмельницький, 14-15 листопада 2019 року, Хмельницький національний університет);

– доповідь на тему «Інформаційна технологія адаптивного тестування рівня знань» на XII Всеукраїнській науково-практичній конференції «Актуальні проблеми комп’ютерних наук АПКН-2020» (Хмельницький, 9-10 листопада 2020 року, Хмельницький національний університет).

Структура та обсяг роботи. Дипломна робота магістра складається з завдання, реферату, змісту, переліку скорочень, вступу, 4 розділів, висновків, переліку посилань із 44 найменувань та 9 додатків. Загальний обсяг дипломної роботи магістра становить 204 сторінки, з них 104 сторінки основного тексту та 95 сторінок додатків. У роботі наведено 32 рисунка та 9 таблиць.

Розділ 1

Сучасний стан та проблеми комп'ютерного тестування

1.1 Використання тестового навчального матеріалу для контролю рівня знань

Розвиток сучасного суспільства і системи освіти пред'являють все більш високі вимоги до якості підготовки студентів. Особливу роль в підвищенні якості освіти відіграє його комп'ютеризація, яка розуміється не як просте передавання навчальної інформації в цифровому вигляді, а як створення спроектованого з психолого-педагогічної та методичної точок зору інформаційно-освітнього навчального середовища [1].

Інструментом здійснення високоякісної освітньої моделі є моніторинг якості освіти, головною складовою якого є моніторинг навчальних досягнень студентів [2]. Найважливішою цінністю сучасного світу є особистість, перед якою відкрита можливість самореалізації та саморозвитку, входження до сучасного суспільства. Побудова освіти в новітньому контексті, модернізація усіх її складових перетворюється на першочергову потребу.

Сучасна практика та проведені дослідження дозволяють стверджувати про відсутність на сьогодні комплексної моделі моніторингу якості освіти студента, яка б дозволяла систематично, прозоро та ефективно відслідковувати динаміку розвитку особистості, встановлювати причини проблем і розробляти прогнози [3]. Якість у цьому контексті набуває особливої актуальності на сучасному етапі. Моніторинг навчальних досягнень студентів повинен характеризуватися систематичністю, тривалістю в часі, прозорістю, ефективною системою відслідковування та ставити завдання встановлення причин і невідповідностей результату поставленим цілям [4].

Контроль рівня знань студентів має опиратися на зміст та матеріал навчального курсу в рамках якого він проводиться. Навчальний курс є одиницею організації навчального процесу, логічно цілісним етапом навчання. Згідно з

визначенням В. Даля [5], для курсу характерно те, що він «закінчується в установленому порядку», тобто заліком, іспитом або іншою формою контролю і оцінювання результатів його проходження. Курс відрізняється від інших процесів навчання зосередженістю на одній темі, плановістю і відносністю тривалісті в часі. Ще однією важливою характеристикою навчального курсу є наявність в ньому чітко визначених навчальних цілей – знань, умінь і компетенцій, які студенти, повинні придбати після проходження курсу.

Основою курсу є навчальний матеріал, що розкриває його проблематику та є базисом при формуванні знань. Навчальний матеріал являє собою складну систему, яка має свою структуру зі специфічними елементами й зв'язками між ними. Будучи основою процесу навчання, навчальний матеріал включає всю інформацію, яка подана для засвоєння й сприяє засвоєнню. Тобто навчальний матеріал розглядається як сукупність двох інформацій: основної та допоміжної. Кінцева мета подання основної інформації є перетворення її в знання або вміння (знання - це сприйнята, зрозуміла, усвідомлена й включена в систему наявних знань інформація). Допоміжна інформація має на меті забезпечення надійності (гарантованості) у засвоєнні основної інформації. Сконцентрований у довідниках, дидактичних матеріалах, навчальних посібниках, підручниках, збірниках задач і вправ, інструкціях, засобах наочності тощо, навчальний матеріал залежно від виконуваних функцій може бути згрупований таким чином:

– інформаційний (поданий звичайно як тексти, малюнки, креслення, схеми та інші форми графічного вираження інформації: таблиці, моделі, географічні карти, твори скульптури й живопису, музичні твори, ноти, реальні об'єкти навколишньої дійсності й т.ін.);

– операційний (задачі, вправи, завдання інтелектуального або практичного змісту, під час виконання яких виробляються уміння та навички);

– контролюючий (завдання, що забезпечують внутрішній і зовнішній зворотний зв’язок);

– актуалізуючий (тексти, завдання, які сприяють актуалізації опорних знань, умінь і навичок, необхідних для розуміння й засвоєння нового матеріалу);

– стимулюючий (тексти, завдання, що викликають потребу набути нові знання або нові способи дій);

– діагностуючий (завдання, які дозволяють виявити прогалини в знаннях, причини неправильних дій учнів) [6].

Одним із найважливіших показників високоякісної освітньої моделі є контроль рівня знань студентів на основі навчального матеріалу, що вивчається. Найбільш об'єктивним засобом оцінювання рівня знань в даний час вважають тестування, що дозволяє неупереджено оцінити навчальні досягнення студентів.

Тестування [7] є методом діагностики із застосуванням стандартизованих запитань та завдань, що мають певну шкалу значень. До нього вдаються для стандартизованого визначення індивідуальних відмінностей особистості в усьому світі.

Тестові методи дають змогу з певною ймовірністю визначити рівень розвитку в індивіда психологічних властивостей (пам'яті, мислення, уяви та ін.), особистісних характеристик, ступінь готовності до певної діяльності, засвоєння знань і навичок тощо.

Тестологія – наука про вимірювання психофізіологічних та особистісних характеристик, а також обсягу та якості знань, умінь, навичок. Тестологи вивчають і створюють способи, методи, технології вимірювань психофізіологічних та особистісних характеристик, а також обсягу та якості знань, умінь, навичок. Тестологи створюють тестові комплекси, у яких реалізовано досягнення тестології у вигляді сукупності технологій, рекомендацій, тестів, автоматизованих систем, пристроїв. Тестові комплекси застосовують для атестації учнів та абітурієнтів, для вимірювання обсягу та якості навичок і умінь при прийомі людей на роботу тощо [8].

Тестове завдання є складовою частиною тесту, що відповідає вимогам до завдань у тестовій формі та пройшла обов'язкову перевірку статистичних властивостей [9].

Тестові завдання повинні відповідати певним вимогам:

– однозначність завдань – текст тестового завдання не повинен допускати вільного трактування;

– однозначність відповідей – повинна бути виключена можливість багатозначних відповідей;

– відповідність вивченому навчальному матеріалу - не можна включати в тестове завдання відповіді, які на момент тестування особа, яка тестується, не може обґрунтувати;

– підбір варіантів відповідей (дистракторів) – неправильні відповіді повинні конструюватися на основі типових помилок та повинні бути правдоподібними;

– унікальність – питання не повинні повторювати формулювань підручника;

– мати складність, що відповідає меті й рівню оцінювання;

– диференційна здатність (достатня варіативність тестових балів);

– позитивна кореляція балів завдань із балами всього тесту;

– відповідати вимогам чистоти форми й предметної чистоти змісту.

Тестові завдання за характером формування відповідей розподіляються на завдання закритої, відкритої та комбінованої форм. У завданні закритої форми особа робить вибір з запропонованого їй готового переліку варіантів відповідей. У завданні відкритої форми особа сама генерує відповідь і додає її у поле тесту за допомогою клавіатури або записує ручкою. Існує ряд типів тестових завдань як відритої, так і закритої форми, але основними із них є завдання логічного типу, завдання з множинним вибором та завдання з відритою короткою відповіддю (таблиця 1.1) [10, 11].

Тестовий діагностуючий навчальний матеріал (ТНМ) є різновидом діагностуючого навчального матеріалу й розглядається як засіб перевірки рівня одержаних з ІНМ знань [12]. У тестології існує проблема рівномірного покриття ТНМ семантики контенту ІНМ. Повнота покриття тестами ключових положень навчального матеріалу може визначатися через аналіз використання у тестових завданнях ключових термінів із навчальних матеріалів. Рівномірність покриття тестами структури ІНМ визначається через аналіз прив'язки тестових завдань до ієрархії структурних елементів навчального матеріалу. Така прив'язка може бути забезпечена аналізом контенту тестових завдань на предмет наявності у них ключових термінів із навчальних матеріалів, за умови прив'язки ключових термінів до ієрархії структурних елементів навчального матеріалу, що є також важливим аспектом при формуванні наборів тестових завдань [13].

Таблиця 1.1 – Основні типи завдань тестових наборів [10, 11]

Тип завдання	**Опис**
Логічний тип	Передбачають наявність двох варіантів відповіді, зокрема «так – ні», «правильно – неправильно». Їх використовують для попередньої перевірки правильності вибору або прийняття рішення за змістом завдання без розкриття його суті.
Вибір з множини	Цей тип завдання передбачає вибір однієї або декількох відповідей із запропонованого набору варіантів.
Вибір відповідності	У завданні цього типу на задану тему створюється множина питань і множина правильних відповідей на ці питання. При тестуванні для кожного питання треба обрати із переліку відповідну правильну відповідь.
Відкрита відповідь	У завданні цього типу потрібно з клавіатури або за допомогою ручки відповісти на поставлене питання – додати необхідне слово чи словосполучення. У завданні можна використовувати графічні зображення. Завдання може мати декілька правильних відповідей з різним значенням оцінок. Відповіді можуть бути чутливими до регістру (великі чи маленькі літери) при комп'ютерному тестуванні.

Процес тестування складається з ряду різних елементів, одну частину яких можна віднести до більш істотних, а іншу – до менш істотних. Залежно від конкретної ситуації, кожен з них може надати той чи інший вплив на такі

ключові критерії оцінки тестових результатів, як надійність, валідність і ефективність. Основними елементами тестового завдання [14] є:

– інструкція;
– завдання (умова);
– варіанти відповідей;
– критерії оцінювання.

Інструкція – визначає, що слід робити випробуваному. Інструкція повинна бути сформульована коротко, чітко і зрозуміло, наприклад, у таких формах: «Вкажіть правильну відповідь», «Вказати всі правильні відповіді», «Доповнити», «Встановити відповідність», «Встановити правильну послідовність» і т.д..

Умова – це стимул для відповіді, яка описує певну проблему і ставить завдання перед особою, що проходить тестування. Умова може містити лише завдання або складатися із вступних відомостей та запитання. Умова може подаватися у формі запитання, у наказовій формі або у формі незавершеного твердження. Формулювання запитання доцільно починати з дієслова. Якщо все ж таки використовується формат незавершеного твердження, пропуск в останньому не повинен бути на початку або в середині, його треба розмістити в кінці фрази. Крім того, навіть у незавершеному твердженні умова має бути завершеною з точки зору змісту, щоб на неї можна було відповісти, незважаючи на перелік варіантів відповідей.

Серед варіантів відповідей до тестового завдання мінімум одна є правильною, решта – дистрактори – неправильними. Усі дистрактори мають бути правдоподібними і однорідними. Під час добору дистракторів доцільно використовувати поширені помилки, хибні уявлення, об'єкти, що відповідають лише частині характеристик, наведених в умові, тощо. Водночас у дистракторах не повинно бути фальшивих та хибних відомостей.

Для забезпечення процесу тестування, важливими параметрами тестового завдання [15], які не стосуються безпосередньо його контенту, є такі: тип

питання, час відповіді, максимальна кількість балів, рівень складності. Хоча передбачається ручне або фіксоване визначення більшості цих параметрів, автоматизація їх обрахунку є окремими задачами. Наприклад, час відповіді на тестове завдання може бути обрахований залежно від типу питання та кількості символів у контенті тестового завдання. Тестове завдання має множину відповідей з параметрами: контент відповіді та оцінка відповіді.

В межах моделі семантичної структури НК, можлива фіксація проміжних даних, які можна розглядати як семантичні параметри тестового завдання, а саме:

– термін, рівень засвоєння якого перевіряється тестовим завданням;

– номер фрагменту (речення), що було використано для створення тестового завдання (тобто знання якого й перевіряється);

– номер правила продукції, використаного для формування тестового завдання;

– номер типу тестового завдання;

– оцінка якості використання правила;

– необов’язковий параметр, що може бути обрахований додатково.

Отже, на сучасному етапі тести є ефективним методом контролю рівня одержаних знань. В процесі тестування слід виконувати перевірку розуміння студентом складових семантичної структури навчального матеріалу у вигляді системи заголовків та відповідних множин ключових термінів. Для цього є перспективним використання можливостей інформаційних технологій.

1.2 Проблеми сучасного комп’ютерного тестування та навчальні середовища

Тестовий комп’ютерний контроль знань може бути використаний під час поточного або підсумкового контролю. Він здійснюється у формі самостійного діалогу студента з комп’ютером у присутності викладача, або без нього, з

можливістю запам’ятовування результатів тестування. Тестування можна проводити на різних етапах навчання: під час повторення, на етапі актуалізації опорних знань, для перевірки домашнього завдання, під час вивчення нового матеріалу, для закріплення вивченого, у вигляді заліків [16].

У зв'язку з недостатнім до останнього часу різним розвитком і доступністю комп'ютерних ресурсів в навчальних закладах, а також практичною відсутністю можливості їх застосування в освітньому процесі ідеї комп'ютерного контролю знань не отримали належного вивчення. Але важливість цього питання в підвищенні мотивації студентів до навчання та активізації пізнавальної роботи доведена багатьма дослідниками, серед яких слід відзначити В. Ю. Бикова [17], М. Ю. Кадемія [18], В. М. Федорака [19], В. Е. Снитюка та К. Н. Юрченко [20], О. В. Бармака [21].

Сучасні тенденції розвитку інформаційного суспільства та проблеми впровадження цифрових технологій у вітчизняній освіті і науці є актуальними [17]. У роботах В. Бикова окреслено ряд пріоритетних дій для трансформації комп’ютерно-технологічної платформи освіти та науки України: стандартизація рівнів електронної досконалості освітніх ресурсів, створення технологічної інфраструктури на базі освітніх web-платформ, впровадження педагогічно доцільного та виваженого застосування ІКТ у навчально процесі та розробка електронних освітніх ресурсів задля ефективнішого контролю знань.

Сьогодення диктує такі умови при яких збільшується кількість форм проведення начального процесу і всі вони в тій чи іншій мірі зумовлюють використання інформаційних технологій. Студенти одержують навчальні матеріали, котрі можуть бути розміщені на YouTube, веб-порталах, що містять тексти лекцій, електронні посібники, відеолекції. З’являється можливість ознайомитися з теоретичними основами дисципліни в будь-де, будь-коли та у власному темпі. У такому випадку оцінювання засвоєння нового матеріалу має бути здійснене також з використанням сучасних технологій, наприклад, за допомогою комп’ютерного тестування [18].

Комп’ютерне тестування дає можливість реалізувати основні дидактичні положення контролю навчання: принцип індивідуального характеру перевірки й оцінки знань; системність перевірки й оцінки знань; принцип тематичності; правило диференційованої оцінки успішності [19]. При цьому підготовка тестових завдань повинна базуватися на сучасних технологіях педагогічних вимірів, що не тільки вимагає від розробників високої кваліфікації з педагогіки та предметних областей, але й спеціальних теоретичних знань з теорії тестування.

Під час проведення тестування є важливою варіація рівня складності та типу самих завдань. Відомо ряд моделей [20], що дозволяють змінювати рівень складності запитань безпосередньо в процесі тестування. Зокрема, пропонується автоматизувати процес контролю знань за допомогою структурування інформації та здійснення переривання процесу тестування, визначаючи при цьому послідовності запитів у реальному часі на основі результатів відповідей на попередні завдання. Проте значним мінусом запропонованої методики є її трудомісткість в процесі побудови онтології та безпосередньої реалізації методу моніторингу знань.

Також слід зазначити, що комп’ютерне тестування може здійснюватися із застосуванням онлайн-систем та додатків, що встановлюються безпосередньо на комп’ютер користувача та можуть функціонувати без використанням мережі Інтернет. Кожен із цих варіантів має ряд переваг та недоліків [21] та можливість реалізації різних форм комп’ютерного тестування.

Комп'ютерне тестування може проводитися в різних формах, що розрізняються за технологією об'єднання завдань в тест [22]. Перша форма – найпростіша (класична). Готовий тест, стандартизований або призначений для поточного контролю, вводиться в спеціальну оболонку, функції якої можуть відрізнятися за ступенем повноти. Зазвичай при підсумковому тестуванні оболонка дозволяє представити завдання на екрані, оцінювати результати їх виконання, формувати матрицю результатів тестування, обробляти її і

градувати первинні бали випробовуваних шляхом перекладу в одну зі стандартних шкал для видачі кожному випробуваному тестового балу та протоколу його оцінок за завданнями тесту.

Друга форма комп'ютерного тестування (параметризована) передбачає автоматизовану генерацію варіантів тесту, здійснювану за допомогою інструментальних засобів. Варіанти створюються перед іспитом або безпосередньо під час його проведення з банку каліброваних тестових завдань з стійкими статистичними характеристиками. Калібрування досягається завдяки тривалої попередньої роботи по формуванню бланка, параметри завдань якого отримують на репрезентативній вибірці студентів, як правило, протягом 3-4 років за допомогою бланкових тестів. Змістовна валідність і паралельність варіантів забезпечуються за рахунок строго регламентованого відбору завдань кожного варіанту відповідно до специфікації тесту.

Третя форма – адаптивне тестування. В основі ідей адаптивності лежать міркування про те, що студенту марно давати завдання тесту, які він виконає напевно правильно без найменших труднощів, або гарантовано не впорається з високими труднощами. Тому пропонується оптимізувати труднощі завдань, адаптуючи їх до рівня підготовленості кожного випробуваного, і скоротити за рахунок виключення частини завдань довжину тесту.

Тобто, адаптивні тести розроблені таким чином, щоб регулювати рівень їх складності, виходячи з наданих відповідей, щоб відповідати знанням та можливостям учасника тесту. Якщо студент дає неправильну відповідь, комп’ютер дає більш просте запитання; якщо студент відповість правильно, наступне запитання буде складніше. Адаптивне тестування дозволяє ефективніше виміряти здібності окремих студентів, уникаючи деяких питань, часто пов'язаних із стандартизованим принципом «один розмір – підходить усім».

Для студентів комп'ютерне адаптивне тестування пропонує коротший сеанс тестування з меншою кількістю запитань, оскільки задаються лише ті

питання, які вважаються посильними для студента. З іншого боку, розробники тестів повинні створити більшу базу тестових завдань, щоб у систем тестування було достатньо питань для відповідності різноманітним можливостям усіх студентів, які мають здавати іспит. Оскільки, адаптивне тестування може бути автоматизоване, викладачі та студенти можуть отримати результати тестів швидше, ніж при проведенні тестування на папері.

Проаналізувавши актуальність застосування в освіті та форми сучасного комп’ютерного тестування рівня знань, можна визначити наступні його проблеми:

– одержана студентом вибірка тестових завдань може не повністю покривати семантичну структуру навчального матеріалу;

– для повного покриття семантичної структури навчального матеріалу потрібна велика кількість тестових завдань;

– при оцінюванні не враховується зростання рівня складності тестових завдань через використання в контенті різних за рівнем семантичної значущості термінів;

– використання фіксованого обсягу тестових завдань незалежно від успішності процесу тестування.

Персональний комп'ютер та мережа Інтернет дали змогу створювати загальнодоступну, надзвичайно інформаційно містку, та, порівняно з іншими інформаційно-технологічними системами, дешеву й швидку інформаційну інфраструктуру. Їх доступність та надійність сприяли входженню у всі сфери суспільства нових інформаційних технологій, які повною мірою забезпечили ріст продуктивності у сфері послуг.

Під час проведення аналізу існуючого програмного забезпечення було виявлено те, що для даного напрямку найбільш поширеними є інтернет-ресурси. На сучасному етапі у навчальних закладах широко використовується набір програмного забезпечення для організації навчання засобами інформаційних

технологій. До найбільш використовуваних належать «Classtime» [23], Moodle [24], Khan Academy [25] тощо.

Так, навчальна система «Classtime» [23] (рисунок 1.1) надає можливість створення тестових завдань та проходження тестів студентами задля перевірки знань.

До переваг даного сервісу можна віднести можливість створення тестових завдань наступних типів: множинний вибір, завдання на відповідність, логічний вибір, завдання з відкритою відповіддю. Після створення тесту за умови, якщо викладач є зареєстрованим у системі, тест зберігається там. Ще одним плюсом є можливість редагувати сформовані тестові завдання.

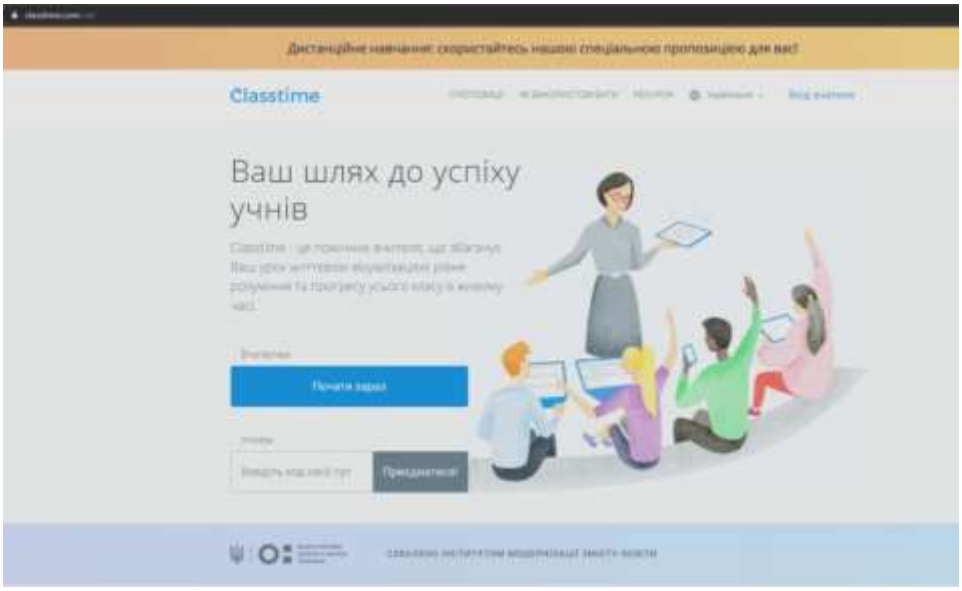


Рисунок 1.1 – Головна сторінка сайту «Classtime» [23]

На рисунку 1.2 зображено приклад демонстрації тестового завдання для студента під час моніторингу його знань на сайті «Classtime».

Щодо мінусів, то не всі можливості сайту є безкоштовними. Система не має функції відображення правильних відповідей після завершення тесту. Також процес тестування можна розпочати лише після того, як викладач надасть код доступу да нього. Порядковість появи тестів в даній системі є сталою, що є негативною рисою стосовно гнучкості системи оцінювання знань.

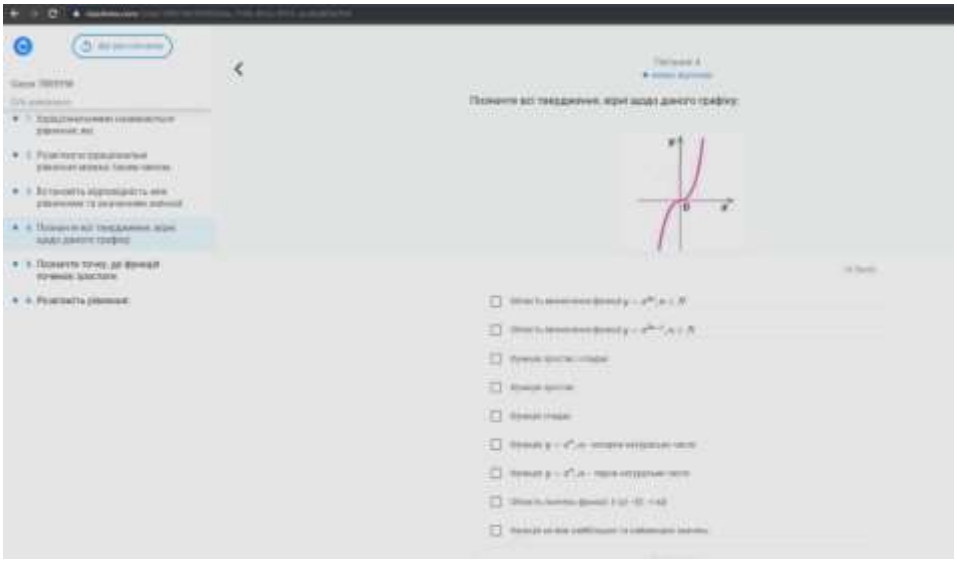


Рисунок 1.2 –Тестове завдання в «Classtime» [23]

Наступний проаналізований інструмент для тестування рівня засвоєних знань є «Moodle», який є безкоштовною, відкритою системою управління навчанням [24] (рисунок 1.3). Дана система дозволяє створювати тестові завдання аналогічних типів, що і попередньо розглянутий інструмент. Швидке та зрозуміле редагування тестових завдань, перегляд результатів тестів та обрахунок статистичних даних, базуючись на проведених тестуваннях, є перевагами даної системи. До плюсів можна віднести випадковий порядок завдань, але при цьому вони не є адаптивними, оскільки усі тестові завдання подаються одразу при початку тестування.

Система Moodle вимагає постійного та стабільного з'єднання з Інтернетом, тому не ефективно використовувати цю систему там, де є проблеми з підключенням до мережі. Вона побудована як серверна система і дозволяє одночасно запускати велику кількість користувачів, для чого потрібно мати потужний сервер, тобто це має під собою фінансові витрати як на придбання серверу так і на його обслуговування. Також процес тестування можна розпочати лише після того, як викладач надасть код доступу да нього.

Незважаючи на витрати на підтримку продуктивності, Moodle є найпоширенішою системою організації навчального процесу. Проте вагомим мінусом цієї системи є відсутність адаптивного методу тестування, яке б доповно його функції та зробило процес перевірки знань більш точним та об'єктивним.

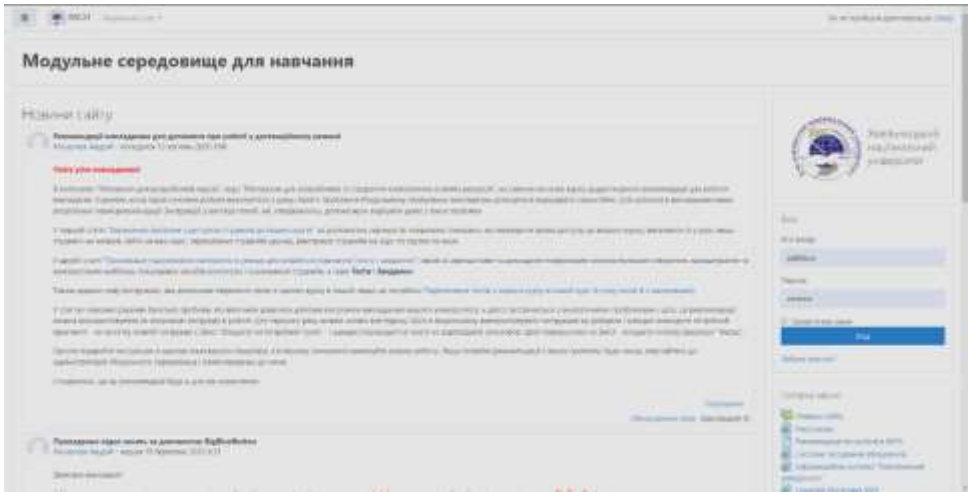


Рисунок 1.3 – Головна сторінка системи «Moodle» Хмельницького національного університету [24]

На рисунку 1.4 з представлено приклад тестового завдання під час моніторингу знань у системі «Moodle».

Рисунок 1.4 –Тестові завдання у системі «Moodle» [24]

Ще один інструментом, що було проаналізовано є система «Khan Academy» [25] (рисунок 1.5). Дана система є деякою бібліотекою тестів із різних галузей знань та предметів. Крім тестів у цій системі подано і навчальні матеріали у вигляді лекцій, наукові статті, відео-матеріали. Ще одним плюсом даної системи є можливість додати існуючу групу із Google Classroom. Незважаючи на достатньо велику базу навчальних матеріалів та тестів, у системі «Khan Academy» є відсутнім функціонал додавання нових тестів та редагування існуючих, що є важливим для подібних систем.

На рисунку 1.6 зображено приклад тестового завдання у системі «Khan Academy».

Також метод генерації порядковості тестів у системі «Khan Academy» не є адаптивним, тобто дана система не є гнучкою, що є одним із її недоліків.

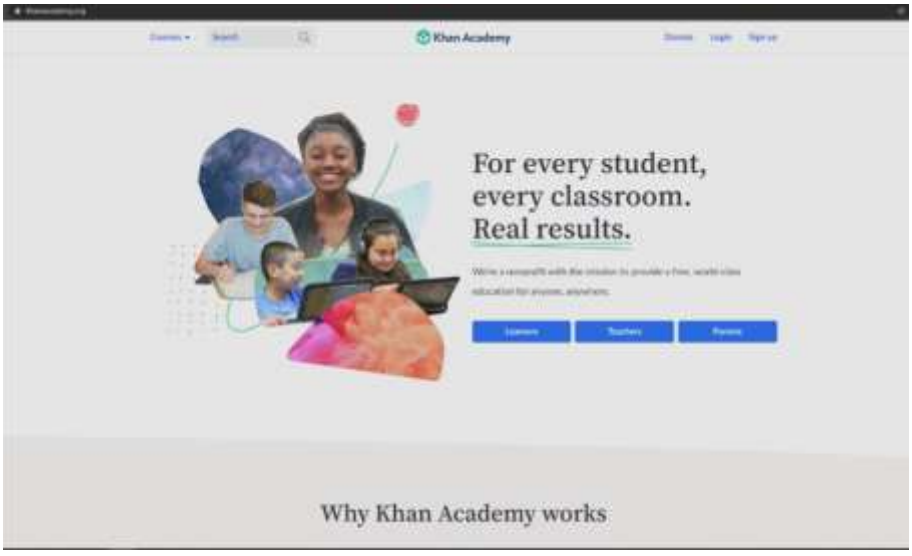


Рисунок 1.5 – Головна сторінка зайту «Khan Academy» [25]

Недоліком є англомовний інтерфейс, що сповільнить роботу користувачів із недостатнім рівнем володіння англійською. Не всі можливості сайту є безкоштовними.

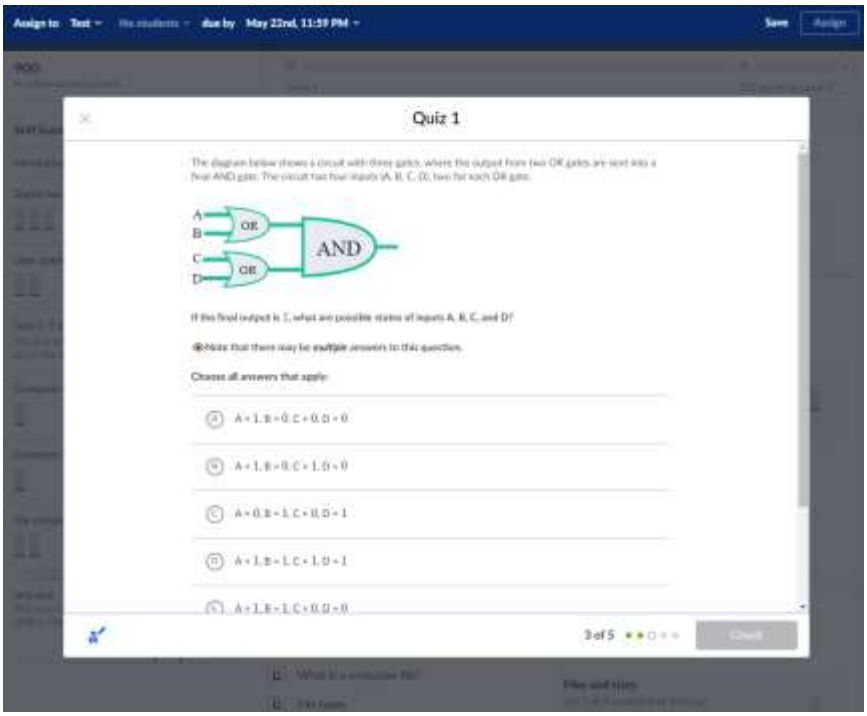


Рисунок 1.6 – Тестове завдання в системі «Khan Academy» [25]

Отже, результат аналізу визначив, що сучасні системи тестування рівня знань, у тому числі в складі навчальних середовищ, є широко вживаними та гнучкими у застосуванні. Пріоритетним завданням освіти є вдосконалення професійної підготовки студентів, яке можна в деякій мірі вирішити шляхом використання інформаційних технологій та програмних комп'ютерно-орієнтованих систем моніторингу та контролю знань. В результаті проведеного аналізу було зроблено висновок про необхідність застосування адаптивної форми тестування, що надасть належну об'єктивність процесу контролю рівня знань.

1.3 Аналіз існуючих алгоритмів адаптивного тестування

Адаптивний тестовий контроль знань дозволяє реалізувати один із основних принципів педагогіки – індивідуальний підхід, що є важливим для підготовки кваліфікованих фахівців. Запитання в адаптивному тесті не пропонуються у чітко визначеному порядку, а залежать від того, яку відповідь

дає студент на попереднє запитання. Переваги адаптивного тесту полягають у тому, що він дозволяє визначити компетенції (бали) того, хто проходить тестові випробування за допомогою меншої кількості питань, ніж тест класичного типу, інколи зменшуючи час проведення тестування до 40%. За рахунок цього той, хто проходить тест отримує більше часу для обмірковування кожного питання [26].

При адаптивному тестуванні рівня знань кількість завдань попередньо невідома. В загальному, якщо було дано правильну відповідь, наступне завдання буде більш складного характеру, при неправильній відповіді – рівень складності наступного завдання буде нижчим. Процес триває доти, доки тестова система не виявить рівень знань реципієнта.

Існує декілька алгоритмів проведення тестування адаптивним шляхом (додаток А). Перший [20] із них передбачає, що студентам на початку тестування даються завдання середнього рівня складності й далі, залежно від відповідей, визначається складнішими чи легшими будуть подальші завдання.

Наступний варіант алгоритму адаптивного тестування [15] починається із перевірки найважливіших термінів ІНМ. У випадку, коли студент відповідає правильно, складність тестових завдань підвищується шляхом перевірки менш важливих термінів.

Відомо також алгоритм [20] при якому існує база знань, що містить у собі розподілені за рівнем складності тестові завдання. При неправильній відповіді наступне питання обирається із вибірки більш легших завдань, при правильній – з більш складних.

Ще один із можливих варіантів здійснення адаптивного тестування рівня знань [20] базується на обході за структурою НК зверху вниз. Тобто, у випадку неправильної відповіді на найлегші тестові завдання, тестування у частині структури, що розташовується нижче – не проводиться.

Наступний різновид алгоритму [20] передбачає, що контроль знань починається з будь-якого рівня складності тестових завдань й далі ітераційним

методом наближається до рівня складності, що є відповідним реальному рівню знань студента.

Отже, існує декілька алгоритмів контолю засвоєння навчальних матеріалів у формі адаптивного тестування рівня знань як технології управління якістю освіти, що дозволяє провести дослідження задля порівняння ефективності наведених вище підходів.

1.4 Застосування адаптивного тестування рівня знань

Адаптивне тестування може використовуватися для найрізноманітніших цілей, включаючи оцінювання, яке надає постійний зворотній зв’язок щодо навчання студентів, що може використовуватися для модифікації методики навчання; і підсумкове оцінювання, яке викладачі використовують для визначення того, що учні засвоїли наприкінці теми, семестру чи навчального року. Вони також використовуються для ідентифікації студентів, яким може знадобитися спеціалізована академічна підтримка з певної теми чи предметної області [27].

Загалом, адаптивне тестування вважається ефективним на основі таких обґрунтувань:

– використовуючи більш точні та ефективні методи оцінювання, які потребують менше часу для виконання, викладачі та студенти матимуть більше часу для викладання та навчання, отримуючи при цьому результати тесту, які є настільки ж точними, як класичні тести, або більш точними;

– тести пристосовують кожне запитання до знань та здібностей учасника тесту, усуваючи необхідність студентам боротися із надто складними питаннями або витрачати час на надто прості питання;

– тести можуть надати більш точну та швидко доступну інформацію про потреби в навчанні студентів, яку викладачі можуть використовувати для адаптації навчання та покращення академічної підтримки студентів;

– захищеність тесту покращується, оскільки не всі учасники тестів бачать однакові елементи;

– тести можуть допомогти визначити рівень навчання студента точніше, ніж іспити з фіксованим запитанням, особливо для студентів задовільного рівня знань та відмінників;

– тести можуть збільшити залучення учнів до процесу тестування і призвести до більш точних результатів, оскільки тести коротші, менш втомлюючі та краще відкалібровані під індивідуальні здібності студента.

Оскільки впровадження адаптивного тестування з метою контролю знань визнано перспективним, предметна область даного напрямку активно досліджується науковцями [15, 20, 21, 22, 28, 29].

Основні принципи, які мають бути притаманні адаптивному тестуванню, повинні збігатися з принципами особистісно-орієнтованого підходу до навчання: принципи гуманізму, мобільності, науковості [28]. Доцільність використання адаптивного контролю випливає з міркувань раціоналізації традиційного тестування. Підготовленому студенту немає необхідності давати легкі завдання, тому що висока ймовірність їх правильного розв’язування. Легкі завдання не мають помітного розвивального потенціалу, в той час як складні завдання у більшості студентів знижують навчальну мотивацію. При цьому через високу ймовірність неправильного рішення немає сенсу давати важкі завдання слабкому студенту. Використання завдань, що відповідають рівню підготовленості студента, істотно підвищує точність вимірювань і мінімізує час індивідуального контролю.

В той же час організація процесу тестування включає інтелектуальні операції із різною насиченістю творчої компоненти: від авторського бачення структури завдань із врахуванням складності до рутинної перевірки правильності їх розв’язання. Виділення та формалізація нетворчих компонентів процесу навчання із наступною їх автоматизацією дозволяє вивільняти час

викладача на посилення творчої компоненти, що покращує розвиток творчих задатків студентів [29].

Також перевагою адаптивного тестування є можливість внесення додаткових питань в області, яку особа знає не дуже добре для більш тонкого з'ясування рівня знань у цих галузях. Дана модель застосовується для тестування студентів за допомогою комп'ютерних засобів, тому що «вручну» неможливо заздалегідь розмістити стільки питань і в тому порядку, скільки і в якому вони повинні бути пред'явлені особі, що тестується [20].

Оскільки адаптивне тестування має доведену ефективність, існують інформаційні системи на основі принципів цієї форми контролю знань, що мають широке застосування у навчальному процесі. Серед них можна виділити ALEKS [30] та Knewton [31].

Інформаційна система ALEKS (рисунок 1.7) оцінює поточні знання студента, ставлячи йому невелику кількість питань (зазвичай 20-30) [30]. Система обирає кожне питання на основі відповідей на всі попередні запитання. Кожен студент, а отже, і кожен набір питань оцінювання, є унікальним. Неможливо передбачити питання, які будуть задані.

Найважливішою особливістю ALEKS є те, що система використовує штучний інтелект для детального відображення рівня знань кожного студента. Саме із застосуванням можливостей штучного інтелекту система визначає стосовно кожної окремо взятої теми чи студент засвоїв цю тему, якщо так, ALEKS визначає, чи готовий студент вивчити наступну тему на поточний момент. Коли студент працює під час курсу, система ALEKS періодично переоцінює знання студента, щоб переконатися у остаточному засвоєнні матеріалу вивчених тем.

На рисунку 1.8 зображено приклад демонстрації тестового завдання для студента під час моніторингу його знань у інформаційній системі «ALEKS».

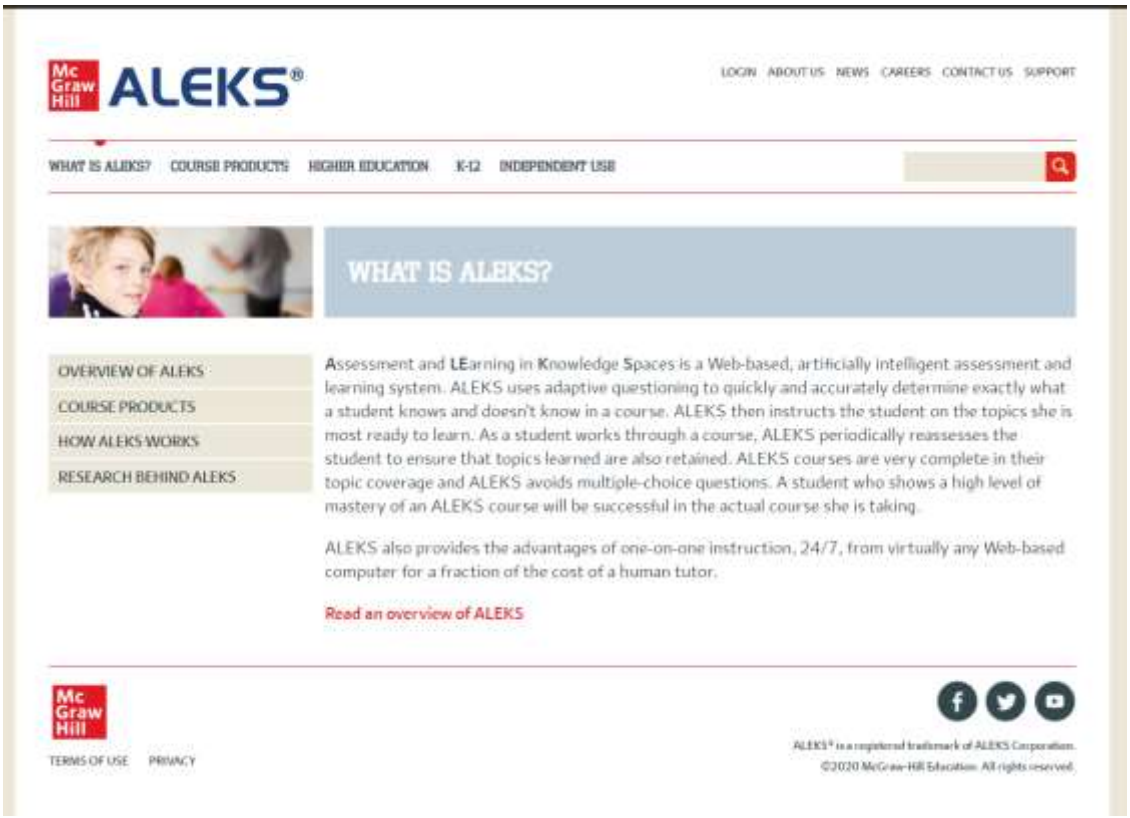


Рисунок 1.7 – Довідкова сторінка сайту «ALEKS» [30]

Рисунок 1.8 – Тестове завдання в системі «ALEKS» [30]

Інформаційна система «Knewton» (рисунок 1.9) базується на двох основних поняттях: технології планування освітньої траєкторії і складної моделі

оцінювання студента. Адаптивний підхід, реалізований в системі Knewton, реагує в реальному часі на результати окремого студента. Наприклад, якщо студент погано справляється з певною вибіркою тестових завдань, то система може встановити, які теми, представлені в цьому списку виявилися незрозумілими, та запропонувати йому контент, який допоможе підвищити рівень розуміння саме цих тем.

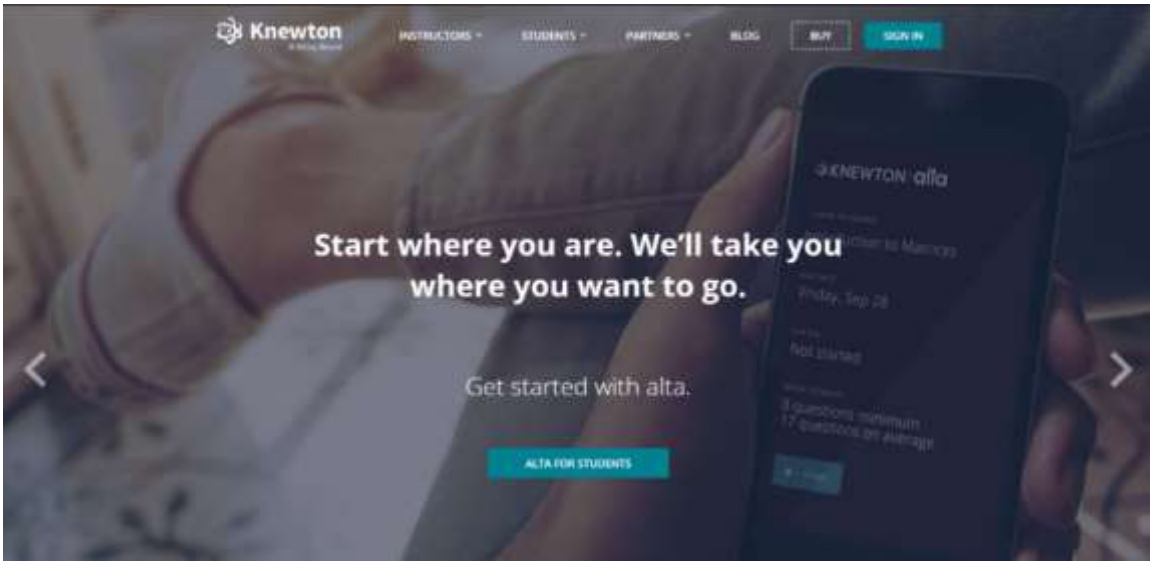


Рисунок 1.9 – Головна сторінка сайту «Knewton» [31]

Також дані про успішність студента аналізуються та створюються рекомендації викладачеві, яку тему потрібно пояснити студенту наступною.

На рисунку 1.10 зображено приклад інтерфейсу редагування тестового завдання викладачем у інформаційній системі «Knewton».

Щодо недоліків цих систем, то вони не мають безкоштовних функцій. Курси у цих системах представленні англійською (та іспанською у системі «ALEKS») мовою, що може створити труднощі у їх використанні. Також, хоча системи базуються на використанні штучного інтелекту у процесі адаптивного тестування, у ній не враховується структура ІНМ за яким відбувається тестування.

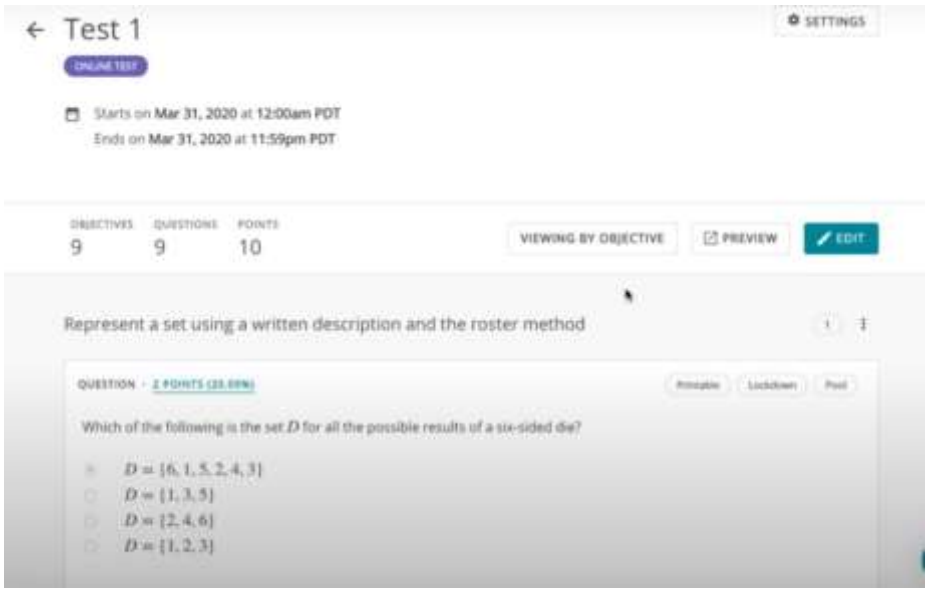


Рисунок 1.10 – Редагування тестового завдання в системі «Knewton» [31]

Отже, адаптивне тестування дозволяє не лише проводити оцінку знань студентів, а й індивідуалізує подання навчального матеріалу. Визначено, що жоден із проаналізованих підходів та систем адаптивного тестування не реалізує метод використання тестових завдань, сформованих на основі ключових термінів вжитих у ІНМ.

1.5 Постановка задачі

Метою дипломної роботи магістра є розробка інформаційної технології для адаптивного тестування знань, що використовує показники семантичної важливості ключових термінів інформаційного навчального матеріалу для адаптивного вибору тестових завдань у процесі тестування. Для досягнення мети необхідно розв’язати наступні *задачі*:

– дослідити відомі підходи до адаптивного тестування рівня знань;

– вдосконалити інформаційну модель адаптивного тесту для формального подання елементів терміноорієнтованого адаптивного тестування рівня знань;

– вдосконалити метод адаптивного вибору тестових запитань для динамічного вибору тестових завдань у процесі тестування;

– розробити інформаційну технологію адаптивного тестування рівня знань, що використовує вдосконалені модель і метод;

– розробити інформаційну систему адаптивного тестування рівня знань на основі розробленої інформаційної технології;

– дослідити практичну ефективність інформаційної технології адаптивного тестування рівня знань шляхом аналізу функціональності й ефективності застосування відповідної інформаційної системи.

Автоматизована система тестування, створена за розробленою інформаційною технологією адаптивного тестування рівня знань, має виконувати наступні основні функції:

– створення користувачів адміністратором (викладачів, студентів);

– формування множини тестових завдань до обраної рубрики викладачем;

– створення цільових вибірок тестових завдань викладачем;

– створення запиту на рівень знань викладачем;

– визначення рівня знань по запиту студентом;

– перевірка результатів проходження запитів на рівень знань викладачем.

Висновки до розділу 1

У розділі було розглянуто проблеми сучасного комп’ютерного тестування та досліджено відомі підходи до адаптивного тестування рівня знань. Контроль рівня знань студентів має опиратися на зміст та матеріал навчального курсу в рамках якого він проводиться. Одним із найважливіших показників високоякісної освітньої моделі є контроль рівня знань студентів на основі навчального матеріалу, що вивчається. Найбільш об'єктивним засобом

оцінювання рівня знань в даний час вважають тестування, що дозволяє неупереджено оцінити навчальні досягнення студентів.

На сучасному етапі виділено наступні проблеми комп'ютерного тестування за класичною формою: одержана студентом вибірка тестових завдань може не повністю покривати семантичну структуру навчального матеріалу; для повного покриття семантичної структури навчального матеріалу потрібна велика кількість тестових завдань; при оцінюванні не враховується зростання рівня складності тестових завдань через використання в контенті різних за рівнем семантичної значущості термінів; використання фіксованого обсягу тестових завдань незалежно від успішності процесу тестування. Таким чином, у процесі тестування слід виконувати перевірку розуміння студентом складових семантичної структури навчального матеріалу у вигляді системи заголовків та відповідних множин ключових термінів. Для цього є перспективним використання можливостей інформаційних технологій.

За проведеними дослідженнями було сформовано мету й завдання, які потрібно вирішити в рамках виконання магістерської дипломної роботи. Метою дипломної роботи магістра визначено розробку інформаційної технології для адаптивного тестування знань, що використовує показники семантичної важливості ключових термінів інформаційного навчального матеріалу для адаптивного вибору тестових завдань у процесі тестування. Також необхідним є дослідження практичної ефективності інформаційної технології адаптивного тестування рівня знань шляхом розробки й аналізу функціональності та ефективності використання відповідної інформаційної системи.

Розділ 2

Розробка інформаційної технології адаптивного тестування рівня знань та її компонентів

2.1 Інформаційна модель адаптивного тесту

Тестовий навчальний матеріал є найпоширенішою варіацією діагностуючого навчального матеріалу й використовується для визначення рівня засвоєння семантичної структури ІНМ (як системи заголовків) та відповідних множин ключових термінів шляхом паперового чи комп'ютерного тестування суб'єкта, що вивчає НК.

Структура інформаційної моделі тестового навчального матеріалу для адаптивного тестування має складатися з наступних множин:

- множина інформаційних навчальних матеріалів M_{IEM};
- множина заголовків ІНМ M_H;
- множина абзаців M_P;
- множина типів абзаців M_{Pt};
- множина речень M_S;
- множина типів речень M_{St};
- множина термінів M_{Term};
- множина ключових термінів M_{KT};
- множина появи терміну у реченнях M_{TS};
- множина тестових завдань M_{TTasks};
- множина типів тестових завдань $M_{TTaskst}$;
- множина відповідей M_A;
- множина вибірок тестових завдань $M_{STTasks}$;
- множина тестових завдань у вибірках $M_{TTasksS}$;
- множина спроб тестування M_{TA};
- множина типів тестування M_{Tt};

– множина результатів проходження одного тестового завдання $M_{R1TTask}$;

– множина користувачів M_U;

– множина типів користувачів M_{Ut}.

Тобто подання інформаційної моделі адаптивного тесту (AT) набуває наступного вигляду:

$$\{M_{IEM} \cup M_H \cup M_P \cup M_{Pt} \cup M_S \cup M_{St} \cup M_{Term} \cup M_{KT} \cup$$
$$\cup M_{TS} \cup M_{TTasks} \cup M_{TTaskst} \cup M_A \cup M_{STTasks} \cup \quad (2.1)$$
$$\cup M_{TTaskssS} \cup M_{TA} \cup M_{Tt} \cup M_{R1TTasks} \cup M_U \cup M_{Ut}\} \subset AT$$

Кожен із елемент відповідних вищеподаних множин в свою чергу має набір параметрів (властивостей), що описують цю множину. Параметрами множини M_{IEM} є унікальний ідентифікатор *ID* та назва ІНМ *Name*, що можна подати у вигляді:

$$M_{IEM} = \{ID, Name\}. \quad (2.2)$$

Властивості множини M_H описуються наступною формулою:

$$M_H = \{ID, Name, Level, Heading\_ID, IEM\_ID\}, \quad (2.3)$$

де *ID* – унікальний ідентифікатор заголовку, *Name* – назва, *Level* – рівень, *Heading_ID* – ідентифікатор заголовку, якому підпорядковується поточний, *IEM_ID* – унікальний ідентифікатор ІНМ.

Множина абзаців M_P характеризується такими властивостями як ідентифікатор *ID*, текст *Text*, номер у межах заголовку *Number*, ідентифікатор заголовку *Heading_ID* та тип абзацу *Type_ID*:

$$M_P = \{ID, Text, Number, Heading\_ID, Type\_ID\}. \quad (2.4)$$

Множина типів абзаців M_{Pt} має наступні параметри: ідентифікатор *ID* та назву *Name* абзацу. Дані властивості можна подати у вигляді:

$$M_{Pt} = \{ID, Name\}. \quad (2.5)$$

Параметрами множини M_S є унікальний ідентифікатор *ID*, текст речення *Text*, номер речення в абзаці *Number*, ідентифікатор абзацу *Paragraph_ID* та ідентифікатор типу речення *Type_ID*, що можна подати у вигляді:

$$M_S = \{ID, Text, Number, Paragraph\_ID, Type\_ID\}. \quad (2.6)$$

Множина типів абзаців M_{St} характеризується такими властивостями як унікальний ідентифікатор *ID* та назва типу *Name*:

$$M_{St} = \{ID, Name\}. \quad (2.7)$$

Властивостями множини M_{Term} є унікальний ідентифікатор *ID* та назва терміну *Name*, що можна описати у вигляді:

$$M_{Term} = \{ID, Name\}. \quad (2.8)$$

Параметри множини ключових термінів M_{KT} можна представити наступною формулою:

$$M_{KT} = \{ID, Term\_ID, Heading\_ID, Estimation\}, \quad (2.9)$$

де *ID* – унікальний ідентифікатор заголовку, *Term_ID* – ідентифікатор терміну, *Heading_ID* – унікальний ідентифікатор заголовку, до якого відноситься поточний ключовий термін, *Estimation* – оцінка важливості ключового терміну.

Множина появи терміну у реченнях M_{TS} характеризується такими властивостями як унікальний ідентифікатор *ID*, поточна форма терміну *CurrentTermForm*, ідентифікатор терміну *Term_ID* та ідентифікатор речення *Sentence_ID*:

$$M_{TS} = \{ID, CurrentTermForm,\ Term\_ID, Sentence\_ID\}. \quad (2.10)$$

Параметрами множини M_{TTasks} є унікальний ідентифікатор *ID*, текст тестового завдання *Text*, ідентифікатор ключового терміну за яким сформоване завдання *KeyTerm_ID*, ідентифікатор речення *Sentence_ID*, що є основою поточного завдання, та ідентифікатор типу тестового завдання *Type_ID*, що можна представити у вигляді наступної формули:

$$M_{TTasks} = \{ID, Text, KeyTerm\_ID, Sentence\_ID, Type\_ID\}. \quad (2.11)$$

Множина типів тестових завдань $M_{TTaskst}$ описується такими властивостями як унікальний ідентифікатор *ID*, назва типу *Name*:

$$M_{TTaskst} = \{ID, Name\}. \quad (2.12)$$

Властивості множини відповідей M_A характеризуються наступною формулою:

$$M_A = \{ID, Text, Sentence\_ID, TestTask\_ID, Correctness\}, \quad (2.13)$$

де *ID* – унікальний ідентифікатор, *Text* – текст відповіді, *Sentence_ID* – ідентифікатор речення, за яким сформовано відповідь, *TestTask_ID* – ідентифікатор завдання до якого відноситься відповідь, *Correctness* – коректність відповіді.

Параметрами множини вибірок тестових завдань $M_{STTasks}$ є унікальний ідентифікатор *ID*, назва вибірки *Name*, ідентифікатор заголовку *Heading_ID*, до якого відноситься вибірка, ідентифікатор викладача *Teacher_ID*, що сформував вибірку та дата і час створення вибірки *DateAndTimeCreation*, що можна подати у вигляді:

$$M_{STTasks} = \{ID, Heading\ ID, Name, Teacher\_ID, DateAndTimeCreation\}. \quad (2.14)$$

Множина тестових завдань у вибірках $M_{TTasksS}$ має наступні параметри: унікальний ідентифікатор *ID*, ідентифікатор вибірки тестових завдань *SelectionOfTestTasks_ID* та ідентифікатор тестового завдання *TestTask_ID*. Дані властивості можна подати у вигляді:

$$M_{TTasksS} = \{ID, SelectionOfTestTasks\_ID, TestTask\_ID\}. \quad (2.15)$$

Множина спроб тестування M_{TA} представляється такими властивостями як унікальний ідентифікатор *ID*, дата та час створення заявки *DateAndTimeAttemptCreation*, дата та час початку спроби *StartDateAndTime*, дата та час завершення спроби *FinishDateAndTime*, ідентифікатор вибірки тестових завдань *SelectionOfTestTasks_ID*, ідентифікатор типу тестування *Type_ID*, ідентифікатор викладача *Teacher_ID*, який створив поточну спробу, ідентифікатор студента *Student_ID*, для якого призначена спроба, показник чи студенту була надіслана спроба до виконання *IsGivenToStudentToPerform*, повідомлення *Message* від викладача студенту, максимальний час *MaxTime* на виконання тесту. Властивості, що визначаються безпосередньо після завершення тестування є оцінка *Score*, результат *Result*, кількість правильних відповідей *NumberOfCorrectAnswers*, кількість неправильних відповідей *NumberOfIncorrectAnswers*, показник затвердження спроби *IsApproved*

викладачем. Описані вище властивості можна представити у вигляді наступної формули:

$$M_{TA} = \{ID, DateAndTimeAttemptCreation, StartDateAndTime, FinishDateAndTime, SelectionOfTestTasks\_ID, Type\_ID, Teacher\_ID, Student\_ID, IsGivenToStudentToPerform, Message, Score, Result, NumberOfCorrectAnswers, NumberOfIncorrectAnswers, IsApproved, MaxTime\}. \quad (2.16)$$

Множина типів тестування M_{Tt} описується такими властивостями як унікальний ідентифікатор *ID*, назва типу *Name*:

$$M_{Tt} = \{ID, Name\}. \quad (2.17)$$

Властивостями множини результатів проходження одного тестового завдання $M_{R1TTask}$ є унікальний ідентифікатор *ID*, ідентифікатор спроби *Attempt_ID*, ідентифікатор тестового завдання *TestTask_ID*, результат *Result*, що можна описати у вигляді:

$$M_{R1TTasks} = \{ID, Attempt\_ID, TestTask\_ID, Result\}. \quad (2.18)$$

Параметри множини користувачів M_U представляються наступною формулою:

$$M_U = \{ID, Login, Password, Type\_ID, LastName, FirstName, SurName, DateAndTimeCreation, PhoneNumber, Activity\}, \quad (2.19)$$

де *ID* – унікальний ідентифікатор, *Login* – логін, *Password* – пароль, *Type_ID* – ідентифікатор типу користувача, *LastName* – прізвище, *FirstName* – ім’я, *SurName* – по батькові користувача, *DateAndTimeCreation* – дата та час створення користувача, *PhoneNumber* – телефон, *Activity* – активність.

Множина типів користувачів M_{Ut} має наступні параметри: унікальний ідентифікатор *ID* та назву *Name*. Дані властивості можна подати у вигляді:

$$M_{Ut} = \{ID, Name\}. \quad (2.20)$$

Таким чином, сформовано формальний опис інформаційної моделі адаптивного тестування рівня знань, що містить у собі множину ІНМ M_{IEM}, множину заголовків ІНМ M_H, множину абзаців M_P, множину типів абзаців M_{Pt},

множину речень M_S, множину типів речень M_{St}, множину термінів M_{Term}, множину ключових термінів M_{KT}, множину появи терміну у реченнях M_{TS}, множину тестових завдань M_{TTasks}, множину типів тестових завдань $M_{TTaskst}$, множину відповідей M_A, множину вибірок тестових завдань $M_{STTasks}$, множину тестових завдань у вибірках $M_{TTasksS}$, множину спроб тестування M_{TA}, множину типів тестування M_{Tt}, множину результатів проходження одного тестового завдання $M_{R1TTask}$, множину користувачів M_U, множину типів користувачів M_{Ut}.

2.2 Метод адаптивного вибору тестових запитань

Метод адаптивного вибору тестових запитань грунтується на даних множин запропонованої інформаційної моделі адаптивного тесту. Загальну схему методу подано на рисунку 2.1.

Вхідними даними розробленого методу є елементи інформаційної моделі адаптивного тесту. На *Кроці 1* із множини ключових термінів M_{KT} обирається поточний термін, розуміння якого буде перевірятися на n-ій ітерації. Далі, коли термін було обрано, на *Кроці 2* формується вибірка всіх тестових завдань в яких є речення, що містять цей термін. Важливою умовою при цьому є те, що ці речення не повинні містити менш важливий термін.

Відібрана множина передається на *Крок 3*, де відбувається селекція тестових завдань за кількома критеріями. Спочатку у вибірку потрапляють тестові завдання, які відповідають умові, що речення, з яких вони були сформовані, ще не використовувалися у процесі тестування. Після того, від множини тестових завдань, що залишилися, відкидаються ті, що уже були задіяні під час поточної сесії тестування. Далі вибірка відбувається відносно типів тестових завдань, що найменше використовувалися. Серед тестових завдань у кінцевій вибірці те, що буде задане, обирається методом рандому.

На *Кроці 4* безпосередньо відбувається процес тестування: тестове запитання надається для опитуваного та фіксується отримана відповідь.

Базуючись, на коректності відповіді, яку дав студент на тестове запитання на *Кроці 5* приймається рішення про рівень важливості ключового терміну, який буде перевірятися наступним: якщо відповідь була правильною, то важливість чергового терміну має бути нижчою, ніж поточного, якщо неправильною – важливість має бути вищою.

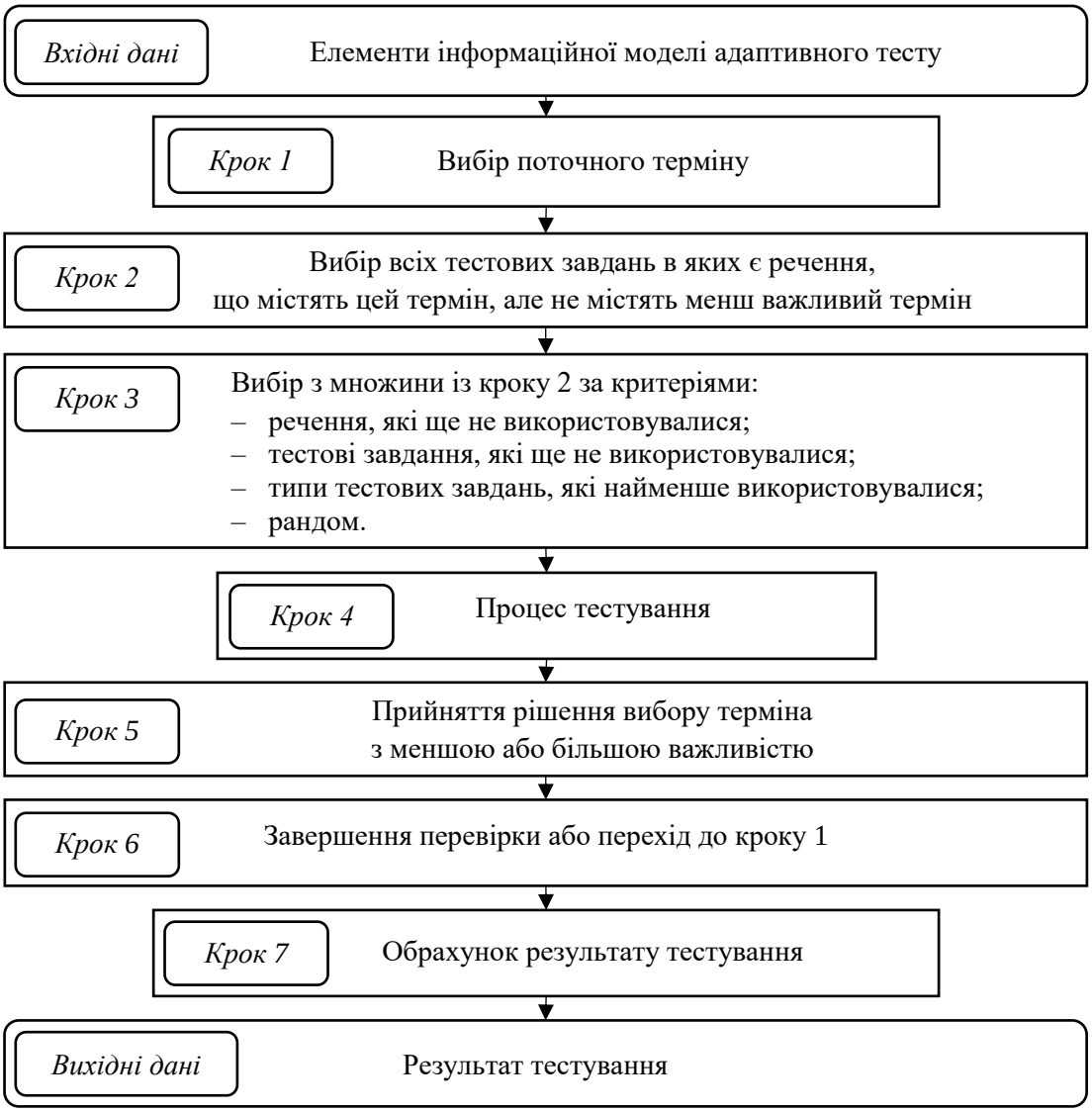


Рисунок 2.1 – Схема методу адаптивного вибору тестових запитань

На *Кроці 6* відбувається завершення перевірки знань або ж здійснюється перехід до *Кроку 1*. За умови завершення процесу тестування на *Кроці 7* обраховується результат на основі кількості даних правильних та неправильних

відповідей на отриманні тестові запитання. *Вихідними даними* методу адаптивного вибору тестових запитань є результат тестування.

Вибір поточного терміну та процес тестування потребують розробки відповідного алгоритмічного забезпечення, що детально описуватиме ці кроки. Використання даного методу дозволяє проводити тестування із врахуванням показників семантичної важливості ключових термінів ІНМ для адаптивного вибору тестових завдань у процесі тестування.

2.3 Алгоритмічне забезпечення для методу адаптивного вибору тестових запитань

Для забезпечення процесу адаптивного тестування, важливим є вибір першого терміну, що буде перевірятися. Можливі три варіанти вибору початкового терміну (типів адаптивного тестування). При першому із них визначається найважливіший термін у межах заголовку і саме він обирається початковим. Далі у процесі тестування будуть задіяні терміни, що мають меншу важливість, ніж початковий термін. Описаний алгоритм називатимемо *регресивним*.

Другий варіант вибору початкового терміну полягає у визначенні найменш важливого терміну в межах заголовку, що береться як початковий. Далі під час тестування терміни, що використовуватимуться, будуть мати більшу важливість, аніж початковий термін. Запропонований алгоритм називатимемо *прогресивним*.

Останній варіант вибору початкового терміну, що матиме назву *медіанного* полягає в тому, що в межах заголовку буде обрано термін, що має середній рівень важливості. Саме з цього терміну буде починатися контроль рівня знань. За наступних ітерацій чергові терміни матимуть більшу або меншу важливість від середнього, що залежить від відповіді даної на поточне тестове запитання.

На початку алгоритму методу адаптивного вибору тестових запитань (рисунок 2.2) визначається за вказаним типом тестування початковий термін: якщо обрано регресивний алгоритм, то початковим терміном буде той, що має найбільшу важливість; якщо обрано прогресивний алгоритм, то початковим терміном буде той, що має найменшу важливість; якщо обрано медіанний алгоритм, то початковим терміном буде той, що має середній рівень важливості.

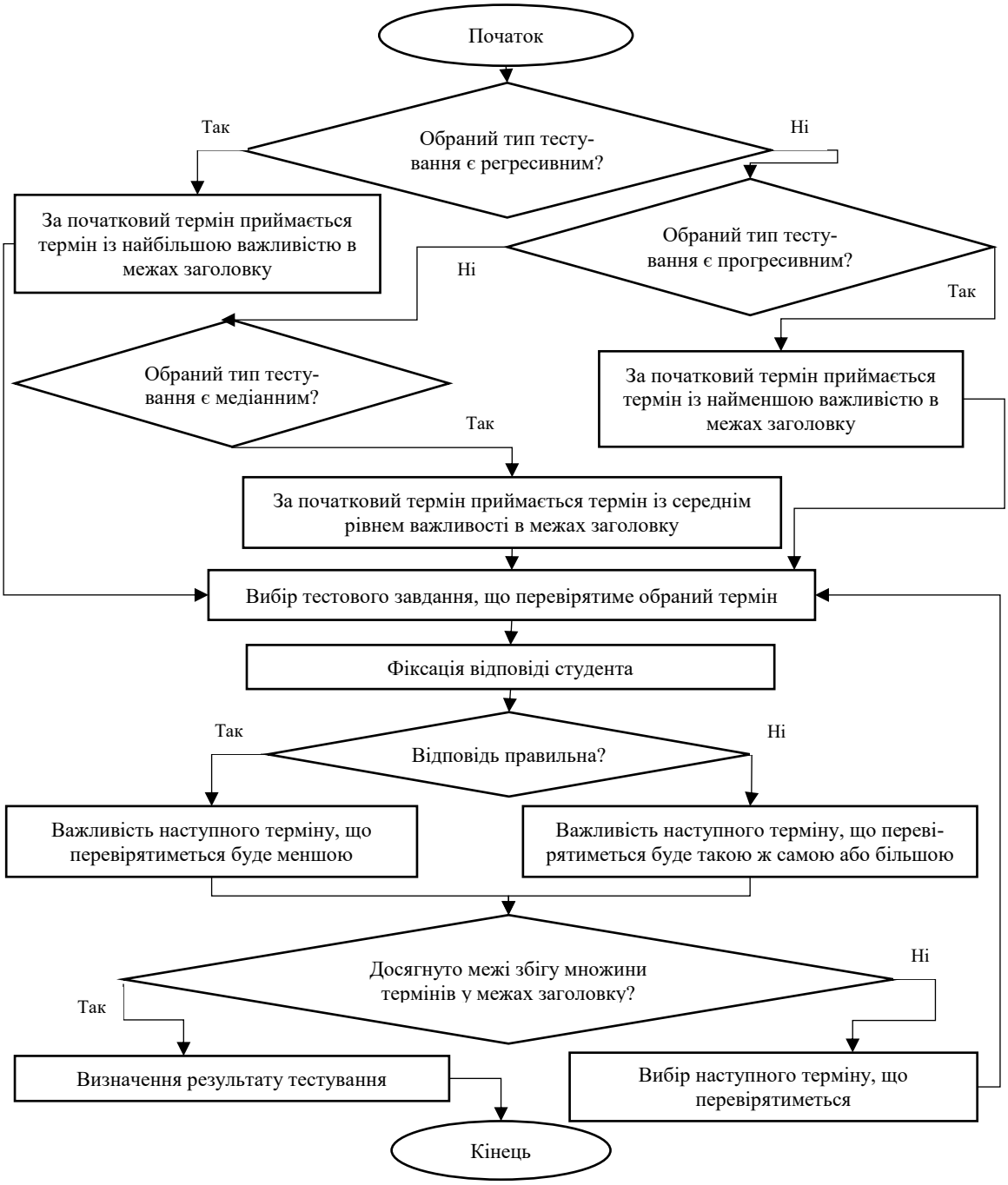


Рисунок 2.2 – Схема адаптивного алгоритму вибору тестових запитань

Далі, базуючись на визначеному початковому ключовому терміні, здійснюється вибір тестового завдання, що його перевірятиме. Наступним кроком є фіксація відповіді студента на обране тестове завдання та залежно від того чи є дана відповідь правильна визначається подальша дія. Якщо відповідь коректна, то важливість наступного терміну, що перевірятиметься, буде меншою за поточну. Якщо відповідь некоректна, то важливість наступного терміну, що перевірятиметься, буде або такою ж самою, або більшою за поточну.

На наступному етапі перевіряється, чи досягнута межа збігу множини термінів у межах заголовку. Якщо межа недосягнута, то відбуваться вибір наступного терміну для перевірки і алгоритм повертається на етап вибору тестового завдання, що перевірятиме обраний термін. Якщо межа досягнута, то обраховується результат тестування та виконання алгоритму завершується.

Алгоритми адаптивного вибору до обробки елементів у залежності від результату аналізу властивостей попередніх елементів, були успішно реалізовані для ряду аналогічних задач [40]. У даному випадку, в залежності від вибору одного з розроблених алгоритмів адаптивного тестування, визначається критерій вибору першого терміну, що буде перевірятися.

2.4 Схема та кроки інформаційної технології адаптивного тестування рівня знань

Інформаційна технологія адаптивного тестування рівня знань (додаток Б) використовує запропоновану у п. 2.1 інформаційну модель та представлений у п. 2.2 метод адаптивного вибору тестових завдань.

Кроки інформаційної технології адаптивного тестування рівня знань зображенно на рисунку 2.3. *Вхідними даними* інформаційної технології є поточний елемент семантичної структури m'_{H}, тип адаптивного тестування t та множина тестових завдань $M_{ТЗ}$. Алгоритм адаптивного тестування

обумовлюється семантичною важливістю початкового ключового терміну (найбільш важливий, найменш важливий, середньої важливості), що була визначена методом дисперсійного оцінювання [32].

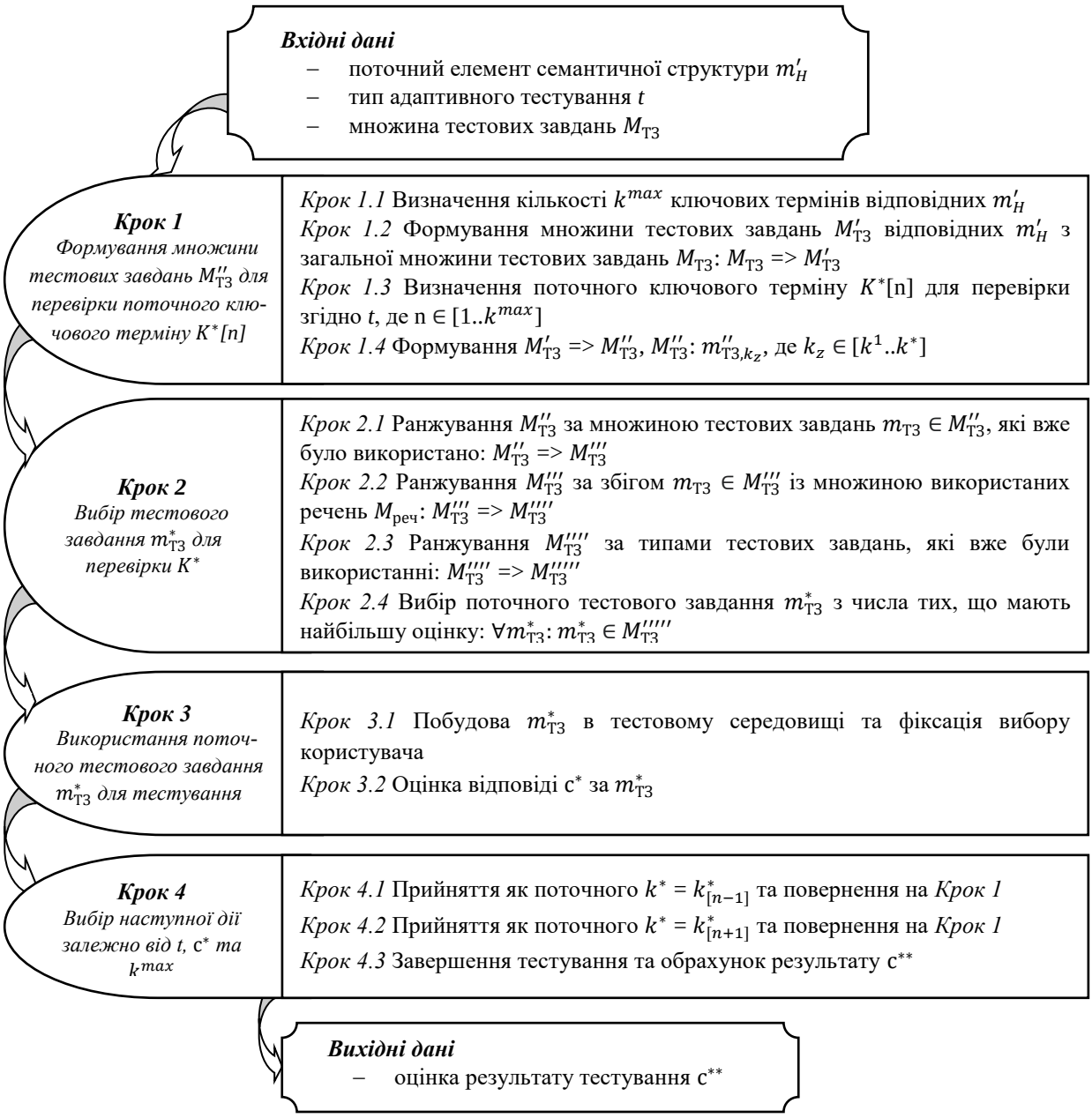


Рисунок 2.3 – Кроки інформаційної технології адаптивного тестування рівня знань

На *Кроці 1* формується множина тестових завдань $M''_{\text{ТЗ}}$ для перевірки поточного ключового терміну K^*[n]. Для цього на *Кроці 1.1* визначається

кількість k^{max} ключових термінів відповідних актуальному елементу семантичної структури m'_H. Далі на *Кроці 1.2* утворюється множина тестових завдань $M'_{\text{ТЗ}}$ відповідних актуальному елементові семантичної структури m'_H з загальної множини тестових завдань $M_{\text{ТЗ}}$ ($M_{\text{ТЗ}}$ => $M'_{\text{ТЗ}}$) та на *Кроці 1.3* визначається поточний ключовий термін K^*[n] для перевірки згідно типу адаптивного тестування t, де n ∈ [1..k^{max}]. Завершальним є *Крок 1.4* на якому формується множина тестових завдань $M''_{\text{ТЗ}}$ ($M'_{\text{ТЗ}}$ => $M''_{\text{ТЗ}}$, $M''_{\text{ТЗ}}$: $m''_{\text{ТЗ},k_z}$, де $k_z \in [k^1..k^*]$).

На *Кроці 2* інформаційної технології відбувається вибір тестового завдання $m^*_{\text{ТЗ}}$ для перевірки поточного ключового терміну K^*. Множина тестових завдань $M''_{\text{ТЗ}}$ ранжується на *Кроці 2.1* за множиною тестових завдань $m_{\text{ТЗ}} \in M''_{\text{ТЗ}}$, які вже було використано, що формує множину $M'''_{\text{ТЗ}}$ ($M''_{\text{ТЗ}}$ => $M'''_{\text{ТЗ}}$). Тестові завдання із множини $M'''_{\text{ТЗ}}$ в свою чергу на *Кроці 2.2* сортуються за збігом $m_{\text{ТЗ}} \in M'''_{\text{ТЗ}}$ із множиною використаних речень $M_{\text{реч}}$ таким чином утворюючи множину $M''''_{\text{ТЗ}}$ ($M'''_{\text{ТЗ}}$ => $M''''_{\text{ТЗ}}$). Далі на *Кроці 2.3* множина тестових завдань $M''''_{\text{ТЗ}}$ ранжується за типами тестових завдань, які вже були використанні, що зумовлює формування множини $M'''''_{\text{ТЗ}}$ ($M''''_{\text{ТЗ}}$ => $M'''''_{\text{ТЗ}}$). На *Кроці 2.4* відбувається вибір поточного тестового завдання $m^*_{\text{ТЗ}}$ з числа тих, що мають найбільшу оцінку $\forall m^*_{\text{ТЗ}}$: $m^*_{\text{ТЗ}} \in M'''''_{\text{ТЗ}}$.

Використання поточного тестового завдання $m^*_{\text{ТЗ}}$ для тестування здійснюється на *Кроці 3*. При цьому відбувається формування $m^*_{\text{ТЗ}}$ в тестовому середовищі, фіксується вибір (відповідь) користувача на *Кроці 3.1* та відбувається оцінка відповіді c^* за поточне тестове завдання $m^*_{\text{ТЗ}}$ на *Кроці 3.2*.

На *Кроці 4* обирається подальша дія залежно від типу адаптивного тестування t, оцінки відповіді c^* та загальної кількості ключових термінів k^{max}: поточний ключовий термін приймається як $k^* = k^*_{[n-1]}$ на *Кроці 4.1* чи $k^* = k^*_{[n+1]}$ на *Кроці 4.2* та відбувається повернення на *Крок 1*. Або на *Кроці 4.3* тестування завершується та обраховується результат c^{**}. *Вихідними даними* інформаційної

технології є оцінка результату тестування c^{**}, що вираховується залежно від відповідей опитуваного впродовж тестування.

Отже, сформована інформаційна технологія, використовуючи інформаційну модель та метод адаптивного тесту, дозволяє проводити контроль рівня знань студентів на основі семантичної структури ІНМ. Для дослідження ефективності запропонованої інформаційної технології потрібно розробити відповідну інформаційну систему адаптивного тестування рівня знань, дотримуючись поданих вище кроків.

Висновки до розділу 2

В розділі було сформовано інформаційну модель адаптивного тесту для формального подання елементів терміноорієнтованого адаптивного тестування рівня знань. Одержана інформаційна модель адаптивного тесту відрізняється тим, що містить формальне подання складових предметної області терміноорієнтованого адаптивного тестування рівня знань. Було вдосконалено метод адаптивного вибору тестових запитань для динамічного вибору тестових завдань у процесі тестування. Цей метод характерний тим, що використовує показники семантичної важливості ключових термінів навчального матеріалу для динамічного вибору тестових завдань у процесі тестування.

Також було розроблено інформаційну технологію адаптивного тестування рівня знань, що використовує вдосконалені інформаційну модель і метод адаптивного вибору тестових запитань й дозволяє за вхідними даними у вигляді пов’язаних із семантичною структурою навчального курсу тестових завдань одержувати в результаті проходження тесту вихідні дані у вигляді оцінки результату тестування, що вираховується залежно від відповідей опитуваного впродовж тестування. Розроблено три алгоритми реалізації методу адаптивного вибору тестових запитань: регресивний, прогресивний та медіанний.

Розділ 3

Розробка інформаційної системи адаптивного тестування рівня знань

3.1 Етапи роботи та функції інформаційної системи адаптивного тестування рівня знань

Основною метою створення інформаційної технології адаптивного тестування знань та системи на її основі є автоматизація процесу тестування із адаптивною складністю завдань, що дає можливість об’єктивного контролю та ефективного управління навчальним процесом.

Користувачами інформаційної системи адаптивного тестування є викладачі та студенти. Також окремим користувачем варто виділити адміністратора, який буде вносити дані про існуючих користувачів та створювати нових.

Робота автоматизованої системи адаптивного тестування знань складається з шести ключових етапів (рисунок 3.1): створення користувачів адміністратором (викладачів, студентів), формування множини тестових завдань до обраної рубрики викладачем, створення цільових вибірок тестових завдань викладачем, створення запиту на рівень знань викладачем, визначення рівня знань по запиту студентом, перевірка результатів проходження запитів на рівень знань викладачем. Етапи пов’язані між собою або через обов’язкову умову дотримання послідовності, або тому що набір вихідних даних попереднього етапу є набором вхідних даних наступного. На кожному з представлених шести етапів роботи автоматизованої системи, обмін даними відбувається з використанням бази даних.

Як було вказано вище, у системі адаптивного тестування рівня знань є три типи користувачів. Кожен тип користувача має притаманні лише йому функції.

До функціоналу, що доступний адміністратору відноситься робота з користувачами, а саме:

– створення нових користувачів;

– редагування даних існуючих користувачів;

– блокування/розблокування існуючих користувачів.

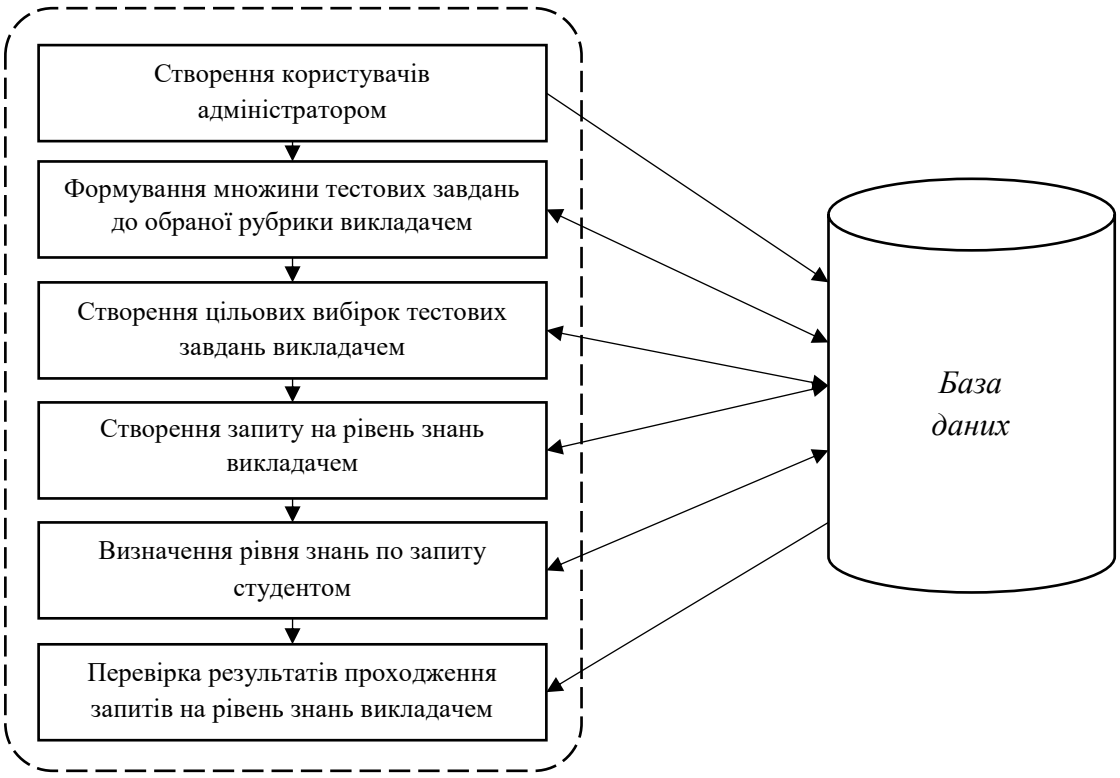


Рисунок 3.1 – Основні етапи роботи автоматизованої системи адаптивного тестування знань

До функцій викладача у інформаційній системі можна віднести роботу з тестовими завданнями, роботу із вибірками тестових завдань, роботу із проходженням тестів. До категорії роботи із тестовими завданнями належить:

– перегляд існуючих тестових завдань;

– редагування існуючих тестових завдань;

– створення нових тестових завдань.

До категорії роботи з вибірками тестових завдань відносяться:

– перегляд існуючих вибірок тестових завдань;

– редагування існуючих вибірок тестових завдань;

– створення нових вибірок тестових завдань.

Підфункціями категорії роботи з проходженням тестів є:

– планування проходження тесту (вибір типу тестування, написання повідомлення);

– скасування проходження тесту;

– перегляд результатів (архів (затвердженні), непройденні, незатвердженні);

– перегляд результатів проходження тестових завдань (правильна/неправильна відповідь).

Серед функцій, що доступні користувачу студент є лише робота з проходженням тестів, що включає в себе наступні підфункції:

– проходження тестування;

– перегляд результатів (архів (затвердженні), непройденні, незатвердженні).

Наступним етапом у створенні інформаційної системи адаптивного тестування рівня знань є розробка її структури відповідно до зазначених вище етапів роботи системи та функцій, що доступні трьом типам користувачів даної системи (адміністратор, викладач та студент).

3.2 Розробка структури інформаційної системи адаптивного тестування рівня знань

Суть інформаційних технологій становлять методи і засоби формування та підтримки інформаційних потоків, їх структура у системах управління об'єктами, у тому числі, при адаптивному тестуванні рівня знань.

Відповідно до поставлених завдань загальну схему роботи автоматизованої системи адаптивного тестування рівня знань можна представити у вигляді функціональних блоків IDEF0 системи, що зображена на рисунку 3.2.

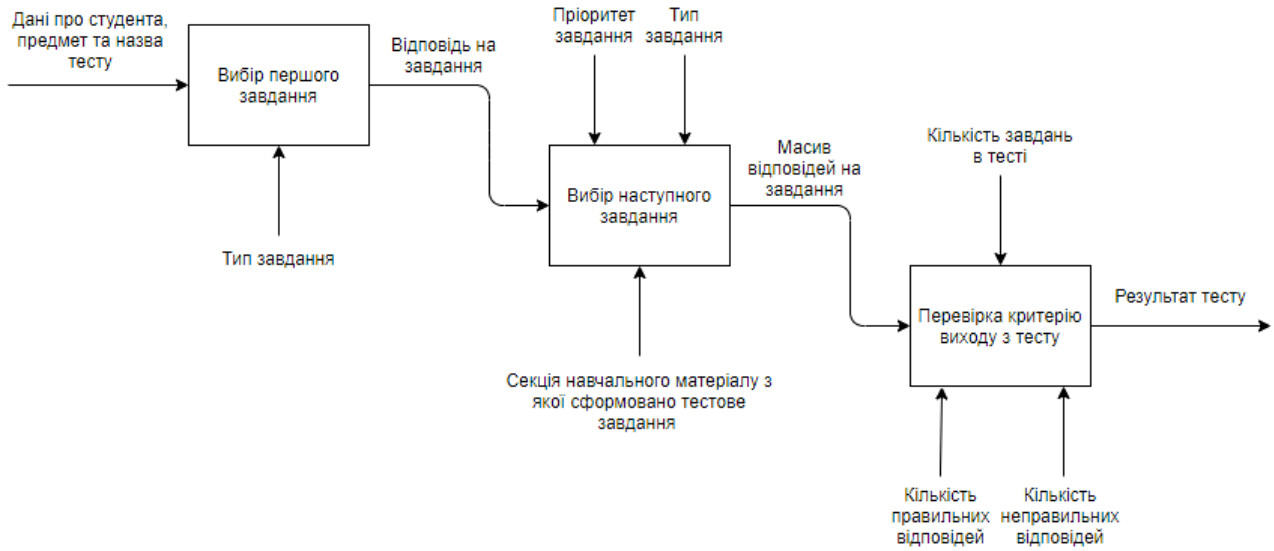


Рисунок 3.2 – IDEF0-модель роботи автоматизованої системи адаптивного тестування рівня знань

Передбачається, що автоматизована система адаптивного тестування рівня знань буде складатися з двох основних складових: програмного додатку та бази даних. Програмний додаток міститиме класи, що відповідатимуть за інтерфейс користувача та класи програмно-апаратної частини сервера: класи компонентів обробки даних з форми входу у систему, класи взаємодії з базою даних, класи роботи з користувачами, класи роботи з тестовими завданнями, класи роботи з вибірками тестових завдань, класи роботи з проходженням тестування, класи безпосереднього керування процесом адаптивного тестування, класи аналізу відповідей на тестові запитання.

Отже, відповідно до поставлених завдань, було скомпоновано схему автоматизованої інформаційної системи адаптивного тестування рівня знань, призначеної для її реалізації та модель взаємодії компонентів (рисунок 3.3).

На початковому етапі функціонування системи відбувається створення користувачів адміністратором. Адміністратор також може оновлювати дані про існуючих користувачів та регулювати активність користувачів у системі. Далі викладач, використовуючи підсистему формування множини тестових задань до

обраного заголовку, обирає семантичний рівень для перевірки знань, операцію, що виконуватиме над тестовими завданнями та коригує семантичну структуру інформаційного навчального матеріалу (через перевизначення показників важливості термінів, видалення термінів з числа ключових).

Рисунок 3.3 – Схема інформаційної системи адаптивного тестування рівня знань

На наступному кроці викладачем створюються цільові вибірки тестових завдань. До існуючих вибірок можна додавати або видаляти тестові завдання. Далі відбувається формування запиту на рівень знань викладачем при цьому

додається повідомлення та визначається тип тестування. До функціоналу підсистеми створення запиту на рівень знань також відноситься формування переліку та детальної картки непройдених запитів на рівень знань.

Після цього у підсистемі визначення рівня знань по запиту для студента формується перелік запитів на рівень знань та відбувається проходження тестування за запитом на рівень знань. По завершенню процесу тестування формується результат поточного тесту та перелік пройдених запитів на рівень знань (затверджених та незатверджених).

Останнім етапом є перевірка результатів проходження запитів на рівень знань викладачем. При цьому формується перелік пройдених запитів для затвердження, доступний детальний перегляд обраного запиту на рівень знань. Після ознайомлення з результатами тестування викладач затверджує обрані запити та відбувається формування переліку затверджених запитів на рівень знань.

Під час адаптивного тестування система повинна використовувати попередньо підготовлений набір тестових завдань для проведення процесу тестування. Система має слідувати ієрархічній структурі та задавати запитання відповідно до рівнів складностей, які зберігаються у тестових запитаннях. Запитання повинні обиратися один за одним. Після відповіді на поставлене запитання система повинна проаналізувати їх і відповідно до результатів аналізу прийняти рішення про рівень складності наступного запитання або ж завершення тесту. Після завершення тесту система має проаналізувати усі результати відповідей та виставити оцінку відповідно до обраного методу оцінювання та вказаної шкали.

3.3 Даталогічна модель даних для адаптивного тестування рівня знань

Для реалізації завдань інформаційної технології, організації даних та збереження проміжних етапів роботи потрібно створити базу даних, структуру якої наведено на рисунку 3.4 та у додатку В.

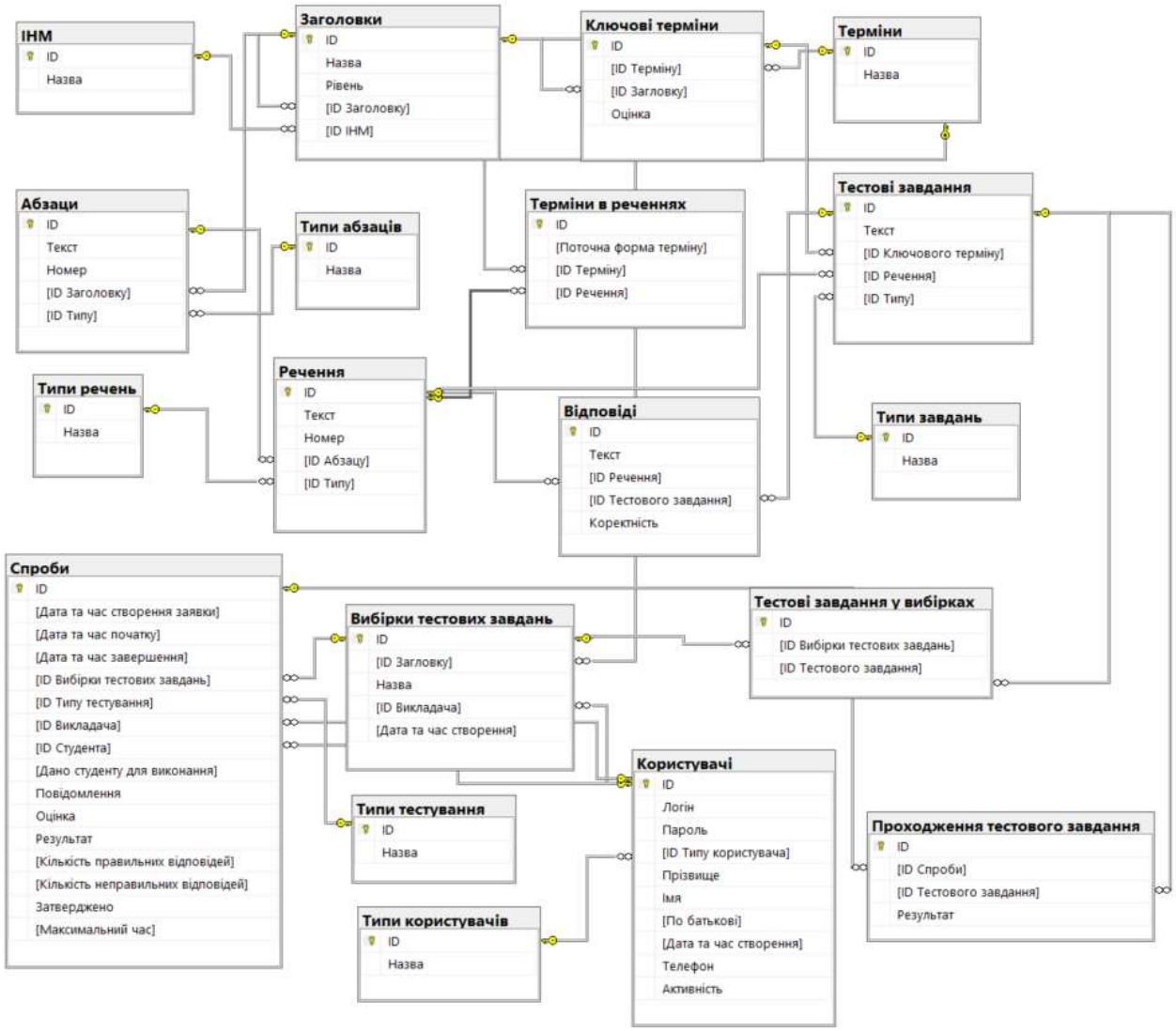


Рисунок 3.4 – Діаграма бази даних інформаційної системи адаптивного тестування рівня знань

Даталогічна модель бази даних розроблюється відповідно до процесів та функцій, що відповідають системі, тому було створено таблиці «Абзаци»,

«Вибірки тестових завдань», «Відповіді», «Заголовки», «ІНМ», «Ключові терміни», «Користувачі», «Проходження тестового завдання», «Речення», «Спроби», «Терміни», «Терміни в реченнях», «Тестові завдання», «Тестові завдання у вибірках», «Типи абзаців», «Типи завдань», «Типи користувачів», «Типи речень», «Типи тестування».

Таблиця «ІНМ» призначена для зберігання даних про інформаційний навчальний матеріал та має два параметри: «ID» - унікальний ідентифікатор, що є первинним ключем та має тип int; «Назва» (тип nvarchar, що має обмеження в 200 символів) – відповідає найменуванню ІНМ.

Таблиця «Заголовки» призначена для зберігання даних про структурну одиницю ІНМ – заголовки. Таблиця «Заголовки» має наступні атрибути: первинний ключ «ID» (int), що є унікальним ідентифікатоом заголовку; «Назва» (тип nvarchar, що має обмеження в 200 символів), що відповідає назві заголовка; «Рівень» (int) – вказує на рівень поточного заголовку відносно документу ІНМ, «ID Заголовку» (int) – зовнішній ключ, що означає зв’язок із таблицею «Заголовки» та вказує на вкладеність заголовків один в один, «ID ІНМ» (int) – зовнішній ключ, що вказує на зв’язок таблиці «Заголовки» та таблиці «ІНМ».

Таблиця «Типи абзаців» має два поля «ID» (унікальний ідентифікатор) та «Назва» (назва типу абзацу). Перше з них буде первинним ключем і матиме цілочисельний тип (int), а інше – стрічковий тип (nvarchar), довжина якого має обмеження у 50 символів.

На поля з даними таблиці «Абзаци», що містить у собі інформацію про абзаци ІНМ, накладаються обмеження у вигляді типу даних, тобто поле «ID» має цілочисельний тип даних (int) та є унікальним ідентифікатором абзацу; поле «Текст» має стрічковий тип, та містить текстове подання абзацу; поле «Номер» типу int, вказує на порядковий номер абзацу в межах заголовку. Поля «ID Заголовку» та «ID Типу» є вторинними ключами, мають цілочисельний тип даних (int) та вказують на з’язок із таблицями «Заголовок» та «Тип абзацу» відповідно.

Таблиця «Типи речень» має атрибути «ID» (унікальний ідентифікатор, первинний ключ) типу int та «Назва» (назва типу речення) стрічкового типу (nvarchar), довжина якого має обмеження у 50 символів.

Таблиця «Речення» містить перелік усіх речень, що додані до бази даних. Зв’язок із таблицею «Абзаци», що здійснюється через вторинний ключ «ID Абзацу», вказує на відношення певного речення до конкретного абзацу. Також через атрибут «ID Типу» реалізується зв’язок із таблицею «Типи речень» (тип int). Крім цього дана таблиця має наступні атрибути: «Текст» (тип nvarchar), «Номер» (тип int), що містять інформацію про текст речення та його порядковий номер у абзаці відповідно.

Таблиця «Терміни» містить дані про терміни, що містяться в ІНМ та має два поля «ID» (унікальний ідентифікатор, первинний ключ типу int) та «Назва» (назва терміну стрічкового тип (nvarchar) з обмеженням у 100 символів).

Ключові терміни та їх параметри зберігаються у таблиці «Ключові терміни». Дана таблиця містить зв’язок із таблицею з даними про заголовки ІНМ «Заголовки» через вторинний ключ «ID Заголовка» типу int; з таблицею «Терміни», через вторинний ключ типу int «ID Терміну». Оцінка важливості терміну зберігається у відповідному атрибуті «Оцінка» типу float.

Таблиця «Терміни у реченнях» є розвідною між таблицями «Терміни» та «Речення». Зв’язок із таблицею «Терміни» забезпечується через вторинний ключ «ID Терміну» типу int, а з таблицею «Речення» через «ID Речення» типу int. Поточній формі терміну відповідає атрибут «Поточна форма терміну» стрічкового типу nvarchar, довжина якого має обмеження у 50 символів.

Окремим блоком розробленої БД слугують таблиці «Тестові завдання», «Відповіді» та «Типи завдань», що орієнтовані на зберігання даних про тестові завдання та відповіді до них. Таблиця «Типи завдань» містить відомості про типи тестових завдань та має атрибути «ID» та «Назва» – унікальний ідентифікатор, первинний ключ типу int та назва типу завдань (тип nvarchar з обмеженням у 50 символів) відповідно. Таблиця «Тестові завдання» має наступні

атрибути: «ID» (унікальний ідентифікатор, первинний ключ типу int), «Текст» (тип nvarchar з обмеженням довжини у 500 символів), «ID Ключового терміну» (вторинний ключ, що пов’язує дану таблицю із таблицею «Ключові терміни», типу int) та «ID Речення» (вторинний ключ, що пов’язує дану таблицю із таблицею «Речення» та вказує на те, з якого речення було сформовано завдання, типу int). У таблиці «Відповіді» дані описуються через атрибути: «ID» – первинний ключ таблиці та унікальний ідентифікатор відповіді у таблиці; «Текст» – текст відповіді стрічкового типу nvarchar, довжина якого має обмеження у 500 символів; «ID Речення» – вторинний ключ, що вказує на зв’язок із таблицею «Речення», типу int; «Коректність» – атрибут у якому зазначається чи правильна поточна відповідь чи ні, типу bit.

Ще одним окремим блоком БД адаптивного тестування рівня знань є таблиці «Користувачі» та «Типи користувачів». Таблиця «Типи користувачів» містить інформацію про типи користувачів у розробленій інформаційній системі (атрибут «ID» є унікальним ідентифікатором, первинним ключем типу int та атрибут «Назва» зберігає назву типу користувача стрічкового типу (nvarchar), довжина якого має обмеження у 50 символів).

Таблиця «Користувачі» призначена для зберігання особистих даних про користувачів системи: унікального ідентифікатора «ID», що є первинним ключем типу int; логіну та паролю, що містяться у атрибутах «Login» «Password» (типу nvarchar з обмеженням довжини у 50 символів) відповідно; вторинного ключа типу int «ID Типу користувача», що вказує на зв’язок із таблицею «Типи користувачів»; прізвища, що зберігається у атрибуті стрічкового типу nvarchar, довжина якого має обмеження у 50 символів «Прізвище»; атрибут «Ім’я» стрічкового типу (nvarchar з обмеженням довжини у 50 символів) відповідає імені користувача; по батькові, що міститься у атрибуті «По батькові» (типу nvarchar з обмеженням довжини у 50 символів); даті та часу створення користувача у системі відповідає атрибут «Дата та час створення» типу datetime; телефон користувача зберігається у атрибуті стрічкового типу (nvarchar)

«Телефон»; атрибут «Активність» (типу bit) вказує на те, чи дозволений користувачу вхід у систему чи користувач є заблокованим.

Таблиця «Типи тестування» має атрибути «ID» (унікальний ідентифікатор типу, первинний ключ) типу int та «Назва» (назва типу тестування) стрічкового типу (nvarchar), довжина якого має обмеження у 50 символів.

У таблиці «Вибірки тестових завдань» описані вибірки, що створенні викладачами із переліку тестових завдань. Зв’язок із таблицею «Заголовок», що здійснюється через вторинний ключ «ID Заголовку», вказує на відношення певної вибірки до конкретного заголовку. Також через атрибут «ID Викладача» реалізується зв’язок із таблицею «Користувачі» (тип int). Крім цього дана таблиця має наступні атрибути: «Назва» (тип nvarchar), «Дата та час створення» (тип datetime), що містять інформацію про назву вибірки та дату і час створення вибірки відповідно.

Таблиця «Тестові завдання у вибірках» є розвідною між таблицями «Тестові завдання» та «Вибірки тестових завдань». Зв’язок із таблицею «Тестові завдання» забезпечується через вторинний ключ «ID Тестового завдання» типу int, а з таблицею «Вибірки тестових завдань» через вторинний ключ «ID Вибірки тестових завдань» типу int.

Таблиця «Спроби» містить перелік усіх спроб тестування, що додані до бази даних. Зв’язок із таблицею «Вибірки тестових завдань», що здійснюється через вторинний ключ «ID Вибірки тестових завдань», вказує на те, що певній спробі відповідає певна вибірка тестових завдань. Також через вторинний ключ «ID Типу» реалізується зв’язок із таблицею «Типи тестування» (тип int). Таблиця «Спроби» має подвійний зв’язок із таблицею «Користувачі» через вторинні ключі «ID Викладача» (типу int) та «ID Студента» (типу int), що надає інформацію про те, який викладач дав поточну спробу на виконання для якого студента. Крім цього дана таблиця має наступні атрибути типу datetime: «Дата та час створення» спроби, «Дата та час початку» спроби, «Дата та час завершення». Також ще є два атрибути типу bit: «Дано студенту для виконання» та

«Затверджено», що містять інформацію про те, чи поточна спроба є надісланою студенту на виконання та чи викладач переглянув результат проходження спроби студентом та затвердив її, відповідно. Атрибут «Повідомлення» типу nvarchar з обмеженням довжини у 50 символів зберігає текст повідомлення від викладача студенту. Атрибутами, що зберігають результуючі дані завершення спроби є «Оцінка» та «Результат» стрічкового типу nvarchar, довжина якого має обмеження у 50 символів. Кількість правильних та неправильних відповідей міститься у відповідних атрибутах «Кількість правильних відповідей» та «Кількість неправильних відповідей» типу int. Максимальний час за якого допускається виконання спроби зберігається у атрибуті «Максимальний час» типу time.

Таблиця «Проходження тестового завдання» є розвідною між таблицями «Тестові завдання» та «Спроби». Зв’язок із таблицею «Тестові завдання» забезпечується через вторинний ключ «ID Тестового завдання» типу int, а з таблицею «Спроби» через вторинний ключ «ID Спроби» типу int. На коректність даної відповіді на тестове запита вказує атрибут «Результат» стрічкового типу nvarchar, довжина якого має обмеження у 50 символів.

Запропонована структура бази даних сприяє реалізації завдань інформаційної технології, дозволяє організувати дані, зберігати проміжні етапи та забезпечує комфортне використання автоматизованої системи.

3.4 Розробка компонентів інформаційної системи адаптивного тестування рівня знань

3.4.1 Аналіз та вибір засобів створення програмного забезпечення

На даний момент найбільш активно розвиваються дві конкуруючі лінії технологій створення програмного забезпечення на основі компонентів – технології Java і .NET. Це Web-додатки, тобто розподілене програмне забезпечення, що використовує базову інфраструктуру Інтернету для зв'язку між

різними своїми компонентами, а стандартні інструменти для навігації по Web-браузерах – як основу для свого призначеного для користувача інтерфейсу.

Технології Java є набором стандартів, інструментів і бібліотек, призначених для розробки додатків різних типів і пов'язаних один з одним використанням мови програмування Java [33]. .NET є схожим набором стандартів, інструментів і бібліотек, але розробка додатків в рамках .NET можлива з використанням різних мов програмування. Основою .NET є віртуальна машина для проміжного мови, в який транслюються всі .NET програми, також звана загальним середовищем виконання CLR, і загальна бібліотека класів (.NET Framework class library), доступна з усіх .NET додатків.

Проміжна мова є повноцінною мовою програмування, але вона не призначена для використання людьми. Однак різні мови досить сильно відрізняються один від одного, і щоб гарантувати можливість з однієї мови працювати з компонентами, написаними на іншій мові, необхідно при розробці цих компонентів дотримуватися загальних правил (CLS), що визначають, якими конструкціями можна користуватися в усіх .NET мовами без втрати можливості взаємодії між результатами. Найбільш близький до проміжної мови C # – ця мова була спеціально розроблений разом з платформою .NET [34].

Microsoft Visual Studio – лінійка продуктів компанії Майкрософт, що включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів [35]. Visual Studio 2017 була випущена 7 березня 2017. Вона володіє досконалого рівня інструментарієм для розробки програмного забезпечення [36].

Для написання програмного продукту необхідно використати об’єктно-орієнтовану мову програмування, до таких, зокрема, належать C++ та C# [37]. C# – це об’єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML.

C# розроблялась як мова програмування прикладного рівня для CLR і тому вона залежить, перш за все, від можливостей самої CLR. Це стосується, перш за все, системи типів C#. CLR надає C#, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких позбавлені «класичні» мови програмування. Методи розширення в C# дозволяють програмістам використовувати статичні методи, як якщо б вони були методами з таблиці методів класу, дозволяючи програмістам додавати методи до об'єкта, який, на їхню думку, повинен існувати на цьому об'єкті і його похідних [38].

Отже, було прийнято рішення використання платформи .NET для реалізації інформаційної системи. Для розробки програмного забезпечення було обрано середовище розробки Visual Studio 2017 та, враховуючи переваги та недоліки об’єктно-орієнтовних мов .NET платформи, для розробки програмного продукту було обрано мову C#.

3.4.2 Структура компонентів інформаційної системи

Для реалізації автоматизованої системи адаптивного тестування рівня знань потрібно розробити компоненти, наступних видів: представлення (рисунок 3.5), та модулі програмно-апаратної частини (рисунок 3.6) згідно алгоритмів інформаційних потоків.

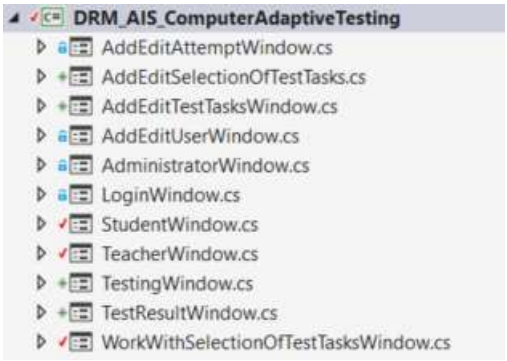


Рисунок 3.5 – Представлення створеного додатку

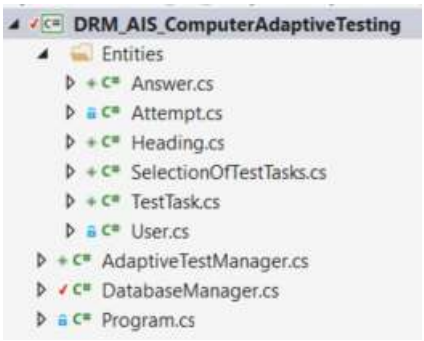


Рисунок 3.6 – Модулі програмно-апаратної частини

Реалізований програмний продукт повинен мати свою власну структуру, яка відповідає поставленим функціям. Діаграма класів автоматизованої системи адаптивного тестування рівня знань представлена та у додатку Г.

Діаграма класів для моделей проекту буде в себе включати моделі програмно-апаратної частини, класи-представлення, що формують UI-частину додатку.

Діаграма класів модулю для роботи з базою даних (рисунок 3.7) містить шість класів, що описують ключові об’єкти системи: спробу (запит на рівень знань), завдання, відповідь, користувача, заголовок як структурну одиницю ІНМ та вибірку тестових завдань. Для класів «Attempt», «TestTask», «Answer», «User», «Heading» та «SelectionOfTestTasks» існують відповідні таблиці у базі даних та більшість властивостей об’єктів цих класів мають рівноцінні колонки у таблицях БД.

Діаграма класів модулю інтерфейсу інформаційної системи (рисунок 3.8) містить класи одинадцяти форм, які є представленням програмного продукту. Клас «LoginWindow» та відповідна форма призначенні для входу користувачів (під час якого, залежно від визначеного типу користувача, буде обиратися наступна форма, що з’явиться).

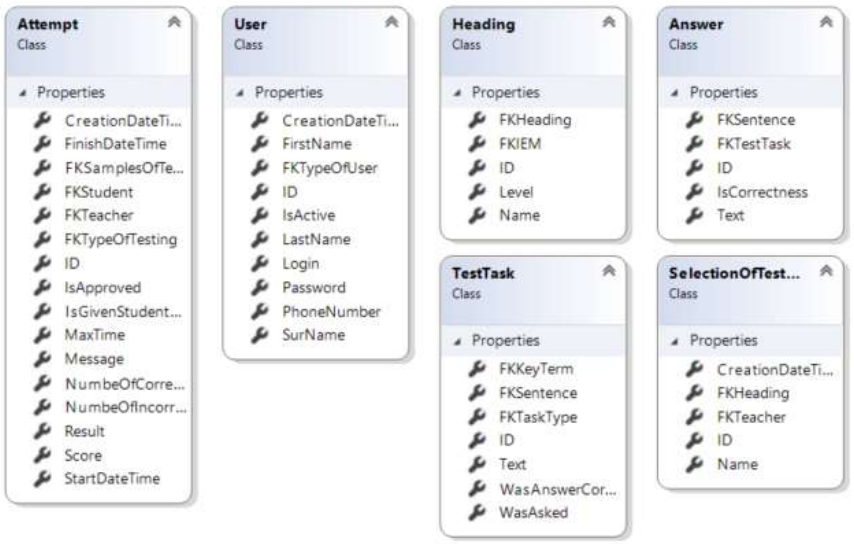


Рисунок 3.7 – Діаграма класів модулю для роботи з базою даних

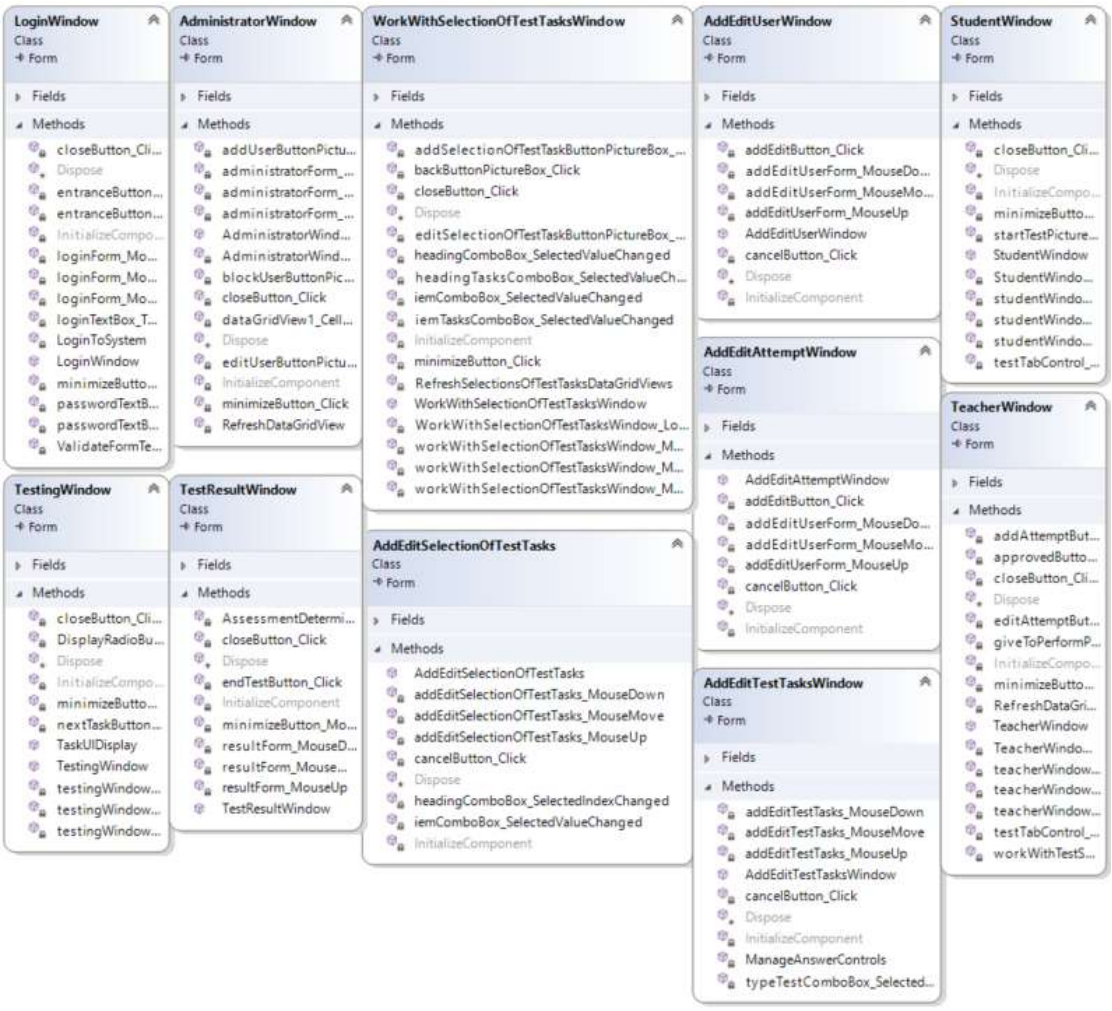


Рисунок 3.8 – Діаграма класів для модулю інтерфейсу інформаційної системи

Залежно від типу користувача, що ввійшов у в систему керування додатком буде передано для класу «AdministratorWindow» (тип користувача адміністратор), «TeacherWindow» (тип користувача викладач) чи «StudentWindow» (тип користувача студент). Клас «AdministratorWindow» реалізує інтерфейс та функціонал для роботи із даними користувачів з використанням класу та форми «AddEditUserWindow», що відповідає саме за додавання та редагування нових користувачів.

Клас «TeacherWindow» реалізує функціонал та інтерфейс у вигляді форми для роботи викладача з тестовими завданнями, з вибірками тестових завдань, з проходженням тестів. Додатково для роботи з тестовими завданнями створено клас «AddEditTestTasksWindow», що дозволяє додавати та редагувати тестові завдання. Для додавання та редагування вибірок тестових завдань створено клас та форму «AddEditSelectionOfTestTasks». Використовуючи інтрефейс форми та функціонал класу «AddEditAttemptWindow» викладач може створювати та редагувати запити на рівень знань студента.

Форма та клас «StudentWindow» реалізує інтерфейс та функціонал для роботи з проходженням тесту студентом, використовуючи який він може переглядати результати тестування та розпочати безпосередньо процес тестування.

Клас та форма «TestingWindow» створюють представлення на якому відбувається сам процес тестування: відображається тестове завдання та варіанти відповіді. Клас «TestResultWindow» та відповідна йому форма створенні для показу результату виконаного запиту на рівень знань студентом.

Діаграма класів модулю забезпечення тестування (рисунок 3.9) містить клас Program з методом Main – точкою входу у програму. Клас «DatabaseManager», що наслідується від інтерфейсу IDisposable відповідає за зв’язок додатку із базою даних та містить реалізацію методів на отримання та додавання даних у БД. Клас «AdaptiveTestManager» містить механізм керування

процесом тестування та відповідає за вибір питань та перевірки умови закінчення тестування.

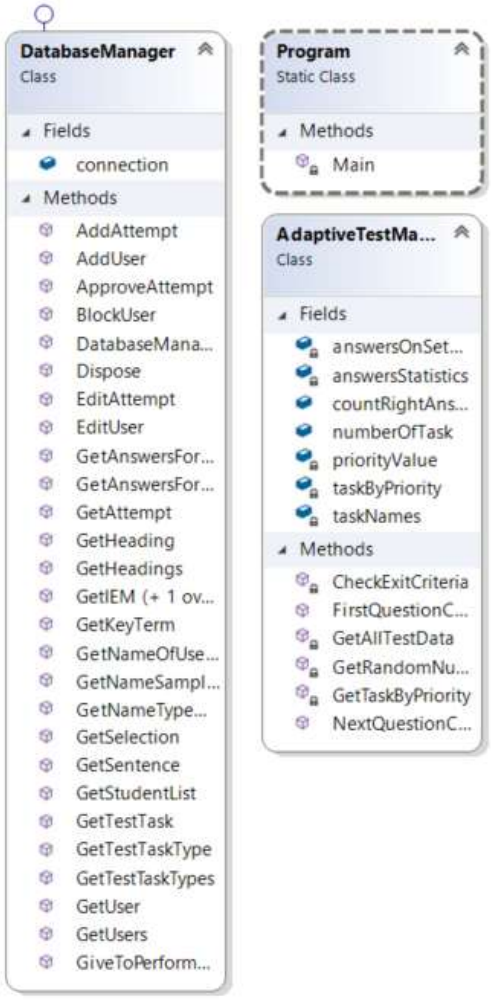


Рисунок 3.9 – Діаграма класів модулю забезпечення тестування

В результаті проведеної роботи було розроблено три програмних модулі та створено користувальницький інтерфейс, що безпосередньо взаємодіє із серверною програмно-апаратною частиною.

Для реалізації автоматизованої системи адаптивного тестування рівня знань було обрано платформу .NET, мову програмування C# та середовище розробки Visual Studio 2017. База даних створювалась з використанням MS SQL Server 2019.

3.4.3 Розробка програмного модулю для роботи з базою даних

У додатку було реалізовано шість класів, що відповідають сутностям бази даних за властивостями задля зручності структурування даних у списки та «словники». Таким чином одному об’єкту будь-якого з цих класів відповідає лише один запис у базі даних, що знаходиться у відповідній таблиці. Процеси роботи із базою даних реалізовані з використанням Entity Framework, що забезпечує використання об’єктно-орієнтованого підходу на основі .NET.

Клас «Answer» розроблений для роботи із таблицею «Відповіді» має наступні властивості: «ID», «Text», «FKSentence», «FKTestTask», «IsCorrect» (таблиця 3.1).

Таблиця 3.1 – Властивості класу Answer

Назва	Тип	Опис
ID	int	Встановлює або повертає значення первинного ключа запису в таблиці «Відповіді».
Text	string	Встановлює або повертає значення тексту формулювання відповіді на тестове завдання.
FKSentence	int	Встановлює або повертає значення вторинного ключа «ID Речення», який містить ID відповідного речення ІНМ, за яким було сформовано відповідь.
FKTestTask	int	Встановлює або повертає значення вторинного ключа «ID Тестового завдання», який містить ID відповідного тестового завдання.
IsCorrect	bool	Встановлює або повертає значення, що вказує на те, чи відповідь є коректною.

Клас «Attempt» розроблений для роботи із таблицею «Спроби» має наступні властивості: «ID», «CreationDateTime», «StartDateTime», «FinishDateTime», «FKSamplesOfTestTasks», «FKTypeOfTesting», «FKTeacher», «FKStudent», «IsGivenStudentToPerform», «Message», «Score», «Result», «NumbeOfCorrectAnswers», «NumbeOfIncorrectAnswers», «IsApproved», «MaxTime» (таблиця 3.2).

Таблиця 3.2 – Властивості класу Attempt

Назва	Тип	Опис
ID	int	Встановлює або повертає значення первинного ключа запису в таблиці «Спроби».
CreationDateTime	DateTime	Встановлює або повертає значення дати та часу створення запиту на рівень знань.
StartDateTime	DateTime	Встановлює або повертає значення дати та часу початку спроби.
FinishDateTime	DateTime	Встановлює або повертає значення дати та часу завершення спроби.
FKSamplesOfTestTasks	int	Встановлює або повертає значення вторинного ключа «ID Вибірки тестових завдань», який містить ID відповідної вибірки тестових завдань.
FKTypeOfTesting	int	Встановлює або повертає значення вторинного ключа «ID Типу тестування», який містить ID відповідного типу тестування.
FKTeacher	int	Встановлює або повертає значення вторинного ключа «ID Викладача», який містить ID користувача (викладача), який створив запит на рівень знань.
FKStudent	int	Встановлює або повертає значення вторинного ключа «ID Студента», який містить ID користувача (студента), який призначений до виконання запиту на рівень знань.
IsGivenStudentToPerform	bool	Встановлює або повертає значення, що вказує на те, чи був надісланий запит на виконання тесту студенту.
Message	string	Встановлює або повертає значення повідомлення викладача для студента.
Score	string	Встановлює або повертає значення оцінки.
Result	string	Встановлює або повертає значення результату тесту.
NumbeOfCorrectAnswers	int	Встановлює або повертає значення кількості правильних відповідей даних студентом під час тестування.
NumbeOfIncorrectAnswers	int	Встановлює або повертає значення кількості неправильних відповідей даних студентом під час тестування.
IsApproved	bool	Встановлює або повертає значення, що вказує на те, чи була спроба затверджена викладачем.
MaxTime	DateTime	Встановлює або повертає значення максимального часу на виконання спроби.

Клас «Heading» розроблений для роботи із таблицею «Заголовки» має наступні властивості: «ID», «Name», «Level», «FKHeading», «FKIEM» (таблиця 3.3).

Таблиця 3.3 – Властивості класу Heading

Назва	Тип	Опис
ID	int	Встановлює або повертає значення первинного ключа запису в таблиці «Заголовки».
Name	string	Встановлює або повертає значення назви заголовку ІНМ.
Level	int	Встановлює або повертає значення рівня заголовку у межах ІНМ.
FKHeading	int	Встановлює або повертає значення вторинного ключа «ID Заголовку», який містить ID заголовку, якому він належить.
FKIEM	int	Встановлює або повертає значення вторинного ключа «ID ІНМ», який містить ID відповідного ІНМ до якого належить заголовок.

Клас «SelectionOfTestTasks» розроблений для роботи із таблицею «Вибірки тестових завдань» має наступні властивості: «ID», «FKHeading», «Name», «FKTeacher», «CreationDateTime» (таблиця 3.4).

Таблиця 3.4 – Властивості класу SelectionOfTestTasks

Назва	Тип	Опис
ID	int	Встановлює або повертає значення первинного ключа запису в таблиці «Вибірки тестових завдань».
FKHeading	int	Встановлює або повертає значення вторинного ключа «ID Заголовку», який містить ID заголовку до якого відноситься вибірка.
Name	string	Встановлює або повертає значення назви вибірки тестових завдань.
FKTeacher	int	Встановлює або повертає значення вторинного ключа «ID Викладача», який містить ID користувача (викладача), який сформував вибірку.
CreationDateTime	DateTime	Встановлює або повертає значення дати та часу створення вибірки тестових завдань.

Клас «TestTask» розроблений для роботи із таблицею «Тестові завдання» має наступні властивості: «ID», «Text», «FKKeyTerm», «FKSentence», «FKTaskType» (таблиця 3.5).

Таблиця 3.5 – Властивості класу TestTask

Назва	Тип	Опис
ID	int	Встановлює або повертає значення первинного ключа запису в таблиці «Тестові завдання».
Text	string	Встановлює або повертає значення тексту формулювання тестового завдання.
FKKeyTerm	int	Встановлює або повертає значення вторинного ключа «ID Ключового терміну», який містить ID відповідного ключового терміну за яким було сформовано тестове завдання.
FKSentence	int	Встановлює або повертає значення вторинного ключа «ID Речення», який містить ID відповідного речення ІНМ, за яким було сформовано тестове завдання.
FKTaskType	int	Встановлює або повертає значення вторинного ключа «ID Типу», який містить ID відповідного типу тестового завдання.
WasAsked	bool	Встановлює або повертає значення, що вказує на те, чи було задано тестове завдання під час тестування.
WasAnswerCorrect	bool	Встановлює або повертає значення, що вказує на те, чи була відповідь дана на тестове завдання коректною.

Таблиця 3.6 – Властивості класу User

Назва	Тип	Опис
ID	int	Встановлює або повертає значення первинного ключа запису в таблиці «Користувачі».
Login	string	Встановлює або повертає значення логіну користувача.
Password	string	Встановлює або повертає значення паролю користувача.
FKTypeOfUser	int	Встановлює або повертає значення вторинного ключа «ID Типу користувача», який містить ID відповідного типу користувача.
LastName	string	Встановлює або повертає значення прізвища користувача.
FirstName	string	Встановлює або повертає значення імені користувача.
SurName	string	Встановлює або повертає значення по батькові користувача.
CreationDateTime	DateTime	Встановлює або повертає значення дати та часу створення користувача.
PhoneNumber	string	Встановлює або повертає значення номеру телефону користувача.
IsActive	bool	Встановлює або повертає значення, що вказує на те, чи є користувач активним у системі.

Властивості «WasAsked» та «WasAnswerCorrect» є властивостями лише класу «TestTask» (відповідників у БД немає), що допомагають позначати вже

задані запитання задля уникнення повторів у завданнях та питання, на які була дана коректна відповідь, задля зручності обрахунку оцінки і результату відповідно.

Клас «User» розроблений для роботи із таблицею «Користувачі» має наступні властивості: «ID», «Login», «Password», «FKTypeOfUser», «LastName», «FirstName», «SurName», «CreationDateTime», «PhoneNumber», «IsActive» (таблиця 3.6).

Отже, модуль для роботи з базою даних містить класи, що описують ключові об’єкти системи: спробу (клас «Attempt»), завдання (клас «TestTask»), відповідь (клас «Answer»), користувача (клас «User»), заголовок (клас «Heading») та вибірку тестових завдань («SelectionOfTestTasks»). Для класів даного модулю існують відповідні таблиці у базі даних та майже всі властивості об’єктів цих класів мають рівнозначні атрибути у таблицях БД.

3.4.4 Розробка програмного модулю забезпечення тестування

Призначенням модулю забезпечення тестування є прикладна організація роботи з базою даних та реалізація процесу адаптивного тестування. До даного модулю відноситься три класи серед яких клас Program з стандартним методом Main, що є точкою входу у програму.

Клас «AdaptiveTestManager» призначений для реалізації процесу тестування відповідно до його типу, тобто саме цей клас керує процесом тестування та відповідає за вибір питань та перевірки умови закінчення тестування. Клас має наступні загальні змінні, що ініціалізованні на рівні класу:

– список taskNames типу TestTask, що містить перелік тестових завдань, які можуть бути використанні під час поточного тестування;

– масив логічних даних answersStatistics, що містить статистику відповідей на тестові завдання;

– статична цілочисельна змінна numberOfTask, що містить значення кількості завдань у сесії тестування;

– статична цілочисельна змінна countRightAnswersOnTasks, що містить значення кількості правильних відповідей даних під час тестування;

– «словник», значеннями якого є список завдань, taskByPriority типу TestTask, що містить перелік тестових завдань та відповідних до них пріоритетів, що визначаються за важливістю ключових термінів, для перевірки яких створенні тестові завдання;

– цілочисельна змінна priorityValue, що містить значення поточного пріоритету тестового завдання.

Клас «AdaptiveTestManager» містить шість методів, що реалізовують логіку за якою відбувається тестування. Це методи:

– GetAllTestData – реалізує функціонал отримання переліку тестових завдань для поточної сесії тестування;

– FirstQuestionChoice – реалізує функціонал вибору першого завдання відповідно до обраного типу тестування вибірки тестових завдань, повертає значення типу TestTask;

– NextQuestionChoice – реалізує функціонал вибору наступного завдання відповідно до обраного типу тестування, повертає значення типу TestTask та має вхідний параметр логічного типу wasAnswerRight;

– GetRandomNumber – реалізує функціонал вибору довільного номеру тестового завдання залежно від кількості countOfTask, що є вхідним параметром, метод повертає цілочисельне значення;

– GetTaskByPriority – реалізує функціонал отримання тестового завдання за пріоритетом, повертає значення типу TestTask та має вхідний цілочисельний параметр priority;

– CheckExitCriteria – реалізує функціонал перевірки умови виходу з сесії тестування та повертає значення логічного типу.

Клас «DatabaseManager» забезпечує зв’язок програми із базою даних та реалізує методи отримання, додавання та зміни даних у БД. Даний клас наслідується від інтерфейсу IDisposable, що оголошує один єдиний метод Dispose, в якому при реалізації інтерфейсу в класі повинно відбуватися звільнення некерованих ресурсів. Клас має лише одну загальну змінну, що ініціалізованна на рівні класу і це змінна connection типу SqlConnection, що зберігає у собі дані про з’єднання додатку до БД.

У класі «DatabaseManager» оголошено та реалізовано двадцять сім методів:

– AddAttempt – реалізує додавання запиту на рівень знань до БД;

– AddUser – реалізує додавання даних про користувача до БД;

– ApproveAttempt – реалізує функціонал оновлення атрибуту «Затверджено» таблиці «Спроби»;

– BlockUser – реалізує функціонал оновлення атрибуту «Активність» таблиці «Користувачі»;

– EditAttempt – реалізує функціонал редагування даних про запит на рівень знань до БД;

– EditUser – реалізує функціонал оновлення даних про користувача в БД;

– GetAnswersForChooseTasks – реалізує отримання відповідей для тестових завдань типів «Множинний вибір» та «Одиничний вибір»;

– GetAnswersForOpenOrLogicalTasks – реалізує функціонал отримання відповідей для тестових завдань типу «Логічний вибір» та «Введення відповіді»;

– GetAttempt – реалізує отримання даних про запит на рівень знань з БД;

– GetHeading – реалізує функціонал отримання даних з БД про заголовок, що є підпорядкований іншому заголовку;

– GetHeadings – реалізує функціонал отримання даних з БД про заголовок, що за структурою належать певному ІНМ;

– GetIEM – реалізує функціонал отримання даних з БД про ІНМ та має дві реалізації (є перевантаженим): отримання переліку усіх записів із таблиці «ІНМ» та отримання назви ІНМ за ID заголовку, що входить до цього ІНМ.

– GetKeyTerm– реалізує функціонал отримання даних з БД про ключові терміни за обраним заголовком ІНМ;

– GetNameOfUserType – реалізує отримання переліку типів користувачів;

– GetNameSamplesOfTestTasks – реалізує функціонал отримання назв вибірок тестових завдань з БД;

– GetNameTypeOfTesting – реалізує функціонал отримання переліку типів тестування;

– GetSelection – реалізує отримання даних про вибірки тестових завдань;

– GetSentence – реалізує функціонал отримання даних з БД про речення;

– GetStudentList – реалізує функціонал отримання даних з БД про користувачі типу «Студент»;

– GetTestTask – реалізує отримання даних про тестове завдання;

– GetTestTaskType – реалізує функціонал отримання типу поточного тестового завдання;

– GetTestTaskTypes – реалізує функціонал отримання переліку типів тестових завдань;

– GetUser – реалізує функціонал отримання даних про поточного користувача системи;

– GetUsers – реалізує функціонал отримання переліку користувачів;

– GiveToPerformAttempt – реалізує функціонал оновлення атрибуту «Дано студенту для виконання» таблиці «Спроби».

До класів модулю забезпечення тестування відноситься клас Program з методом Main, що є точкою входу у програму. Методи класу «DatabaseManager», реалізують зв’язок інформаційної системи із базою даних через механізм отримання та додавання даних. Методи класу «AdaptiveTestManager» містять

механізм керування процесом тестування та відповідають за вибір питань та перевірки умови закінчення тестування.

3.4.5 Розробка програмного модулю інтерфейсу інформаційної системи

Модуль «Представлення» складається з одинадцяти форм та відповідних до них класів. Дані класи відповідають як і за графічне представлення інтерфейсу користувача так і за логіку виконання більшості функцій, що вказанні у п.3.1. Усі класи даного модулю мають три однакових змінних, що ініціалізованні на рівні цих класів: логічна змінна dragging, що вказує на те, чи форма в певний момент часу переноситься у межах екрану; змінні dragCursorPoint та dragFormPoint типу Point, що містять координати точки розташування курсору та форми відповідно.

Клас «LoginWindow» призначений для виконання операцій, що пов’язані із входом у систему. Даний клас містить шість методів (таблиця 3.7), що реалізовують логіку за якою відбувається тестування. Основні методи з них: entranceButton_KeyPress, entranceButton_MouseClick, loginTextBox_ TextChanged – методи, які викликають одну і ту ж функцію процесу входу у систему; LoginToSystem – реалізує функціонал входу у систему; ValidateFormTextBoxes – реалізує функціонал валідації даних у полях паролю та логіну.

Клас «AdministratorWindow» реалізує функції, що необхідні для роботи адміністратора та містить наступні методи:

– addUserButtonPictureBox_Click – реалізує функціонал виклику форми «AddEditUserWindow» в режимі додавання нового користувача;

– AdministratorWindow_Load – реалізує функціонал виклику методу RefreshDataGridView при завантаженні вікна;

– blockUserButtonPictureBox_Click – реалізує функціонал блокування користувача;

– dataGridView1_CellFormatting – метод, що реалізує приховування значення паролю у таблиці «Користувачі» символом *;

– editUserButtonPictureBox_Click – реалізує функціонал виклику форми «AddEditUserWindow» в режимі редагування даних існуючого користувача;

– RefreshDataGridView – реалізує функціонал оновлення даних у контролі userTableDataGridView, що є фактично таблицею із даними користувачів.

Клас «AddEditUserWindow» реалізує додавання та редагування нових користувачів. Клас має наступні загальні змінні, що ініціалізованні на рівні класу: «словник» з стрічковими значеннями typeOfUsersNames, що містить перелік типів користувачів, стрічкова змінна типу workWithUserMode, що зберігає значення режиму у якому працює вікно (додавання чи редагування) та цілочисельна змінна currentUserId, що містить значення унікального ідентифікатора користувача. Клас «AddEditUserWindow» містить наступні методи: addEditButton_Click – реалізує функціонал додавання чи оновлення даних користувача; AddEditUserWindow – конструктор класу «AddEditUserWindow».

Клас «TeacherWindow» призначений для виконання операцій, які відповідають за роботу викладача з тестовими завданнями, з вибірками тестових завдань, з проходженням тестів. У класі «TeacherWindow» оголошено та реалізовано чотирнадцять методів, опис деяких із них:

– addAttemptButtonPictureBox_Click – реалізує функціонал виклику форми «AddEditAttemptWindow» в режимі додавання нового запиту на рівень знань;

– approvedButtonPictureBox_Click – реалізує функціонал затвердження виконаного запиту на рівень знань;

– editAttemptButtonPictureBox_Click – реалізує функціонал виклику форми «AddEditAttemptWindow» в режимі редагування даних існуючого запиту на рівень знань;

– giveToPerformPictureBox_Click – реалізує функціонал відправлення запиту рівня знань студенту;

– RefreshDataGridViews - реалізує функціонал оновлення даних у контролах newTestDataGridView, notApprovedTestDataGridView, approvedTestDataGridView, що є фактично таблицями із даними про запити на визначення рівня знань;

– TeacherWindow_Load – реалізує функціонал виклику методу RefreshDataGridViews при завантаженні вікна;

– testTabControl_SelectedIndexChanged – реалізує функціонал керування доступності кнопок на панелі керування;

– workWithTestSamplesButtonPictureBox_Click – реалізує функціонал виклику форми «WorkWithSelectionOfTestTasksWindow» для роботи з вибірками тестових завдань та самими тестовими завданнями.

Клас «AddEditAttemptWindow» реалізує додавання та редагування нових запитів на рівень знань. Клас має наступні загальні змінні, що ініціалізованні на рівні класу: «словники» з стрічковими значеннями samplesOfTestTasksNames, typeOfTestingNames, studentsList, що містять перелік тестових завдань поточної вибірки, типів тестування та список студентів; стрічкова змінна типу workWithAttemptMode, що зберігає значення режиму у якому працює вікно (додавання чи редагування); цілочисельні змінні currentAttemptId та currentUserId, що містить значення унікального ідентифікатора запиту на рівень знань та користувача відповідно. Клас «AddEditAttemptWindow» містить наступні методи: AddEditAttemptWindow – конструктор класу «AddEditAttemptWindow»; addEditButton_Click – реалізує функціонал додавання чи оновлення даних про запит на рівень знань.

Клас «WorkWithSelectionOfTestTasksWindow» реалізує функції, що необхідні для роботи викладача з вибірками тестових завдань та з тестовими завданнями. Клас має наступні загальні змінні, що ініціалізованні на рівні класу: цілочисельну змінну currentUserId, що містить значення унікального

ідентифікатора користувача; стрічкові змінні currentUserLastName, currentUserFirstName, currentUserSurName, що відповідно містять інформацію про прізвище, ім’я та по батькові поточного користувача; «словники» з стрічковими значеннями iemList, headingsListSelectedTestTasks, headingsListTestTasks, що містять перелік ІНМ, заголовків та тестових завдань відповідно.

Клас «WorkWithSelectionOfTestTasksWindow» містить наступні методи:

– addSelectionOfTestTaskButtonPictureBox_Click – реалізує функціонал виклику форми «AddEditSelectionOfTestTasks» в режимі додавання даних про нову вибірку тестових завдань;

– backButtonPictureBox_Click – реалізує функціонал повернення до вікна «TeacherWindow»;

– editSelectionOfTestTaskButtonPictureBox_Click – реалізує функціонал виклику форми «AddEditSelectionOfTestTasks» в режимі редагування даних існуючої вибірки тестових завдань;

– headingComboBox_SelectedValueChanged – реалізує отримання переліку вибірок тестових завдань, що відносяться до обраного заголовку;

– headingTasksComboBox_SelectedValueChanged – реалізує функціонал отримання переліку тестових завдань, що відносяться до обраного заголовку;

– iemComboBox_SelectedValueChanged та iemTasksComboBox_ SelectedValueChanged – реалізують функціонал отримання переліку заголовків, що відносяться до обраного ІНМ;

– RefreshSelectionsOfTestTasksDataGridViews – реалізує функціонал оновлення даних у контролах newTestDataGridView, selectionsOfTestTasksDataGridView, testTasksDataGridView, що є фактично таблицями із даними про вибірки тестових завдань та тестові завдання;

– WorkWithSelectionOfTestTasksWindow_Load – реалізує функціонал виклику методу RefreshSelectionsOfTestTasksDataGridViews при завантаженні вікна.

Клас «AddEditSelectionOfTestTasks» реалізує додавання та редагування нових вибірок тестових завдань. Клас має наступні загальні змінні, що ініціалізованні на рівні класу: «словники» з стрічковими значеннями iemList, headingsListSelectedTestTasks, що містять перелік ІНМ та заголовків; стрічкова змінна типу workWithAttemptMode, що зберігає значення режиму у якому працює вікно (додавання чи редагування); цілочисельні змінні currentSelectionId, що містить значення унікального ідентифікатора вибірки тестових завдань. Клас «AddEditSelectionOfTestTasks» містить наступні методи: addEditButton_Click – реалізує функціонал додавання чи оновлення даних про вибірку тестових завдань; headingComboBox_SelectedIndexChanged – реалізує функціонал отримання переліку тестових завдань, що відносяться до обраного заголовку; iemComboBox_SelectedValueChanged - реалізує функціонал отримання переліку заголовків, що відносяться до обраного ІНМ.

Клас «AddEditTestTasksWindow» реалізує додавання та редагування нових тестових завдань. Даний клас містить наступні методи: addEditButton_Click – реалізує функціонал додавання чи оновлення даних про тестове завданя; addEditTestTasks_MouseDown, addEditTestTasks_MouseMove, addEditTestTasks_MouseUp – методи, що забезпечують можливість пересування вікна «AddEditTestTasksWindow» по екрану; ManageAnswerControls – реалізує функціонал керування контролів відповідей на формі; typeTestComboBox_SelectedValueChanged – реалізує функціонал виклику методу ManageAnswerControls при зміні значення типу тестового завдання.

Клас «StudentWindow» призначений для виконання операцій, які відповідають за реалізацію процесу проходженням тестування студентом. У класі «StudentWindow» оголошено та реалізовано дев’ять методів, опис деяких із них: startTestPictureBox_Click – реалізує функціонал виклику форми «TestingWindow» для проходження тестування; StudentWindow_Load – реалізує функціонал оновлення даних у контролах newTestDataGridView, notApprovedTestDataGridView, approvedTestDataGridView, що є фактично

таблицями із даними про запити на визначення рівня знань; testTabControl_SelectedIndexChanged – реалізує функціонал керування доступності кнопок на панелі керування.

Клас «TestingWindow» призначений для реалізації операцій процесу тестування, що пов’язані з інтерфейсом користувача. Клас має загальні змінні, що ініціалізованні на рівні класу: змінна adaptiveTestManager типу AdaptiveTestManager – об’єкт класу, що реалізує обчилення процесу тестування, список answerTexts типу Answer, що містить відповіді на поточне запитання та змінна currentTask типу TestTask, що містить формулювання поточного тестового завдання.

У класі «TestingWindow» оголошено та реалізовано дев’ять методів, опис деяких із них: DisplayRadioButtons – реалізує функціонал керування видимості радіо-кнопок; nextTaskButton_Click – реалізує функціонал переходу до наступного тестового завдання; TaskUIDisplay – реалізує функціонал відображення відповідних контролів до типу тестового завдання.

Клас «TestResultWindow» призначений для відображення результату тестування. Даний клас містить вісім методів (таблиця 3.7), що реалізують логіку за якою відбувається тестування. Найважливіші з них методи: AssessmentDetermining – реалізує функціонал обрахунку оцінки студента; resultForm_MouseDown, resultForm_MouseMove, resultForm_MouseUp – методи, що забезпечують можливість пересування вікна «TestResultWindow» по екрану.

Отже, методи класу «LoginWindow» модулю інтерфейсу інформаційної системи призначенні для реалізації входу користувачів. Функціонал адміністратора реалізують методи класу «AdministratorWindow», викладача – методи класу «TeacherWindow» та студента – методи класу «StudentWindow». Клас «AddEditUserWindow», через методи реалізує додавання та редагування нових користувачів. Для роботи з тестовими завданнями створено методи класу «AddEditTestTasksWindow», а для додавання та редагування вибірок тестових завдань – «AddEditSelectionOfTestTasks». Методи класу

«AddEditAttemptWindow» відповідають за створення та редагування запитів на рівень знань студента. У класі «TestingWindow» реалізується процес тестування, а у класі «TestResultWindow» – показ результату тестування. Лістинги програмної реалізації вищеописаних класів інформаційної системи адаптивного контролю знань розташовано у додатку Д.

Висновки до розділу 3

В розділі визначено етапи роботи та функції інформаційної системи, що реалізує розроблену інформаційну технологію адаптивного тестування знань. Робота інформаційної системи адаптивного тестування складається з шести етапів: створення користувачів, формування множини тестових завдань до обраної рубрики, створення цільових вибірок тестових завдань, створення запиту на рівень знань, визначення рівня знань по запиту, перевірка результатів проходження запитів на рівень знань.

Функцією адміністратора є робота з даними користувачів. До функціоналу, розробленого для викладача, можна віднести роботу з тестовими завданнями, роботу із вибірками тестових завдань, роботу із проходженням тестів. До функцій, що доступні студенту, відноситься робота з проходженням тестів. Відповідно до функцій та етапів роботи, розроблено структуру інформаційної системи, яка дає більш детальний опис етапів її роботи відносно призначених користувачів.

Аналіз функціональності й ефективності застосування інформаційної системи дозволить визначити практичну ефективність відповідної інформаційної технології адаптивного тестування рівня знань.

Розділ 4

Дослідження ефективності інформаційної технології адаптивного тестування рівня знань

4.1 Дослідження коректності виконання функцій проходження тестів у системі

Для дослідження коректності виконання функцій проходження тестів у розробленій інформаційній системі адаптивного тестування рівня знань було розроблено чотири тестових випадки (тест-кейси). У першому тестовому випадку (таблиця 4.1) перевіряється процес входу у інформаційну систему використовуючи дані заблокованого користувача. Верифікація даних відбувається при вході у обліковий запис. Тобто, якщо ввести значення у поле логіну та у поле паролю користувача, що є заблокований адміністратором, з'явиться діалог із помилкою «Ваш обліковий запис заблокований. Будь ласка, зверніться до адміністратора!» (рисунок 4.1).

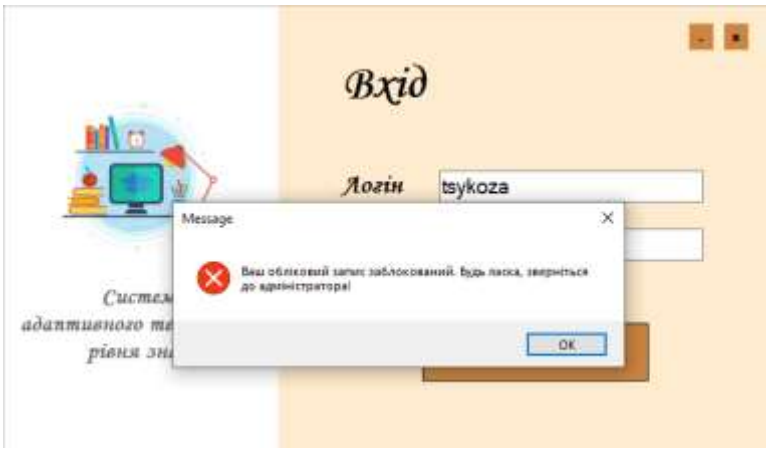


Рисунок 4.1 – Помилка у випадку, якщо користувач заблокований у системі

У другому тестовому випадку (таблиця 4.2) перевіряється функціонал перегляду результатів. При переході на вкладку «Незатвердженні тести» (рисунок 4.2) у таблиці мають відображатися лише записи із значенням у

колонці «Затверджено» = false, а кнопка «Почати тестування» повинна бути недоступною, оскільки тестування за даними запитами вже завершене, але ще не переглянуто викладачем.

Таблиця 4.1 – Тест-кейс AT0001

Тест-кейс ID: AT0001	**Пріоритет: 3**	**Створено:** 08.11.2020, Г.Білоус
Назва: Перевірка логіну заблокованого користувача **Вхідні дані:** Логін = «tsykoza», Пароль = «tsykoza»		
Кроки	**Очікуваний результат**	
Передумова: користувач з логіном tsykoza має бути заблокований адміністратором (Активність=false), а користувач з логіном «kovalchuk» - активним (Активність=true) 1. Запустити додаток 2. Ввести значення в поля логіну та паролю із вхідних даних 3. Натиснути кнопку «Вхід» 4. Порівняти фактичний результат з очікуваним 5. Закрити діалог із помилкою 6. Ввести значення в поля логіну та паролю значення «kovalchuk» 7. Натиснути кнопку «Вхід» 8. Порівняти фактичний результат з очікуваним	Діалог із помилкою: «Ваш обліковий запис заблокований. Будь ласка, зверніться до адміністратора!». Вікно входу у систему закрилось. Вікно з інтерфейсом студента відкрито.	
Результат виконання тест-кейсу: пройдено успішно		

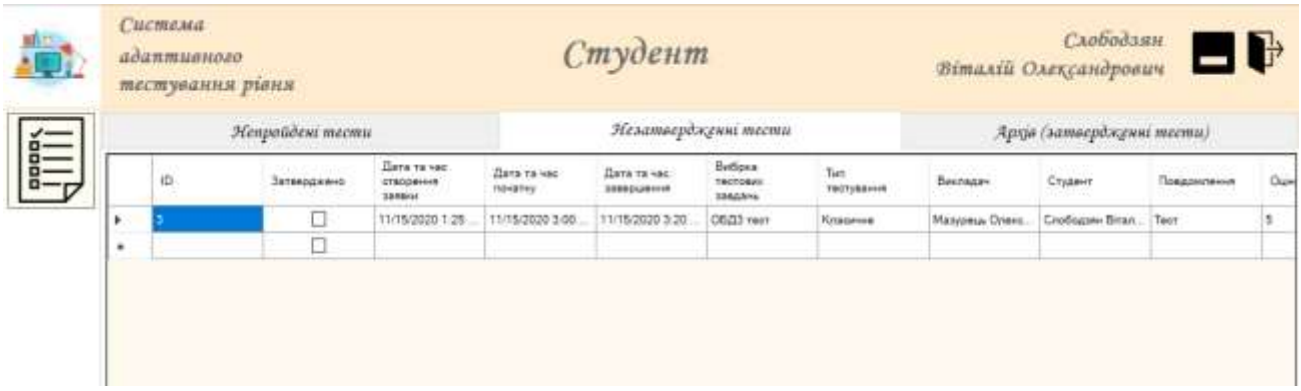


Рисунок 4.2 – Вкладка «Незатверджені тести»

На вкладці «Архів (затверджені тести)» (рисунок 4.3) у таблиці повинні відображатися лише ті записи, що мають значення у колонці «Затверджено» = true. Кнопка «Почати тестування» теж недоступна, тому що тестування даними запитами також вже завершене та переглянуто викладачем.

Рисунок 4.3 – Вкладка «Архів (затверджені тести)»

На вкладці «Непройдені тести» (рисунок 4.4) знаходиться таблиця з обмеженою кількістю колонок, тому що не всі дані про спробу тестування є відомими, оскільки дані запити на визначення рівня знань студента ще не були пройдені. Кнопка «Почати тестування» у даному випадку доступна.

Рисунок 4.4 – Вкладка «Непройдені тести»

Третій тестовий випадок (таблиця 4.3) перевіряє функціонал проходження тестування. При настисканні на кнопку «Почати тестування», що знаходиться на панелі керування вікна, відкривається вікно «Тестування»

(рисунок 4.5). У вікні представлено тестове завдання, що відповідає тематиці вибірки тестових завдань рядка таблиці «Непройдені тести», що був обраний перед натисканням на кнопку «Почати тестування».

Таблиця 4.2 – Тест-кейс AT0002

<table>
<tr><td>Тест-кейс ID: AT0002</td><td>Пріоритет: 1</td><td>Створено: 08.11.2020, Г.Білоус</td></tr>
<tr><td colspan="3">Назва: Перевірка перегляду запитів на рівень знань
Вхідні дані: Логін = «bilous», Пароль = «bilous»</td></tr>
<tr><td colspan="2">Кроки</td><td>Очікуваний результат</td></tr>
<tr><td colspan="2">1. Запустити додаток
2. Ввести значення у поле логіну та паролю із вхідних даних
3. Натиснути кнопку «Вхід»
4. Порівняти фактичний результат з очікуваним
5. Перейти на вкладку «Незатверджені тести»
6. Порівняти фактичний результат з очікуваним
7. Перейти на вкладку «Архів (затверджені тести)»
8. Порівняти фактичний результат з очікуваним
9. Повернутися на вкладку «Непройдені тести»
10. Порівняти фактичний результат з очікуваним</td><td>Вікно входу у систему закрилось. Вікно з інтерфейсом студента відкрито.

Таблиця з тестами містить лише записи із значенням у колонці «Затверджено» = false. Кнопка «Почати тестування» недоступна.

Таблиця з тестами містить лише записи із значенням у колонці «Затверджено» = true. Кнопка «Почати тестування» недоступна.

Таблиця з тестами містить обмежену кількість колонок: ID, Дата та час створення заявки, Вибірка тестових завдань, Тип тестування, Викладач, Студент, Повідомлення, Максимальний час. Кнопка «Почати тестування» доступна.</td></tr>
<tr><td colspan="3">Результат виконання тест-кейсу: пройдено успішно</td></tr>
</table>

Четвертий тестовий випадок (таблиця 4.4) перевіряє функціонал презентування результату по завершенню тестування. Після прохоження тесту для студента має бути показаний результат тестування, що містить назву тесту, оцінку, час виконання, кількість питань, що була задана та кількість правильних відповідей (рисунок 4.6).

Таблиця 4.3 – Тест-кейс AT0003

Тест-кейс ID: AT0003	**Пріоритет: 1**	**Створено:** 08.11.2020, Г.Білоус
Назва: Перевірка функціоналу проходження тестування **Вхідні дані:** Логін = «bilous», Пароль = «bilous»		
Кроки	**Очікуваний результат**	
1. Запустити додаток 2. Ввести значення у поле логіну та паролю із вхідних даних 3. Натиснути кнопку «Вхід» 4. Порівняти фактичний результат з очікуваним 5. Обрати перший рядок у таблиці тестів 6. Натиснути кнопку «Почати тестування» 7. Порівняти фактичний результат з очікуваним 8. Дати відповідь на запитання 9. Натиснути кнопку «Наступне запитання» 10. Порівняти фактичний результат з очікуваним	Вікно входу у систему закрилось. Вікно з інтерфейсом студента відкрито. Відкрито вікно «Тестування». У вікні представлено тестове завдання, що відповідає тематиці вибірки тестових завдань першого рядка таблиці «Непройдені тести». Тестове запитання було змінено.	
Результат виконання тест-кейсу: пройдено успішно		

Рисунок 4.5 – Вікно «Тестування»

Рисунок 4.6 – Вікно «Результат»

Таблиця 4.4 – Тест-кейс AT0004

Тест-кейс ID: AT0004	**Пріоритет: 1**	**Створено:** 08.11.2020, Г.Білоус
Назва: Перевірка презентування результату по завершенню тестування **Вхідні дані:** Логін = «bilous», Пароль = «bilous»		
Кроки		**Очікуваний результат**
1. Запустити додаток 2. Ввести значення у поле логіну та паролю із вхідних даних 3. Натиснути кнопку «Вхід» 4. Порівняти фактичний результат з очікуваним 5. Обрати перший рядок у таблиці тестів 6. Натиснути кнопку «Почати тестування» 7. Порівняти фактичний результат з очікуваним 8. Пройти тестування. 9. Порівняти фактичний результат з очікуваним		Вікно входу у систему закрилось. Вікно з інтерфейсом студента відкрито. Відкрито вікно «Тестування», де представлено тестове завдання відповідно тематиці вибірки тестових завдань першого рядка таблиці «Непройдені тести». По завершенню тестування відкрилось вікно «Результат» із даними про результат поточного тестування.
Результат виконання тест-кейсу: пройдено успішно		

Отже, дослідження коректності виконання функцій проходження тестів у інформаційній системі успішно пройдено. Наочно доведено, що функції користувача студент вказані у п.3.1 було реалізовано у інформаційній системі.

Дослідження коректності виконання функцій адміністрування системи, проведене аналогічним чином, наведено у Додатку Е, а дослідження коректності виконання функцій викладача системи наведено у Додатку Ж.

4.2 Дослідження ефективності за часом, що витрачається під час адаптивного тестування

Як зазначалося у п. 1.3 перевагою адаптивного тестування над класичним тестуванням є те, що використання першого зменшує час проведення процесу тестування. Для дослідження ефективності за часом, що витрачається під час адаптивного тестування за розробленою інформаційною технологією, порівняно із класичним тестуванням було використано реалізовану інформаційну систему адаптивного контролю знань.

Як вхідні дані дослідження було використано множини тестових завдань, згенеровані за допомогою інформаційної системи автоматизованої генерації тестових завдань до навчальних матеріалів [39] на основі семантичної структури ІНМ у вигляді системи заголовків та відповідних множин ключових термінів для навчального курсу «Організація баз даних та знань». Застосовуючи одну й ту ж множину тестових завдань, було запропоновано за власним бажанням студентам пройти тестування з використанням класичного алгоритму проходження тесту та з використанням адаптивних алгоритмів (регресивним, прогресивним, медіанним) контролю знань у розробленій інформаційній системі адаптивного тестування.

Дослідження ефективності за часом, що витрачається під час тестування проведено шляхом оцінки різниці часу на виконання тесту за наведеними чотирма алгоритмами, що обраховується за наступними формулами:

$$\Delta T1_i = \frac{TK_i - TA1_i}{TK_i} \cdot 100,\ \overline{\Delta T1_i} = \frac{\sum_{j=1}^{n} \Delta T1_{i,j}}{n}, \quad (4.1)$$

$$\Delta T2_i = \frac{TK_i - TA2}{TK_i} \cdot 100,\ \overline{\Delta T2_i} = \frac{\sum_{j=1}^{n} \Delta T2_{i,j}}{n}, \quad (4.2)$$

$$\Delta T3_i = \frac{TK_i - TA3_i}{TK_i} \cdot 100,\ \overline{\Delta T3_i} = \frac{\sum_{j=1}^{n} \Delta T3_{i,j}}{n}, \quad (4.3)$$

де TK_i – час, що витрачається за класичного алгоритму на тестування i-м студентом; $TA1_i$ – витрачений час на тестування за регресивного адаптивного алгоритму i-м студентом; $TA2_i$ – витрачений час на тестування за прогресивного адаптивного алгоритму i-м студентом; $TA3_i$ – витрачений час на тестування за медіанним адаптивним алгоритмом i-м студентом; $\overline{\Delta T1_i}$ – середнє значення для вибірки з n проходжень тесту за класичним та регресивним адаптивним алгоритмами; $\overline{\Delta T2_i}$ – середнє значення для вибірки з n проходжень тесту за класичним та прогресивним адаптивним алгоритмами; $\overline{\Delta T3_i}$ – середнє значення для вибірки з n проходжень тесту за класичним та медіанним адаптивним алгоритмами.

На *Emani 1* дослідження ефективності за часом, що витрачається під час тестування різними алгоритмами викладач запланує проходження тестування за певною вибіркою тестових завдань (рисунок 4.7). Далі на *Emani 2 відбувається* проходження тестування реціпієнтами за класичним алгоритмом, на *Emani 3* – за регресивним адаптивним алгоритмом, на *Emani 4* за прогресивним адаптивним алгоритмом та на *Emani 5* за медіанним адаптивним алгоритмом (рисунок 4.8).

На *Emani 6* відбувається визначення різниці в часі тестування для кожного студента та на *Emani 7* обчислюються середні та граничні показники для різниці в часі тестування за тестовою вибіркою. Останнім є *Eman 8* на якому відбувається трактування результатів дослідження.

У таблиці 4.5 представлено фрагмент результатів дослідження під час проходження тесту із використанням однакового набору тестових завдань, що студенти проходили за класичного алгоритму та за адаптивних алгоритмів

(регресивним, прогресивним, медіанним) контролю знань у розробленій інформаційній системі тестування.

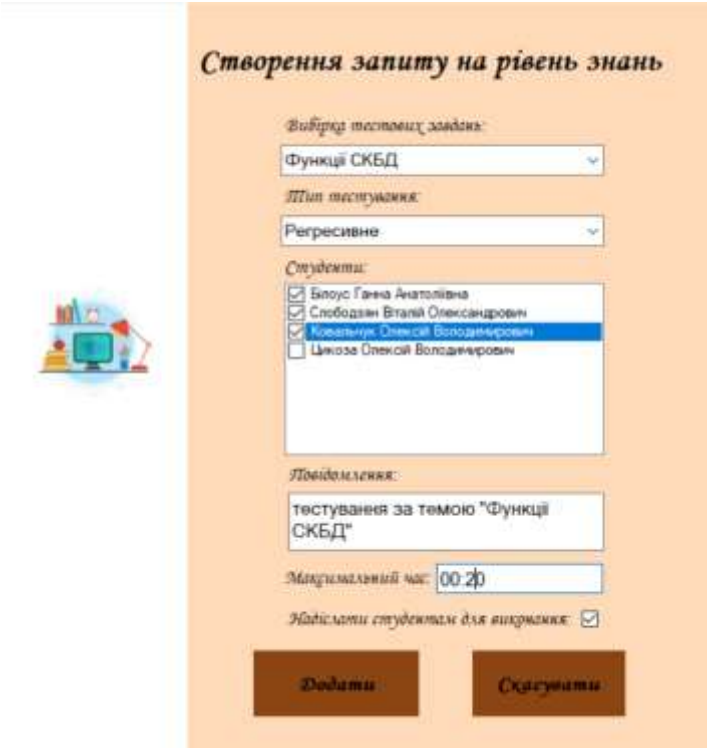


Рисунок 4.7 – Планування проходження тестування за певною вибіркою тестових завдань

Рисунок 4.8 – Приклад тестового завдання у розробленій інформаційній системі адаптивного тестування

Таблиця 4.5 – Фрагмент результатів дослідження за часом, що витрачається під час адаптивного тестування

№ п/п	Суб'єкт тесту-вання	Класичний алгоритм		Адаптивне регресивне тестування		Адаптивне прогресивне тестування		Адаптивне медіанне тестування	
		Витрачений час, хв.	Оцінка	Витрачений час, хв.	Оцінка	Витрачений час, хв.	Оцінка	Витрачений час, хв.	Оцінка
1.	Студент A	27	3,25	19	3,30	18	3,15	21	3,35
2.	Студент B	23	4,45	17	4,30	17	4,35	20	4,30
3.	Студент C	34	3,00	26	3,15	24	3,20	28	3,05
4.	Студент B	21	4,85	13	4,90	15	4,95	17	4,85
5.	Студент E	30	2,65	21	2,70	24	2,6	22	2,55
6.	Студент F	26	3,45	18	3,60	21	3,65	23	3,60
7.	Студент G	28	3,70	21	3,55	20	3,60	22	3,60
8.	Студент H	24	4,90	17	5,00	17	4,95	19	4,90
…	…	…	…	…	…	…	…	…	…

Провівши аналіз отриманих результатів дослідження можна визначити, що середній час проходження вибірки тестових завдань для 30 студентів за класифікацією відносно одержаної за національною шкалою оцінки при класичному алгоритмі інформаційної системи становить: для оцінки «відмінно» $\overline{TK_в}$ = 22,69 хв., для оцінки «добре» $\overline{TK_д}$ = 25,29 хв., для оцінки «задовільно» $\overline{TK_з}$ = 31,42 хв., для оцінки «незадовільно» $\overline{TK_н}$ = 29,13 хв., середній для всіх категорій $\overline{TK_с}$ = 27,13 хв..

Середній час проходження вибірки тестових завдань за аналогічних умов з використанням регресивного адаптивного алгоритму проходження тесту склав: для оцінки «відмінно» $\overline{TA1_в}$ = 18,45 хв., для оцінки «добре» $\overline{TA1_д}$ = 17,97 хв., для оцінки «задовільно» $\overline{TA1_з}$ = 25,08 хв., для оцінки «незадовільно» $\overline{TA1_н}$ = 21,79 хв., середній для всіх категорій $\overline{TA1_с}$ = 20,82 хв..

Середній час проходження тесту за аналогічних умов з використанням прогресивного адаптивного алгоритму становить: для оцінки «відмінно» $\overline{TA2_в}$ = 17,86 хв., для оцінки «добре» $\overline{TA2_д}$ = 17,56 хв., для оцінки «задовільно» $\overline{TA2_з}$ = 24,12 хв., для оцінки «незадовільно» $\overline{TA2_н}$ = 21,85 хв., середній для всіх категорій $\overline{TA2_с}$ = 20,34 хв..

Середній час тестування за тотожних умов з використанням медіанного адаптивного алгоритму становить: для оцінки «відмінно» $\overline{TA3_в}$ = 19,04 хв., для оцінки «добре» $\overline{TA3_д}$ = 21,69 хв., для оцінки «задовільно» $\overline{TA3_з}$ = 24,91 хв., для оцінки «незадовільно» $\overline{TA3_н}$ = 23,46 хв., середній для всіх категорій $\overline{TA2_c}$ = 22,28 хв..

Графічне подання результатів дослідження тестування за середнім часом зображено на рисунку 4.9.

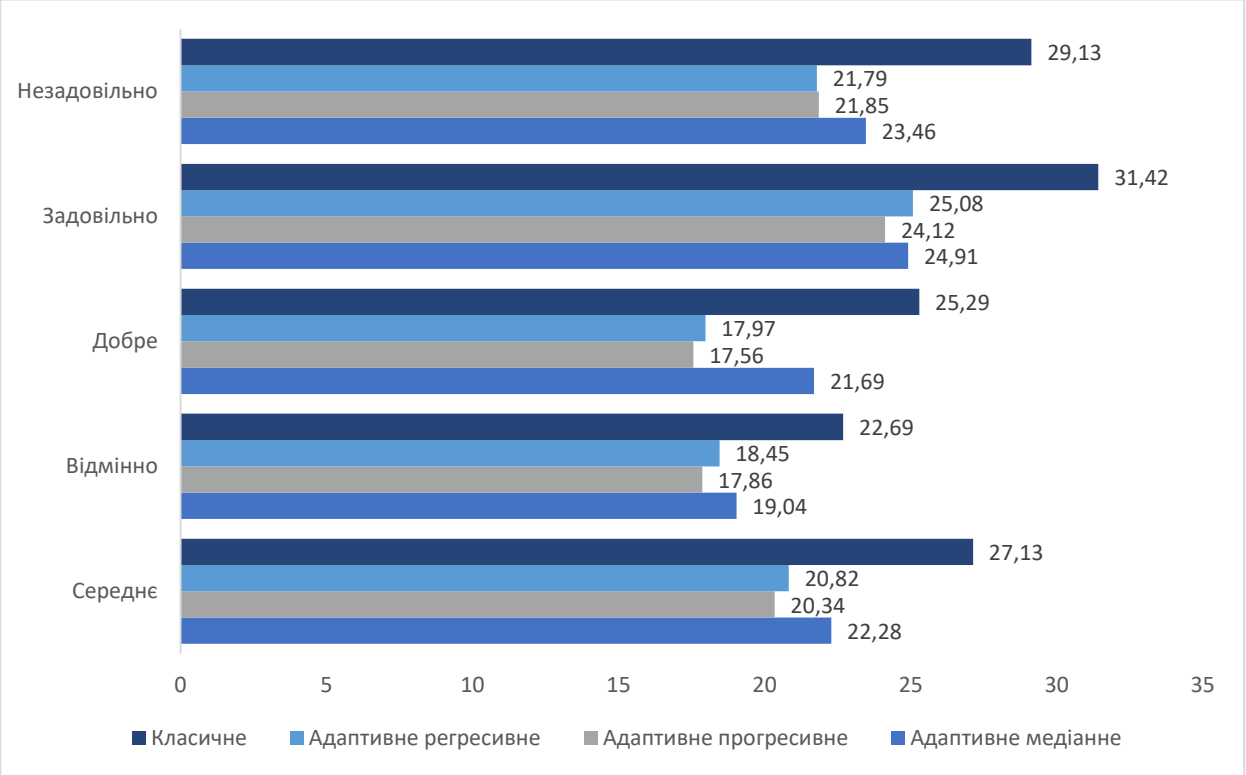


Рисунок 4.9 – Середній час тестування, хв.

Отриманні результати дозволили визначити ефективність від використання регресивного адаптивного тестування порівняно із класичним за різницею в часі тестування: для оцінки «відмінно» $\overline{\Delta T1_в}$ = 18,69 %; для оцінки «добре» $\overline{\Delta T1_д}$ = 28,94 %; для оцінки «задовільно» $\overline{\Delta T1_з}$ = 20,18 %; для оцінки «незадовільно» $\overline{\Delta T1_н}$ = 25,2 %; середнє значення за всіма категоріями $\overline{\Delta T1c}$ = 23,26 %.

Ефективність прогресійного адаптивного тестування у порівнянні із класичним за різницею в часі тестування наступна: для оцінки «відмінно» $\overline{\Delta T2_в}$ = 21,29 %; для оцінки «добре» $\overline{\Delta T2_д}$ = 30,56 %; для оцінки «задовільно» $\overline{\Delta T2_з}$ = 23,23 %; для оцінки «незадовільно» $\overline{\Delta T2_н}$ = 24,99 %; середнє значення за всіма категоріями $\overline{\Delta T2c}$ = 25,03 %.

При порівнянні результатів за медіанним адаптивним тестуванням та класичним за різницею в часі тестування ефективність наступна: для оцінки «відмінно» $\overline{\Delta T3_в}$ = 16,09 %; для оцінки «добре» $\overline{\Delta T3_д}$ = 14,23 %; для оцінки «задовільно» $\overline{\Delta T3_з}$ = 20,72 %; для оцінки «незадовільно» $\overline{\Delta T3_н}$ = 19,46 %; середнє значення за всіма категоріями $\overline{\Delta T3c}$ = 17,88 %.

Результати дослідження ефективності за часом, що витрачається під час адаптивного тестування за розробленою інформаційною технологією, підтвердили перевагу тестування за адаптивними алгоритмами над класичним. Використання адаптивних алгоритмів тестування зменшує час проведення процесу тестування в середньому на 22,06 %.

4.3 Дослідження ефективності за кількістю одержаних тестових завдань під час адаптивного тестування

Необхідна кількість тестових завдань за класичного алгоритму тестування визначається шляхом їх випадкового додавання до тесту допоки не закінчиться вказаний максимальний час або їх кількість фіксована та відома зазделегіть. Необхідна кількість тестових завдань за адаптивного алгоритму знаходиться шляхом визначення кінцевого результату за рівнем знань відповідно до кожного із заголовків за структурою ІНМ.

Для дослідження ефективності за кількістю одержаних тестових завдань під час адаптивного тестування за інформаційною технологією у порівнянні із класичним тестуванням використовувалася розроблена інформаційна система

адаптивного контролю знань. Як матеріал дослідження використовувалися множини тестових завдань, сформовані інформаційною системою автоматизованої генерації тестових завдань до навчальних матеріалів [39] на основі семантичної структури ІНМ та відповідних множин ключових термінів для навчального курсу «Організація баз даних та знань».

Використовуючи одну й ту ж множину тестових завдань, було запропоновано за власним бажанням студентам пройти тестування за класичним алгоритмом та за адаптивними алгоритмами (регресивним, прогресивним, медіанним) контролю знань у реалізованій інформаційній системі адаптивного тестування рівня знань. Ефективність використання адаптивного тестування за різницею в кількості отриманих при тестуванні завдань визначається так:

$$\Delta N1_i = \frac{NK_i - NA1_i}{NK_i} \cdot 100,\ \overline{\Delta N1_i} = \frac{\sum_{j=1}^{n} \Delta N1_{i,j}}{n}, \quad (4.4)$$

$$\Delta N2_i = \frac{NK_i - NA2_i}{NK_i} \cdot 100,\ \overline{\Delta N2_i} = \frac{\sum_{j=1}^{n} \Delta N2_{i,j}}{n}, \quad (4.5)$$

$$\Delta N3_i = \frac{NK_i - NA3_i}{NK_i} \cdot 100,\ \overline{\Delta N3_i} = \frac{\sum_{j=1}^{n} \Delta N3_{i,j}}{n}, \quad (4.6)$$

де NK_i – кількість завдань виконаних при тестуванні за класичним алгоритмом i-м студентом; $NA1_i$ – кількість завдань, одержана під час тестування за регресивного адаптивного алгоритму i-м студентом; $NA2_i$ – кількість завдань, одержана під час тестування за прогресійного адаптивного алгоритму i-м студентом; $NA3_i$ – кількість завдань, одержана під час тестування за медіанного адаптивного алгоритму i-м студентом; $\overline{\Delta N1_i}$ – середнє значення для вибірки з n проходжень тесту за класичним та регресивним адаптивним алгоритмами; $\overline{\Delta N2_i}$ – середнє значення для вибірки з n проходжень тесту за класичним та прогресійним адаптивним алгоритмами; $\overline{\Delta N3_i}$ – середнє значення для вибірки з n проходжень тесту за класичним та медіанним адаптивним алгоритмами.

Етапи 1-5 дослідження ефективності за кількістю одержаних тестових завдань під час адаптивного тестування за інформаційною технологією є аналогічним до *Етапів 1-5* дослідження ефективності за часом, що витрачається

під час адаптивного тестування за інформаційною технологією. Далі на *Emani 6* відбувається визначення різниці кількості одержаних тестових завдань для кожного студента та на *Emani 7* обчислюються середні та граничні показники для різниці кількості одержаних тестових завдань за тестовою вибіркою. Останнім є *Eman 8* на якому відбувається інтерпретація результатів дослідження.

У таблиці 4.6 представлено фрагмент результатів дослідження проходження тесту із застосуванням одного й того ж набору тестових завдань, що студенти проходили за класичного алгоритму та адаптивних алгоритмів (регресивним, прогресивним, медіанним) контролю знань у розробленій інформаційній системі.

Таблиця 4.6 – Фрагмент результатів дослідження ефективності за кількістю одержаних тестових завдань під час адаптивного тестування

№ п/п	Суб'єкт тесту-вання	Класичний алгоритм		Адаптивне регресивне тестування		Адаптивне прогресивне тестування		Адаптивне медіанне тестування	
		Кількість завдань	Оцінка	Кількість завдань	Оцінка	Кількість завдань	Оцінка	Кількість завдань	Оцінка
1.	Студент A	24	3,25	18	3,30	16	3,15	17	3,35
2.	Студент B	25	4,45	20	4,30	21	4,35	19	4,30
3.	Студент C	25	3,00	16	3,15	15	3,20	17	3,05
4.	Студент B	24	4,85	19	4,90	15	4,95	18	4,85
5.	Студент E	25	2,65	11	2,70	13	2,6	12	2,55
6.	Студент F	25	3,45	18	3,60	15	3,65	17	3,60
7.	Студент G	23	3,70	21	3,55	20	3,60	19	3,60
8.	Студент H	25	4,90	15	5,00	17	4,95	16	4,90
…	…	…	…	…	…	…	…	…	…

Проаналізувавши результати дослідження, можна визначити, що середня кількість отриманих тестових завдань за використання класичного алгоритму склала: для оцінки «відмінно» $\overline{NK_в}$ = 24,90 од., для оцінки «добре» $\overline{NK_д}$ = 24,68 од., для оцінки «задовільно» $\overline{NK_з}$ = 24,10 од., для оцінки «незадовільно» $\overline{NK_н}$ = 23,31 од., середній для всіх категорій $\overline{NK_1}$ = 24,25 од..

Середня кількість одержаних тестових завдань за аналогічних умов з використанням регресивного адаптивного алгоритму проходження тесту становить: для оцінки «відмінно» $\overline{NA1_в}$ = 15,21 од., для оцінки «добре» $\overline{NA1_д}$ =

19,52 од., для оцінки «задовільно» $\overline{NA1_з}$ = 18,76 од., для оцінки «незадовільно» $\overline{NA1_н}$ = 12,94 од., середній для всіх категорій $\overline{NA1_с}$ = 16,61 од..

Середня кількість одержаних тестових завдань за тотожних умов з використанням прогресивного адаптивного алгоритму проходження тесту становить: для оцінки «відмінно» $\overline{NA2_в}$ = 16,45 од., для оцінки «добре» $\overline{NA2_д}$ = 20,02 од., для оцінки «задовільно» $\overline{NA2_з}$ = 19,13 од., для оцінки «незадовільно» $\overline{NA2_н}$ = 13,24 од., середній для всіх категорій $\overline{NA2_с}$ = 17,21 од..

Середня кількість одержаних тестових завдань за аналогічних умов з використанням регресивного адаптивного алгоритму проходження тесту склала: для оцінки «відмінно» $\overline{NA3_в}$ = 15,96 од., для оцінки «добре» $\overline{NA3_д}$ = 21,31 од., для оцінки «задовільно» $\overline{NA3_з}$ = 20,42 од., для оцінки «незадовільно» $\overline{NA3_н}$ = 14,66 од., середній для всіх категорій $\overline{NA3_с}$ = 18,09 од..

Графічне подання результатів дослідження тестування за середньою кількістю отриманих тестових завдань зображено на рисунку 4.10.

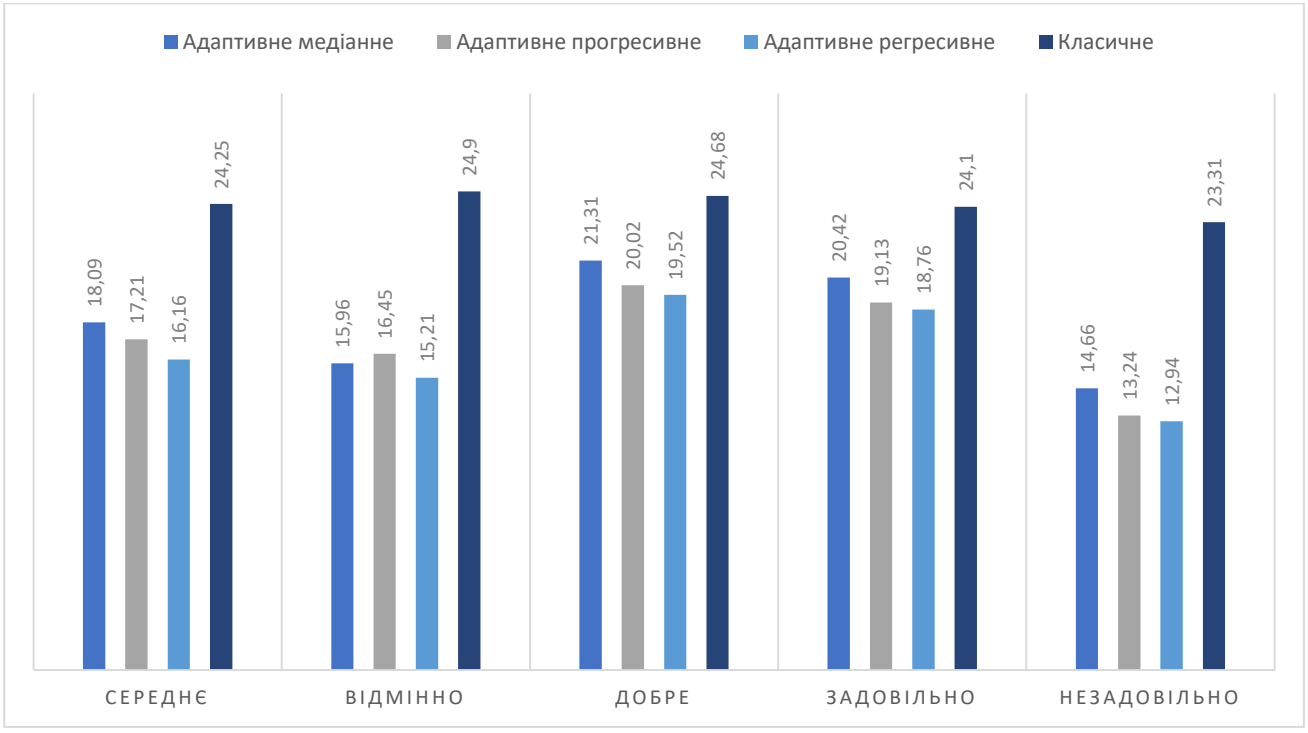


Рисунок 4.10 – Середня кількість одержаних тестових завдань, од.

Отримані результати дослідження дозволили визначити ефективність від використання регресивного адаптивного тестування у порівнянні із класичним тестуванням за кількістю одержаних тестових завдань: для оцінки «відмінно» $\overline{\Delta N1_в}$ = 38,92 %, для оцінки «добре» $\overline{\Delta N1_д}$ = 20,91 %, для оцінки «задовільно» $\overline{\Delta N1_з}$ = 22,16 %, для оцінки «незадовільно» $\overline{\Delta N1_н}$ = 44,49 %, середнє значення за всіма категоріями $\overline{\Delta N1c}$ = 33,36 %.

Ефективність прогресивного адаптивного тестування у порівнянні із класичним за кількістю отриманих тестових завдань під час тестування наступна: для оцінки «відмінно» $\overline{\Delta N2_в}$ = 33,94 %, для оцінки «добре» $\overline{\Delta N2_д}$ = 18,88 %, для оцінки «задовільно» $\overline{\Delta N2_з}$ = 20,62 %, для оцінки «незадовільно» $\overline{\Delta N2_н}$ = 43,2 %, середнє значення за всіма категоріями $\overline{\Delta N2c}$ = 29,03 %.

При порівнянні результатів за медіанним адаптивним тестуванням та класичним за кількістю отриманих тестових завдань ефективність наступна: для оцінки «відмінно» $\overline{\Delta N3_в}$ = 35,9 %, для оцінки «добре» $\overline{\Delta N3_д}$ = 13,65 %, для оцінки «задовільно» $\overline{\Delta N3_з}$ = 15,27 %, для оцінки «незадовільно» $\overline{\Delta N3_н}$ = 37,11 %, середнє значення за всіма категоріями $\overline{\Delta N3c}$ = 25,4 %.

Результати дослідження ефективності за кількістю одержаних тестових завдань під час адаптивного тестування за розробленою інформаційною технологією, підтвердили те, що при тестуванні за адаптивними алгоритмами кількість завдань даних під час тестування буде меншою, ніж при класичному алгоритмі. Використання адаптивних алгоритмів тестування зменшує кількість тестових завдань в середньому на 29,26 %.

4.4 Результати використання інформаційної технології

Проведені дослідження довели ефективність адаптивних алгоритмів порівняно із класичним алгоритмом за різницею у витраченому часі на процес тестування (в середньому час зменшується на $\overline{\Delta Tc}$ = 22,06 %) та за різницею

отриманих тестових завдань студентами (в середньому для визначення рівня знань знадобилося використання на $\overline{\Delta Nc}$ = 29,26 % меншої кількості завдань).

Під час проведення досліджень було виявлено, що чим вищим є рівень знань студента, тим менше завдань потрібно для визначення рівня знань за адаптивного тестування (це притаманно для кожного із трьох запропонованих алгоритмів), але при цьому більше часу було витрачено на їх вирішення (що пояснюється різними типами завдань, які даються для підтвердження розуміння ключових термінів). Тобто, якщо для оцінки «задовільно» кількість отриманих під час тестування запитань зменшилось на $\overline{Nc_з}$ = 4,66 одн. та витрачений час зменшився на $\overline{Tc_з}$ = 6,72 хв, то для оцінки «відмінно» відповідно ці дані зменшилися на $\overline{Nc_в}$ = 9,03 одн. та $\overline{Tc_в}$ = 4,24 хв.

В той час як за класичного алгоритму різниця між кількістю отриманих тестових завдань становить менше 1 одн. для студентів із «задовільним» та «відмінним» рівнем знань. Проте різниця у часі витраченому на процес тестування за класичного алгоритму є також значною в протилежну до адаптивного тестування сторону: студентам із «задовільним» рівнем знань потрібно на 8,73 хв більше часу, ніж для відмінників.

Також під час досліджень було виявлено, що чим нижчим є рівень знань опитуваного (до порівняння приймаються результати з оцінками «незадовільно» та «добре»), тим також менше тестових завдань знадобиться для визначення його рівня знань алгоритмами адаптивного тестування, але при цьому часу у середньому витрачається майже однаково. Тобто, якщо для оцінки «добре» витрачений час зменшився на $\overline{Tc_д}$ = 6,22 хв та кількість одержаних тестових запитань під час тестування зменшилась в середньому на $\overline{Nc_д}$ = 4,4 одн., то для оцінки «незадовільно» відповідно ці дані зменшилися на $\overline{Tc_н}$ = 6,76 хв та $\overline{Nc_н}$ = 9,7 одн..

На об’єктивність результатів має значний вплив коректність сформованих тестових завдань, оскільки для формування тестового набору, тестові завдання створюються так, щоб вони рівномірно покривали ІНМ. Маючи коректно сформованований набір тестових завдань, запропонований підхід дозволяє

точніше виявляти рівномірність рівня отриманих знань та визначати прогалини в розумінні вивченого матеріалу.

Висновки до розділу 4

Під час дослідження функціональності розробленої інформаційної системи адаптивного тестування рівня знань було підтверджено, що функції користувачів типу адміністратор, викладач, студент було реалізовано: до функцій, що доступні адміністратору, належить робота з даними про користувачів; викладач може виконувати роботу з тестовими завданнями, роботу із вибірками тестових завдань та роботу із проходженням тестів; студенту доступний функціонал для роботи з проходженням тестів.

Також з метою визначення ефективності за різницею у часі та кількості отриманих тестових завдань регресивного, прогресивного та медіанного алгоритмів адаптивного тестування в порівнянні із класичним, було проведено дослідження із використанням реалізованої інформаційної системи адаптивного контролю знань. Алгоритми адаптивного тестування забезпечують в середньому на 22,06 % більш швидке проведення процесу тестування, при цьому для визначення рівня знань студентів знадобилося в середньому на 29,26 % менше тестових завдань.

Коректність сформованих тестових завдань має великий вплив на об’єктивність результатів адаптивного тестування, оскільки для формування тесту тестові завдання необхідно створювати так чином, щоб вони рівномірно охоплювали ІНМ. Завдяки коректно сформованому набору тестових завдань, запропонований метод адаптивного тестування дозволяє точніше виявити єдність рівнів знань та виявити прогалини в розумінні вивчених матеріалів.

Загальні висновки

Робота розв’язує науково-технічну задачу створення інформаційної технології для забезпечення адаптивного тестування рівня знань. *Результатом дипломної роботи магістра* є розроблені інформаційна технологія та автоматизована система адаптивного тестування рівня знань. В ході виконання дипломної роботи магістра було вирішено наступні завдання:

– досліджено відомі підходи до адаптивного тестування рівня знань;

– вдосконалено інформаційну модель адаптивного тесту для формального подання елементів терміноорієнтованого адаптивного тестування рівня знань;

– вдосконалено метод адаптивного вибору тестових запитань для динамічного вибору тестових завдань у процесі тестування;

– розроблено інформаційну технологію адаптивного тестування рівня знань, що використовує вдосконалені модель і метод;

– розроблено інформаційну систему адаптивного тестування рівня знань на основі розробленої інформаційної технології;

– досліджено практичну ефективність інформаційної технології адаптивного тестування рівня знань шляхом аналізу функціональності й ефективності застосування відповідної інформаційної системи.

Було виконано дослідження ефективності інформаційної технології адаптивного тестування рівня знань під час яких було визначено, що алгоритми адаптивного тестування забезпечують в середньому на 22,06 %. більш швидке проведення процесу тестування, ніж при класичному тестуванні, при цьому для визначення рівня знань студентів знадобилося в середньому на 29,26 % менше тестових завдань.

В результаті проведеної роботи були отримані наступні наукові положення:

1. Вдосконалено інформаційну модель адаптивного тесту, яка відрізняється тим, що містить формальне подання складових предметної області терміноорієнтованого адаптивного тестування рівня знань.

2. Вдосконалено метод адаптивного вибору тестових запитань, який відрізняється тим, що використовує показники семантичної важливості ключових термінів навчального матеріалу для динамічного вибору тестових завдань у процесі тестування, застосовуючи один із розроблених алгоритмів адаптивного вибору тестових завдань: регресивний, прогресивний та медіанний.

3. Розроблено нову інформаційну технологію адаптивного тестування рівня знань, що використовує вдосконалені інформаційну модель і метод адаптивного вибору тестових запитань й дозволяє за вхідними даними у вигляді пов’язаних із семантичною структурою навчального курсу тестових завдань одержувати в результаті проходження тесту вихідні дані у вигляді оцінки результату тестування, що вираховується залежно від відповідей опитуваного впродовж тестування.

4. Розроблено нову інформаційну систему адаптивного тестування рівня знань за розробленою інформаційною технологією, що дозволяє працювати з тестовим матеріалом, створювати цільові вибірки тестових завдань, формувати індивідуальні запити на рівень знань, проводити адаптивне тестування за одним із доступних алгоритмів та обраховувати результати проходження запитів на рівень знань.

За темою дипломної роботи магістра автором виконано п’ять наукових публікацій [40, 41, 42, 43, 44]. Основні наукові та практичні результати доповідалися на конференціях: VII Міжнародній науково-практичної конференції «Інформаційні управляючі системи та технології ICST-ODESSA-2018», Всеукраїнській науково-практичній конференції «Інтелектуальний потенціал – 2018», XI Всеукраїнській науково-практичній конференції «Актуальні проблеми комп’ютерних наук АПКН-2019» (Хмельницький, 14-15 листопада 2019 року, Хмельницький національний університет), XII Всеукраїнській науково-практичній конференції «Актуальні проблеми комп’ютерних наук АПКН-2020».

Перелік посилань

1. Комп’ютерне тестування як одна з форм діагностики та контролю якості знань студентів – [Електронний ресурс]. – Режим доступу: https://naurok.com.ua/stattya-komp-yuterne-testuvannya-yak-odna-z-form-diagnostiki-ta-kontrolyu-yakosti-znan-studentiv-28883.html

2. Концептуальні основи оцінювання якості шкільної освіти – [Електронний ресурс]. – Режим доступу: http://www.ippobuk.cv.ua/images/Шепенюк_7.pdf

3. Мазурець О. В. Особливості автоматизованого формування тестових завдань / О. В. Мазурець // Матеріали V Міжнародної науково-практичної конференції «Інформаційні управляючі системи та технології ICST-ODESSA-2016». Одеса – 2016. – С.71-73.

4. Конструювання тестів – [Електронний ресурс]. – Режим доступу: http://moodle.ndu.edu.ua/pluginfile.php/889/mod_page/content/1/kt.pdf

5. Курсы и дисциплины – [Електронний ресурс]. – Режим доступу: http://edu.tltsu.ru/er/book_view.php?book_id=964&page_id=5312

6. Навчальний матеріал та його структура – [Електронний ресурс]. – Режим доступу: http://www.rusnauka.com/19_DSN_2010/Pedagogica/69745.doc.htm

7. Метод опитування – [Електронний ресурс]. – Режим доступу: https://studopedia.org/5-41976.html

8. Тестологія як наука – [Електронний ресурс]. – Режим доступу: https://moodle-student.fi.npu.edu.ua/pluginfile.php/2085/mod_resource/content/1/Л_01_1.doc

9. Тестові завдання – [Електронний ресурс]. – Режим доступу: http://www.zhu.edu.ua/mk_school/mod/page/view.php?id=2027&lang=ru

10. Форми тестових завдань. Форма подання тестового завдання – [Електронний ресурс]. – Режим доступу: https://pidru4niki.com/12590605/informatika/formi_testovih_zavdan_forma_podannya_testovogo_zavdannya

11. Тестові завдання системи Moodle – [Електронний ресурс]. – Режим доступу: http://virtuni.education.zp.ua/edu_cpu/mod/resource/view.php?id=72101

12. Крак Ю. В. Інформаційна модель семантичної структури навчального курсу для генерації тестових завдань / Ю. В. Крак, О. В. Бармак, О. В. Мазурець // Матеріали XIX Міжнародної науково-практичної конференції «Моделювання та дослідження стійкості динамічних систем DSMSI-2019». Київ – 2019. – С.365-367.

13. Бармак О. В. Інформаційна модель семантичної структури навчального курсу / О. В. Бармак, О. В. Мазурець // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2018, №6, Т.1. – С.92-97.

14. Методичні рекомендації для розробників тестових завдань– [Електронний ресурс]. – Режим доступу: https://disted.edu.vn.ua/media/dlia_rosrobnukiv_testiv.pdf

15. Мазурець О. В. Інформаційна технологія автоматизованого структурування навчальних матеріалів та створення тестів для адаптивного контролю рівня знань / О. В. Мазурець // Дис. канд. техн. наук: спец. 05.13.05 «Інформаційні технології». Тернопільський національний економічний університет. – Тернопіль, 2019. – 217 с.

16. Комп’ютерне тестування як засіб оцінювання рівня компетентності учнів – [Електронний ресурс]. – Режим доступу: http://timso.koippo.kr.ua/hmura9/kompyuterne-testuvannya-yak-zasib-otsinyuvannya-rivnya-kompetentnosti-uchniv/

17. Биков В. Ю. Цифрова трансформація суспільства і розвиток комп’ютерно-технологічної платформи освіти і науки україни / В. Ю. Биков, В. В. Лапінський // Матеріали методологічного семінару НАПН України «Інформаційно-цифровий освітній простір України: трансформаційні процеси і перспективи розвитку» – 2019. – С.20-26.

18. Кадемія М. Ю. Дуальна освіта та інноваційні технології навчання / М. Ю. Кадемія, В. М. Кобися, А. П. Кобися // Збірник наукових праць «сучасні

інформаційні технології та інноваційні методики навчання в підготовці фахівців: методологія, теорія, досвід, проблеми». Київ - Вінниця – 2019. – С.99-104.

19. Федорак В. М. Комп'ютерне тестування – інноваційний метод контролю знань, навчальних досягнень студентів / В.М. Федорак // Галицький лікарський вісник. Івано-Франківськ– 2015. – № 3. – С.99-101.

20. Снитюк В. Е. Интеллектуальное управление оцениванием знаний / В. Е. Снитюк, К. Н. Юрченко // Черкассы, 2013. – 262 с.

21. Бармак О. В. Застосування інформаційної технології гнучкого тестування рівня знань у середовищі Moodle / О. В. Бармак, О. В. Мазурець, А. О. Матвійчук // Науковий журнал «Вісник Хмельницького національного університету" серія: Технічні науки. Хмельницький, 2017, №3. – С.103-115.

22. Загребельний С. Л. Адаптивне тестування як один із способів перевірки знань студентів у технічному вузі / С. Л. Загребельний, М. В. Брус // Науковий Вісник Донбаської державної машинобудівної академії. № 1 (22Е). Краматорськ – 2017. – С.155-162.

23. Classtime – [Електронний ресурс]. – Режим доступу: https://www.classtime.com/uk/

24. Moodle – [Електронний ресурс]. – Режим доступу: https://moodle.org/?lang=uk

25. Khan Academy – [Електронний ресурс]. – Режим доступу: https://www.khanacademy.org /

26. Тестовий контроль знань студентів у системі Moodle– [Електронний ресурс]. – Режим доступу: http://elibrary.kubg.edu.ua/10373/1/D_Bodnenko_L_Varchenko_O_Zhylcov_Testovy_kontrol.pdf

27. Computer-adaptive test– [Електронний ресурс]. – Режим доступу: https://www.edglossary.org/computer-adaptive-test/

28. Триус Ю. В. Засоби комп'ютеризованого адаптивного навчання і тестування у внз / Ю. В . Триус, О. О. Сотуленко // Матеріали другої

міжнародної конференції з адаптивних технологій управління навчанням ATL-2016. Одеса – 2016. – С. 103-106.

29. Мельник В. Д. Інтелектуальні технології контролю знань в комп'ютеризованому навчанні / В. Д. Мельник // Ukraine–EU. Modern Technology, Business and Law. – 2015. – С. 60-64.

30. ALEKS – [Електронний ресурс]. – Режим доступу: https://www.aleks.com/

31. Knewton – [Електронний ресурс]. – Режим доступу: https://www.knewton.com/

32. Слободзян В. О. Аналіз результатів автоматизованого пошуку ключових термінів у навчальних матеріалах / В. О. Слободзян, О. В. Мазурець // Збірник наукових праць за матеріалами XII всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2020» – Хмельницький, 2020, Т.1. – С.269-274.

33. Мова програмування Java – [Електронний ресурс]. – Режим доступу: http://avj.uuu.in.ua/articles/mova-programuvannja-java.html

34. Основні конструкції мов програмування Java та C# – Інтуіт [Електронний ресурс]. – Режим доступу: http://www.intuit.ru/studies/courses/64/64/lecture/1884

35. Вибір мови програмування та середовища розробки – Студопедія [Електронний ресурс]. – Режим доступу: https://studopedia.org/5-67082.html

36. Visual Studio 2017 – Вікіпедія [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio#Visual_Studio_2017

37. Обґрунтування вибору мови програмування – [Електронний ресурс]. – Режим доступу: https://studopedia.ru/15_59823_obruntuvannya-viboru-movi-programuvannya.html

38. C Sharp– Вікіпедія [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/ C_Sharp

39. Ковальчук О. В. Метод генерації тестових завдань до навчальних матеріалів на основі продукційних правил / О. В. Ковальчук, О. В. Мазурець //

Збірник наукових праць за матеріалами XII всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2020» – Хмельницький, 2020, Т.1. – С.33-142

40. Білоус Г. А. Розбиття 3d-об'єктів на тетраедри із заданим ступенем дискретності/ Г. А. Білоус, Т. К. Скрипник, Н. К. Медведчук // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2019, №3. – С.59-62.

41. Білоус Г. А. Параметризація моделі тестового завдання при автоматизованому формуванні тестів / Г. А. Білоус, О. В. Мазурець // Матеріали VII Міжнародної науково-практичної конференції «Інформаційні управляючі системи та технології ICST-ODESSA-2018». Одеса – 2018. – С.64-66.

42. Мазурець О. В. Метод формального опису елементів моделей автоматизованого формування тестових завдань / О. В. Мазурець, О. В. Ковальчук, В. О. Слободзян, Г. А. Білоус // Збірник наукових праць за матеріалами Всеукраїнської науково-практичної конференції «Інтелектуальний потенціал – 2018». Хмельницький, 2018, Ч.1. – С.51-56.

43. Ковальчук О. В. Використання програмного розширення spire.doc для автоматизації роботи з цифровими документами / О. В. Ковальчук, Г. А. Білоус, В. О. Слободзян // Збірник наукових праць за матеріалами XI всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2019» – Хмельницький, 2019, Т.1. – С.116-122.

44. Білоус Г. А. Інформаційна технологія адаптивного тестування рівня знань / Г. А. Білоус, О. В. Мазурець // Збірник наукових праць за матеріалами XII всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2020» – Хмельницький, 2020, Т.1. – С.33-41.

ДОДАТКИ

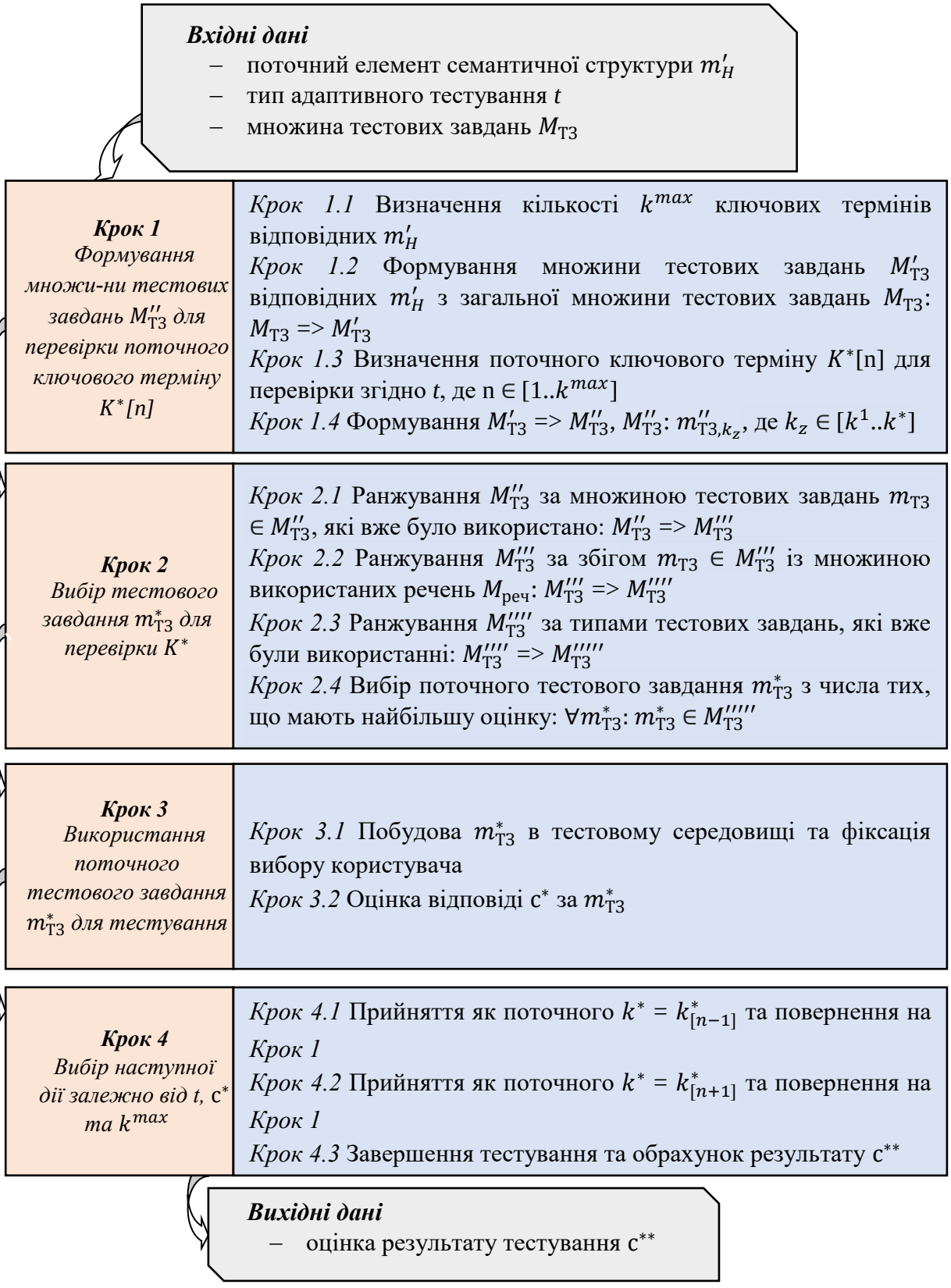
Додаток А

Результати аналізу відомих алгоритмів адаптивного тестування рівня знань

Назва	Опис
Алгоритм адаптивного тестування, при якому на початку тестування даються завдання середнього рівня складності	При використанні даного алгоритму тестування розпочинається із завдання середнього рівня складності й далі, залежно від відповідей, визначається складнішими чи легшими будуть подальші завдання. Недоліком даного алгоритму є застосування більшої кількості завдань у процесі тестування, аніж при алгоритмах, що починаються із перевірки найважливіших або найменш важливих термінів.
Алгоритм адаптивного тестування, що починається із перевірки найважливіших термінів	В ході тестування у випадку, коли студент відповідає правильно, складність тестових завдань підвищується шляхом перевірки менш важливих термінів. Недоліком є те, що коли студент має хороші знання, то він отримує більшу кількість завдань, аніж студент, що має задовільний рівень знань.
Алгоритм адаптивного тестування, при якому існує база знань, що містить у собі розподілені за рівнем складності тестові завдання	При використанні даного алгоритму за умови, що студент дав неправильну відповідь, наступне завдання обирається із вибірки більш легших завдань, при правильній – з більш складних. Недоліком алгоритму є те, що при визначені складності завдань не враховується семантична структура ІНМ, а, отже, неможливо повноцінно перевірити володіння навчальним матеріалом.
Алгоритм адаптивного тестування, що базується на обході за структурою НК зверху вниз	За умови застосування даного алгоритму, у випадку неправильної відповіді на найлегші тестові завдання, тестування у частині структури, що розташовується нижче – не проводиться. Недолік – при проходженні тестування студентами з рівнем знань відмінно, не перевірятиметься повноцінне володіння матеріалом НК.
Алгоритм адаптивного тестування, що передбачає, що контроль знань починається з будь-якого рівня складності тестових завдань	Даний алгоритм передбачає, що контроль знань починається з будь-якого рівня складності тестових завдань й далі ітераційним методом наближається до рівня складності, що є відповідним реальному рівню знань студента. Недоліком методу є те, що тестування може розпочинатися із різних рівнів складності для різних студентів, що може вплинути на результат тестування.

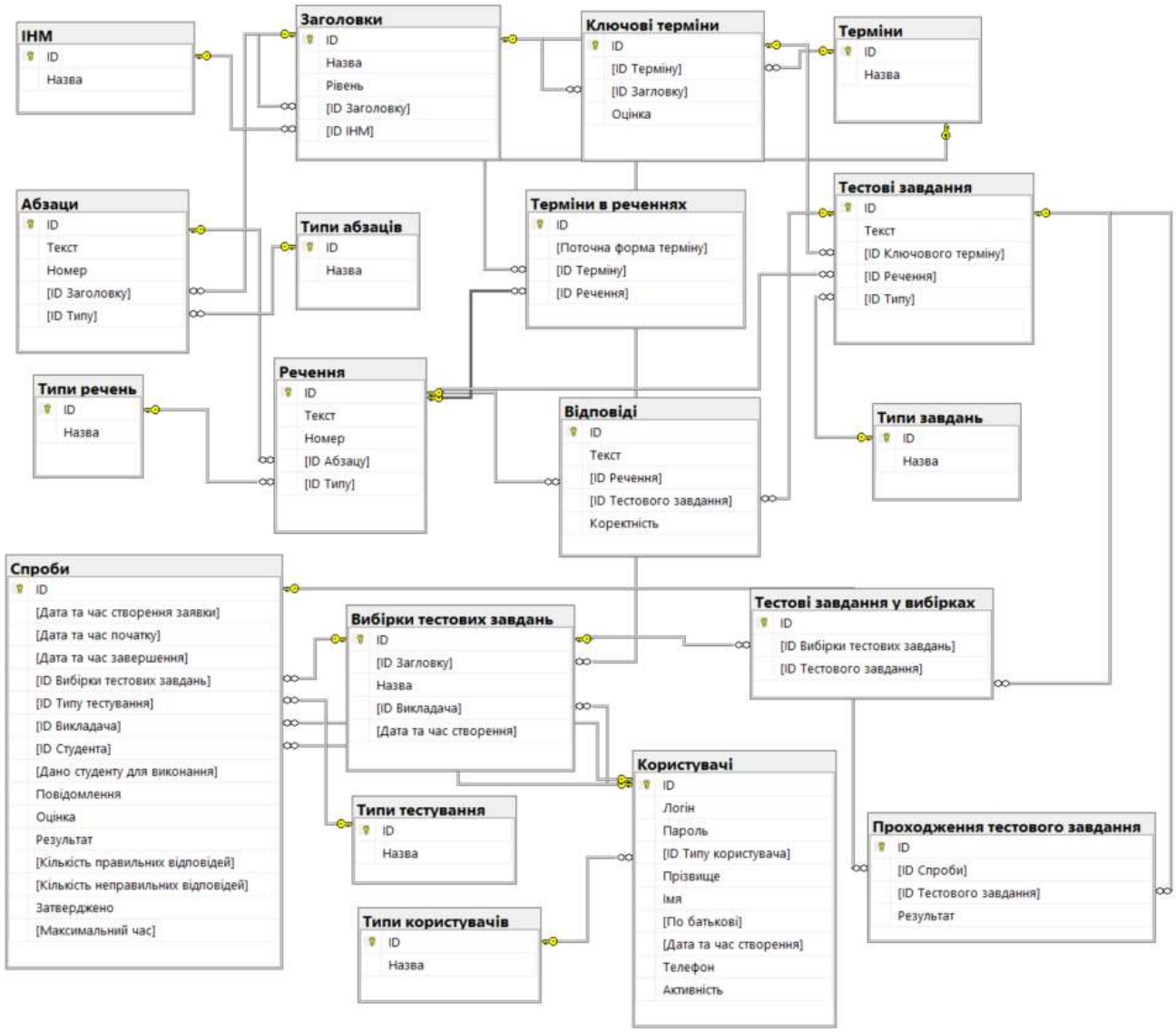
Додаток Б

Кроки інформаційної технології адаптивного тестування рівня знань



Додаток В

Структура бази даних інформаційної системи адаптивного тестування рівня знань



Додаток Г

Розгорнута структура класів автоматизованої системи

Додаток Д

Програмні коди

Лістинг Answer.cs:

```
namespace DRM_AIS_ComputerAdaptiveTesting.Entities
{
	public class Answer
	{
		public int ID { get; set; }
		public string Text { get; set; }
		public int FKSentence { get; set; }
		public int FKTestTask { get; set; }
		public bool IsCorrectness { get; set; }
	}
}
```

Лістинг Attempt.cs:

```
using System;
namespace DRM_AIS_ComputerAdaptiveTesting.Entities
{
	public class Attempt
	{
		public int ID { get; set; }
		public DateTime CreationDateTime { get; set;
}
		public DateTime StartDateTime { get; set; }
		public DateTime FinishDateTime { get; set; }
		public int FKSamplesOfTestTasks { get; set;
}
		public int FKTypeOfTesting { get; set; }
		public int FKTeacher { get; set; }
		public int FKStudent { get; set; }
		public bool IsGivenStudentToPerform { get;
set; }
		public string Message { get; set; }
		public string Score { get; set; }
		public string Result { get; set; }
		public int NumbeOfCorrectAnswers { get; set;
}
		public int NumbeOfIncorrectAnswers { get;
set; }
		public bool IsApproved { get; set; }
		public DateTime MaxTime { get; set; }
	}
}
```

Лістинг Heading.cs:

```
namespace DRM_AIS_ComputerAdaptiveTesting.Entities
{
	public class Heading
	{
		public int ID { get; set; }
		public string Name { get; set; }
		public int Level { get; set; }
		public int FKHeading { get; set; }
		public int FKIEM { get; set; }
	}
}
```

Лістинг SelectionOfTestTasks.cs:

```
using System;
namespace DRM_AIS_ComputerAdaptiveTesting.Entities
{
	public class SelectionOfTestTasks
	{
		public int ID { get; set; }
		public int FKHeading { get; set; }
		public string Name { get; set; }
		public int FKTeacher { get; set; }
		public DateTime CreationDateTime { get; set;
}
	}
}
```

Лістинг TestTask.cs:

```
namespace DRM_AIS_ComputerAdaptiveTesting.Entities
{
	public class TestTask
	{
		public int ID { get; set; }
		public string Text { get; set; }
		public string FKKeyTerm { get; set; }
		public int FKSentence { get; set; }
		public int FKTaskType { get; set; }
		public bool WasAsked { get; set; }
		public bool WasAnswerCorrect { get; set; }
	}
}
```

Лістинг User.cs:

```
using System;

namespace DRM_AIS_ComputerAdaptiveTesting.Entities
{
	public class User
	{
		public int ID { get; set; }
		public string Login { get; set; }
		public string Password { get; set; }
		public int FKTypeOfUser { get; set; }
		public string LastName { get; set; }
		public string FirstName { get; set; }
		public string SurName { get; set; }
		public DateTime CreationDateTime { get; set;
}
		public string PhoneNumber { get; set; }
		public bool IsActive { get; set; }
	}
}
```

Лістинг AdaptiveTestManager.cs:

```
using DRM_AIS_ComputerAdaptiveTesting.Entities;
using System;
using System.Collections.Generic;
using System.Linq;
using Task =
DRM_AIS_ComputerAdaptiveTesting.Entities.TestTask;

namespace DRM_AIS_ComputerAdaptiveTesting
{
	public class AdaptiveTestManager
	{
		List<Task> taskNames = new List<Task>();
		bool[] answersStatistics,
answersOnSetQuestions;
		public static int numberOfTask;
		public static int countRightAnswersOnTasks =
1;
		Dictionary<int, List<Task>> taskByPriority =
new Dictionary<int, List<Task>>();
		int priorityValue;

		private void GetAllTestData()
		{
			using (DatabaseManager dbManager =
new DatabaseManager())
			{
				taskNames =
dbManager.GetTestTask(Test.ID);
			}
			answersStatistics = new
bool[taskNames.Count];

			var maxValueOfPriority =
taskNames.Max(t => t.Priority);

			for (int i = 0; i <
maxValueOfPriority; i++)
			{
				taskByPriority.Add(i + 1,
taskNames.Where(t => t.Priority == i + 1).ToList());
			}
		}

		public Task FirstQuestionChoice()
		{
			GetAllTestData();
			Random random = new Random();
			//Task firstTask =
taskByPriority[1].First();
			Task firstTask =
taskByPriority[1][random.Next(taskByPriority[1].Count)];

			numberOfTask = 0;
			firstTask.WasAsked = true;
			return firstTask;
		}

		public Task NextQuestionChoice(bool
wasAnswerRight)
		{
			answersStatistics[numberOfTask] =
wasAnswerRight;

			Task nextTask = new Task();
			if (numberOfTask == 0)
			{
				// should add the check
on 'was asked' criteria
				//int randomNumber = 0;
				//do
				//{
				//	randomNumber =
GetRandomNumber(taskByPriority[1].Count);
				//} while
(!taskByPriority[1][randomNumber].WasAnswerCorrect);

				//nextTask =
taskByPriority[1][randomNumber];
				//nextTask.WasAsked =
true;
				priorityValue = 1;
```

```
                                nextTask                =
GetTaskByPriority(priorityValue);
                        }
                        else
                        {
                                if (CheckExitCriteria())
                                {

        answersOnSetQuestions = new bool[numberOfTask + 1];

        Array.Copy(answersStatistics,  answersOnSetQuestions,
numberOfTask + 1);

        countRightAnswersOnTasks                        =
answersOnSetQuestions.Where(a => a).Count();
                                        numberOfTask++;
                                        return       new
Task();
                                }

                                if (wasAnswerRight)
                                {
                                        if
(answersStatistics[numberOfTask - 1])
                                        {

        priorityValue       =       (priorityValue       !=
taskByPriority.Count) ? priorityValue + 1 : priorityValue;

        nextTask = GetTaskByPriority(priorityValue);
                                        }
                                        else
                                        {

        nextTask = GetTaskByPriority(priorityValue);
                                        }
                                }
                                else
                                {
                                        if
(answersStatistics[numberOfTask - 1])
                                        {

        nextTask = GetTaskByPriority(priorityValue);
                                        }
                                        else
                                        {

        priorityValue = (priorityValue != 1) ? priorityValue
- 1 : priorityValue;

        nextTask = GetTaskByPriority(priorityValue);
                                        }
                                }
                        }

                        numberOfTask++;
                        return nextTask;
                }

                private int GetRandomNumber(int countOfTask)
                {
                        Random random = new Random();
                        return random.Next(countOfTask);
                }

                private Task GetTaskByPriority(int priority)
                {
                        int randomNumber = 0;
                        do
                        {
                                randomNumber             =
GetRandomNumber(taskByPriority[priority].Count);
                        }                                       while
(taskByPriority[priority][randomNumber].WasAsked);


        taskByPriority[priority][randomNumber].WasAsked     =
true;
                        return
taskByPriority[priority][randomNumber];
                }

                private bool CheckExitCriteria()
                {
                        return numberOfTask == 15;
                }
        }
}
```

Лістинг AddEditAttemptWindow.cs:

```
using DRM_AIS_ComputerAdaptiveTesting.Entities;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace DRM_AIS_ComputerAdaptiveTesting
{
        public partial class AddEditAttemptWindow : Form
        {
                private bool dragging = false;
                private Point dragCursorPoint;
                private Point dragFormPoint;
                Dictionary<int,                         string>
samplesOfTestTasksNames, typeOfTestingNames, studentsList;
                string workWithAttemptMode;
                int currentAttemptId, currentUserId;

                public AddEditAttemptWindow(string workMode,
int attemptId, int userId, int rowIndex)
                {
                        InitializeComponent();

                        workWithAttemptMode = workMode;
                        currentUserId = userId;

                        using (DatabaseManager dbManager =
new DatabaseManager())
                        {
                                samplesOfTestTasksNames  =
dbManager.GetNameSamplesOfTestTasks();
                                foreach               (var
samplesOfTestTasksName in samplesOfTestTasksNames)
                                {

        sampleTestTasksComboBox.Items.Add(samplesOfTestTasksN
ame.Value);
                                }
                        }

                        using (DatabaseManager dbManager =
new DatabaseManager())
                        {
                                studentsList             =
dbManager.GetStudentList();
                                foreach  (var  student  in
studentsList)
                                {

        studentCheckedListBox.Items.Add(student.Value);
                                }
                        }

                        using (DatabaseManager dbManager =
new DatabaseManager())
                        {
                                typeOfTestingNames       =
dbManager.GetNameTypeOfTesting();
                                foreach               (var
typeOfTestingName in typeOfTestingNames)
                                {

        typeOfTestComboBox.Items.Add(typeOfTestingName.Value)
;
                                }
                        }

                        if  (workWithAttemptMode  ==  "Edit
attempt")
                        {
                                currentAttemptId         =
attemptId;

                                using     (DatabaseManager
dbManager = new DatabaseManager())
                                {
                                        Attempt
currentAttempt = dbManager.GetAttempt(attemptId);


        sampleTestTasksComboBox.Text                        =
samplesOfTestTasksNames.Values.ElementAt(currentAttempt.FKSamp
lesOfTestTasks - 1);

        typeOfTestComboBox.Text                             =
typeOfTestingNames.Values.ElementAt(currentAttempt.FKTypeOfTes
ting - 1);

        studentCheckedListBox.Text                          =
studentsList.Values.ElementAt(rowIndex);

        studentCheckedListBox.SetItemChecked(rowIndex, true);

        messageTextBox.Text = currentAttempt.Message;

        maxTimeTextBox.Text                                 =
currentAttempt.MaxTime.Minute.ToString() + " хвилин";
                                }

                                addEditButton.Text       =
"Відредагувати";
                                titleLabel.Text          =
"Відредагувати запит на рівень знань";
                        }
                }

                private                                 void
addEditUserForm_MouseDown(object sender, MouseEventArgs e)
                {
                        dragging = true;
                        dragCursorPoint = Cursor.Position;
                        dragFormPoint = this.Location;
                }

                private                                 void
addEditUserForm_MouseMove(object sender, MouseEventArgs e)
                {
                        if (dragging)
                        {
                                Point    newPosition    =
Point.Subtract(Cursor.Position, new Size(dragCursorPoint));
```

```
                                this.Location = Point.Add(dragFormPoint, new Size(newPosition));
                        }
                }

                private void addEditUserForm_MouseUp(object sender, MouseEventArgs e)
                {
                        dragging = false;
                }

                private void cancelButton_Click(object sender, EventArgs e)
                {
                        this.Close();
                }

                private void addEditButton_Click(object sender, EventArgs e)
                {
                        for (int i = 0; i < studentCheckedListBox.CheckedItems.Count; i++)
                        {
                                Attempt attempt = new Attempt();
                                attempt.CreationDateTime = DateTime.Now;
        attempt.FKSamplesOfTestTasks = samplesOfTestTasksNames.FirstOrDefault(x => x.Value == sampleTestTasksComboBox.Text).Key;
                                attempt.FKTypeOfTesting = typeOfTestingNames.FirstOrDefault(x => x.Value == typeOfTestComboBox.Text).Key;
                                attempt.FKTeacher = currentUserId;
                                attempt.FKStudent = studentsList.FirstOrDefault(x => x.Value == studentCheckedListBox.CheckedItems[i]).Key;
        attempt.IsGivenStudentToPerform = giveToPerformCheckBox.Checked;
                                attempt.Message = messageTextBox.Text;
                                attempt.MaxTime = Convert.ToDateTime(maxTimeTextBox.Text);

                                using (DatabaseManager dbManager = new DatabaseManager())
                                {
                                        if (workWithAttemptMode == "Add attempt")
                                        {
        dbManager.AddAttempt(attempt);
                                        }
                                        else
                                        {
                                                if (i==0)
                                                {
        dbManager.EditAttempt(currentAttemptId, attempt);
                                                }
                                                else
                                                {
        dbManager.AddAttempt(attempt);
                                                }
                                        }
                                }

                                this.Close();
                        }
                }
}
```

Лістинг AddEditSelectionOfTestTasks.cs:

```
using DRM_AIS_ComputerAdaptiveTesting.Entities;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace DRM_AIS_ComputerAdaptiveTesting
{
        public partial class AddEditSelectionOfTestTasks : Form
        {
                private bool dragging = false;
                private Point dragCursorPoint;
                private Point dragFormPoint;
                Dictionary<int, string> iemList;
                Dictionary<int, string> headingsListSelectedTestTasks = new Dictionary<int, string>();
                string workWithAttemptMode;
                int currentSelectionId;

                public AddEditSelectionOfTestTasks(string workMode, int selectionId)
                {
                        InitializeComponent();
                        workWithAttemptMode = workMode;
                        currentSelectionId = selectionId;

                        using (DatabaseManager dbManager = new DatabaseManager())
                        {
                                iemList = dbManager.GetIEM();

                                foreach (var iem in iemList)
                                {
        iemComboBox.Items.Add(iem.Value);
                                }
                        }

                        if (workWithAttemptMode == "Edit selection")
                        {
                                using (DatabaseManager dbManager = new DatabaseManager())
                                {
        SelectionOfTestTasks currentSelectionOfTestTasks = dbManager.GetSelection(selectionId);
                                        iemComboBox.Text = dbManager.GetIEM(currentSelectionOfTestTasks.FKHeading);
        headingComboBox.Text = dbManager.GetHeading(currentSelectionOfTestTasks.FKHeading);
                                        nameTextBox.Text = currentSelectionOfTestTasks.Name;


                                        DatabaseManager databaseManager = new DatabaseManager();
                                        SqlDataAdapter testTasksSqlDataAdapter = new SqlDataAdapter(string.Format($"SELECT testTask.[ID], testTask.[Текст], term.Назва [Ключовий термін], sentence.Текст [Речення], tasksTypes.Назва [Тип завдання] " +
                                                $"FROM [VAO Master Work].[dbo].[Тестові завдання] testTask " +
                                                $"join [VAO Master Work].[dbo].[Ключові терміни] keyTerm on testTask.[ID Ключового терміну] = keyTerm.id " +
                                                $"join [VAO Master Work].[dbo].[Терміни] term on keyTerm.[ID Терміну] = term.ID " +
                                                $"join [VAO Master Work].[dbo].[Речення] sentence on testTask.[ID Речення] = sentence.id " +
                                                $"join [VAO Master Work].[dbo].[Типи завдань] tasksTypes on testTask.[ID Типу] = tasksTypes.id " +
                                                $"WHERE keyTerm.[ID Загловку] = {currentSelectionOfTestTasks.FKHeading}"), databaseManager.connection);

                                        DataSet data = new DataSet();
        testTasksSqlDataAdapter.Fill(data, "Тестові завдання");
        testTasksDataGridView.DataSource = data;
        testTasksDataGridView.DataMember = "Тестові завдання";
                                }

                                addEditButton.Text = "Відредагувати";
                                titleLabel.Text = "Відредагувати вибірку тестових завдань";
                        }
                }

                private void addEditSelectionOfTestTasks_MouseDown(object sender, MouseEventArgs e)
                {
                        dragging = true;
                        dragCursorPoint = Cursor.Position;
                        dragFormPoint = this.Location;
                }

                private void addEditSelectionOfTestTasks_MouseMove(object sender, MouseEventArgs e)
                {
                        if (dragging)
                        {
                                Point newPosition = Point.Subtract(Cursor.Position, new Size(dragCursorPoint));
                                this.Location = Point.Add(dragFormPoint, new Size(newPosition));
                        }
                }

                private void addEditSelectionOfTestTasks_MouseUp(object sender, MouseEventArgs e)
                {
```

```
                dragging = false;
        }

        private    void    cancelButton_Click(object
sender, EventArgs e)
        {
                this.Close();
        }

        private                                  void
iemComboBox_SelectedValueChanged(object sender, EventArgs e)
        {
                List<Heading> headingsList;
                headingComboBox.Items.Clear();

    headingsListSelectedTestTasks.Clear();

                if (iemComboBox.Text == "")
                {
                        headingComboBox.Enabled  =
false;
                        headingComboBox.Text     =
"";
                }
                else
                {
                        using     (DatabaseManager
dbManager = new DatabaseManager())
                        {
                                headingsList    =
dbManager.GetHeadings(iemList.FirstOrDefault(x  =>  x.Value  ==
iemComboBox.Text).Key);
                                foreach      (var
heading in headingsList)
                                {

    headingComboBox.Items.Add(heading.Level             +
heading.Name);

    headingsListSelectedTestTasks.Add(heading.ID,
heading.Level + heading.Name);
                                }
                        }

                        headingComboBox.Enabled  =
true;
                }
        }

        private                                  void
headingComboBox_SelectedIndexChanged(object  sender,  EventArgs
e)
        {
                if (iemComboBox.Text != "")
                {
                        DatabaseManager
databaseManager = new DatabaseManager();
                        SqlDataAdapter
testTasksSqlDataAdapter                =                  new
SqlDataAdapter(string.Format($"SELECT          testTask.[ID],
testTask.[Текст],      term.Назва      [Ключовий      термін],
tasksTypes.Назва [Тип завдання] " +
                        $"FROM    [VAO    Master
Work].[dbo].[Тестові завдання] testTask " +
                        $"join    [VAO    Master
Work].[dbo].[Ключові    терміни]    keyTerm    on    testTask.[ID
Ключового терміну] = keyTerm.id " +
                        $"join    [VAO    Master
Work].[dbo].[Терміни] term on  keyTerm.[ID Терміну] = term.ID
" +
                        $"join    [VAO    Master
Work].[dbo].[Типи завдань] tasksTypes on testTask.[ID Типу] =
tasksTypes.id " +
                        $"WHERE      keyTerm.[ID
Загловку] = {headingsListSelectedTestTasks.FirstOrDefault(x =>
x.Value          ==          headingComboBox.Text).Key}"),
databaseManager.connection);
                        DataSet  testTasksData  =
new DataSet();

    testTasksSqlDataAdapter.Fill(testTasksData,  "Тестові
завдання");

    testTasksDataGridView.DataSource = testTasksData;

    testTasksDataGridView.DataMember      =      "Тестові
завдання";
                }
        }
    }
}
```

Лістинг AddEditTestTasksWindow.cs:

```
using DRM_AIS_ComputerAdaptiveTesting.Entities;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;

namespace DRM_AIS_ComputerAdaptiveTesting
{
    public partial class AddEditTestTasksWindow : Form
    {
        private bool dragging = false;
        private Point dragCursorPoint;
        private Point dragFormPoint;
        Dictionary<int,     string>     keyTermList,
testTasksTypeList;
        Dictionary<int,                      string>
headingsListSelectedTestTasks = new Dictionary<int, string>();
        string workWithAttemptMode;
        int currentTestTaskId;

        public          AddEditTestTasksWindow(string
workMode, int testTaskId)
        {
            InitializeComponent();

            workWithAttemptMode = workMode;
            currentTestTaskId = testTaskId;

            using (DatabaseManager dbManager =
new DatabaseManager())
            {
                keyTermList             =
dbManager.GetKeyTerm();

                foreach  (var  keyTerm  in
keyTermList)
                {

    keyTermComboBox.Items.Add(keyTerm.Value);
                }
            }

            using (DatabaseManager dbManager =
new DatabaseManager())
            {
                testTasksTypeList       =
dbManager.GetTestTaskTypes();

                foreach               (var
testTasksType in testTasksTypeList)
                {

    typeTestComboBox.Items.Add(testTasksType.Value);
                }
            }

            if  (workWithAttemptMode  ==  "Edit
task")
            {
                currentTestTaskId       =
testTaskId;

                using     (DatabaseManager
dbManager = new DatabaseManager())
                {
                    TestTask
currentTestTask = dbManager.GetTestTask(testTaskId);


    testTaskTextTextBox.Text = currentTestTask.Text;

    keyTermComboBox.Text = currentTestTask.FKKeyTerm;

    sentenceTextBox.Text = currentTestTask.Text;

    typeTestComboBox.Text                           =
dbManager.GetTestTaskType(currentTestTask.FKTaskType);


    ManageAnswerControls();

                    if
(typeTestComboBox.Text == "Логічний")
                    {

    logicalAnswerComboBox.Text                      =
dbManager.GetAnswersForOpenOrLogicalTasks(testTaskId);
                    }
                    else           if
(typeTestComboBox.Text == "Одиничного вибору")
                    {

    List<Answer>                 answers            =
dbManager.GetAnswersForChooseTasks(testTaskId);

    firstAnswerTextBox.Text = answers[0].Text;

    sentenceFirstAnswerTextBox.Text                 =
answers[0].FKSentence         ==         0?         ""         :
dbManager.GetSentence(answers[0].FKSentence);

    correctnessFirstAnswerCheckBox.Enabled          =
answers[0].IsCorrectness;

    textSecondAnswerTextBox.Text = answers[1].Text;

    sentenceSecondAnswerTextBox.Text                =
answers[1].FKSentence       ==       0       ?       ""       :
dbManager.GetSentence(answers[1].FKSentence);

    correctnessSecondAnswerCheckBox.Enabled         =
answers[1].IsCorrectness;


    textThirdAnswerTextBox.Text = answers[2].Text;

    sentenceThirdAnswerTextBox.Text                 =
answers[2].FKSentence       ==       0       ?       ""       :
dbManager.GetSentence(answers[2].FKSentence);
```

```
        correctnessSecondAnswerCheckBox.Enabled             =
answers[2].IsCorrectness;

        testForthAnswerTextBox.Text = answers[3].Text;

        sentenceForthAnswerTextBox.Text                     =
answers[3].FKSentence       ==       0       ?       ""       :
dbManager.GetSentence(answers[3].FKSentence);

        correctnessForthAnswerCheckBox.Enabled              =
answers[3].IsCorrectness;
                                        }
                                        else           if
(typeTestComboBox.Text == "Множиного вибору")
                                        {
                                        }
                                        else           if
(typeTestComboBox.Text == "Введення відповіді")
                                        {

        openAnswerTextBox.Text                              =
dbManager.GetAnswersForOpenOrLogicalTasks(testTaskId);
                                        }
                                }

                                addEditButton.Text      =
"Відредагувати";
                                titleLabel.Text         =
"Відредагувати тестове завдання";
                        }
                }
                private                                 void
addEditTestTasks_MouseDown(object sender, MouseEventArgs e)
                {
                        dragging = true;
                        dragCursorPoint = Cursor.Position;
                        dragFormPoint = this.Location;
                }

                private                                 void
addEditTestTasks_MouseMove(object sender, MouseEventArgs e)
                {
                        if (dragging)
                        {
                                Point    newPosition    =
Point.Subtract(Cursor.Position, new Size(dragCursorPoint));
                                this.Location           =
Point.Add(dragFormPoint, new Size(newPosition));
                        }
                }

                private void addEditTestTasks_MouseUp(object
sender, MouseEventArgs e)
                {
                        dragging = false;
                }

                private void ManageAnswerControls()
                {
                        if     (typeTestComboBox.Text     ==
"Логічний")
                        {

        logicalAnswerLabel.Visible = true;
        logicalAnswerComboBox.Visible = true;
        logicalAnswerComboBox.Items.Add("Так");
        logicalAnswerComboBox.Items.Add("Ні");

        firstAnswerTextBox.Enabled = false;
        sentenceFirstAnswerTextBox.Enabled = false;
        correctnessFirstAnswerCheckBox.Enabled = false;
        textSecondAnswerTextBox.Enabled = false;
        sentenceSecondAnswerTextBox.Enabled = false;
        correctnessSecondAnswerCheckBox.Enabled = false;

        textThirdAnswerTextBox.Enabled = false;
        sentenceThirdAnswerTextBox.Enabled = false;
        correctnessSecondAnswerCheckBox.Enabled = false;
        testForthAnswerTextBox.Enabled = false;
        sentenceForthAnswerTextBox.Enabled = false;
        correctnessForthAnswerCheckBox.Enabled = false;

                        openAnswerLabel.Visible = false;
                        openAnswerTextBox.Visible = false;
                        }
                        else  if  (typeTestComboBox.Text  ==
"Одиничного вибору")
                        {
        logicalAnswerLabel.Visible = false;
        logicalAnswerComboBox.Visible = false;
        firstAnswerTextBox.Enabled = true;
        sentenceFirstAnswerTextBox.Enabled = true;
        correctnessFirstAnswerCheckBox.Enabled = true;
        textSecondAnswerTextBox.Enabled = true;
        sentenceSecondAnswerTextBox.Enabled = true;
        correctnessSecondAnswerCheckBox.Enabled = true;

        textThirdAnswerTextBox.Enabled = true;
        sentenceThirdAnswerTextBox.Enabled = true;
        correctnessSecondAnswerCheckBox.Enabled = true;
        testForthAnswerTextBox.Enabled = true;
        sentenceForthAnswerTextBox.Enabled = true;
        correctnessForthAnswerCheckBox.Enabled = true;

                                openAnswerLabel.Visible  =
false;
                                openAnswerTextBox.Visible
= false;
                        }
                        else  if  (typeTestComboBox.Text  ==
"Множиного вибору")
                        {
        logicalAnswerLabel.Visible = false;
        logicalAnswerComboBox.Visible = false;

        firstAnswerTextBox.Enabled = true;
        sentenceFirstAnswerTextBox.Enabled = true;
        correctnessFirstAnswerCheckBox.Enabled = true;
        textSecondAnswerTextBox.Enabled = true;
        sentenceSecondAnswerTextBox.Enabled = true;
        correctnessSecondAnswerCheckBox.Enabled = true;

        textThirdAnswerTextBox.Enabled = true;
        sentenceThirdAnswerTextBox.Enabled = true;
        correctnessSecondAnswerCheckBox.Enabled = true;
        testForthAnswerTextBox.Enabled = true;
        sentenceForthAnswerTextBox.Enabled = true;
        correctnessForthAnswerCheckBox.Enabled = true;

                                openAnswerLabel.Visible  =
false;
                                openAnswerTextBox.Visible
= false;
                        }
                        else  if  (typeTestComboBox.Text  ==
"Введення відповіді")
                        {
        logicalAnswerLabel.Visible = false;
        logicalAnswerComboBox.Visible = false;

        firstAnswerTextBox.Enabled = false;
        sentenceFirstAnswerTextBox.Enabled = false;
        correctnessFirstAnswerCheckBox.Enabled = false;
        textSecondAnswerTextBox.Enabled = false;
        sentenceSecondAnswerTextBox.Enabled = false;
        correctnessSecondAnswerCheckBox.Enabled = false;

        textThirdAnswerTextBox.Enabled = false;
        sentenceThirdAnswerTextBox.Enabled = false;
        correctnessSecondAnswerCheckBox.Enabled = false;
        testForthAnswerTextBox.Enabled = false;
        sentenceForthAnswerTextBox.Enabled = false;
        correctnessForthAnswerCheckBox.Enabled = false;

                        openAnswerLabel.Visible = true;
                        openAnswerTextBox.Visible = true;
                        }
                }

                private                                 void
typeTestComboBox_SelectedValueChanged(object sender, EventArgs
e)
                {
                        ManageAnswerControls();
                }

                private    void    cancelButton_Click(object
sender, EventArgs e)
                {
                        this.Close();
                }
        }
}
```

Лістинг AddEditUserWindow.cs:

```
using DRM_AIS_ComputerAdaptiveTesting.Entities;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace DRM_AIS_ComputerAdaptiveTesting
{
        public partial class AddEditUserWindow : Form
        {
                private bool dragging = false;
                private Point dragCursorPoint;
                private Point dragFormPoint;
                Dictionary<int, string> typeOfUsersNames;
                string workWithUserMode;
                int currentUserId;

                public   AddEditUserWindow(string   workMode,
int userId)
                {
                        InitializeComponent();

                        workWithUserMode = workMode;

                        using (DatabaseManager dbManager =
new DatabaseManager())
                        {
                                typeOfUsersNames        =
dbManager.GetNameOfUserType();
                                foreach                (var
typeOfUsersName in typeOfUsersNames)
                                {
        typeOfUsersComboBox.Items.Add(typeOfUsersName.Value);
                                }
                        }
```

```
                        if   (workWithUserMode   ==   "Edit
user")
                        {
                                currentUserId = userId;

                                using     (DatabaseManager
dbManager = new DatabaseManager())
                                {
                                        User  currentUser
= dbManager.GetUser(userId);

        loginTextBox.Text = currentUser.Login;
        passwordTextBox.Text = currentUser.Password;
        typeOfUsersComboBox.Text                        =
typeOfUsersNames.Values.ElementAt(currentUser.FKTypeOfUser    -
1);
        lastNameTextBox.Text = currentUser.LastName;
        firstNameTextBox.Text = currentUser.FirstName;
        surnameTextBox.Text = currentUser.SurName;
        phopeNamberTextBox.Text = currentUser.PhoneNumber;
        activeCheckBox.Checked = currentUser.IsActive;
                                }

                addEditButton.Text = "Відредагувати";
                titleLabel.Text       =       "Відредагувати
користувача";
                        }
                }

                private                                 void
addEditUserForm_MouseDown(object sender, MouseEventArgs e)
                {
                        dragging = true;
                        dragCursorPoint = Cursor.Position;
                        dragFormPoint = this.Location;
                }

                private                                 void
addEditUserForm_MouseMove(object sender, MouseEventArgs e)
                {
                        if (dragging)
                        {
                                Point    newPosition    =
Point.Subtract(Cursor.Position, new Size(dragCursorPoint));
                                this.Location           =
Point.Add(dragFormPoint, new Size(newPosition));
                        }
                }

                private  void  addEditUserForm_MouseUp(object
sender, MouseEventArgs e)
                {
                        dragging = false;
                }

                private    void    cancelButton_Click(object
sender, EventArgs e)
                {
                        this.Close();
                }

                private    void    addEditButton_Click(object
sender, EventArgs e)
                {
                        User user = new User();
                        user.Login = loginTextBox.Text;
                        user.Password                   =
passwordTextBox.Text;
                        user.FKTypeOfUser               =
typeOfUsersNames.FirstOrDefault(x      =>      x.Value      ==
typeOfUsersComboBox.Text).Key;
                        user.LastName                   =
lastNameTextBox.Text;
                        user.FirstName                  =
firstNameTextBox.Text;
                        user.SurName = surnameTextBox.Text;
                        user.CreationDateTime           =
DateTime.Now;
                        user.PhoneNumber                =
phopeNamberTextBox.Text;
                        user.IsActive                   =
activeCheckBox.Checked;

                        using (DatabaseManager dbManager =
new DatabaseManager())
                        {
                                if   (workWithUserMode   ==
"Add user")
                                {

        dbManager.AddUser(user);
                                }
                                else
                                {

        dbManager.EditUser(currentUserId, user);
                                }
                        }

                        this.Close();
                }
        }
}
```

Лістинг AdministratorWindow.cs:

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Windows.Forms;

namespace DRM_AIS_ComputerAdaptiveTesting
{
        public partial class AdministratorWindow : Form
        {
                private bool dragging = false;
                private Point dragCursorPoint;
                private Point dragFormPoint;

                public            AdministratorWindow(string
userLastName, string userFirstName, string userSurName)
                {
                        InitializeComponent();
                        lastNameTextBox.Text              =
userLastName;
                        firstNameTextBox.Text             =
userFirstName + " " + userSurName;
                }

                private                                 void
administratorForm_MouseDown(object sender, MouseEventArgs e)
                {
                        dragging = true;
                        dragCursorPoint = Cursor.Position;
                        dragFormPoint = this.Location;
                }

                private                                 void
administratorForm_MouseMove(object sender, MouseEventArgs e)
                {
                        if (dragging)
                        {
                                Point    newPosition    =
Point.Subtract(Cursor.Position, new Size(dragCursorPoint));
                                this.Location           =
Point.Add(dragFormPoint, new Size(newPosition));
                        }
                }

                private                                 void
administratorForm_MouseUp(object sender, MouseEventArgs e)
                {
                        dragging = false;
                }

                private   void   minimizeButton_Click(object
sender, EventArgs e)
                {
                        this.WindowState                  =
FormWindowState.Minimized;
                }

                private    void    closeButton_Click(object
sender, EventArgs e)
                {
                        this.Close();
                }

                private void RefreshDataGridView()
                {
                        DatabaseManager  databaseManager  =
new DatabaseManager();
                        SqlDataAdapter sqlDataAdapter = new
SqlDataAdapter("Select      *      FROM      [VAO      Master
Work].[dbo].[Користувачі]", databaseManager.connection);
                        DataSet data = new DataSet();
                        sqlDataAdapter.Fill(data,
"Користувачі");
                        userTableDataGridView.DataSource  =
data;
                        userTableDataGridView.DataMember  =
"Користувачі";
                }

                private void AdministratorWindow_Load(object
sender, EventArgs e)
                {
                        RefreshDataGridView();
                }

                private                                 void
blockUserButtonPictureBox_Click(object sender, EventArgs e)
                {
                        int           rowindex            =
userTableDataGridView.CurrentCell.RowIndex;
                        int           userId              =
Convert.ToInt32(userTableDataGridView.Rows[rowindex].Cells[0].
Value);

                        var         isUserActive          =
Convert.ToInt32(userTableDataGridView.Rows[rowindex].Cells[use
rTableDataGridView.ColumnCount - 1].Value);

                        using (DatabaseManager dbManager =
new DatabaseManager())
                        {

        dbManager.BlockUser(userId,  isUserActive  ==  1?  0  :
1);
                        }

                        RefreshDataGridView();
                }
```

```
                private                                 void
addUserButtonPictureBox_Click(object sender, EventArgs e)
                {
                        AddEditUserWindow addEditUserWindow
= new AddEditUserWindow("Add user", 0);
                        addEditUserWindow.ShowDialog();
                        RefreshDataGridView();
                }

                private                                 void
editUserButtonPictureBox_Click(object sender, EventArgs e)
                {
                        AddEditUserWindow addEditUserWindow
=          new          AddEditUserWindow("Edit          user",
Convert.ToInt32(userTableDataGridView.Rows[userTableDataGridVi
ew.CurrentCell.RowIndex].Cells[0].Value));
                        addEditUserWindow.ShowDialog();
                        RefreshDataGridView();
                }

                private                                 void
dataGridView1_CellFormatting(object                     sender,
DataGridViewCellFormattingEventArgs e)
                {
                        if (e.ColumnIndex == 2 && e.Value
!= null)
                        {
                                e.Value = new String('*',
e.Value.ToString().Length);
                        }
                }
        }
}
```

Лістинг DatabaseManager.cs:

```
using DRM_AIS_ComputerAdaptiveTesting.Entities;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Globalization;

namespace DRM_AIS_ComputerAdaptiveTesting
{
        public class DatabaseManager : IDisposable
        {
                public readonly SqlConnection connection;

                public DatabaseManager()
                {
                        string  connectionString  =  @"Data
Source=HBILOUS;Initial          Catalog=VAO          Master
Work;Trusted_Connection=true";
                        this.connection        =        new
SqlConnection(connectionString);
                        this.connection.Open();
                }

                public List<User> GetUsers()
                {
                        List<User>     users     =     new
List<User>();
                        string  sqlExpression  =  "Select  *
FROM [VAO Master Work].[dbo].[Користувачі]";

                        SqlCommand     command     =     new
SqlCommand(sqlExpression, connection);
                        SqlDataReader       reader       =
command.ExecuteReader();

                        if (reader.HasRows)
                        {
                                while (reader.Read())
                                {
                                        users.Add(new
User {
                                                ID     =
Convert.ToInt32(reader["ID"]),
                                                Login =
reader["Логін"].ToString(),

        Password = reader["Пароль"].ToString(),

        FKTypeOfUser    =    Convert.ToInt32(reader["ID    Типу
користувача"]),

        LastName = reader["Прізвище"].ToString(),

        FirstName = reader["Імя"].ToString(),
                                                SurName
= reader["По батькові"].ToString(),

        CreationDateTime = Convert.ToDateTime(reader["Дата та
час створення"]),

        PhoneNumber = reader["Телефон"].ToString(),

        IsActive = Convert.ToBoolean(reader["Активність"])
                                        });
                                }

                                reader.Close();
                        }
                        return users;
                }

                public User GetUser(int userId)
                {
                        User user = new User();
                        string       sqlExpression       =
string.Format($"Select      *      FROM      [VAO      Master
Work].[dbo].[Користувачі] Where ID = {userId}");

                        SqlCommand     command     =     new
SqlCommand(sqlExpression, connection);
                        SqlDataReader       reader       =
command.ExecuteReader();

                        if (reader.HasRows)
                        {
                                while (reader.Read())
                                {
                                        user.ID        =
Convert.ToInt32(reader["ID"]);
                                        user.Login     =
reader["Логін"].ToString();
                                        user.Password  =
reader["Пароль"].ToString();

        user.FKTypeOfUser  =  Convert.ToInt32(reader["ID  Типу
користувача"]);
                                        user.LastName  =
reader["Прізвище"].ToString();
                                        user.FirstName =
reader["Імя"].ToString();
                                        user.SurName   =
reader["По батькові"].ToString();

        user.CreationDateTime                               =
Convert.ToDateTime(reader["Дата та час створення"]);
                                        user.PhoneNumber
= reader["Телефон"].ToString();
                                        user.IsActive  =
Convert.ToBoolean(reader["Активність"]);
                                }

                                reader.Close();
                        }
                        return user;
                }

                public     string     GetNameOfUserType(int
fkTypeOfUsert)
                {
                        string       typeOfUserName       =
string.Empty;
                        string       sqlExpression       =
string.Format($"Select  *  From  [VAO  Master  Work].[dbo].[Типи
користувачів] Where ID = {fkTypeOfUsert}");

                        SqlCommand     command     =     new
SqlCommand(sqlExpression, connection);
                        SqlDataReader       reader       =
command.ExecuteReader();

                        if (reader.HasRows)
                        {
                                while (reader.Read())
                                {
                                        typeOfUserName  =
reader["Назва"].ToString();
                                }

                                reader.Close();
                        }
                        return typeOfUserName;
                }

                public        Dictionary<int,        string>
GetNameOfUserType()
                {
                        Dictionary<int, string> typeOfUsers
= new Dictionary<int, string>();
                        string       sqlExpression       =
string.Format($"Select  *  From  [VAO  Master  Work].[dbo].[Типи
користувачів]");

                        SqlCommand     command     =     new
SqlCommand(sqlExpression, connection);
                        SqlDataReader       reader       =
command.ExecuteReader();

                        if (reader.HasRows)
                        {
                                while (reader.Read())
                                {

        typeOfUsers.Add(Convert.ToInt32(reader["ID"]),
reader["Назва"].ToString());
                                }

                                reader.Close();
                        }
                        return typeOfUsers;
                }

                public void AddUser(User userData)
                {
                        string       sqlExpression       =
string.Format($"INSERT        INTO        [VAO        Master
Work].[dbo].[Користувачі] " +

        $"(Логін, Пароль, [ID Типу користувача], Прізвище,
```

```
Імя, [По батькові], [Дата та час створення], Телефон, Активність) " +

        $"VALUES ('{userData.Login}', '{userData.Password}', {userData.FKTypeOfUser}, N'{userData.LastName}', N'{userData.FirstName}', " +

        $"N'{userData.SurName}', '{userData.CreationDateTime}', '{userData.PhoneNumber}', {(userData.IsActive ? 1 : 0)})");
                        SqlCommand command = new SqlCommand(sqlExpression, connection);
                        command.ExecuteNonQuery();
                }

                public void EditUser(int userId, User userData)
                {
                        string sqlExpression = string.Format($"UPDATE [VAO Master Work].[dbo].[Користувачі] SET Логін = '{userData.Login}', Пароль = '{userData.Password}', " +
                                                                $" [ID Типу користувача] = {userData.FKTypeOfUser}, Прізвище = N'{userData.LastName}', Імя = N'{userData.FirstName}', " +
                                                                $"[По батькові] = N'{userData.SurName}', Телефон = '{userData.PhoneNumber}', Активність = {(userData.IsActive ? 1 : 0)} WHERE ID = {userId}");
                        SqlCommand command = new SqlCommand(sqlExpression, connection);
                        command.ExecuteNonQuery();
                }

                public void BlockUser(int userId, int userActivity)
                {
                        string sqlExpression = string.Format($"UPDATE [VAO Master Work].[dbo].[Користувачі] SET Активність = {userActivity} WHERE ID = {userId}");
                        SqlCommand command = new SqlCommand(sqlExpression, connection);
                        command.ExecuteNonQuery();
                }

                public void ApproveAttempt(int attemptId, int isApproved)
                {
                        string sqlExpression = string.Format($"UPDATE [VAO Master Work].[dbo].[Спроби] SET Затверджено = {isApproved} WHERE ID = {attemptId}");
                        SqlCommand command = new SqlCommand(sqlExpression, connection);
                        command.ExecuteNonQuery();
                }

                public Dictionary<int, string> GetNameSamplesOfTestTasks()
                {
                        Dictionary<int, string> samplesOfTestTasks = new Dictionary<int, string>();
                        string sqlExpression = string.Format($"Select [ID], [Назва] From [VAO Master Work].[dbo].[Вибірки тестових завдань]");

                        SqlCommand command = new SqlCommand(sqlExpression, connection);
                        SqlDataReader reader = command.ExecuteReader();

                        if (reader.HasRows)
                        {
                                while (reader.Read())
                                {
        samplesOfTestTasks.Add(Convert.ToInt32(reader["ID"]), reader["Назва"].ToString());
                                }

                                reader.Close();
                        }
                        return samplesOfTestTasks;
                }

                public Dictionary<int, string> GetNameTypeOfTesting()
                {
                        Dictionary<int, string> typeOfTesting = new Dictionary<int, string>();
                        string sqlExpression = string.Format($"Select * From [VAO Master Work].[dbo].[Типи тестування]");

                        SqlCommand command = new SqlCommand(sqlExpression, connection);
                        SqlDataReader reader = command.ExecuteReader();

                        if (reader.HasRows)
                        {
                                while (reader.Read())
                                {
        typeOfTesting.Add(Convert.ToInt32(reader["ID"]), reader["Назва"].ToString());
                                }

                                reader.Close();
                        }
                        return typeOfTesting;
                }

                public Dictionary<int, string> GetStudentList()
                {
                        Dictionary<int, string> studentsList = new Dictionary<int, string>();
                        string sqlExpression = string.Format($"Select [ID], [Прізвище], [Імя], [По батькові] From [VAO Master Work].[dbo].[Користувачі] Where [ID Типу користувача] = '3'");

                        SqlCommand command = new SqlCommand(sqlExpression, connection);
                        SqlDataReader reader = command.ExecuteReader();

                        if (reader.HasRows)
                        {
                                while (reader.Read())
                                {
        studentsList.Add(Convert.ToInt32(reader["ID"]), (reader["Прізвище"].ToString() + ' ' + reader["Імя"].ToString() + ' ' + reader["По батькові"].ToString()));
                                }

                                reader.Close();
                        }
                        return studentsList;
                }

                public Attempt GetAttempt(int attemptId)
                {
                        Attempt attempt = new Attempt();
                        string sqlExpression = string.Format($"Select * FROM [VAO Master Work].[dbo].[Спроби] Where ID = {attemptId}");

                        SqlCommand command = new SqlCommand(sqlExpression, connection);
                        SqlDataReader reader = command.ExecuteReader();

                        if (reader.HasRows)
                        {
                                while (reader.Read())
                                {
                                        attempt.ID = Convert.ToInt32(reader["ID"]);

        attempt.FKSamplesOfTestTasks = Convert.ToInt32(reader["ID Вибірки тестових завдань"]);

        attempt.FKTypeOfTesting = Convert.ToInt32(reader["ID Типу тестування"]);

        attempt.FKStudent = Convert.ToInt32(reader["ID Студента"]);
                                        attempt.Message = reader["Повідомлення"].ToString();

        attempt.IsGivenStudentToPerform = Convert.ToBoolean(reader["Дано студенту для виконання"]);
                                        attempt.MaxTime = DateTime.ParseExact(reader["Максимальний час"].ToString(), "HH:mm:ss", CultureInfo.InvariantCulture);
                                }

                                reader.Close();
                        }
                        return attempt;
                }

                public void AddAttempt(Attempt attemptData)
                {
                        string sqlExpression = string.Format($"INSERT INTO [VAO Master Work].[dbo].[Спроби] " +

        $"([Дата та час створення заявки], [ID Вибірки тестових завдань], [ID Типу тестування], [ID Викладача], [ID Студента], [Дано студенту для виконання], [Повідомлення], [Максимальний час]) " +

        $"VALUES ('{attemptData.CreationDateTime}', '{attemptData.FKSamplesOfTestTasks}', '{attemptData.FKTypeOfTesting}', {attemptData.FKTeacher}," +
                                                                $" {attemptData.FKStudent}, {attemptData.IsGivenStudentToPerform}, N'{attemptData.Message}', '{ "00:" + attemptData.MaxTime.Minute.ToString() + ":00" }')");
```

```
                        SqlCommand     command     =     new
SqlCommand(sqlExpression, connection);
                        command.ExecuteNonQuery();
                }

                public   void   EditAttempt(int   attemptId,
Attempt attemptData)
                {
                        string        sqlExpression        =
string.Format($"UPDATE  [VAO  Master  Work].[dbo].[Користувачі]
SET      [ID       Вибірки      тестових      завдань]      =
'{attemptData.FKSamplesOfTestTasks}', " +
                                                             $"[ID
Типу   тестування]   =   '{attemptData.FKTypeOfTesting}',   [ID
Студента]  =  '{attemptData.FKStudent}',  [Дано  студенту  для
виконання] = '{attemptData.IsGivenStudentToPerform}', " +


        $"[Повідомлення]      =      '{attemptData.Message}',
[Максимальний       час]       =       '{       "00:"       +
attemptData.MaxTime.Minute.ToString()  +  ":00"  }'  WHERE  ID  =
{attemptId}");
                        SqlCommand     command     =     new
SqlCommand(sqlExpression, connection);
                        command.ExecuteNonQuery();
                }

                public     void     GiveToPerformAttempt(int
attemptId, int isGiveToPerformChecked)
                {
                        string        sqlExpression        =
string.Format($"UPDATE  [VAO  Master  Work].[dbo].[Спроби]  SET
[Дано студенту для виконання] = {isGiveToPerformChecked} WHERE
ID = {attemptId}");
                        SqlCommand     command     =     new
SqlCommand(sqlExpression, connection);
                        command.ExecuteNonQuery();
                }

                public Dictionary<int, string> GetIEM()
                {
                        Dictionary<int, string> iem = new
Dictionary<int, string>();
                        string        sqlExpression        =
string.Format($"Select * From [VAO Master Work].[dbo].[IHM]");

                        SqlCommand     command     =     new
SqlCommand(sqlExpression, connection);
                        SqlDataReader        reader        =
command.ExecuteReader();

                        if (reader.HasRows)
                        {
                                while (reader.Read())
                                {

         iem.Add(Convert.ToInt32(reader["ID"]),
reader["Назва"].ToString());
                                }

                                reader.Close();
                        }
                        return iem;
                }

                public List<Heading> GetHeadings(int iemId)
                {
                        List<Heading>    heading    =    new
List<Heading>();
                        string        sqlExpression        =
string.Format($"Select      *      From      [VAO      Master
Work].[dbo].[Заголовки]  WHERE  [ID  IHM]  =  {iemId}  AND  [ID
Заголовку] IS NOT NULL");

                        SqlCommand     command     =     new
SqlCommand(sqlExpression, connection);
                        SqlDataReader        reader        =
command.ExecuteReader();

                        if (reader.HasRows)
                        {
                                while (reader.Read())
                                {
                                        heading.Add(new
Heading()
                                        {
                                                ID     =
Convert.ToInt32(reader["ID"]),
                                                Name   =
reader["Назва"].ToString(),
                                                Level =
Convert.ToInt32(reader["Рівень"]),

         FKHeading = Convert.ToInt32(reader["ID Заголовку"]),
                                                FKIEM =
Convert.ToInt32(reader["ID IHM"])
                                        });
                                }

                                reader.Close();
                        }
                        return heading;
                }

                public SelectionOfTestTasks GetSelection(int
selectionId)
                {
                        SelectionOfTestTasks
selectionOfTestTasks = new SelectionOfTestTasks();
                        string        sqlExpression        =
string.Format($"Select * FROM [VAO Master Work].[dbo].[Вибірки
тестових завдань] Where ID = {selectionId}");

                        SqlCommand     command     =     new
SqlCommand(sqlExpression, connection);
                        SqlDataReader        reader        =
command.ExecuteReader();

                        if (reader.HasRows)
                        {
                                while (reader.Read())
                                {

         selectionOfTestTasks.ID                                =
Convert.ToInt32(reader["ID"]);

         selectionOfTestTasks.FKHeading                         =
Convert.ToInt32(reader["ID Загловку"]);

         selectionOfTestTasks.Name                              =
reader["Назва"].ToString();

         selectionOfTestTasks.FKTeacher                         =
Convert.ToInt32(reader["ID Викладача"]);
                                }

                                reader.Close();
                        }
                        return selectionOfTestTasks;
                }

                public string GetHeading(int fkHeading)
                {
                        string headingName = string.Empty;
                        string        sqlExpression        =
string.Format($"Select      *      From      [VAO      Master
Work].[dbo].[Заголовки] Where ID = {fkHeading}");

                        SqlCommand     command     =     new
SqlCommand(sqlExpression, connection);
                        SqlDataReader        reader        =
command.ExecuteReader();

                        if (reader.HasRows)
                        {
                                while (reader.Read())
                                {
                                        headingName     =
reader["Назва"].ToString();
                                }

                                reader.Close();
                        }
                        return headingName;
                }

                public string GetIEM(int headingId)
                {
                        string iemName = string.Empty;
                        string        sqlExpression        =
string.Format($"Select    iem.Назва    From    [VAO    Master
Work].[dbo].[Заголовки]    heading    join    [VAO    Master
Work].[dbo].[IHM]  iem  on  heading.[ID  IHM]  =  iem.id  WHERE
heading.[ID] = {headingId}");

                        SqlCommand     command     =     new
SqlCommand(sqlExpression, connection);
                        SqlDataReader        reader        =
command.ExecuteReader();

                        if (reader.HasRows)
                        {
                                while (reader.Read())
                                {
                                        iemName         =
reader["Назва"].ToString();
                                }

                                reader.Close();
                        }
                        return iemName;
                }

                public Dictionary<int, string> GetKeyTerm()
                {
                        Dictionary<int, string> keyTerms =
new Dictionary<int, string>();
                        string        sqlExpression        =
string.Format($"SELECT  DISTINCT  term.[ID]  ID,  term.[Назва]
Назва FROM [VAO Master Work].[dbo].[Ключові терміни] keyTerm
join  [VAO  Master  Work].[dbo].[Терміни]  term  on  keyTerm.[ID
Терміну] = term.id");

                        SqlCommand     command     =     new
SqlCommand(sqlExpression, connection);
                        SqlDataReader        reader        =
command.ExecuteReader();

                        if (reader.HasRows)
                        {
                                while (reader.Read())
                                {
```

```
        keyTerms.Add(Convert.ToInt32(reader["ID"]),
reader["Назва"].ToString());
                                }

                                reader.Close();
                        }
                        return keyTerms;
                }

                public        Dictionary<int,        string>
GetTestTaskTypes()
                {
                        Dictionary<int,             string>
testTasksTypes = new Dictionary<int, string>();
                        string        sqlExpression        =
string.Format($"SELECT   [ID],   [Назва]   FROM   [VAO   Master
Work].[dbo].[Типи завдань]");

                        SqlCommand     command     =     new
SqlCommand(sqlExpression, connection);
                        SqlDataReader        reader        =
command.ExecuteReader();

                        if (reader.HasRows)
                        {
                                while (reader.Read())
                                {

        testTasksTypes.Add(Convert.ToInt32(reader["ID"]),
reader["Назва"].ToString());
                                }

                                reader.Close();
                        }
                        return testTasksTypes;
                }

                public TestTask GetTestTask(int testTasksId)
                {
                        TestTask     testTasks     =     new
TestTask();
                        string        sqlExpression        =
string.Format($"SELECT     testTask.[ID],     testTask.[Текст],
term.Назва    [Ключовий    термін],    testTask.[ID    Речення],
testTask.[ID Типу] " +


$"FROM [VAO Master Work].[dbo].[Тестові завдання] testTask " +


$"join  [VAO  Master  Work].[dbo].[Ключові  терміни]  keyTerm  on
testTask.[ID Ключового терміну] = keyTerm.id " +


$"join [VAO Master Work].[dbo].[Терміни] term on  keyTerm.[ID
Терміну] = term.ID " +


$"WHERE testTask.[ID] = {testTasksId}");

                        SqlCommand     command     =     new
SqlCommand(sqlExpression, connection);
                        SqlDataReader        reader        =
command.ExecuteReader();

                        if (reader.HasRows)
                        {
                                while (reader.Read())
                                {
                                        testTasks.ID    =
Convert.ToInt32(reader["ID"]);

        testTasks.FKKeyTerm         =         reader["Ключовий
термін"].ToString();
                                        testTasks.Text  =
reader["Текст"].ToString();

        testTasks.FKSentence   =   (reader["ID   Речення"]   ==
DBNull.Value) ? Convert.ToInt32(reader["ID Речення"]) : 0;

        testTasks.FKTaskType   =   Convert.ToInt32(reader["ID
Типу"]);
                                }

                                reader.Close();
                        }
                        return testTasks;
                }

                public      string      GetTestTaskType(int
testTaskId)
                {
                        string testTaskName = string.Empty;
                        string        sqlExpression        =
string.Format($"Select  *  From  [VAO  Master  Work].[dbo].[Типи
завдань] Where ID = {testTaskId}");

                        SqlCommand     command     =     new
SqlCommand(sqlExpression, connection);
                        SqlDataReader        reader        =
command.ExecuteReader();

                        if (reader.HasRows)
                        {
                                while (reader.Read())
                                {
                                        testTaskName    =
reader["Назва"].ToString();
                                }

                                reader.Close();
                        }
                        return testTaskName;
                }

                public                                  string
GetAnswersForOpenOrLogicalTasks(int testTaskId)
                {
                        string answerName = string.Empty;
                        string        sqlExpression        =
string.Format($"Select      *      From      [VAO      Master
Work].[dbo].[Відповіді]   Where   [ID   Тестового   завдання]   =
{testTaskId} AND [Коректність] = 1");

                        SqlCommand     command     =     new
SqlCommand(sqlExpression, connection);
                        SqlDataReader        reader        =
command.ExecuteReader();

                        if (reader.HasRows)
                        {
                                while (reader.Read())
                                {
                                        answerName      =
reader["Текст"].ToString();
                                }

                                reader.Close();
                        }
                        return answerName;
                }

                public                              List<Answer>
GetAnswersForChooseTasks(int testTaskId)
                {
                        List<Answer>    answers    =    new
List<Answer>();
                        string        sqlExpression        =
string.Format($"Select      *      From      [VAO      Master
Work].[dbo].[Відповіді]   Where   [ID   Тестового   завдання]   =
{testTaskId}");

                        SqlCommand     command     =     new
SqlCommand(sqlExpression, connection);
                        SqlDataReader        reader        =
command.ExecuteReader();

                        if (reader.HasRows)
                        {
                                while (reader.Read())
                                {
                                        answers.Add(new
Answer()
                                        {
                                                ID     =
Convert.ToInt32(reader["ID"]),
                                                Text   =
reader["Текст"].ToString(),

        FKSentence = (reader["ID Речення"] == DBNull.Value)?
Convert.ToInt32(reader["ID Речення"]) : 0,

        FKTestTask   =   Convert.ToInt32(reader["ID   Тестового
завдання"]),

        IsCorrectness                                   =
Convert.ToBoolean(reader["Коректність"])
                                        });
                                }

                                reader.Close();
                        }
                        return answers;
                }

                public string GetSentence(int sentenceId)
                {
                        string senteceText = string.Empty;
                        string        sqlExpression        =
string.Format($"Select * From [VAO Master Work].[dbo].[Текст]
Where ID = {sentenceId}");

                        SqlCommand     command     =     new
SqlCommand(sqlExpression, connection);
                        SqlDataReader        reader        =
command.ExecuteReader();

                        if (reader.HasRows)
                        {
                                while (reader.Read())
                                {
                                        senteceText     =
reader["Текст"].ToString();
                                }

                                reader.Close();
                        }
                        return senteceText;
                }

                public void Dispose()
                {
                        this.connection.Close();
```

```
                }
        }
}
```

Лістинг LoginWindow.cs:

```
using DRM_AIS_ComputerAdaptiveTesting.Entities;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace DRM_AIS_ComputerAdaptiveTesting
{
        public partial class LoginWindow : Form
        {
                private bool dragging = false;
                private Point dragCursorPoint;
                private Point dragFormPoint;

                public LoginWindow()
                {
                        InitializeComponent();
                }

                private   void   loginForm_MouseDown(object
sender, MouseEventArgs e)
                {
                        dragging = true;
                        dragCursorPoint = Cursor.Position;
                        dragFormPoint = this.Location;
                }

                private   void   loginForm_MouseMove(object
sender, MouseEventArgs e)
                {
                        if (dragging)
                        {
                                Point    newPosition    =
Point.Subtract(Cursor.Position, new Size(dragCursorPoint));
                                this.Location           =
Point.Add(dragFormPoint, new Size(newPosition));
                        }
                }

                private    void    loginForm_MouseUp(object
sender, MouseEventArgs e)
                {
                        dragging = false;
                }

                private   void   minimizeButton_Click(object
sender, EventArgs e)
                {
                        this.WindowState                =
FormWindowState.Minimized;
                }

                private    void    closeButton_Click(object
sender, EventArgs e)
                {
                        this.Close();
                }

                ErrorProvider    errorProvider    =    new
ErrorProvider();
                private  void  ValidateFormTextBoxes(TextBox
fieldName, string errorMessage)
                {
                        if
(string.IsNullOrEmpty(fieldName.Text))
                        {

        errorProvider.SetError(fieldName,   "Введіть,   будь
ласка, " + errorMessage);
                        }
                        else
                        {

        errorProvider.SetError(fieldName, null);
                        }
                }

                private void loginTextBox_TextChanged(object
sender, EventArgs e)
                {
                        ValidateFormTextBoxes(loginTextBox,
"логін");
                }

                private                                 void
passwordTextBox_TextChanged(object sender, EventArgs e)
                {

        ValidateFormTextBoxes(passwordTextBox, "пароль");
                }

                private void LoginToSystem ()
                {
                        Control[]    formfields    =    {
loginTextBox, passwordTextBox };
                        bool isAllFiledFilled = true;
                        foreach    (var    formfield    in
formfields)
                        {
                                //startTestButtonEnabled
=                  startTestButtonEnabled                &&
string.IsNullOrEmpty(errorProvider.GetError(formfield));
                                if
(!string.IsNullOrEmpty(errorProvider.GetError(formfield))   ||
string.IsNullOrEmpty(formfield.Text))
                                {

        MessageBox.Show("Облікові дані введено не повністю.
Будь    ласка,    заповніть    усі    поля!",    "Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                                        isAllFiledFilled
= false;
                                        break;
                                }
                        }

                        if (isAllFiledFilled)
                        {
                                List<User> users;
                                using    (DatabaseManager
dbManager = new DatabaseManager())
                                {
                                        users          =
dbManager.GetUsers();
                                }

                                if(users.Select(x       =>
x.Login).ToList().Contains(loginTextBox.Text))
                                {
                                        var  currentUser
=         users.Where(x         =>         x.Login         ==
loginTextBox.Text).FirstOrDefault();

                                        if
(currentUser.Password      ==      passwordTextBox.Text      &&
currentUser.IsActive)
                                        {

        this.Hide();
                                        string typeOfCurrentUser;
                                        using (DatabaseManager dbManager =
new DatabaseManager())
                                                {

        typeOfCurrentUser                                =
dbManager.GetNameOfUserType(currentUser.FKTypeOfUser);
                                                }
                                                if
(typeOfCurrentUser == "Адміністратор")
                                                {

        AdministratorWindow    administratorWindow    =    new
AdministratorWindow(currentUser.LastName,
currentUser.FirstName, currentUser.SurName);

        administratorWindow.Closed    +=    (s,    args)    =>
this.Close();

        administratorWindow.Show();
                                                }
                                                else if
(typeOfCurrentUser == "Викладач")
                                                {

        TeacherWindow        teacherWindow        =        new
TeacherWindow(currentUser.ID,           currentUser.LastName,
currentUser.FirstName, currentUser.SurName);

        teacherWindow.Closed += (s, args) => this.Close();

        teacherWindow.Show();
                                                }
                                                else if
(typeOfCurrentUser == "Студент")
                                                {

        StudentWindow        studentWindow        =        new
StudentWindow(currentUser.ID,           currentUser.LastName,
currentUser.FirstName, currentUser.SurName);

        studentWindow.Closed += (s, args) => this.Close();

        studentWindow.Show();
                                                }
                                        }
                                        else           if
(currentUser.Password != passwordTextBox.Text)
                                        {

        MessageBox.Show("Введено не правильний пароль. Будь
ласка,      перевірте      введені      дані!",      "Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                                        }
                                        else           if
(!currentUser.IsActive)
                                        {

        MessageBox.Show("Ваш обліковий запис заблокований.
Будь   ласка,   зверніться   до   адміністратора!",   "Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                                        }
                                }
                                else
                                {

        MessageBox.Show("Введено не правильний логін. Будь
ласка,      перевірте      введені      дані!",      "Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
                }
            }
        }

        private void entranceButton_MouseClick(object sender, MouseEventArgs e)
        {
            LoginToSystem();
        }

        private void passwordTextBox_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == (char)Keys.Enter)
            {
                LoginToSystem();
            }
        }

        private void entranceButton_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (e.KeyChar == (char)Keys.Enter)
            {
                LoginToSystem();
            }
        }
    }
}
```

Лістинг Program.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DRM_AIS_ComputerAdaptiveTesting
{
    static class Program
    {
        /// <summary>
        ///  The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();

            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new LoginWindow());
        }
    }
}
```

Лістинг StudentWindow.cs:

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Windows.Forms;

namespace DRM_AIS_ComputerAdaptiveTesting
{
    public partial class StudentWindow : Form
    {
        private bool dragging = false;
        private Point dragCursorPoint;
        private Point dragFormPoint;
        int currentUserId;

        public StudentWindow(int userId, string userLastName, string userFirstName, string userSurName)
        {
            InitializeComponent();

            lastNameTextBox.Text = userLastName;
            firstNameTextBox.Text = userFirstName + " " + userSurName;
            currentUserId = userId;
        }

        private void studentWindow_MouseDown(object sender, MouseEventArgs e)
        {
            dragging = true;
            dragCursorPoint = Cursor.Position;
            dragFormPoint = this.Location;
        }

        private void studentWindow_MouseMove(object sender, MouseEventArgs e)
        {
            if (dragging)
            {
                Point newPosition = Point.Subtract(Cursor.Position, new Size(dragCursorPoint));
                this.Location = Point.Add(dragFormPoint, new Size(newPosition));
            }
        }

        private void studentWindow_MouseUp(object sender, MouseEventArgs e)
        {
            dragging = false;
        }

        private void minimizeButton_Click(object sender, EventArgs e)
        {
            this.WindowState = FormWindowState.Minimized;
        }

        private void closeButton_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void StudentWindow_Load(object sender, EventArgs e)
        {
            DatabaseManager databaseManager = new DatabaseManager();
            SqlDataAdapter newTestSqlDataAdapter = new SqlDataAdapter(string.Format($"SELECT attempt.[ID], attempt.[Дата та час створення заявки], testTasksSample.Назва [Вибірка тестових завдань], typeTask.Назва [Тип тестування], " +
                $"teacher.Прізвище + ' ' + teacher.Імя + ' ' + teacher.[По батькові] Викладач, " +
                $"student.Прізвище + ' ' + student.Імя + ' ' + student.[По батькові] Студент, " +
                $"attempt.[Повідомлення], attempt.[Максимальний час] " +
                $"FROM [VAO Master Work].[dbo].[Спроби] attempt " +
                $"join[VAO Master Work].[dbo].[Вибірки тестових завдань] testTasksSample " +
                $"on attempt.[ID Вибірки тестових завдань] = testTasksSample.id " +
                $"join[VAO Master Work].[dbo].[Типи тестування] typeTask " +
                $"on attempt.[ID Типу тестування] = typeTask.id " +
                $"join[VAO Master Work].[dbo].[Користувачі] teacher " +
                $"on attempt.[ID Викладача] = teacher.id " +
                $"join[VAO Master Work].[dbo].[Користувачі] student " +
                $"on attempt.[ID Студента] = student.id " +
                "WHERE ((attempt.Затверджено = 0 OR attempt.Затверджено IS NULL) AND attempt.Оцінка IS NULL AND attempt.Результат IS NULL) " +
                $"AND student.id = {currentUserId}"), databaseManager.connection);
            DataSet newTestData = new DataSet();

            newTestSqlDataAdapter.Fill(newTestData, "Спроби");
            newTestDataGridView.DataSource = newTestData;
            newTestDataGridView.DataMember = "Спроби";

            SqlDataAdapter notApprovedTestSqlDataAdapter = new SqlDataAdapter(string.Format($"SELECT attempt.[ID], attempt.[Затверджено], attempt.[Дата та час створення заявки], attempt.[Дата та час початку], attempt.[Дата та час завершення], " +
                $"testTasksSample.Назва [Вибірка тестових завдань], typeTask.Назва [Тип тестування], " +
                $"teacher.Прізвище + ' ' + teacher.Імя + ' ' + teacher.[По батькові] Викладач, " +
                $"student.Прізвище + ' ' + student.Імя + ' ' + student.[По батькові] Студент, " +
                $"attempt.[Повідомлення], attempt.[Оцінка], attempt.[Результат], attempt.[Кількість правильних відповідей], " +
                $"attempt.[Кількість неправильних відповідей], attempt.[Максимальний час] " +
                $"FROM [VAO Master Work].[dbo].[Спроби] attempt " +
                $"join [VAO Master Work].[dbo].[Вибірки тестових завдань] testTasksSample " +
                $"on attempt.[ID Вибірки тестових завдань] = testTasksSample.id " +
                $"join [VAO Master Work].[dbo].[Типи тестування] typeTask " +
                $"on attempt.[ID Типу тестування] = typeTask.id " +
                $"join [VAO Master Work].[dbo].[Користувачі] teacher " +
                $"on attempt.[ID Викладача] = teacher.id " +
                $"join [VAO Master Work].[dbo].[Користувачі] student " +
                $"on attempt.[ID Студента] = student.id " +
                $"WHERE (attempt.Затверджено = 0 OR attempt.Затверджено IS NULL) AND attempt.Оцінка IS NOT NULL AND attempt.Результат IS NOT NULL " +
```

```
                                        $"AND student.id = {currentUserId}"), databaseManager.connection);
                        DataSet notApprovedTestData = new DataSet();

        notApprovedTestSqlDataAdapter.Fill(notApprovedTestData, "Спроби");

        notApprovedTestDataGridView.DataSource = notApprovedTestData;

        notApprovedTestDataGridView.DataMember = "Спроби";

                        SqlDataAdapter approvedTestSqlDataAdapter = new SqlDataAdapter(string.Format($"SELECT attempt.[ID], attempt.[Затверджено], attempt.[Дата та час створення заявки], attempt.[Дата та час початку], attempt.[Дата та час завершення], " +
        $"testTasksSample.Назва [Вибірка тестових завдань], typeTask.Назва [Тип тестування], " +
        $"teacher.Прізвище + ' ' + teacher.Імя + ' ' + teacher.[По батькові] Викладач, " +
        $"student.Прізвище + ' ' + student.Імя + ' ' + student.[По батькові] Студент, " +
        $"attempt.[Повідомлення], attempt.[Оцінка], attempt.[Результат], attempt.[Кількість правильних відповідей], " +
        $"attempt.[Кількість неправильних відповідей], attempt.[Максимальний час] " +
                        $"FROM [VAO Master Work].[dbo].[Спроби] attempt " +
                        $"join [VAO Master Work].[dbo].[Вибірки тестових завдань] testTasksSample " +
                        $"on attempt.[ID Вибірки тестових завдань] = testTasksSample.id " +
                                        $"join [VAO Master Work].[dbo].[Типи тестування] typeTask " +
                                        $"on attempt.[ID Типу тестування] = typeTask.id " +
                                        $"join [VAO Master Work].[dbo].[Користувачі] teacher " +
                                        $"on attempt.[ID Викладача] = teacher.id " +
                                        $"join [VAO Master Work].[dbo].[Користувачі] student " +
                                        $"on attempt.[ID Студента] = student.id " +
                        $"WHERE attempt.Затверджено = 1 AND attempt.Оцінка IS NOT NULL AND attempt.Результат IS NOT NULL " +
                                        $"AND student.id = {currentUserId}"), databaseManager.connection);
                        DataSet approvedTestData = new DataSet();

        approvedTestSqlDataAdapter.Fill(approvedTestData, "Спроби");
                        approvedTestDataGridView.DataSource = approvedTestData;
                        approvedTestDataGridView.DataMember = "Спроби";
                }

                private void testTabControl_SelectedIndexChanged(object sender, EventArgs e)
                {
                        startTestPictureBox.Enabled = testTabControl.SelectedIndex == 0;

                        var ivoryColor = startTestPictureBox.BackColor;
                        startTestPictureBox.BackColor = startTestPictureBox.Enabled ? ivoryColor : Color.FromArgb(222, 222, 222);
                }

                private void startTestPictureBox_Click(object sender, EventArgs e)
                {

                }
        }
}
```

Лістинг TeacherWindow.cs:

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Windows.Forms;

namespace DRM_AIS_ComputerAdaptiveTesting
{
        public partial class TeacherWindow : Form
        {
                private bool dragging = false;
                private Point dragCursorPoint;
                private Point dragFormPoint;
                int currentUserId;
                string currentUserLastName, currentUserFirstName, currentUserSurName;

                public TeacherWindow(int userId, string userLastName, string userFirstName, string userSurName)
                {
                        InitializeComponent();

                        lastNameTextBox.Text = userLastName;
                        firstNameTextBox.Text = userFirstName + " " + userSurName;
                        currentUserId = userId;
                        currentUserLastName = userLastName;
                        currentUserFirstName = userFirstName;
                        currentUserSurName = userSurName;
                }

                private void teacherWindow_MouseDown(object sender, MouseEventArgs e)
                {
                        dragging = true;
                        dragCursorPoint = Cursor.Position;
                        dragFormPoint = this.Location;
                }

                private void teacherWindow_MouseMove(object sender, MouseEventArgs e)
                {
                        if (dragging)
                        {
                                Point newPosition = Point.Subtract(Cursor.Position, new Size(dragCursorPoint));
                                this.Location = Point.Add(dragFormPoint, new Size(newPosition));
                        }
                }

                private void teacherWindow_MouseUp(object sender, MouseEventArgs e)
                {
                        dragging = false;
                }

                private void minimizeButton_Click(object sender, EventArgs e)
                {
                        this.WindowState = FormWindowState.Minimized;
                }

                private void closeButton_Click(object sender, EventArgs e)
                {
                        this.Close();
                }

                private void RefreshDataGridViews()
                {
                        DatabaseManager databaseManager = new DatabaseManager();
                        SqlDataAdapter newTestSqlDataAdapter = new SqlDataAdapter("SELECT attempt.[ID], attempt.[Дата та час створення заявки], testTasksSample.Назва [Вибірка тестових завдань], typeTask.Назва [Тип тестування], " +
                "teacher.Прізвище + ' ' + teacher.Імя + ' ' + teacher.[По батькові] Викладач, " +
                "student.Прізвище + ' ' + student.Імя + ' ' + student.[По батькові] Студент, " +
                "attempt.[Дано студенту для виконання], attempt.[Повідомлення], attempt.[Максимальний час] " +
                                "FROM [VAO Master Work].[dbo].[Спроби] attempt " +
                "join [VAO Master Work].[dbo].[Вибірки тестових завдань] testTasksSample " +
                                "on attempt.[ID Вибірки тестових завдань] = testTasksSample.id " +
                "join [VAO Master Work].[dbo].[Типи тестування] typeTask " +
                                "on attempt.[ID Типу тестування] = typeTask.id " +
                "join [VAO Master Work].[dbo].[Користувачі] teacher " +
                                "on attempt.[ID Викладача] = teacher.id " +
                "join [VAO Master Work].[dbo].[Користувачі] student " +
                                "on attempt.[ID Студента] = student.id " +
                                "WHERE (attempt.Затверджено = 0 OR attempt.Затверджено IS NULL) AND attempt.Оцінка IS NULL AND attempt.Результат IS NULL", databaseManager.connection);
                        DataSet newTestData = new DataSet();

                newTestSqlDataAdapter.Fill(newTestData, "Спроби");
                        newTestDataGridView.DataSource = newTestData;
                        newTestDataGridView.DataMember = "Спроби";

                        SqlDataAdapter notApprovedTestSqlDataAdapter = new SqlDataAdapter("SELECT attempt.[ID], attempt.[Затверджено], attempt.[Дата та час створення заявки], attempt.[Дата та час початку], attempt.[Дата та час завершення], " +
                                "testTasksSample.Назва [Вибірка тестових завдань], typeTask.Назва [Тип тестування], " +
                                "teacher.Прізвище + ' ' + teacher.Імя + ' ' + teacher.[По батькові] Викладач, " +
```

```
                        "student.Прізвище + ' ' + student.Ім'я + ' ' + student.[По батькові] Студент, " +
                        "attempt.[Повідомлення], attempt.[Оцінка], attempt.[Результат], attempt.[Кількість правильних відповідей], " +
                        "attempt.[Кількість неправильних відповідей], attempt.[Максимальний час] " +
                    "FROM [VAO Master Work].[dbo].[Спроби] attempt " +
                        "join [VAO Master Work].[dbo].[Вибірки тестових завдань] testTasksSample " +
                        "on attempt.[ID Вибірки тестових завдань] = testTasksSample.id " +
                        "join [VAO Master Work].[dbo].[Типи тестування] typeTask " +
                        "on attempt.[ID Типу тестування] = typeTask.id " +
                        "join [VAO Master Work].[dbo].[Користувачі] teacher " +
                        "on attempt.[ID Викладача] = teacher.id " +
                        "join [VAO Master Work].[dbo].[Користувачі] student " +
                        "on attempt.[ID Студента] = student.id " +
                    "WHERE (attempt.Затверджено = 0 OR attempt.Затверджено IS NULL) AND attempt.Оцінка IS NOT NULL AND attempt.Результат IS NOT NULL", databaseManager.connection);
                DataSet notApprovedTestData = new DataSet();

        notApprovedTestSqlDataAdapter.Fill(notApprovedTestData, "Спроби");

        notApprovedTestDataGridView.DataSource = notApprovedTestData;

        notApprovedTestDataGridView.DataMember = "Спроби";

                SqlDataAdapter approvedTestSqlDataAdapter = new SqlDataAdapter("SELECT attempt.[ID], attempt.[Затверджено], attempt.[Дата та час створення заявки], attempt.[Дата та час початку], attempt.[Дата та час завершення], " +
                    "testTasksSample.Назва [Вибірка тестових завдань], typeTask.Назва [Тип тестування], " +
                    "teacher.Прізвище + ' ' + teacher.Ім'я + ' ' + teacher.[По батькові] Викладач, " +
                    "student.Прізвище + ' ' + student.Ім'я + ' ' + student.[По батькові] Студент, " +
                    "attempt.[Повідомлення], attempt.[Оцінка], attempt.[Результат], attempt.[Кількість правильних відповідей], " +
                    "attempt.[Кількість неправильних відповідей], attempt.[Затверджено], attempt.[Максимальний час] " +
                "FROM [VAO Master Work].[dbo].[Спроби] attempt " +
                    "join [VAO Master Work].[dbo].[Вибірки тестових завдань] testTasksSample " +
                        "on attempt.[ID Вибірки тестових завдань] = testTasksSample.id " +
                    "join [VAO Master Work].[dbo].[Типи тестування] typeTask " +
                        "on attempt.[ID Типу тестування] = typeTask.id " +
                    "join [VAO Master Work].[dbo].[Користувачі] teacher " +
                        "on attempt.[ID Викладача] = teacher.id " +
                    "join [VAO Master Work].[dbo].[Користувачі] student " +
                        "on attempt.[ID Студента] = student.id " +
                "WHERE attempt.Затверджено = 1 AND attempt.Оцінка IS NOT NULL AND attempt.Результат IS NOT NULL", databaseManager.connection);
                DataSet approvedTestData = new DataSet();

        approvedTestSqlDataAdapter.Fill(approvedTestData, "Спроби");
                approvedTestDataGridView.DataSource = approvedTestData;
                approvedTestDataGridView.DataMember = "Спроби";
            }

            private void TeacherWindow_Load(object sender, EventArgs e)
            {
                RefreshDataGridViews();

        testTabControl_SelectedIndexChanged(sender, e);
            }

            private void testTabControl_SelectedIndexChanged(object sender, EventArgs e)
            {
                if (testTabControl.SelectedIndex == 0)
                {
        addAttemptButtonPictureBox.Enabled = true;
        editAttemptButtonPictureBox.Enabled = true;
        approvedButtonPictureBox.Enabled = false;
        giveToPerformPictureBox.Enabled = true;
        viewTestResultButtonPictureBox.Enabled = false;
        workWithTestSamplesButtonPictureBox.Enabled = true;
                }
                else if (testTabControl.SelectedIndex == 1)
                {
        addAttemptButtonPictureBox.Enabled = true;
        editAttemptButtonPictureBox.Enabled = false;
        approvedButtonPictureBox.Enabled = true;
        giveToPerformPictureBox.Enabled = false;
        viewTestResultButtonPictureBox.Enabled = true;
        workWithTestSamplesButtonPictureBox.Enabled = true;
                }
                else if (testTabControl.SelectedIndex == 2)
                {
        addAttemptButtonPictureBox.Enabled = true;
        editAttemptButtonPictureBox.Enabled = false;
        approvedButtonPictureBox.Enabled = true;
        giveToPerformPictureBox.Enabled = false;
        viewTestResultButtonPictureBox.Enabled = true;
        workWithTestSamplesButtonPictureBox.Enabled = true;
                }
                approvedButtonPictureBox.Visible = approvedButtonPictureBox.Enabled;
                giveToPerformPictureBox.Visible = giveToPerformPictureBox.Enabled;
                var ivoryColor = addAttemptButtonPictureBox.BackColor;
        addAttemptButtonPictureBox.BackColor = addAttemptButtonPictureBox.Enabled? ivoryColor : Color.FromArgb(222, 222, 222);
        editAttemptButtonPictureBox.BackColor = editAttemptButtonPictureBox.Enabled ? ivoryColor : Color.FromArgb(222, 222, 222);
                approvedButtonPictureBox.BackColor = approvedButtonPictureBox.Enabled ? ivoryColor : Color.FromArgb(222, 222, 222);
        viewTestResultButtonPictureBox.BackColor = viewTestResultButtonPictureBox.Enabled ? ivoryColor : Color.FromArgb(222, 222, 222);
        workWithTestSamplesButtonPictureBox.BackColor = workWithTestSamplesButtonPictureBox.Enabled ? ivoryColor : Color.FromArgb(222, 222, 222);
            }
            private void approvedButtonPictureBox_Click(object sender, EventArgs e)
        {
                int attemptId = 0, isApproved = 0;
                if (testTabControl.SelectedIndex == 1)
                {
                    int rowIndex = notApprovedTestDataGridView.CurrentCell.RowIndex;
                    int columnIndex = notApprovedTestDataGridView.Columns["Затверджено"].Index;
                    attemptId = Convert.ToInt32(notApprovedTestDataGridView.Rows[rowIndex].Cells[0].Value);
                    isApproved = Convert.ToInt32(notApprovedTestDataGridView.Rows[rowIndex].Cells[columnIndex].Value);
                }
                else if (testTabControl.SelectedIndex == 2)
                {
                    int rowIndex = approvedTestDataGridView.CurrentCell.RowIndex;
                    int columnIndex = approvedTestDataGridView.Columns["Затверджено"].Index;
                    attemptId = Convert.ToInt32(approvedTestDataGridView.Rows[rowIndex].Cells[0].Value);
                    isApproved = Convert.ToInt32(approvedTestDataGridView.Rows[rowIndex].Cells[columnIndex].Value);
                }

                using (DatabaseManager dbManager = new DatabaseManager())
                {
        dbManager.ApproveAttempt(attemptId, isApproved == 1 ? 0 : 1);
                }

                RefreshDataGridViews();
            }
            private void addAttemptButtonPictureBox_Click(object sender, EventArgs e)
            {
                AddEditAttemptWindow addEditAttemptWindow = new AddEditAttemptWindow("Add attempt", 0, currentUserId, newTestDataGridView.CurrentCell.RowIndex);
                addEditAttemptWindow.ShowDialog();
                RefreshDataGridViews();
            }
            private void editAttemptButtonPictureBox_Click(object sender, EventArgs e)
            {
                AddEditAttemptWindow addEditAttemptWindow = new AddEditAttemptWindow("Edit attempt",
Convert.ToInt32(newTestDataGridView.Rows[newTestDataGridView.CurrentCell.RowIndex].Cells[0].Value), currentUserId, newTestDataGridView.CurrentCell.RowIndex);
                addEditAttemptWindow.ShowDialog();
                RefreshDataGridViews();
            }
```

```
                private                                         void
giveToPerformPictureBox_Click(object sender, EventArgs e)
                {
                        int             rowindex            =
newTestDataGridView.CurrentCell.RowIndex;
                        int           columnIndex           =
newTestDataGridView.Columns["Дано          студенту          для
виконання"].Index;
                        int           attemptId             =
Convert.ToInt32(newTestDataGridView.Rows[rowindex].Cells[0].Va
lue);

                        var     isGiveToPerformChecked     =
Convert.ToInt32(newTestDataGridView.Rows[rowindex].Cells[colum
nIndex].Value        ==        DBNull.Value?        0        :
newTestDataGridView.Rows[rowindex].Cells[columnIndex].Value);

                        using (DatabaseManager dbManager =
new DatabaseManager())
                        {

        dbManager.GiveToPerformAttempt(attemptId,
isGiveToPerformChecked == 1 ? 0 : 1);
                        }

                        RefreshDataGridViews();
                }

                private                                     void
workWithTestSamplesButtonPictureBox_Click(object       sender,
EventArgs e)
                {
                        WorkWithSelectionOfTestTasksWindow
workWithSelectionOfTestTasksWindow            =             new
WorkWithSelectionOfTestTasksWindow(currentUserId,
currentUserLastName,                      currentUserFirstName,
currentUserSurName);
                        this.Hide();

        workWithSelectionOfTestTasksWindow.Show();
                }
        }
}
```

Лістинг TestingWindow.cs:

```
using DRM_AIS_ComputerAdaptiveTesting.Entities;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace DRM_AIS_ComputerAdaptiveTesting
{
        public partial class TestingWindow : Form
        {
                private bool dragging = false;
                private Point dragCursorPoint;
                private Point dragFormPoint;
                public                    AdaptiveTestManager
adaptiveTestManager = new AdaptiveTestManager();
                List<Answer> answerTexts;
                TestTask currentTask;

                public TestingWindow()
                {
                        InitializeComponent();

        TaskUIDisplay(adaptiveTestManager.FirstQuestionChoice
());

                        subjectToolStripStatusLabel.Text  =
Test.Subject;
                        testNameToolStripStatusLabel.Text =
Test.Name;

        studentNameToolStripStatusLabel.Text                =
Student.LastName + " " + Student.FirstName;
        formulationOfTaskLabel.Text  =  "Питання  1.  Оберіть
правильну відповідь.";
                        testTaskToolStripStatusLabel.Text =
"1";
                }

                private  void  testingWindow_MouseDown(object
sender, MouseEventArgs e)
                {
                        dragging = true;
                        dragCursorPoint = Cursor.Position;
                        dragFormPoint = this.Location;
                }
                private  void  testingWindow_MouseMove(object
sender, MouseEventArgs e)
                {
                        if (dragging)
                        {
                                Point        dif        =
Point.Subtract(Cursor.Position, new Size(dragCursorPoint));
                                this.Location           =
Point.Add(dragFormPoint, new Size(dif));
                        }
                }
                private   void   testingWindow_MouseUp(object
sender, MouseEventArgs e)
                {
                        dragging = false;
                }
                private   void   minimizeButton_Click(object
sender, EventArgs e)
                {
                        this.WindowState                    =
FormWindowState.Minimized;
                }
                private    void    closeButton_Click(object
sender, EventArgs e)
                {
                        this.Close();
                }
                public void TaskUIDisplay(TestTask task)
                {
                        currentTask = task;
                        //  condition  should  be  changed
after rewriting of SQL request by join
                        if  (task.FKTypeOfTask  ==  "1"  ||
task.FKTypeOfTask == "2")
                        {
                                //choisePanel.Visible    =
true;
                                //openAnswerPanel.Visible
= false;
                                firstRadioButton.Visible
= secondRadioButton.Visible = true;
                                openAnswerTextBox.Visible
= false;
                                questionTextBox.Text     =
task.TaskName;

                                List<Answer> answerTexts;
                                using     (DatabaseManager
dbManager = new DatabaseManager())
                                {
                                        answerTexts     =
dbManager.GetAswers(task.ID);
                                }

                                // should be rewritten to
more адекватніший variant
                        DisplayRadioButtons(answerTexts);

                        formulationOfTaskLabel.Text                         =
formulationOfTaskLabel.Text    +    ".    Оберіть    правильну
відповідь.";
                        }
                        else
                        {
                                firstRadioButton.Visible
=   secondRadioButton.Visible   =   thirdRadioButton.Visible   =
fourthRadioButton.Visible = false;
                                openAnswerTextBox.Visible
= true;
                                questionTextBox.Text     =
task.TaskName;

                        formulationOfTaskLabel.Text                         =
formulationOfTaskLabel.Text    +    ".    Введіть    правильну
відповідь.";
                        }
                }
                private                                     void
DisplayRadioButtons(List<Answer> answers)
                {
                        answerTexts = answers;
                        if (answers.Count == 2)
                        {
                                firstRadioButton.Text    =
answers[0].Text;
                                secondRadioButton.Text   =
answers[1].Text;
                                thirdRadioButton.Visible
= fourthRadioButton.Visible = false;
                                firstRadioButton.Checked
= secondRadioButton.Checked = false;
                        }
                        else if (answers.Count == 3)
                        {
                                firstRadioButton.Text    =
answers[0].Text;
                                secondRadioButton.Text   =
answers[1].Text;
                                thirdRadioButton.Text    =
answers[2].Text;
                                thirdRadioButton.Visible
= true;
                                fourthRadioButton.Visible
= false;
                                firstRadioButton.Checked
=   secondRadioButton.Checked   =   thirdRadioButton.Checked   =
false;
                        }
                        else if (answers.Count == 4)
                        {
                                firstRadioButton.Text    =
answers[0].Text;
                                secondRadioButton.Text   =
answers[1].Text;
                                thirdRadioButton.Text    =
answers[2].Text;
                                fourthRadioButton.Text   =
answers[3].Text;
                                thirdRadioButton.Visible
= fourthRadioButton.Visible = true;
                                firstRadioButton.Checked
=   secondRadioButton.Checked   =   thirdRadioButton.Checked   =
fourthRadioButton.Checked = false;
```

```
                        }
                }

                private   void   nextTaskButton_Click(object
sender, EventArgs e)
                {
                        bool isAnswerRight = false;
                        if (currentTask.FKTaskType == "1"
|| currentTask.FKTaskType == "2")
                        {
                                Answer    rightAnswer    =
answerTexts.Where(a => a.IsCorrect == true).FirstOrDefault();

                                foreach          (Control
radioButton in choisePanel.Controls)
                                {
                                        isAnswerRight   =
(radioButton          is          RadioButton)          &&
((RadioButton)radioButton).Checked   &&   (radioButton.Text   ==
rightAnswer.Text);
                                        if
(isAnswerRight)
                                        {
                                                break;
                                        }
                                }
                        }
                        else
                        {
                                List<Answer> answerTexts;
                                using     (DatabaseManager
dbManager = new DatabaseManager())
                                {
                                        answerTexts     =
dbManager.GetAswers(currentTask.ID);
                                }
                                isAnswerRight          =
openAnswerTextBox.Text == answerTexts.FirstOrDefault().Text ||
openAnswerTextBox.Text                                  ==
answerTexts.FirstOrDefault().Text.ToLower();
                        }
                        TestTask         nextTask         =
adaptiveTestManager.NextQuestionChoice(isAnswerRight);
                        if
(!string.IsNullOrEmpty(nextTask.TaskName))
                        {
                                TaskUIDisplay(nextTask);

        formulationOfTaskLabel.Text    =    "Питання    "    +
(AdaptiveTestManager.numberOfTask + 1).ToString();

        testTaskToolStripStatusLabel.Text                    =
nextTask.Priority.ToString();
                        }
                        else
                        {
                                this.Hide();

        CompletedStudentsTest.EndDateAndTime = DateTime.Now;
                                TestResultWindow
resultForm = new TestResultWindow();
                                resultForm.Closed  +=  (s,
args) => this.Close();
                                resultForm.Show();
                        }
                }
        }
}
```

Лістинг TestResultWindow.cs:

```
using DRM_AIS_ComputerAdaptiveTesting.Entities;
using System;
using System.Drawing;
using System.Windows.Forms;

namespace DRM_AIS_ComputerAdaptiveTesting
{
        public partial class TestResultWindow : Form
        {
                private bool dragging = false;
                private Point dragCursorPoint;
                private Point dragFormPoint;

                public TestResultWindow()
                {
                        InitializeComponent();

                        //testNameLabel.Text += Test.Name;
                        //pointLabel.Text                +=
AssessmentDetermining(AdaptiveTestManager.countRightAnswersOnT
asks, AdaptiveTestManager.numberOfTask).ToString();
                        //TimeSpan      completeTime      =
CompletedStudentsTest.EndDateAndTime.Subtract(CompletedStudent
sTest.StartDateAndTime);
                        //completeTimeLabel.Text         +=
completeTime.Minutes.ToString()         +         ":"         +
completeTime.Seconds.ToString();
                        //countAllTasksLabel.Text        +=
AdaptiveTestManager.numberOfTask.ToString() + " запитання (-
нь)";
                        //countRightAnswersLabel.Text    +=
AdaptiveTestManager.countRightAnswersOnTasks.ToString();
                }

                private   void   resultForm_MouseDown(object
sender, MouseEventArgs e)
                {
                        dragging = true;
                        dragCursorPoint = Cursor.Position;
                        dragFormPoint = this.Location;
                }

                private   void   resultForm_MouseMove(object
sender, MouseEventArgs e)
                {
                        if (dragging)
                        {
                                Point    newPosition    =
Point.Subtract(Cursor.Position, new Size(dragCursorPoint));
                                this.Location           =
Point.Add(dragFormPoint, new Size(newPosition));
                        }
                }
                private    void    resultForm_MouseUp(object
sender, MouseEventArgs e)
                {
                        dragging = false;
                }
                private                                 void
minimizeButton_MouseClick(object sender, MouseEventArgs e)
                {
                        this.WindowState                =
FormWindowState.Minimized;
                }
                private    void    closeButton_Click(object
sender, EventArgs e)
                {
                        this.Close();
                }
                private   void   endTestButton_Click(object
sender, EventArgs e)
                {
                        this.Close();
                }

                private    int    AssessmentDetermining(int
countRightAnswers, int countAllTasks)
                {
                        double percentageOfCorrectAnswers =
(double)countRightAnswers / (double)countAllTasks;
                        int score = 2;
                        if  (percentageOfCorrectAnswers  >=
0.4 && percentageOfCorrectAnswers <= 0.6)
                        {
                                score = 3;
                        }
                        else if (percentageOfCorrectAnswers
>= 0.6 && percentageOfCorrectAnswers <= 0.8)
                        {
                                score = 4;
                        }
                        else if (percentageOfCorrectAnswers
> 0.8 && percentageOfCorrectAnswers <= 1)
                        {
                                score = 5;
                        }
                        return score;
                }
        }
}
```

Лістинг WorkWithSelectionOfTestTasksWindow.cs:

```
using DRM_AIS_ComputerAdaptiveTesting.Entities;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace DRM_AIS_ComputerAdaptiveTesting
{
        public                  partial                  class
WorkWithSelectionOfTestTasksWindow : Form
        {
                private bool dragging = false;
                private Point dragCursorPoint;
                private Point dragFormPoint;
                Dictionary<int, string> iemList;
                Dictionary<int,                        string>
headingsListSelectedTestTasks = new Dictionary<int, string>();
                Dictionary<int,                        string>
headingsListTestTasks = new Dictionary<int, string>();
                int currentUserId;
                string                  currentUserLastName,
currentUserFirstName, currentUserSurName;

                public
WorkWithSelectionOfTestTasksWindow(int      userId,      string
userLastName, string userFirstName, string userSurName)
                {
                        InitializeComponent();

                        lastNameTextBox.Text              =
userLastName;
                        firstNameTextBox.Text             =
userFirstName + " " + userSurName;
                        currentUserId = userId;
                        currentUserLastName = userLastName;
                        currentUserFirstName              =
userFirstName;
                        currentUserSurName = userSurName;
```

```
                        using (DatabaseManager dbManager = new DatabaseManager())
                        {
                                iemList = dbManager.GetIEM();
        iemComboBox.Items.Add("");
        iemTasksComboBox.Items.Add("");
                                foreach (var iem in iemList)
                                {
        iemComboBox.Items.Add(iem.Value);
        iemTasksComboBox.Items.Add(iem.Value);
                                }
                        }
                }

                private void workWithSelectionOfTestTasksWindow_MouseDown(object sender, MouseEventArgs e)
                {
                        dragging = true;
                        dragCursorPoint = Cursor.Position;
                        dragFormPoint = this.Location;
                }

                private void workWithSelectionOfTestTasksWindow_MouseMove(object sender, MouseEventArgs e)
                {
                        if (dragging)
                        {
                                Point newPosition = Point.Subtract(Cursor.Position, new Size(dragCursorPoint));
                                this.Location = Point.Add(dragFormPoint, new Size(newPosition));
                        }
                }

                private void workWithSelectionOfTestTasksWindow_MouseUp(object sender, MouseEventArgs e)
                {
                        dragging = false;
                }

                private void backButtonPictureBox_Click(object sender, EventArgs e)
                {
                        TeacherWindow teacherWindow = new TeacherWindow(currentUserId, currentUserLastName, currentUserFirstName, currentUserSurName);
                        this.Close();
                        teacherWindow.Show();
                }

                private void RefreshSelectionsOfTestTasksDataGridViews()
                {
                        DatabaseManager databaseManager = new DatabaseManager();
                        SqlDataAdapter selectionsOfTestTasksSqlDataAdapter = new SqlDataAdapter("SELECT selections.[ID], heading.[Назва] [Назва заголовку], heading.[Рівень] [Рівень заголовку], selections.[Назва], " +
                "teacher.Прізвище + ' ' + teacher.Імя + ' ' + teacher.[По батькові] Викладач, " +
        "selections.[Дата та час створення] " +
                                "FROM [VAO Master Work].[dbo].[Вибірки тестових завдань] selections " +
                                "join [VAO Master Work].[dbo].[Заголовки] heading on selections.[ID Загловку] = heading.id " +
                                "join [VAO Master Work].[dbo].[Користувачі] teacher " +
                                "on selections.[ID Викладача] = teacher.id  ", databaseManager.connection);
                        DataSet newTestData = new DataSet();

        selectionsOfTestTasksSqlDataAdapter.Fill(newTestData, "Вибірки тестових завдань");

        selectionsOfTestTasksDataGridView.DataSource = newTestData;

        selectionsOfTestTasksDataGridView.DataMember = "Вибірки тестових завдань";

                        SqlDataAdapter testTasksSqlDataAdapter = new SqlDataAdapter("SELECT testTask.[ID], testTask.[Текст], term.Назва [Ключовий термін], sentence.Текст [Речення], tasksTypes.Назва [Тип завдання] " +
                                                "FROM [VAO Master Work].[dbo].[Тестові завдання] testTask " +
                                                "join [VAO Master Work].[dbo].[Ключові терміни] keyTerm on testTask.[ID Ключового терміну] = keyTerm.id " +
                                                "join [VAO Master Work].[dbo].[Терміни] term on keyTerm.[ID Терміну] = term.ID " +
                                                "join [VAO Master Work].[dbo].[Речення] sentence on testTask.[ID Речення] = sentence.id " +
                                                "join [VAO Master Work].[dbo].[Типи завдань] tasksTypes on testTask.[ID Типу] = tasksTypes.id ", databaseManager.connection);
                        DataSet testTasksData = new DataSet();

        testTasksSqlDataAdapter.Fill(testTasksData, "Тестові завдання");
                        testTasksDataGridView.DataSource = testTasksData;
                        testTasksDataGridView.DataMember = "Тестові завдання";
                }

                private void WorkWithSelectionOfTestTasksWindow_Load(object sender, EventArgs e)
                {
                RefreshSelectionsOfTestTasksDataGridViews();
                }

                private void headingComboBox_SelectedValueChanged(object sender, EventArgs e)
                {
                        if (iemComboBox.Text != "")
                        {
                                DatabaseManager databaseManager = new DatabaseManager();
                                SqlDataAdapter selectionsOfTestTasksSqlDataAdapter = new SqlDataAdapter(string.Format($"SELECT selections.[ID], heading.[Назва] [Назва заголовку], heading.[Рівень] [Рівень заголовку], selections.[Назва], " +
                                        $"teacher.Прізвище + ' ' + teacher.Імя + ' ' + teacher.[По батькові] Викладач, " +
                                        $"selections.[Дата та час створення] " +
                                $"FROM [VAO Master Work].[dbo].[Вибірки тестових завдань] selections " +
                                        $"join [VAO Master Work].[dbo].[Заголовки] heading on selections.[ID Загловку] = heading.id " +
                                        $"join [VAO Master Work].[dbo].[Користувачі] teacher " +
                                        $"on selections.[ID Викладача] = teacher.id " +
                                $"WHERE selections.[ID Загловку] = {headingsListSelectedTestTasks.FirstOrDefault(x => x.Value == headingComboBox.Text).Key}"), databaseManager.connection);
                                DataSet newTestData = new DataSet();

                        selectionsOfTestTasksSqlDataAdapter.Fill(newTestData, "Вибірки тестових завдань");

                        selectionsOfTestTasksDataGridView.DataSource = newTestData;

                        selectionsOfTestTasksDataGridView.DataMember = "Вибірки тестових завдань";
                        }
                        else
                        {

                        RefreshSelectionsOfTestTasksDataGridViews();
                        }
                }

                private void headingTasksComboBox_SelectedValueChanged(object sender, EventArgs e)
                {
                        if (iemComboBox.Text != "")
                        {
                                DatabaseManager databaseManager = new DatabaseManager();
                                SqlDataAdapter testTasksSqlDataAdapter = new SqlDataAdapter(string.Format($"SELECT testTask.[ID], testTask.[Текст], term.Назва [Ключовий термін], sentence.Текст [Речення], tasksTypes.Назва [Тип завдання] " +
                                        $"FROM [VAO Master Work].[dbo].[Тестові завдання] testTask " +
                                        $"join [VAO Master Work].[dbo].[Ключові терміни] keyTerm on testTask.[ID Ключового терміну] = keyTerm.id " +
                                        $"join [VAO Master Work].[dbo].[Терміни] term on keyTerm.[ID Терміну] = term.ID " +
                                        $"join [VAO Master Work].[dbo].[Речення] sentence on testTask.[ID Речення] = sentence.id " +
                                        $"join [VAO Master Work].[dbo].[Типи завдань] tasksTypes on testTask.[ID Типу] = tasksTypes.id " +
                                        $"WHERE keyTerm.[ID Загловку] = {headingsListTestTasks.FirstOrDefault(x => x.Value == headingTasksComboBox.Text).Key}"), databaseManager.connection);
                                DataSet testTasksData = new DataSet();
```

```
        testTasksSqlDataAdapter.Fill(testTasksData,  "Тестові
завдання");

        testTasksDataGridView.DataSource = testTasksData;

        testTasksDataGridView.DataMember      =      "Тестові
завдання";
                        }
                        else
                        {

        RefreshSelectionsOfTestTasksDataGridViews();
                        }
                }

                private                                 void
iemComboBox_SelectedValueChanged(object sender, EventArgs e)
                {
                        List<Heading> headingsList;
                        headingComboBox.Items.Clear();

        headingsListSelectedTestTasks.Clear();

                        if (iemComboBox.Text == "")
                        {
                                headingComboBox.Enabled  =
false;
                                headingComboBox.Text     =
"";
                        }
                        else
                        {
                                using     (DatabaseManager
dbManager = new DatabaseManager())
                                {
                                        headingsList    =
dbManager.GetHeadings(iemList.FirstOrDefault(x  =>  x.Value  ==
iemComboBox.Text).Key);
                                        foreach      (var
heading in headingsList)
                                        {

        headingComboBox.Items.Add(heading.Level             +
heading.Name);

        headingsListSelectedTestTasks.Add(heading.ID,
heading.Level + heading.Name);
                                        }
                                }

                                headingComboBox.Enabled  =
true;
                        }
                }

                private                                 void
iemTasksComboBox_SelectedValueChanged(object sender, EventArgs
e)
                {
                        List<Heading> headingsList;
                        headingTasksComboBox.Items.Clear();
                        headingsListTestTasks.Clear();

                        if (iemTasksComboBox.Text == "")
                        {

        headingTasksComboBox.Enabled = false;
                                headingTasksComboBox.Text
= "";
                        }
                        else
                        {

        headingTasksComboBox.Items.Add("");
                                using     (DatabaseManager
dbManager = new DatabaseManager())
                                {
                                        headingsList    =
dbManager.GetHeadings(iemList.FirstOrDefault(x  =>  x.Value  ==
iemTasksComboBox.Text).Key);
                                        foreach      (var
heading in headingsList)
                                        {

        headingTasksComboBox.Items.Add(heading.Level + " " +
heading.Name);

        headingsListTestTasks.Add(heading.ID, heading.Level +
" " + heading.Name);
                                        }
                                }
        headingTasksComboBox.Enabled = true;
                        }
                }

                private                                 void
addSelectionOfTestTaskButtonPictureBox_Click(object     sender,
EventArgs e)
                {
                        if (testTabControl.SelectedIndex ==
0)
                        {

        AddEditSelectionOfTestTasks
addEditSelectionOfTestTasks               =               new
AddEditSelectionOfTestTasks("Add selection", 0);

        addEditSelectionOfTestTasks.ShowDialog();

        RefreshSelectionsOfTestTasksDataGridViews();
                        }
                        else
if(testTabControl.SelectedIndex == 1)
                        {
                                AddEditTestTasksWindow
addEditTestTasksWindow   =   new   AddEditTestTasksWindow("Add
task", 0);

        addEditTestTasksWindow.ShowDialog();

        RefreshSelectionsOfTestTasksDataGridViews();
                        }
                }

                private                                 void
editSelectionOfTestTaskButtonPictureBox_Click(object    sender,
EventArgs e)
                {
                        if (testTabControl.SelectedIndex ==
0)
                        {
                                int     columnIndex     =
selectionsOfTestTasksDataGridView.Columns["ID"].Index;

        AddEditSelectionOfTestTasks
addEditSelectionOfTestTasks               =               new
AddEditSelectionOfTestTasks("Edit                selection",
Convert.ToInt32(selectionsOfTestTasksDataGridView.Rows[selecti
onsOfTestTasksDataGridView.CurrentCell.RowIndex].Cells[columnI
ndex].Value));

        addEditSelectionOfTestTasks.ShowDialog();

        RefreshSelectionsOfTestTasksDataGridViews();
                        }
                        else                            if
(testTabControl.SelectedIndex == 1)
                        {
                                int     columnIndex     =
testTasksDataGridView.Columns["ID"].Index;
                                AddEditTestTasksWindow
addEditTestTasksWindow   =   new   AddEditTestTasksWindow("Edit
task",
Convert.ToInt32(testTasksDataGridView.Rows[testTasksDataGridVi
ew.CurrentCell.RowIndex].Cells[columnIndex].Value));

        addEditTestTasksWindow.ShowDialog();

        RefreshSelectionsOfTestTasksDataGridViews();
                        }
                }

                private   void   minimizeButton_Click(object
sender, EventArgs e)
                {
                        this.WindowState                =
FormWindowState.Minimized;
                }

                private    void    closeButton_Click(object
sender, EventArgs e)
                {
                        this.Close();
                }
        }
}
```

Додаток Е

Дослідження коректності виконання функцій адміністрування системи

Дослідження функціональності тестової інформаційної системи – це процес перевірки відповідності вимог, що були заявлені до системи, та фактично реалізованої функціональності. Дослідження відбувається шляхом спостереження за процесами системи в штучно створених ситуаціях і на обмеженому наборі тестових випадків, обраних певним чином.

Для дослідження функціональності підсистеми адміністрування розробленої інформаційної системи адаптивного тестування рівня знань було розроблено чотири тестових випадки (тест-кейси). У першому тестовому випадку (таблиця Е.1) перевіряється процес входу у інформаційну систему. При вході у обліковий запис діє валідація даних. Тобто, якщо ввести значення у поле логіну, але не ввести пароль, справа від поля для введення паролю з’явиться знак попередження із текстом «Введіть, будь ласка, пароль» (рисунок Е.1).

Рисунок Е.1 – Попередження у випадку, якщо поле не заповнено

У випадку, якщо було введено не правильне значення у полі логіну, при натисканні на кнопку «Вхід» з’явиться діалог із помилкою «Введено не правильний логін. Будь ласка, перевірте введені дані!» (рисунок Е.2), при не правильному значенні паролю буде показано аналогічну помилку.

Таблиця Е.1 – Тест-кейс АТ0005

Тест-кейс ID: АТ0005	**Пріоритет:** 1	**Створено:** 08.11.2020, Г.Білоус
Назва: Перевірка логіну у обліковий запис адміністратора **Вхідні дані:** Логін = «admin», Пароль = «admin»		
Кроки		**Очікуваний результат**
1. Запустити додаток 2. Ввести значення у поле логіну із вхідних даних 3. Натиснути кнопку «Вхід» 4. Порівняти фактичний результат з очікуваним 5. Ввести значення в поле логіну = «adm», а у поле паролю із вхідних даних 6. Натиснути кнопку «Вхід» 7. Порівняти фактичний результат з очікуваним 8. Закрити діалог із помилкою 9. Ввести значення в поле логіну із вхідних даних, а у поле паролю = «adm» 10. Натиснути кнопку «Вхід» 11. Порівняти фактичний результат з очікуваним 12. Закрити діалог із помилкою 13. Ввести значення в поля логіну та паролю із вхідних даних 14. Натиснути кнопку «Вхід» 15. Порівняти фактичний результат з очікуваним		Біля поля паролю є знак попередження із текстом: «Введіть, будь ласка, пароль». Діалог із помилкою: «Введено не правильний логін. Будь ласка, перевірте введені дані!». Діалог із помилкою: «Введено не правильний пароль. Будь ласка, перевірте введені дані!». Вікно входу у систему закрилось. Вікно з інтерфейсом адміністратора відкрито.
Результат виконання тест-кейсу: пройдено успішно		

Якщо у поля логіну та паролю введено коректні дані, то при натисканні на кнопку «Вхід» вікно входу закриється та з’явиться вікно адміністрування (рисунок Е.3).

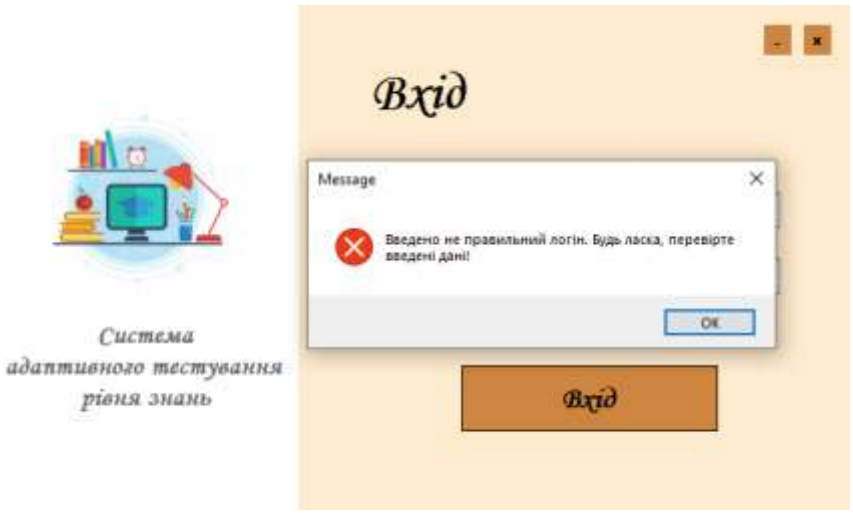


Рисунок Е.2 – Помилка у випадку, якщо введено не правильне значення у полі логіну

Рисунок Е.3 – Вікно адміністрування

Другий тестовий випадок (таблиця Е.2) перевіряє функціонал створення нового користувача. При настисканні на кнопку «Додати користувача», що знаходиться на панелі керування вікна адміністрування, відкривається вікно «Додати користувача» (рисунок Е.4).

Рисунок Е.4 – Вікно «Додати користувача»

Після заповнення полей на формі та натисканні на кнопку «Додати», вікно закривається, а у таблиці із користувачами з’явиться новий рядок, що містить дані введені на формі вікна «Додати користувача» (рисунок Е.5).

Рисунок Е.5 – Оновленні дані у таблиці користувачів

Таблиця Е.2 – Тест-кейс АТ0006

Тест-кейс ID: АТ0006	**Пріоритет:** 2	**Створено:** 08.11.2020, Г.Білоус
Назва: Перевірка функціоналу створення нового користувача **Вхідні дані:** Логін = «admin», Пароль = «admin»		
Кроки		**Очікуваний результат**
1. Запустити додаток 2. Ввести значення у поле логіну та паролю із вхідних даних 3. Натиснути кнопку «Вхід» 4. Порівняти фактичний результат з очікуваним 5. Натиснути кнопку «Додати користувача» 6. Порівняти фактичний результат з очікуваним 7. Ввести довільні дані про нового користувача у форму 8. Натиснути кнопку «Додати» 9. Порівняти фактичний результат з очікуваним		Вікно входу у систему закрито. Вікно з інтерфейсом адміністратора відкрито. Відкрито вікно «Додати користувача». Вікно «Додати користувача» закрито. У таблиці із користувачами з’явився новий рядок, що містить дані введені на кроці 7.
Результат виконання тест-кейсу: пройдено успішно		

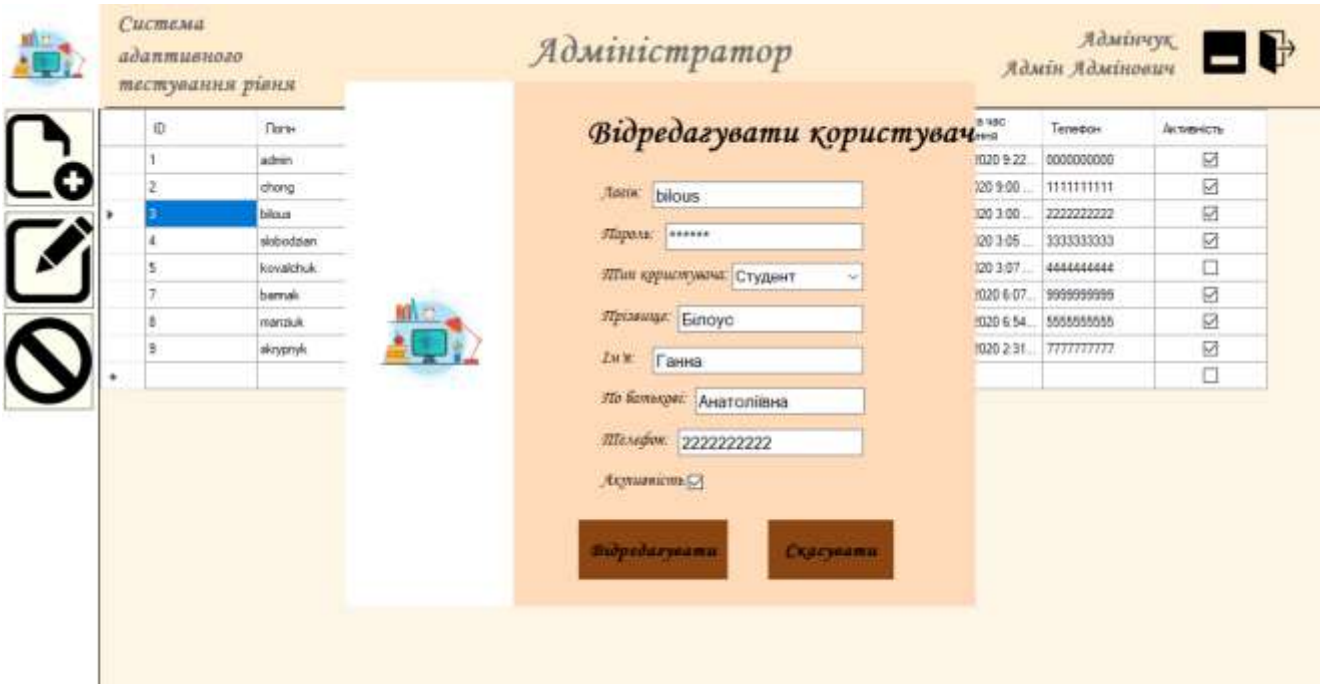


Рисунок Е.6 – Вікно «Відредагувати користувача»

Третій тестовий випадок (таблиця Е.3) перевіряє функціонал редагування даних про існуючого користувача. При настисканні на кнопку «Відредагувати користувача», що знаходиться на панелі керування вікна адміністрування, відкривається вікно «Відредагувати користувача», форма якого містить дані про користувача обраного попередньо у таблиці (рисунок Е.6).

Таблиця Е.3 – Тест-кейс АТ0007

Тест-кейс ID: АТ0007	**Пріоритет:** 2	**Створено:** 08.11.2020, Г.Білоус
Назва: Перевірка функціоналу редагування існуючого користувача **Вхідні дані:** Логін = «admin», Пароль = «admin»		
Кроки	**Очікуваний результат**	
1. Запустити додаток 2. Ввести значення у поле логіну та паролю із вхідних даних 3. Натиснути кнопку «Вхід» 4. Порівняти фактичний результат з очікуваним	Вікно входу у систему закрито. Вікно з інтерфейсом адміністратора відкрито.	
5. Обрати другий рядок у таблиці користувачів 6. Натиснути кнопку «Відредагувати користувача» 7. Порівняти фактичний результат з очікуваним	Відкрито вікно «Відредагувати користувача».	
8. Змінити телефон та тип користувача 9. Натиснути кнопку «Відредагувати» 10. Порівняти фактичний результат з очікуваним	Вікно «Відредагувати користувача» закрито. У таблиці із користувачами дані у другому рядку, змінено на дані введені на кроці 7.	
Результат виконання тест-кейсу: пройдено успішно		

Після редагування полей на формі та натисканні на кнопку «Відредагувати», вікно закриється, а у таблиці із користувачами оновляться дані у рядку, що відповідний користувачу, який редагувався.

У четвертому тестовому випадку (таблиця Е.4) перевіряється функціонал блокування користувача. Якщо у колонці «Активність» стоїть прапорець для певного користувача, це означає, що він активний, у випадку якщо прапорець

знято – для користувача заблокований вхід у систему. Описаний функціонал регулює кнопка «Блокувати користувача».

Таблиця Е.4 – Тест-кейс АТ0008

<table>
<tr><td colspan="2">Тест-кейс ID: АТ0008</td><td>Пріоритет: 3</td><td>Створено: 08.11.2020, Г.Білоус</td></tr>
<tr><td colspan="4">Назва: Перевірка функціоналу блокування існуючого користувача
Вхідні дані: Логін = «admin», Пароль = «admin»</td></tr>
<tr><td colspan="2">Кроки</td><td colspan="2">Очікуваний результат</td></tr>
<tr><td colspan="2">1. Запустити додаток
2. Ввести значення у поле логіну та паролю із вхідних даних
3. Натиснути кнопку «Вхід»
4. Порівняти фактичний результат з очікуваним
5. Обрати третій рядок у таблиці користувачів
6. Натиснути кнопку «Блокувати користувача»
7. Порівняти фактичний результат з очікуваним
8. Обрати третій рядок у таблиці користувачів
9. Натиснути кнопку «Блокувати користувача»
10. Порівняти фактичний результат з очікуваним</td><td colspan="2">Вікно входу у систему закрито. Вікно з інтерфейсом адміністратора відкрито.
Прапорець у колонці «Активність» третього рядка було знято.
Прапорець у колонці «Активність» третього рядка було поставлено.</td></tr>
<tr><td colspan="4">Результат виконання тест-кейсу: пройдено успішно</td></tr>
</table>

Отже, дослідження функціональності підсистеми адміністрування розробленої інформаційної системи адаптивного тестування рівня знань було успішно пройдено. Наочно доведено, що функції користувача адміністратор вказані у п.3.1 було реалізовано у інформаційній системі.

Додаток Ж

Дослідження коректності виконання функцій викладача системи

Для дослідження функціональності підсистеми викладача розробленої інформаційної системи адаптивного тестування рівня знань було розроблено дев’ять тестових випадків (тест-кейсів). У першому тестовому випадку (таблиця Ж.1) перевіряється можливість перегляду запитів на рівень знань. На вкладці «Незатвердженні тести» (рисунок Ж.1) у таблиці мають відображатися лише записи із значенням у колонці «Затверджено» = false. Кнопка «Редагувати запит на рівень знань» недоступна (оскільки тестування за даними запитами вже завершене), а решта кнопок доступні.

Рисунок Ж.1 – Вкладка «Незатвердженні тести»

На вкладці «Архів (затверджені тести)» (рисунок Ж.2) у таблиці повинні відображатися лише ті записи, що мають значення у колонці «Затверджено» = true, кнопка «Редагувати запит на рівень знань» недоступна (оскільки тестування за даними запитами вже завершене). Решта кнопок на панелі керування доступні.

Рисунок Ж.2 – Вкладка «Архів (затверджені тести)»

На вкладці «Непройдені тести» (рисунок Ж.3) знаходиться таблиця з меншою кількістю колонок, ніж таблиці на вкладках «Архів (затверджені тести)» та «Незатверджені тести», тому що не всі дані про запит тестування є відомими, оскільки дані запити на визначення рівня знань студента ще не були пройдені. Кнопка «Перегляд тестового результату» у даному випадку недоступна (решта кнопок доступні).

Таблиця Ж.1 – Тест-кейс AT0009

<table>
<tr><td>Тест-кейс ID: AT0009</td><td>Пріоритет: 1</td><td>Створено: 08.11.2020, Г.Білоус</td></tr>
<tr><td colspan="3">Назва: Перевірка перегляду запитів на рівень знань
Вхідні дані: Логін = «chong», Пароль = «chong»</td></tr>
<tr><td colspan="2">Кроки</td><td>Очікуваний результат</td></tr>
<tr><td colspan="2">1. Запустити додаток
2. Ввести значення у поле логіну та паролю із вхідних даних
3. Натиснути кнопку «Вхід»
4. Порівняти фактичний результат з очікуваним
5. Перейти на вкладку «Незатверджені тести»
6. Порівняти фактичний результат з очікуваним
7. Перейти на вкладку «Архів (затверджені тести)»
8. Порівняти фактичний результат з очікуваним
9. Повернутися на вкладку «Непройдені тести»
10. Порівняти фактичний результат з очікуваним</td><td>Вікно входу у систему закрилось. Вікно з інтерфейсом викладача відкрито.

Таблиця з тестами містить лише записи із значенням у колонці «Затверджено» = false. Кнопка «Редагувати запит на рівень знань» недоступна. Решта кнопок доступні.

Таблиця з тестами містить лише записи із значенням у колонці «Затверджено» = true. Кнопка «Редагувати запит на рівень знань» недоступна. Решта кнопок доступні.

Таблиця з тестами містить обмежену кількість колонок: ID, Дата та час створення заявки, Вибірка тестових завдань, Тип тестування, Викладач, Студент, Дано на виконання студенту, Повідомлення, Максимальний час. Кнопка «Перегляд тестового результату» недоступна. Решта кнопок доступні.</td></tr>
<tr><td colspan="3">Результат виконання тест-кейсу: пройдено успішно</td></tr>
</table>

Рисунок Ж.3 – Вкладка «Непройдені тести»

Другий тестовий випадок (таблиця Ж.2) перевіряє функціонал створення нового запиту на рівень знань. При настисканні на кнопку «Додати запит на рівень знань», що знаходиться на панелі керування, відкривається вікно «Створення запиту на рівень знань» (рисунок Ж.4).

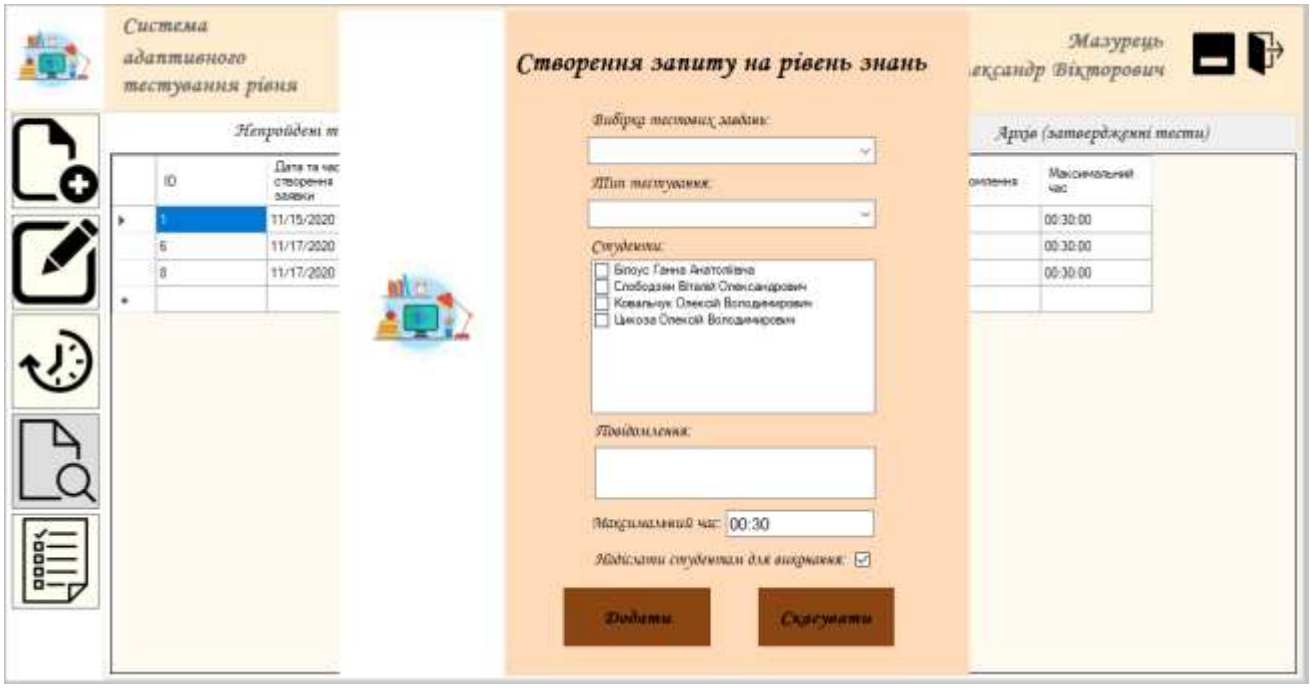


Рисунок Ж.4 – Вікно «Додати запит на рівень знань»

Після заповнення полей на формі та натисканні на кнопку «Додати», вікно закривається, а у таблиці із запитами на рівень знань (таблиця «Непройдені

тести») з’явився новий рядок, що містить дані введені на формі вікна «Додати запит на рівень знань» (рисунок Ж.5).

Рисунок Ж.5 – Оновленні дані у таблиці «Непройдені тести»

Таблиця Ж.2 – Тест-кейс AT0010

Тест-кейс ID: AT0010	**Пріоритет:** 1	**Створено:** 08.11.2020, Г.Білоус
Назва: Перевірка функціоналу створення нового запиту на рівень знань **Вхідні дані:** Логін = «chong», Пароль = «chong»		
Кроки		**Очікуваний результат**
1. Запустити додаток 2. Ввести значення у поле логіну із вхідних даних 3. Натиснути кнопку «Вхід» 4. Порівняти фактичний результат з очікуваним		Вікно входу у систему закрилось. Вікно з інтерфейсом викладача відкрито.
5. Натиснути кнопку «Додати запит на рівень знань» 6. Порівняти фактичний результат з очікуваним		Відкрито вікно «Створення запиту на рівень знань».
7. Ввести довільні дані про новий запит на рівень знань у форму 8. Натиснути кнопку «Додати» 9. Порівняти фактичний результат з очікуваним		Вікно «Створення запиту на рівень знань» закрито. У таблиці «Непройдені тести» з’явився новий рядок, що містить дані введені на кроці 7.
Результат виконання тест-кейсу: пройдено успішно		

Третій тестовий випадок (таблиця Ж.3) перевіряє функціонал редагування існуючого запиту на рівень знань. При настисканні на кнопку «Відредагувати

користувача», що знаходиться на панелі керування, відкривається вікно «Відредагувати запит на рівень знань», форма якого містить дані про запит на рівень знань, що обраний попередньо у таблиці (рисунок Ж.6).

Після редагування полей на формі та натисканні на кнопку «Відредагувати», вікно закриється, а у таблиці «Непройдені тести» оновляться дані у рядку, що відповідний запиту на рівень знань, який редагувався.

Таблиця Ж.3 – Тест-кейс AT0011

Тест-кейс ID: AT0011	**Пріоритет:** 2	**Створено:** 08.11.2020, Г.Білоус
Назва: Перевірка функціоналу редагування існуючого запиту на рівень знань **Вхідні дані:** Логін = «chong», Пароль = «chong»		
Кроки		**Очікуваний результат**
1. Запустити додаток 2. Ввести значення у поле логіну та паролю із вхідних даних 3. Натиснути кнопку «Вхід» 4. Порівняти фактичний результат з очікуваним		Вікно входу у систему закрито. Вікно з інтерфейсом викладача відкрито.
5. Обрати другий рядок у таблиці «Непойдені тести» 6. Натиснути кнопку «Відредагувати запит на рівень знань» 7. Порівняти фактичний результат з очікуваним		Відкрито вікно «Відредагувати запит на рівень знань».
8. Змінити тип тестування та повідомлення 9. Натиснути кнопку «Відредагувати» 10. Порівняти фактичний результат з очікуваним		Вікно «Відредагувати запит на рівень знань» закрито. У таблиці «Непройдені тести» дані у другому рядку, змінено на дані введені на кроці 7.
Результат виконання тест-кейсу: пройдено успішно		

У четвертому тестовому випадку (таблиця Ж.4) перевіряється функціонал затвердження пройденого тесту. Якщо викладач не переглядав результат тестування, то такий запит на рівень знань знаходиться у таблиці «Незатвердженні тести». Для того, щоб прийняти тест як виконаний, викладачу

потрібно натиснути кнопку «Затвердити запит на рівень знань» на панелі керування. Після цього затверджений запит з’являється у таблиці «Архів (затвердженні тести)».

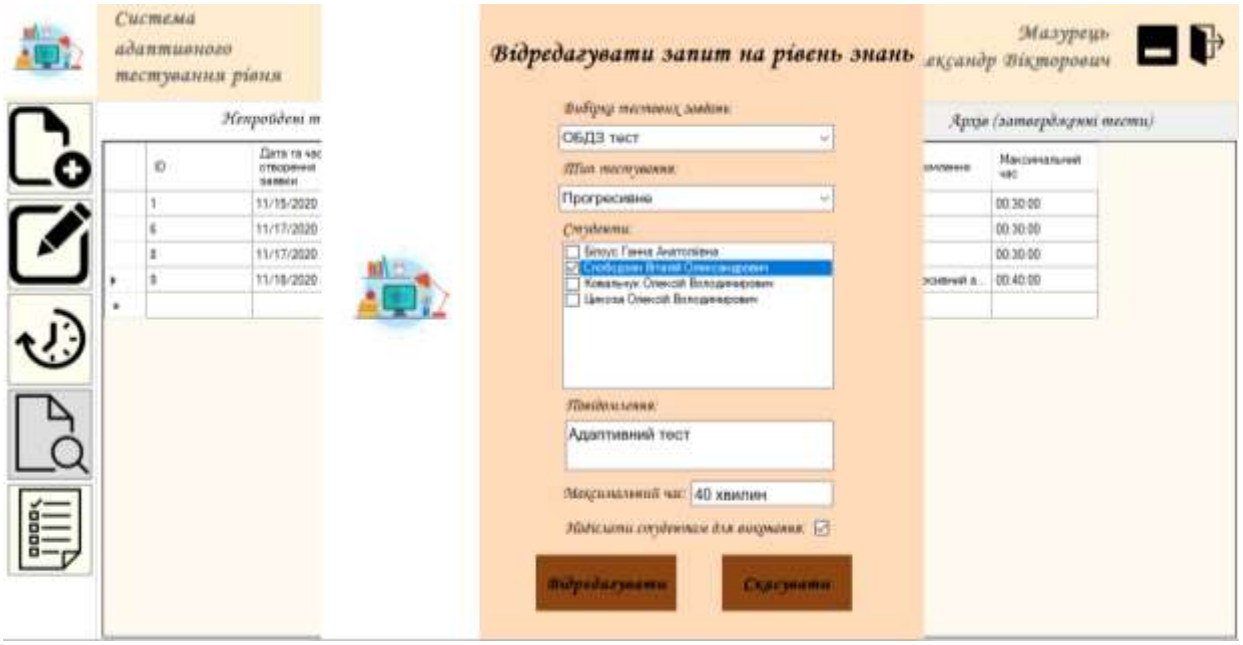


Рисунок Ж.6 – Вікно «Відредагувати запит на рівень знань»

Скасувати затвердження тесту як виконаного можна натиснувши кнопку «Скасувати затвердження запиту на рівень знань», що доступна при активній вкладці «Архів (затвердженні тести)». Після цього запит на рівень знань повертається до таблиці «Незатвердженні тести».

Наступний тест-кейс (таблиця Ж.5) перевіряє перехід до функціоналу для роботи із вибіркою тестових завдань. Для того, щоб перейти до роботи із вибірками тестових завдань та самими тестовими завданнями потірбно натиснути кнопку «Перейти до роботи із вибірками тестових завдань» на панелі керування. При цьому Вікно «Робота із запитами на рівень знань» буде згорнуто, а вікно «Робота із вибірками тестових завдань» відкрито. Активна таблицею у вікні «Робота із вибірками тестових завдань» буде «Вибірки тестових завдань» (рисунок Ж.7).

Таблиця Ж.4 – Тест-кейс AT0012

Тест-кейс ID: AT0012	**Пріоритет:** 2	**Створено:** 08.11.2020, Г.Білоус
Назва: Перевірка функціоналу затвердження пройденого тесту **Вхідні дані:** Логін = «chong», Пароль = «chong»		
Кроки	**Очікуваний результат**	
1. Запустити додаток 2. Ввести значення у поле логіну та паролю із вхідних даних 3. Натиснути кнопку «Вхід» 4. Порівняти фактичний результат з очікуваним 5. Перейти на вкладку «Незатверджені тести» 6. Обрати перший рядок у таблиці «Незатверджені тести» 7. Натиснути кнопку «Затвердити запит на рівень знань» 8. Порівняти фактичний результат з очікуваним 9. Перейти на вкладку «Архів (затверджені тести)» 10. Порівняти фактичний результат з очікуваним 11. Обрати перший рядок у таблиці «Архів (затверджені тести)» 12. Натиснути кнопку «Скасувати затвердження запиту на рівень знань» 13. Порівняти фактичний результат з очікуваним 14. Перейти на вкладку «Незатверджені тести» 15. Порівняти фактичний результат з очікуваним	Вікно входу у систему закрито. Вікно з інтерфейсом викладача відкрито. Запит, що затверджувався зник із таблиці «Незатверджені тести». Запит, що затверджувався з’явився у таблиці «Архів (затверджені тести)». Запит, для якого скасовувалося затвердження зник у таблиці «Архів (затверджені тести)». Запит, для якого скасовувалося затвердження з’явився у таблиці «Незатверджені тести».	
Результат виконання тест-кейсу: пройдено успішно		

На вкладці «Тестові завдання» (рисунок Ж.8) вікна «Робота із вибірками тестових завдань» представленно перелік тестових завдань, що можна сортувати відповідно до ІНМ та заголовку. Для того, щоб перейти назад до роботи із запитами на рівень знань, потрібно натиснути на кнопку «Повернутися до роботи із запитами на рівень знань». Внаслідок цього вікно «Робота із вибірками

тестових завдань» буде закрито, а вікно «Робота із запитами на рівень знань» буде відкрито.

Рисунок Ж.7 – Вікно «Робота із вибірками тестових завдань»

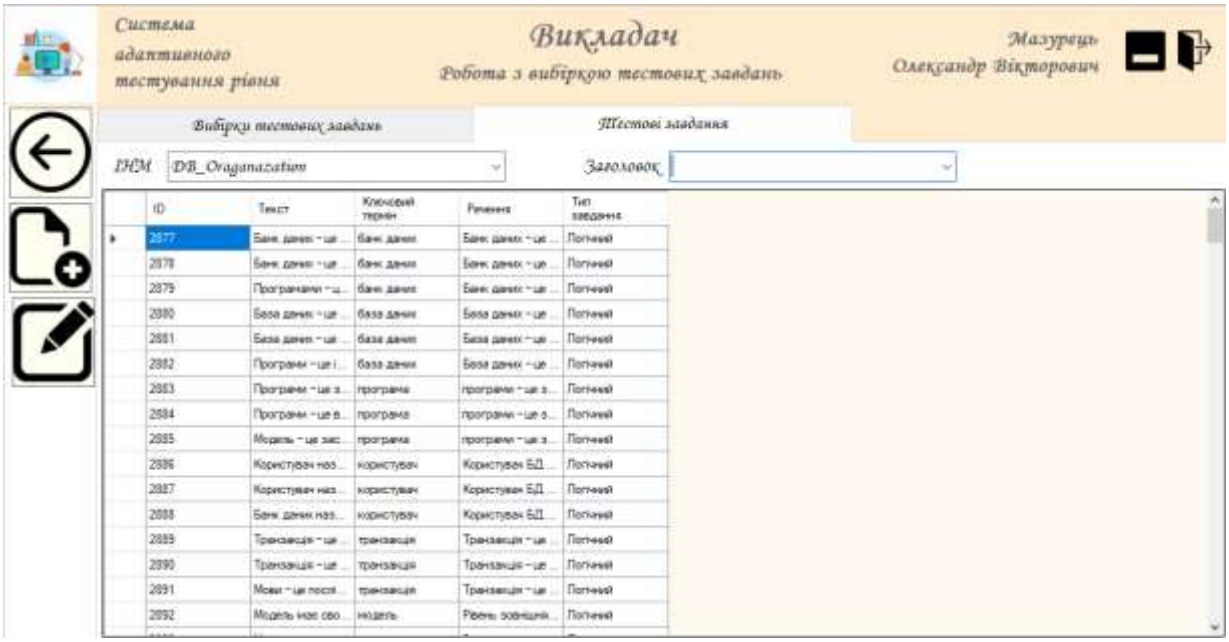


Рисунок Ж.8 – Вкладка «Тестові завдання» вікна «Робота із вибірками тестових завдань»

Шостий тестовий випадок (таблиця Ж.6) перевіряє функціонал створення нової вибірки тестових завдань. При настисканні на кнопку «Додати вибірку тестових завдань», що знаходиться на панелі керування, відкривається вікно «Додати вибірку тестових завдань» (рисунок Ж.9).

Таблиця Ж.5 – Тест-кейс AT0013

Тест-кейс ID: AT0013	**Пріоритет:** 3	**Створено:** 08.11.2020, Г.Білоус
Назва: Перевірка переходу до функціоналу для роботи із вибіркою тестових завдань **Вхідні дані:** Логін = «chong», Пароль = «chong»		
Кроки		**Очікуваний результат**
1. Запустити додаток 2. Ввести значення у поле логіну та паролю із вхідних даних 3. Натиснути кнопку «Вхід» 4. Порівняти фактичний результат з очікуваним 5. Натиснути кнопку «Перейти до роботи із вибірками тестових завдань» 6. Порівняти фактичний результат з очікуваним 7. Перейти на вкладку «Тестові завдання» 8. Порівняти фактичний результат з очікуваним 9. Натиснути кнопку «Повернутися до роботи із запитами на рівень знань» 10. Порівняти фактичний результат з очікуваним		Вікно входу у систему закрито. Вікно з інтерфейсом викладача відкрито. Вікно «Робота із запитами на рівень знань» згорнуто. Активна таблиця «Вибірки тестових завдань». Вікно «Робота із вибірками тестових завдань» відкрито. Активна таблиця «Тестові завдання». Вікно «Робота із вибірками тестових завдань» закрито. Вікно «Робота із запитами на рівень знань» відкрито.
Результат виконання тест-кейсу: пройдено успішно		

Рисунок Ж.9 – Вікно «Додати вибірку тестових завдань»

Таблиця Ж.6 – Тест-кейс AT0014

Тест-кейс ID: AT0014	**Пріоритет:** 1	**Створено:** 08.11.2020, Г.Білоус
Назва: Перевірка функціоналу створення нової вибірки тестових завдань **Вхідні дані:** Логін = «chong», Пароль = «chong»		
Кроки		**Очікуваний результат**
1. Запустити додаток 2. Ввести значення у поле логіну із вхідних даних 3. Натиснути кнопку «Вхід» 4. Порівняти фактичний результат з очікуваним 5. Натиснути кнопку «Перейти до роботи із вибірками тестових завдань» 6. Натиснути кнопку «Додати вибірку тестових завдань» 7. Порівняти фактичний результат з очікуваним 8. Ввести довільні дані про нову вибірку тестових завдань у форму 9. Натиснути кнопку «Додати» 10. Порівняти фактичний результат з очікуваним		Вікно входу у систему закрилось. Вікно з інтерфейсом викладача відкрито. Відкрито вікно «Додати вибірку тестових завдань». Вікно «Додати вибірку тестових завдань» закрито. У таблиці «Вибірки тестових завдань» з’явився новий рядок, що містить дані введені на кроці 8.
Результат виконання тест-кейсу: пройдено успішно		

Після заповнення полей на формі та натисканні на кнопку «Додати», вікно закривається, а у таблиці із вибірками тестових завдань з’явився новий рядок, що містить дані введені на формі вікна «Додати вибірку тестових завдань» (рисунок Ж.10).

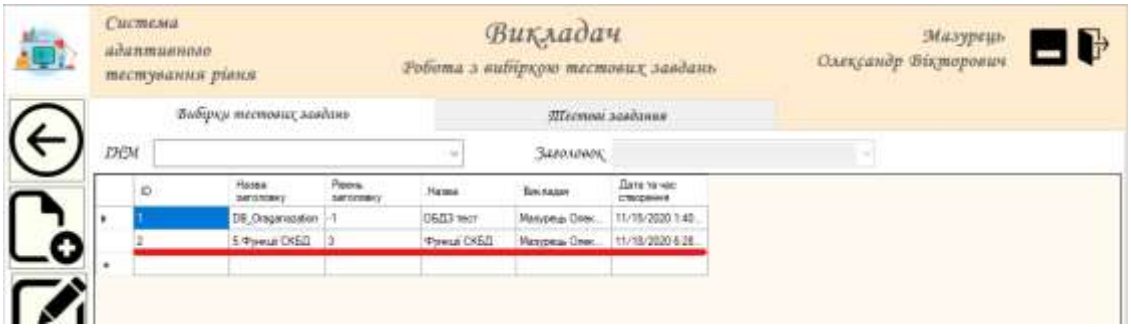


Рисунок Ж.10 – Оновленні дані у таблиці «Вибірки тестових завдань»

Сьомий тестовий випадок (таблиця Ж.7) перевіряє функціонал редагування існуючої вибірки тестових завдань. При настисканні на кнопку «Відредагувати вибірку тестових завдань», що знаходиться на панелі керування, відкривається вікно «Відредагувати вибірку тестових завдань», форма якого містить дані про вибірку тестових завдань, що обрана попередньо у таблиці (рисунок Ж.11).

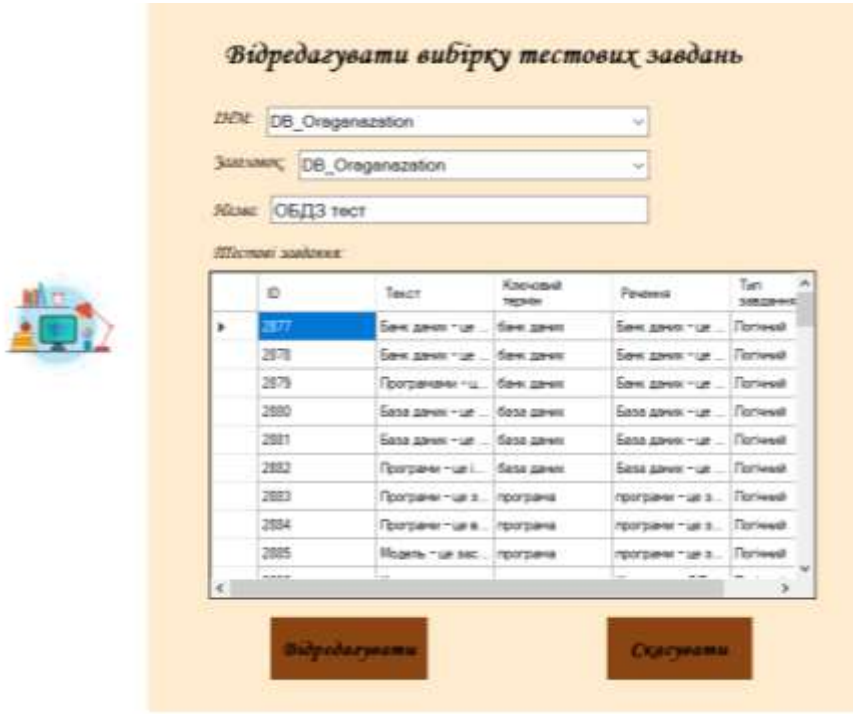


Рисунок Ж.11 – Вікно «Відредагувати вибірку тестових завдань»

Після редагування полей на формі та натисканні на кнопку «Відредагувати», вікно закриється, а у таблиці із вибірками тестових завдань оновляться дані у рядку, що відповідний вибірці, яка редагувалася.

Наступний тестовий випадок (таблиця Ж.8) перевіряє функціонал створення нового тестового завдання. При настисканні на кнопку «Додати тестове завдання», що знаходиться на панелі керування, відкривається вікно «Додати тестове завдання» (рисунок Ж.12).

Таблиця Ж.7 – Тест-кейс AT0015

Тест-кейс ID: AT0015	**Пріоритет:** 2	**Створено:** 08.11.2020, Г.Білоус
Назва: Перевірка функціоналу редагування існуючої вибірки тестових завдань **Вхідні дані:** Логін = «chong», Пароль = «chong»		
Кроки		**Очікуваний результат**
1. Запустити додаток 2. Ввести значення у поле логіну та паролю із вхідних даних 3. Натиснути кнопку «Вхід» 4. Порівняти фактичний результат з очікуваним 5. Натиснути кнопку «Перейти до роботи із вибірками тестових завдань» 6. Обрати перший рядок у таблиці «Вибірки тестових завдань» 7. Натиснути кнопку «Відредагувати вибірку тестових завдань» 8. Порівняти фактичний результат з очікуваним 9. Змінити назву вибірки та тестові завдання, що входять до неї 10. Натиснути кнопку «Відредагувати» 11. Порівняти фактичний результат з очікуваним		Вікно входу у систему закрито. Вікно з інтерфейсом викладача відкрито. Відкрито вікно «Відредагувати вибірку тестових завдань». Вікно «Відредагувати вибірку тестових завдань» закрито. У таблиці «Вибірки тестових завдань» назва вибірки у першому рядку, змінено на назву введену на кроці 9.
Результат виконання тест-кейсу: пройдено успішно		

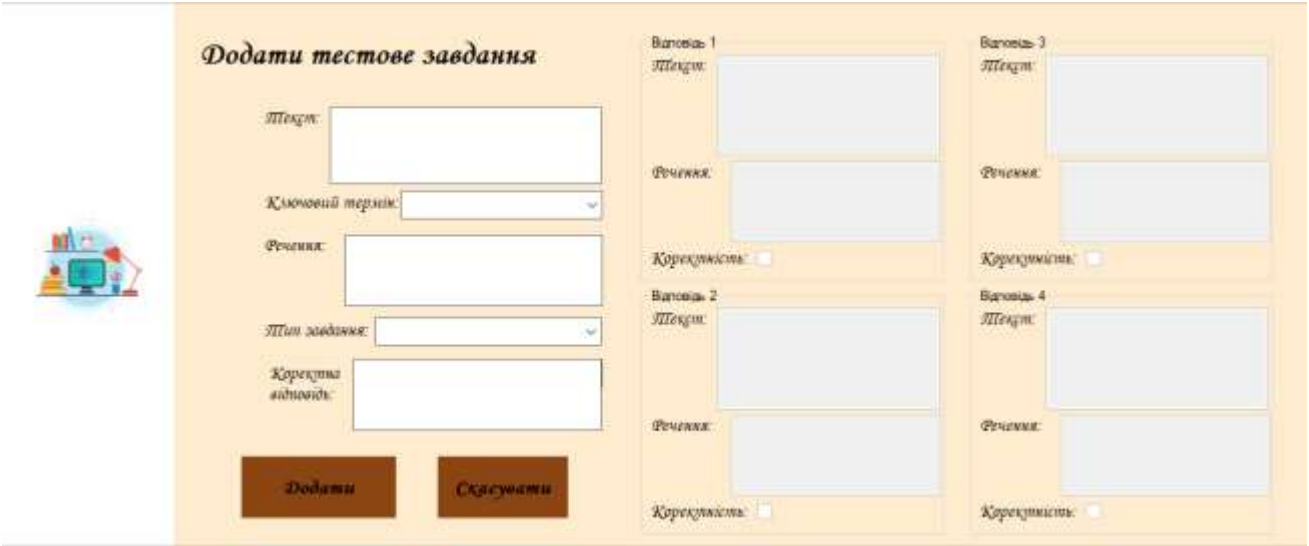


Рисунок Ж.12 – Вікно «Додати тестове завдання»

Таблиця Ж.8 – Тест-кейс AT0016

<table>
<tr><td>Тест-кейс ID: AT0016</td><td>Пріоритет: 1</td><td>Створено: 08.11.2020, Г.Білоус</td></tr>
<tr><td colspan="3">Назва: Перевірка функціоналу створення нового тестового завдання
Вхідні дані: Логін = «chong», Пароль = «chong»</td></tr>
<tr><td colspan="2">Кроки</td><td>Очікуваний результат</td></tr>
<tr><td colspan="2">1. Запустити додаток
2. Ввести значення у поле логіну із вхідних даних
3. Натиснути кнопку «Вхід»
4. Порівняти фактичний результат з очікуваним
5. Натиснути кнопку «Перейти до роботи із вибірками тестових завдань»
6. Перейти на вкладку «Тестові завдання»
7. Натиснути кнопку «Додати тестове завдання»
8. Порівняти фактичний результат з очікуваним
9. Ввести довільні дані про нове тестове завдання у форму
10. Натиснути кнопку «Додати»
11. Порівняти фактичний результат з очікуваним</td><td>Вікно входу у систему закрилось. Вікно з інтерфейсом викладача відкрито.

Відкрито вікно «Додати тестове завдання».

Вікно «Додати тестове завдання» закрито. У таблиці «Тестові завдання» з’явився новий рядок, що містить дані введені на кроці 9.</td></tr>
<tr><td colspan="3">Результат виконання тест-кейсу: пройдено успішно</td></tr>
</table>

Після заповнення полей на формі та натисканні на кнопку «Додати», вікно закривається, а у таблиці із тестовими завданнями з’являться новий рядок, що містить дані введені на формі вікна «Додати тестове завдання».

Дев’ятий тест-кейс (таблиця Ж.9) перевіряє функціонал редагування існуючого тестового завдання. При настисканні на кнопку «Відредагувати тестове завдання», відкривається вікно «Відредагувати тестове завдання», форма якого містить дані про тестове завдання, що обране попередньо у таблиці (рисунок Ж.13).

Таблиця Ж.9 – Тест-кейс AT0017

Тест-кейс ID: AT0017	**Пріоритет:** 2	**Створено:** 08.11.2020, Г.Білоус
Назва: Перевірка функціоналу редагування існуючого тестового завдання **Вхідні дані:** Логін = «chong», Пароль = «chong»		
Кроки		**Очікуваний результат**
1. Запустити додаток 2. Ввести значення у поле логіну та паролю із вхідних даних 3. Натиснути кнопку «Вхід» 4. Порівняти фактичний результат з очікуваним 5. Натиснути кнопку «Перейти до роботи із вибірками тестових завдань» 6. Перейти на вкладку «Тестові завдання» 7. Обрати перший рядок у таблиці «Тестові завдання» 8. Натиснути кнопку «Відредагувати тестове завдання» 9. Порівняти фактичний результат з очікуваним 10. Змінити ключовий термін 11. Натиснути кнопку «Відредагувати» 12. Порівняти фактичний результат з очікуваним		Вікно входу у систему закрито. Вікно з інтерфейсом викладача відкрито. Відкрито вікно «Відредагувати тестове завдання». Вікно «Відредагувати тестове завдання» закрито. У таблиці «Вибірки тестових завдань» ключовий термін завдання у першому рядку, змінено на ключовий термін введений на кроці 10.
Результат виконання тест-кейсу: пройдено успішно		

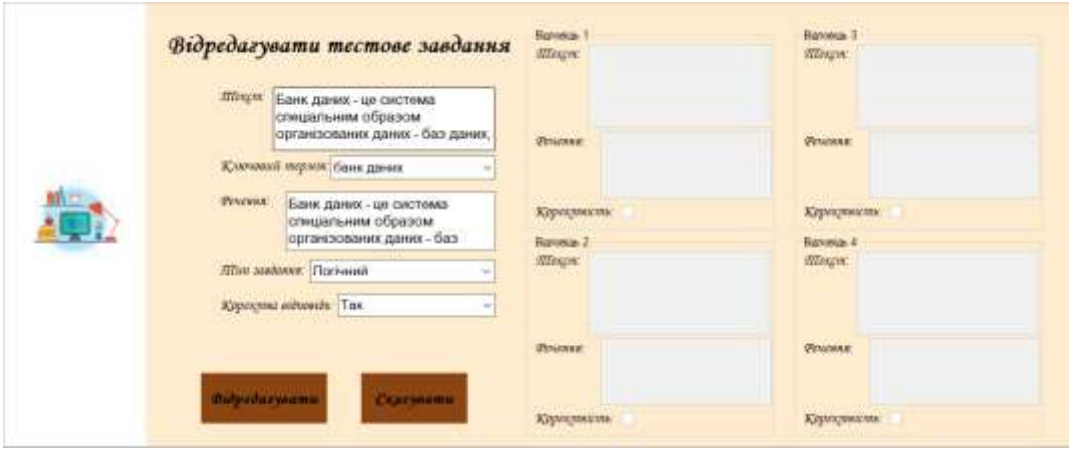


Рисунок Ж.13 – Вікно «Відредагувати тестове завдання»

Після редагування полей на формі та натисканні на кнопку «Відредагувати», вікно закриється, а у таблиці із тестовими завданнями оновляться дані у рядку, що відповідний завданню, яке редагувалося.

Отже, дослідження функціональності підсистеми викладача розробленої інформаційної системи адаптивного тестування рівня знань було успішно пройдено. Наочно доведено, що функції користувача викладач вказані у п.3.1 було реалізовано у інформаційній системі.

Додаток З
Ксерокопії наукових публікацій, виконаних при роботі над дипломною роботою магістра

Перелік наукових публікацій:

1. Білоус Г. А. Розбиття 3D-об’єктів на тетраедри із заданим ступенем дискретності/ Г. А. Білоус, Т. К. Скрипник, Н. К. Медведчук // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2019, №3. – С.59-62.
2. Білоус Г. А., Параметризація моделі тестового завдання при автоматизованому формуванні тестів / Г. А. Білоус, О. В. Мазурець // Матеріали VII Міжнародної науково-практичної конференції «Інформаційні управляючі системи та технології ICST-ODESSA-2018». Одеса – 2018. – С.64-66.
3. Білоус Г. А. Метод формального опису елементів моделей автоматизованого формування тестових завдань / О. В. Мазурець, О. В. Ковальчук, В. О. Слободзян, Г. А. Білоус // Збірник наукових праць за матеріалами Всеукраїнської науково-практичної конференції «Інтелектуальний потенціал – 2018». Хмельницький, 2018, Ч.1. – С.51-56.
4. Білоус Г. А. Використання програмного розширення spire.doc для автоматизації роботи з цифровими документами / О. В. Ковальчук, Г. А. Білоус, В. О. Слободзян // Збірник наукових праць за матеріалами XI всеукраїнської науково-практичної конференції «Актуальні проблеми комп’ютерних наук АПКН-2019» – Хмельницький, 2019, Т.1. – С.116-122.
5. Білоус Г. А. Інформаційна технологія адаптивного тестування рівня знань / Г. А. Білоус, О. В. Мазурець // Збірник наукових праць за матеріалами XII всеукраїнської науково-практичної конференції «Актуальні проблеми комп’ютерних наук АПКН-2020» – Хмельницький, 2020, Т.1. – С.33-41.

ISSN 2307-5732
DOI 10.31891/2307-5732

НАУКОВИЙ ЖУРНАЛ **3.2019**

ВІСНИК

Хмельницького національного університету

Технічні науки

Technical sciences

SCIENTIFIC JOURNAL
HERALD OF KHMELNYTSKYI NATIONAL UNIVERSITY

2019, Issue 3, Volume 273

Хмельницький

ВІСНИК
ХМЕЛЬНИЦЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ
серія: Технічні науки

Затверджений як фахове видання (перереєстрація)
Наказ МОН 04.07.2014 №793

Засновано в липні 1997 р. ***Виходить 6 разів на рік***

Хмельницький, 2019, № 3 (273)

Засновник і видавець: Хмельницький національний університет
(до 2005 р. – Технологічний університет Поділля, м. Хмельницький)

Включено до науково-метричних баз:

Google Scholar	http://scholar.google.com.ua/citations?hl=uk&user=alUP9OYAAAAJ
Index Copernicus	http://jml2012.indexcopernicus.com/passport.php?id=4538&id_lang=3
РИНЦ	http://elibrary.ru/title_about.asp?id=37650
Polish Scholarly Bibliography	https://pbn.nauka.gov.pl/journals/46221

Рекомендовано до друку рішенням вченої ради Хмельницького національного університету, протокол № 11 від 31.05.2019 р.

Адреса редакції: редакція журналу "Вісник Хмельницького національного університету"
Хмельницький національний університет
вул. Інститутська, 11, м. Хмельницький, Україна, 29016

☎ (038-2) 67-51-08 **web:** http://journals.khnu.km.ua/vestnik
e-mail: visnyk.khnu@gmail.com http://vestnik.ho.com.ua
http://lib.khnu.km.ua/visnyk_tup.htm

Зареєстровано Міністерством України у справах преси та інформації.
Свідоцтво про державну реєстрацію друкованого засобу масової інформації
Серія КВ № 9722 від 29 березня 2005 року

ЗМІСТ

МАШИНОЗНАВСТВО ТА ОБРОБКА МАТЕРІАЛІВ В МАШИНОБУДУВАННІ

ТЕХНОЛОГІЇ ЛЕГКОЇ ПРОМИСЛОВОСТІ

РАДІОТЕХНІКА, ЕЛЕКТРОНІКА ТА ТЕЛЕКОМУНІКАЦІЇ

ОБМІН ПРАКТИЧНИМ ДОСВІДОМ, ТЕХНОЛОГІЯМИ ТА ОБГОВОРЕННЯ

DOI 10.31891/2307-5732-2019-273-3-59-62
УДК 004.9

Г.А. БІЛОУС, Т.К. СКРИПНИК, Н.К. МЕДВЕДЧУК
Хмельницький національний університет

РОЗБИТТЯ 3D-ОБ'ЄКТІВ НА ТЕТРАЕДРИ ІЗ ЗАДАНИМ СТУПЕНЕМ ДИСКРЕТНОСТІ

В роботі визначається проблема із методами тріангулювання 3D-об'єктів та пропонується інформаційна технологія розбиття із заданим ступенем дискретності контурного 3D-об'єкта на тетраедри, особливістю якої є робота з STL-файлами тривимірних об'єктів як вхідних та вихідних даних, що грунтується на алгоритмі масштабування заданої неструктурованої розрахункової трикутної сітки.

Ключові слова: тріангуляція, STL-формат, тетраедри, 3D-об'єкт, метод скінченних елементів, дискретність контурної моделі.

H.A. BILOUS, T.K. SKRYPNYK, N.K. MEDVEDCHUK
Khmelnytskyi National University

FRAGMENTATION OF 3D-OBJECTS ON TETRAHEDRONS WITH TARGETED DEGREE OF DISCRETENESS

Abstract - Three-dimensional geometry in leading 3D CAD systems is described by surfaces of the high order, and when being triangulated the surface of the model is divided into small triangles - facets. Currently, large number of research and commercial software packages are developed on the basis of one or another iterative method that implement the building of grids in automatic mode. However, only some software makes it based on STL-files, and even more rarely similar software packages support two types of STL-files. That's why the work is aimed to develop the information technology of the distribution with a given degree of discreteness of the contour 3D model STL into the tetrahedron. To confirm the effectiveness of information technology, an experimental software product was created that is used to generate a tetrahedron's grid based on data from STL-files. Within the framework of the proposed information technology, an algorithm for scaling a given unstructured calculated triangular grid has been developed.

Keywords: triangulation, STL-format, tetrahedron, 3D-object, finite element method, discreteness of contour model

Вступ

Для вирішення широкого кола задач моделювання, візуалізації та проектування виробів у сфері механічної інженерії часто застосовується метод скінченних елементів. Метод скінченних елементів є інженерним аналізом, що полягає в апроксимації суцільного середовища з нескінченно великими числами ступенів свободи сукупністю елементів, що мають скінченне число ступенів свободи, й між цими елементами встановлюється взаємозв'язок [1].

При використанні методу кінцевих елементів у інформаційних технологіях є проблема роботи із файлами STL-формату. STL (StereoLithography) є «мозаїчним» форматом, в якому для представлення форми цифрової 3D-моделі використовується послідовність трикутників (фасетів). STL-формат використовується в сфері прототипування, а саме в стереолітографії, у ньому міститься інформація, що застосовується в розробці різних об'ємних деталей, які можна навіть роздрукувати на 3D-принтері

Тривимірна геометрія в провідних 3D CAD-системах описується поверхнями високого порядку, а при тріангуляції поверхня моделі розбивається на маленькі трикутники – фасети. Кожен фасет описується чотирма наборами даних: координати X, Y, Z кожної з трьох вершин і нормальний вектор, який описує орієнтацію фасета, вказуючи назовні моделі [2].

Тріангуляцією тривимірного об'єкта є його розбиття на тетраедри, що розташовуються один біля одного. Існує два класи методів тривимірної тріангуляції: прямі та ітераційні. Ітераційні мають достатню універсальність і тому, на відміну від прямих, можуть бути використані для тріангуляції об'єктів довільного вигляду, проте вони характеризуються споживанням ресурсів і більш трудомісткою реалізацією методу в конкретному алгоритмі.

Сітки, побудовані ітераційними методами, як правило, неструктуровані й неоднорідні. Неструктурованість обумовлена тим, що топологія сітки формується в процесі побудови, і тому може варіюватися навіть в межах однієї підобласті. З цієї ж причини однорідність може виникнути тільки випадково. Оскільки перед побудовою сітки нічого не можна сказати про її майбутню структуру, не можна гарантувати і її якості. Часто побудовану сітку можна істотно поліпшити за допомогою одного з численних методів оптимізації. Цією можливістю зазвичай не нехтують, благо що час, що витрачається на оптимізацію, як правило, істотно менше часу, що витрачається на побудову [3].

В даний час розроблено велику кількість дослідницьких та комерційних програмних пакетів на основі того чи іншого ітераційного методу, що реалізують побудову сіток (частково або повністю) в автоматичному режимі [4]. Більшість з них грунтуються на використанні критерію Делоне. Тобто, трикутна сітка на площині відповідає критерію Делоне, якщо всередину кола, описаного навколо будь-якого трикутника, не потрапляють ніякі інші вузли цієї сітки. Проте лише деяке програмне забезпечення, що реалізує 3D-тріангуляцію, робить це на основі файлів STL-формату. Ще рідше подібні програмні пакети підтримують два типи STL-файлів, дані у яких можуть зберігатися як в текстовому ASCII-форматі, так і в двійковому вигляді (бінарний формат), що забезпечує більшу швидкодію.

Тому ***метою роботи*** є розробка інформаційної технології розбиття із заданим ступенем

дискретності контурної 3D-моделі STL на тетраедри.

Основна частина

Загальну схему інформаційної технології для створення сітки тетраедрів із даних файлу STL-формату представлено на рисунку 1.

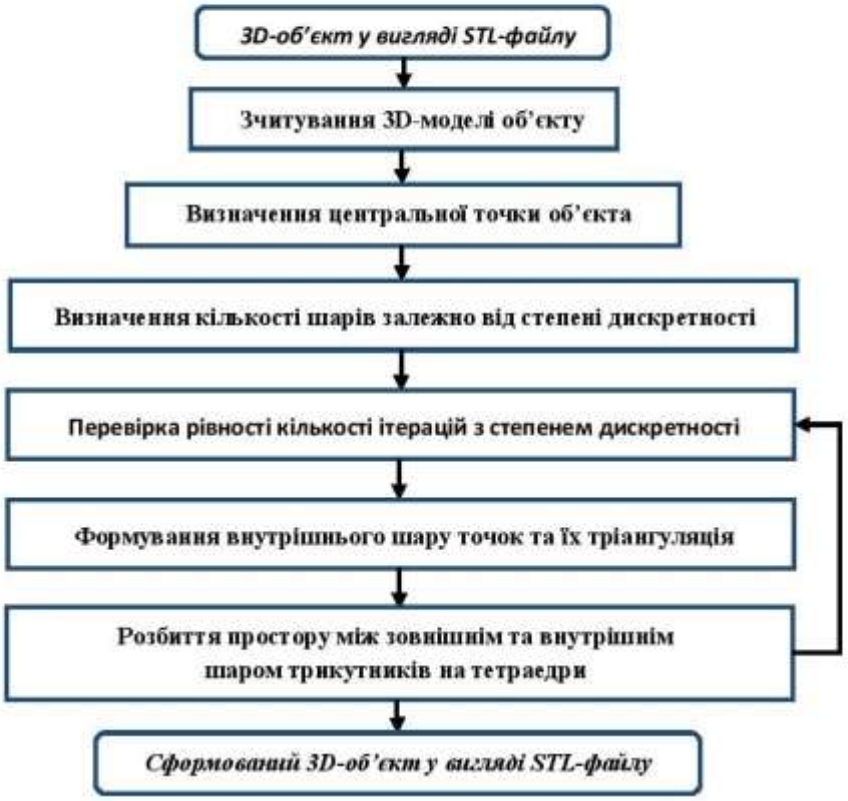


Рис. 1. Схема інформаційної технології розбиття контурної 3D-моделі STL на тетраедри

В рамках запропонованої інформаційної технології, спочатку вводяться вхідні дані через командний рядок: ім'я вхідного файлу в STL-форматі, тип STL-файлу, ім'я вихідного STL-файлу, що містить сітку тетраедрів, та параметри розбиття (ступінь дискретності). Далі зчитується 3D-об'єкт, який розбивається на тетраедри із заданим ступенем дискретності. Це відбувається за алгоритмом масштабування заданої неструктурованої розрахункової трикутної сітки. Для цього спочатку знаходяться координати точки центру фігури, в яку буде переноситися центр системи координат. Формується внутрішній шар точок, який тріангулюється і таким чином будується внутрішній тріангульований шар. Простір між зовнішнім і внутрішнім шаром розбивається на тетраедри. В результаті поєднання зовнішнього й внутрішнього шарів трикутників формуються трикутні призми, які розбиваються на тетраедри. На останній ітерації крайній внутрішній шар з'єднується з центром фігури, створюючи таким чином шар із тетраедрів.

Вихідний файлу зберігається в ASCII-форматі з заданим ім'ям файлу. STL-файли з вхідними даними і результатами створення сітки тетраедрів можна переглядати за допомогою 3D-редакторів з підтримкою STL «MeshLab» або «3D Viewer for Google Chrome».

Для підтвердження ефективності інформаційної технології було створено експериментальний програмний продукт, який застосовується для генерування сітки тетраедрів за даними файлів STL-формату. Даний програмний продукт базується на запропонованій інформаційній технології та алгоритму масштабування заданої неструктурованої розрахункової трикутної сітки, яка зчитується з вхідного STL-файлу. Запропонований алгоритм дозволяє розбити 3D-об'єкт, заданий замкнутим набором трикутних граней на тетраедри (кінцева сітка).

Розроблений додаток складається з наступних модулів (рисунок 2):

1. Модуль інтерфейсу користувача. Даний модуль реалізується у класі Program.

2. Модуль інтерфейсів додатку. Модуль містить у собі інтерфейси класів модуля роботи з STL-файлами та класу модуля розрахунків:

- IstlReader – інтерфейс, який визначає загальні властивості методів читання STL-файлів в залежності від типу файлу (бінарний або ASCII);
- IstlWriter – інтерфейс, що визначає клас запису в файл STL-формату (ASCII-тип);
- ItetrahedralMeshes – інтерфейс, який визначає алгоритм масштабування заданої неструктурованої розрахункової трикутної сітки і розбивки фігури на тетраедри.

3. Модуль структур. Даний модуль відображено у наступних структурах даних:

- StlVector – описує точку з координатами X, Y, Z;
- StlTriangle – описує структуру трикутника, що складається з трьох вершин і вектора нормалі.

4. Модуль роботи з STL-файлами. Реалізує читання даних із двох типів файлів даного формату та запис вихідних даних:

- StlReader – клас реалізує інтерфейс IstlReader. Містить два методи читання файлів STL-формату і метод визначення імені користувача;
- StlWriter – клас реалізує інтерфейс IstlWriter. Містить метод запису файлів STL-формату.

5. Модуль розрахунків. Представлений класом TetrahedralMeshes, який реалізує інтерфейс ItetrahedralMeshes. Клас містить методи реалізації алгоритму масштабування заданої фігури у вигляді сітки трикутників і розбивки фігури на тетраедри.

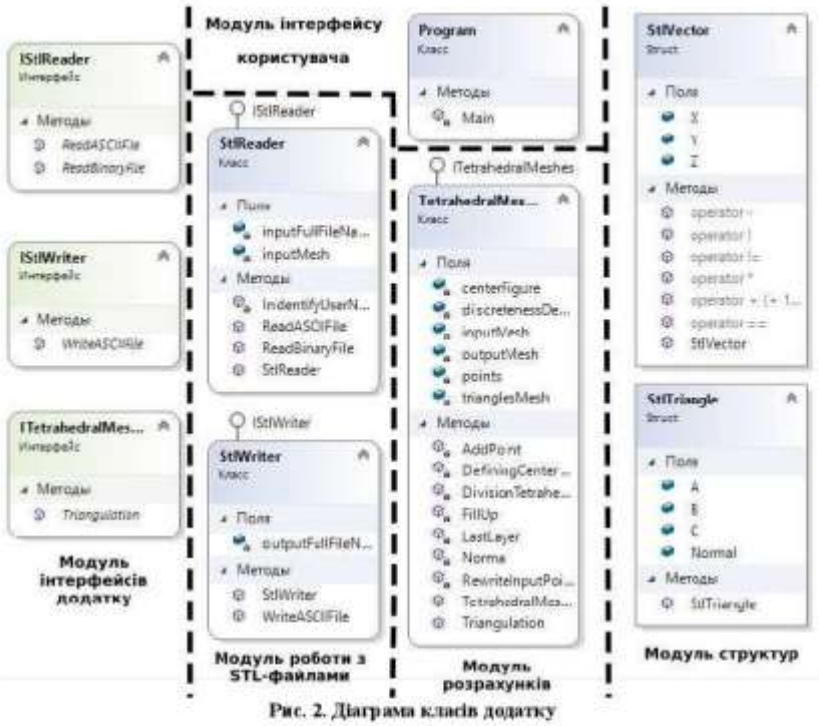


Рис. 2. Діаграма класів додатку

Інтерфейсом користувача для роботи з додатком є консоль. Для запуску побудови сітки необхідно ввести такі параметри як: STL-файл для зчитування поверхні, тип STL-файлу (бінарний або ASCII), назва файлу для запису створеної сітки, ступінь дискретності.

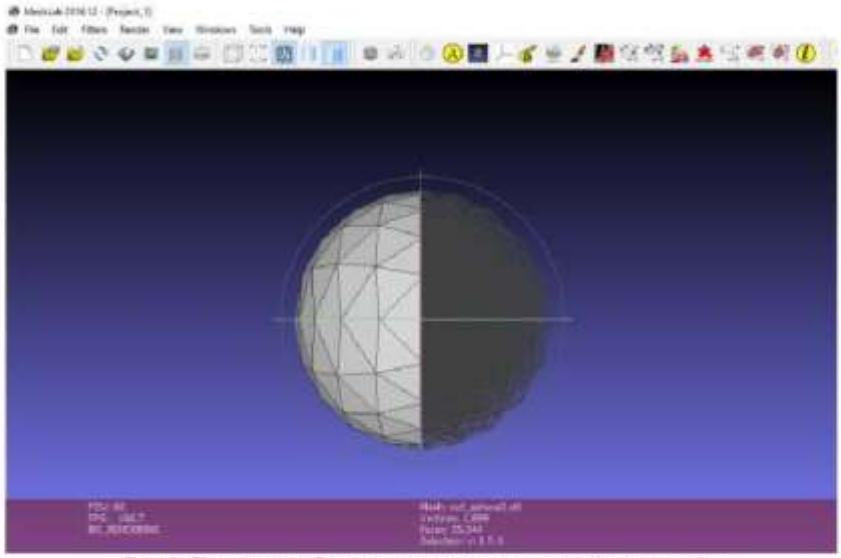


Рис. 3. Результат роботи додатку з генерування сітки тетраедрів

Результатом виконання програми є STL-файл, який містить у собі сітку трикутників, що формують

тетраедри (рисунок 3).

Висновки

Аналіз використання методу кінцевих елементів у інформаційних технологіях визначив, що існує проблема роботи із файлами STL-формату, що містять у собі моделі 3D-об'єктів. Тому запропоновано інформаційну технологію розбиття із заданим ступенем дискретності контурного 3D-об'єкта на тетраедри, особливістю якої є саме робота з STL-файлами тривимірних об'єктів як вхідних та вихідних даних. В рамках запропонованої інформаційної технології, було розроблено алгоритм масштабування заданої неструктурованої розрахункової трикутної сітки.

Розглянуто роботу програмного продукту на основі даної інформаційної технології, його алгоритмічну складову та структуру. Встановлено, що даний програмний продукт дійсно дозволяє формувати сітку тетраедрів для трьохвимірних геометричних тіл обертання, що підтверджує ефективність та функціональність запропонованої інформаційної технології. Подальші дослідження спрямовані на покращення алгоритму задля тріангулювання складних 3D-об'єктів, що містять у собі заглибини та випуклості різної складності: тора, моделей автомобілів, будівель, тощо.

Література

1. Зенкевич О. К. Метод скінченних елементів у техніці / О. К. Зенкевич // Науковий журнал «Світ», Москва, 1975, №6. – С.541.

2. Що таке формат файлу STL. [Електронний ресурс]. – Режим доступу: https://3dnetprint.com/blog/stl-file-what-is-it

3. Development and Implementation of Algorithms for Constrained Volume Triangulations: Iterative Algorithms [Електронний ресурс]. – Режим доступу: http://keldysh.ru/papers/2006/prep09/prep2006_09.html

4. A Quality Tetrahedral Mesh Generator and a 3D Delaunay Triangulator [Електронний ресурс]. – Режим доступу: http://wias-berlin.de/software/tetgen/

References

1. Zenkevych O. K. Finite Element Method in Engineering / O. K. Zenkevych // Naukovyy zhurnal „World", 1975, №6. – P.541.

2. What is the format of the STL file: https://3dnetprint.com/blog/stl-file-what-is-it

3. Development and Implementation of Algorithms for Constrained Volume Triangulations: Iterative Algorithms: http://keldysh.ru/papers/2006/prep09/prep2006_09.html

4. A Quality Tetrahedral Mesh Generator and a 3D Delaunay Triangulator: http://wias-berlin.de/software/tetgen/

Рецензія/Peer review : 16.5.2019 р. Надрукована/Printed : 2.6.2019 р.
Рецензент: д.т.н., проф. Сорокатий Р.В.

Odessa National Polytechnic University
Odessa National Marine University
Kharkov National University of Radio Electronics
Admiral S.O. Makarov National University of Shipbuilding
Lodz University of Technology

MATERIALS OF THE VII INTERNATIONAL SCIENTIFIC- PRACTICAL CONFERENCE

«Information Control Systems and Technologies»
17th – 18th September, 2018

Odessa
2018

СОДЕРЖАНИЕ

Materials of the VII International Scientific Conference
«Information-Management Systems and Technologies»
17th – 18th September, 2018, Odessa

Materials of the VII International Scientific Conference
«Information-Management Systems and Technologies»
17th – 18th September, 2018, Odessa

Materials of the VII International Scientific Conference
«Information-Management Systems and Technologies»
17th – 18th September, 2018, Odessa

Materials of the VII International Scientific Conference
«Information-Management Systems and Technologies»
17th – 18th September, 2018, Odessa

УДК 004.91

Білоус Г.А., Мазурець О.В.

ПАРАМЕТРИЗАЦІЯ МОДЕЛІ ТЕСТОВОГО ЗАВДАННЯ ПРИ АВТОМАТИЗОВАНОМУ ФОРМУВАННІ ТЕСТІВ

Bilous H.A., Mazurets A.V.

PARAMETRIZATION OF THE MODEL OF TEST TASK AT AUTOMATIZED TEST FORMATION

Контроль знань є важливою частиною процесу навчання і дозволяє отримати об'єктивну оцінку рівня знань студентів. Однією з форм контролю знань, що добре себе зарекомендувала, є тестування. Тестові технології, які застосовуються в системі вищої професійної освіти, покликані забезпечувати отримання оперативної та достовірної інформації про якість навчальних досягнень студентів. Тестова перевірка включає в себе набір тестових завдань із різними параметрами, що робить результат тестування більш об'єктивним. Інформаційні технології на сучасному етапі широко використовуються для забезпечення комп'ютерного тестування рівня знань. Зокрема, функція тестування реалізована в відомих системах дистанційного навчання: ATutor, Claroline LMS, Dokeos, eFront, ILIAS, Moodle, OLAT, Open Elms, OpenACS, Sakai, TrainingWare Class, WebTutor тощо. Найбільш широко на сучасному етапі використовується середовище Moodle – безкоштовна, відкрита система управління навчанням, що реалізує взаємодію між викладачами та учнями через мережу Інтернет. В рамках розробки інформаційної технології гнучкого тестування рівня знань [1], яка забезпечує формування репрезентативних наборів тестових завдань та адаптивно обирає тестові завдання в процесі тестування, вирішується проблема рівномірного використання тестових завдань в кожній окремій вибірці тесту за рядом параметрів. При вирішенні цієї проблеми є актуальною задача визначення параметрів тестів, значення яких слід

автоматизовано враховувати при формуванні збалансованих вибірок тестів. Частину актуальних для задачі параметрів тестів визначає безпосередньо середовище Moodle.

До них належать: тип питання, бал за замовчуванням і кількість правильних відповідей. Тип питання у середовищі Moodle є первинним класифікатором тестових завдань. Передбачено питання множинного вибору, логічного вибору (Так/Ні), відповідності, короткої відповіді, числової відповіді, есе та вбудованої відповіді. Параметр типу питання впливає на логічну однорідність структури тесту. Бал за замовчуванням є оцінкою за питання у разі правильної відповіді й штрафним балом при неправильній відповіді, параметр істотно впливає на результуючу оцінку й як правило визначає складність тестового завдання. Кількість правильних відповідей може варіюватися в залежності від типу завдання й виділяється одна або кілька правильних відповідей. Цей параметр впливає на логічну однорідність структури тесту та складність тестового завдання. Іншу частину актуальних для задачі параметрів тестів визначає його семантичне ядро. Семантичне ядро тесту пропорційно якості тесту в максимально можливій мірі відповідає семантичному ядру навчальних матеріалів, по яких проводиться тестування.

До параметрів тестового завдання, що стосуються семантичного ядра тесту, належать важливість терміна та кількість використаних термінів.

Важливість терміна, засвоєння сенсу якого пріоритетно перевіряється – визначається шляхом ручного чи автоматизованого семантичного аналізу контенту відповідних навчальних матеріалів [2].

Кількість термінів, засвоєння яких перевіряється – визначається кількістю термінів із числа ключових для відповідних навчальних матеріалів, що використані при композиції тестового завдання.

Врахування значень наведених параметрів тестових завдань при формуванні збалансованих вибірок тестів дозволить досягти високої якості автоматизованого формування репрезентативних наборів тестових завдань.

За допомогою такого конструювання тесту можна забезпечити відповідний рівень дискримінативності та репрезентативності наборів

тестових завдань, що формуються в системах тестування сучасних навчальних середовищ.

Література

1. Бармак О. В., Мазурець О. В., Матвійчук А. О. Застосування інформаційної технології гнучкого тестування рівня знань у середовищі Moodle / О. В. Бармак, О. В. Мазурець, А. О. Матвійчук // Науковий журнал „Вісник Хмельницького національного університету" серія: Технічні науки. Хмельницький, 2017.- №3. – С.103-115.

2. Мазурець О. В. Інформаційна технологія автоматизованого визначення семантичних термінів в елементах навчальних матеріалів / О. В. Мазурець // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2018.- №3.–С.223-230.

УДК 519.6

Косенко Е. Д.

ПРОБЛЕМЫ РАЗВИТИЯ ОБЛАЧНЫХ ТЕХНОЛОГИЙ В УКРАИНЕ

Kosenko E.

PROBLEMS OF DEVELOPMENT OF CLOUD CALCULATIONS IN UKRAINE

В последние десять лет облачные технологии, или вычисления (англ . Cloud Computing) стали одним из основных трендов развития IT-технологий. Эффективность облачных технологий стала практически общепризнанной, в странах ЕС и США проработаны юридические аспекты работы облачных систем и созданы новые экономические модели использования IT-услуг. Активно внедряют облачные технологии в свои разработки такие компании, как Microsoft, Apple, Google, Yahoo,

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Хмельницький національний університет

Військовий інститут Київського національного університету ім.Тараса Шевченка

ПВНЗ "Університет економіки і підприємництва"

Тернопільський інститут агропромислового виробництва

Інтелектуальний потенціал - 2018

збірник наукових праць молодих науковців і студентів

Присвячується 30-річчю підготовки ІТ- фахівців в Хмельницькому національному університеті

сформовано за матеріалами
Всеукраїнської науково-практичної конференції
молодих науковців і студентів «Інтелектуальний потенціал – 2018»

14-16 листопада 2018р.

Частина 1
Комп'ютерні науки та інформаційні технології проектування

Хмельницький
2018

ББК 74.480.278
С.88

«Інтелектуальний потенціал – 2018» - збірник наукових праць молодих науковців і студентів з нагоди 30-річчя підготовки ІТ- фахівців в ХНУ/Колектив авторів – Хмельницький: ПВНЗ УЕП, 2018. – Ч.1: Комп'ютерні науки та інформаційні технології проектування. – 132 с.

ЗМІСТ

Метод формального опису елементів моделей автоматизованого формування тестових завдань

Мазурець О.В., Ковальчук О.В., Слободзян В.О., Білоус Г.А.
Хмельницький національний університет

Одним із основних способів контролю знань в навчальних інформаційних системах є комп'ютерне тестування, яке крім контрольної функції застосовується для навчання, тренінгу, розвитку здібностей. Інформаційні технології дають можливість суттєво зменшити трудові затрати на створення тестових завдань з можливістю їх постійного оновлення, що формує актуальний напрямок наукових досліджень.

Інформаційна технологія автоматизованого формування тестових завдань на основі контенту навчальних матеріалів [1] дозволяє на основі вхідних даних у вигляді контенту файлу формату .docx навчальних матеріалів або його визначеної частини автоматизовано формувати вихідні дані у вигляді файлу з тестами для імпорту у віртуальне навчальне середовище. Розв'язок задачі автоматизації формування тестових завдань містить ряд послідовних етапів перетворення інформації, до яких відносяться визначення семантичних термінів у контенті навчальних матеріалів [2], формування структури навчальних матеріалів [3], безпосередньо автоматизоване формування тестових завдань [1] й перенесення результатів у область взаємодії з кінцевим користувачем – підсистему тестування середовища Moodle [4].

Характерною рисою автоматизованого формування тестових завдань за даною інформаційною технологією є використання набору моделей тестових завдань для перетворення фрагментів контенту у тестові завдання.

Модель тестового завдання являє собою структуру, що складається з наступних функціональних елементів:

– маска для фрагменту тексту з поняттям, що призначена для ідентифікації фрагментів контенту із заданим терміном, до яких воно може бути застосовано;

– параметри моделі тестового завдання, що визначають особливості й ефективність його застосування: тип запитання, що може бути створено за цією моделлю; базова оцінка очікуваної якості застосування моделі у діапазоні (0;1), що одержується методом експертної оцінки результатів; набір правил корекції базової оцінки цього правила конвертації, що в залежності від вказаних особливостей фрагменту можуть знизити базову оцінку, при активації виступаючи множниками у діапазоні (0;1);

– маска для формування тестового завдання, що містить алгоритм перетворення даного фрагменту у елементи тестового завдання.

Тобто модель у даному випадку представляє собою структуру, в яку входять: набір з'єднувачів (слів або символів які вирізняють певний фрагмент тексту як релевантний відносно терміну); правило конвертації фрагментів з тексту у тестове завдання, та дані про словоформу

(відмінок/рід). На вхід моделі завжди дається певна зв'язка [термін – слово маркер – теза], за якою здійснюється формування тестового завдання та підбір варіантів відповідей.

При використанні даного підходу спершу встановлюються вимоги до набору тестових завдань, визначаються актуальні моделі тестових завдань й на основі найбільш прийнятних із них формується набір тестових завдань. В результаті за кожною з обраних моделей тестових завдань формується одне тестове запитання у форматі .gift, що може бути конвертоване у середовище Moodle. Кінцевий результат зберігається в одному файлі тесту.

Таким чином, при формуванні тестових завдань різних типів використовуються моделі, наділені параметрами та критеріями, характерними лише для конкретного типу питання та будови терміну. В межах розглянутої інформаційної технології, з метою формування моделей тестових завдань є необхідною розробка набору тегів для формального опису елементів правил конвертації.

Метою роботи є розробка множини тегів як методу формального опису елементів моделей автоматизованого формування тестових завдань. Це дозволить проводити як зручне й ефективне створення нових чи редагування існуючих моделей формування тестових завдань, так і компактне зберігання розроблених моделей для подальшого використання.

Відповідно до розглянутих складових моделей формування тестових завдань, виділяються теги для ідентифікації елементів контенту та теги для формування тестів. Теги для ідентифікації елементів контенту використовуються в масках ідентифікації й наведені в таблиці 1.

Таблиця 1 – Теги для ідентифікації елементів контенту

Тег	Опис
[TermGroup]	Фрагмент тексту, який являє собою термін, що складається з набору слів
[ThesisGroup]	Фрагмент тексту (теза), який дає визначення терміну
[RandomTermGroup]	Випадково обраний інший термін
[RandomThesisGroup]	Випадково обране визначення іншого терміну
[Connector]	Слово або символ із тексту, що поєднує термін з тезою (– , – це, є, називається, тощо)
[BeginSentence]	Фрагмент тексту від початку речення до TermGroup або ThesisGroup (може бути null)
[Inflexion]	Тег повертає нормальну форму наступного елементу
[Caps]	Переведення першої букви до верхнього регістру
[ReCaps]	Переведення першої букви до нижнього регістру

Нижче наведений приклад для ідентифікації двох моделей з істинним твердженням, базової (Basic) та реверсивної (Reversed):

```
[Caps][TermGroup][connector][ThesisGroup]. {TRUE}
[Caps][ThesisGroup][connector][TermGroup]. {TRUE}
```

Теги для формування тестів використовуються в масках формування й наведені у таблиці 2.

Таблиця 2 – Теги для формування тестів

Тег	Опис
{Header}	Візуальний визначник для тексту питання
{Answer}	Візуальний визначник для варіанту відповіді
{FALSE}	Вказівка на неправильний варіант відповіді
{TRUE}	Вказівка на правильний варіант відповіді

Нижче зображений приклад моделі формування тестового завдання одиничного вибору, де в якості тексту питання (Header) вказана теза, а у варіантах відповідей (Answer 1 – N) назви термінів:

```
Header -> [Caps][ThesisGroup][“-, - це”]
Answer 1 -> [TermGroup]              {TRUE}
Answer 2 -> [RandomTermGroup]        {FALSE}
Answer 3 -> [RandomTermGroup]        {FALSE}
Answer N -> [RandomTermGroup]        {FALSE}
```

Наведені теги можна віднести до елементів псевдокоду, або змінних, які під час роботи алгоритму мають різні значення, а іноді можуть не мають жодних (null). У випадках, коли тег може повернути порожнє значення, він вказується з додаванням знака дорівнює і нуля, наступним чином: [Tag = 0].

Наведений підхід дозволяє відкрите програмування алгоритму роботи різноманітних моделей генерування тестових завдань, що визначає множину формалізованих таким чином моделей як базу знань відповідної інформаційної системи для автоматизованого формування тестових завдань за навчальними матеріалами.

Для прикладу наведено процес формування тестового завдання типу «із введенням тексту» з використанням однієї з моделей формування тестових завдань для відповідного типу завдань.

Моделі формування завдань із введенням тексту полягають у формулюванні одного твердження з відсутнім ключовим словом та формуванні можливих варіантів відповідей, які не пропонуються користувачеві, а використовуються лише для перевірки введеного тексту. Відповідно, можлива модель, мета якої – забезпечити максимальну кількість правильних відповідей для коректної перевірки тексту, введеного користувачем.

При формуванні маски ідентифікації, за основу твердження береться певна пара [термін – теза], причому важливо, щоб термін був у початковій формі (називному відмінку однини), щоб виключити можливість ігнорування правильних відповідей через несхожість у закінченнях або словоформах. Далі відбувається формування набору правильних відповідей. Необхідним і

Відповідний сформований фрагмент GIFT-файлу, який сформований за даною моделлю формування тестового завдання типу «із введенням тексту» із відкритою відповіддю, наступний:

```
// question
Множина однотипних даних, що утворюють одну з граней гіперкуба це -
{
=%100% Вимірювання#
=%100% Dimension#
}
```

Отже, розроблені теги для моделей формування тестових завдань дозволяють формально описати процес формування тестових завдань із рахуванням всіх особливостей та параметрів, й забезпечити автоматизацію імпорту доступних тестових завдань у середовищі Moodle. Тег формального опису моделей є елементом псевдокоду, який призначений для формального опису структури моделі, її вхідних та вихідних параметрів.

За умови достатньої відповідності семантичним та структурним вимогам і коректного співвідношення між обсягом контенту навчального матеріалу та параметрів генерації набору тестових завдань, одержується репрезентативний тест, що може бути використаний як безпосередньо для тестування, так і як сировина для подальшої роботи розробника тестів.

Література

1. Бармак О. В., Мазурець О. В., Кліменко В. І. Інформаційна технологія автоматизованого формування тестових завдань / О. В. Бармак, О. В. Мазурець, В. І. Кліменко // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2017, №5. – С.93-103.
2. Мазурець О. В. Інформаційна технологія автоматизованого визначення семантичних термінів в елементах навчальних матеріалів / О. В. Мазурець // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2018, №3. – С.223-230
3. Мазурець О. В. Онтологічний підхід до побудови семантичної моделі навчальних матеріалів / О. В. Мазурець // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2017, №6. – С.223-229.
4. Moodle – Open-source learning platform. – [Електронний ресурс]. – Режим доступу: https://moodle.org/

достатнім є хоча б один варіант відповіді, якщо він повністю покриває можливі форми вживання даного терміну. Множина варіантів відповідей формуються на підставі параметрів моделі.

При формуванні правильних відповідей необхідно враховувати: абревіатуру терміну; скорочення; слова іншомовного походження (якщо є у тексті); відмінок/рід терміну.

Оскільки варіанти відповідей завжди сховані, функції цієї моделі не включають у себе підбір хибних термінів, а навпаки – модель має гарантувати, що всі можливі форми вживання цього поняття будуть розпізнані як правильна відповідь.

Отже, для сформованого на рисунку 1 тестового питання типу «Коротка відповідь» було використано термін «Вимірювання» та відповідну модель формування тестових завдань із відкритою відповіддю. Термін взятий з фрагменту тексту: «Вимірювання – це множина однотипних даних, що утворюють одну з граней гіперкуба» з теми «Моделі даних» навчального матеріалу «Організація баз даних та знань».

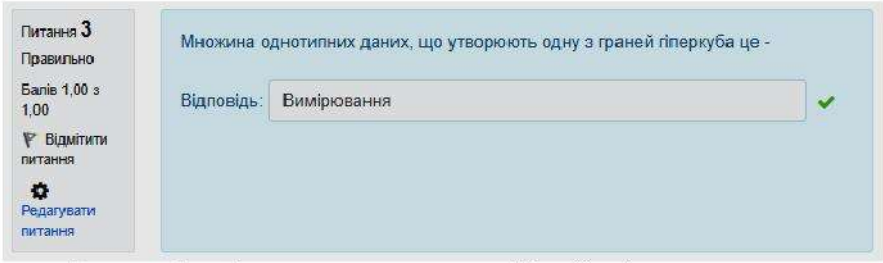


Рисунок 1 – Приклад використання у Moodle сформованого тестового завдання типу «Коротка відповідь»

Необхідно відмітити, що правильними відповідями є обидві з наведених, а для повної відповіді на питання достатньо вказати одну з них.

В даному випадку актуальний фрагмент тексту був знайдений наступною маскою ідентифікації:

```
[BeginSentence][ReCaps][TermGroup][Connector][ThesisGroup].
```

А генерація тестового завдання за актуалізованою моделлю відбулася шляхом використання відповідної маски формування тестового завдання із таким кодом:

```
Header -> [Caps][_____][connector][ThesisGroup]:
Answer 1 -> [Abbreviation] {TRUE}
Answer 2 -> [TermGroup1] {TRUE}
Answer N -> [TermGroupN] {TRUE}
```

Міністерство освіти і науки України
Хмельницький національний університет

АПКН 2019

АКТУАЛЬНІ ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК

ЗБІРНИК НАУКОВИХ ПРАЦЬ
за матеріалами XI всеукраїнської науково-практичної конференції
«Актуальні проблеми комп'ютерних наук АПКН-2019»

14-15 листопада 2019

Том 1

Роботи студентів та молодих вчених Факультету програмування та комп'ютерних і телекомунікаційних систем ХНУ

Хмельницький 2019

Актуальні проблеми комп'ютерних наук. Збірник наукових праць за матеріалами XI всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2019» – Хмельницький: ХНУ, 2019, Т.1. – 248с.

У збірнику наукових праць представлені перспективні практичні розробки аспірантів, магістрів та здобувачів в області сучасних інформаційних технологій. Розглянуто актуальні проблеми комп'ютерних наук, прикладної математики й інформатики, приведено ряд робіт по впровадженню інформаційних технологій у виробництво та управління. Висвітлено перспективні розробки з сучасних систем пошуку й обробки інформації, САПР та математичного, комп'ютерного і нейромережевого моделювання.

АКТУАЛЬНІ ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК - 2019

XI всеукраїнська науково-практична конференція

Метою конференції є висвітлення актуальних проблем комп'ютерних наук, інформатики та інформаційних технологій.

Основні напрямки роботи конференції:

1. Прикладні інформаційні технології.
2. Сучасні системи пошуку, захисту і обробки інформації.
3. Математичне, комп'ютерне і нейромережеве моделювання.
4. САПР, математичні моделі і методи рішення інженерних задач.
5. Проблеми впровадження інформаційних технологій у виробництво та управління.

Робочі мови конференції: українська, англійська.

КЕРІВНИЦТВО ОРГКОМІТЕТУ:

СИНЮК О. М.
голова оргкомітету, проректор Хмельницького національного університету з наукової роботи, доктор технічних наук, професор

СОРОКАТИЙ Р. В.
заступник голови оргкомітету, доктор технічних наук, завідувач кафедри Комп'ютерних наук та інформаційних технологій ХНУ, професор

БАРМАК О. В.
заступник голови оргкомітету, доктор технічних наук, професор

СЕКРЕТАРІАТ КОНФЕРЕНЦІЇ:

Мазурець О. В.
секретар конференції, старший викладач каф. КНІТ ХНУ

КОНТАКТНА ІНФОРМАЦІЯ:
e-mail для листування: *apkt.khnu@gmail.com*

ЗМІСТ

УДК 004.4

Ковальчук О.В., Білоус Г.А., Слободзян В.О.

Хмельницький національний університет

ВИКОРИСТАННЯ ПРОГРАМНОГО РОЗШИРЕННЯ SPIRE.DOC ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ З ЦИФРОВИМИ ДОКУМЕНТАМИ

З метою визначення найбільш ефективного програмного розширення проведено аналіз сучасних існуючих спеціалізованих програмних розширень для автоматизації обробки електронних документів: Microsoft.Office.Interap.Word.dll, DocumentFormat.OpenXml.dll та Spire.Doc.dll. В результаті аналізу розглянутих бібліотек, було визначено Spire.Doc.dll оптимальним варіантом для використання в автоматизації обробки електронних документів. Встановлено, що перенесення функцій автоматичного співставлення стилів текстових блоків їх властивостям з рівня функціоналу програмного коду застосунка на рівень функціоналу бібліотеки дозволяє спростити як роботу застосунка з електронним документом, так і процес програмування.

Microsoft.Office.Interop.Word.dll, DocumentFormat.OpenXml.dll and Spire.Doc.dll were useing for chooce the most effective program components for automation work with electronic documents. In result audit this libraries there was chosen Spire.Doc.dll for use to automation of processing of electronic documents. It is established that in result transfer of functions for automatic matching styles of text block to their properties from the level of the functional code of the application to the level of the functional library simplify work of the application with electronic document and programming process.

Для ефективної реалізації програмної обробки цифрових документів є доцільним використання спеціалізованих програмних комплексів, що надають об'єктно-орієнтований інструментарій для програмної роботи з контентом відповідних файлів [1]. Для реалізації програмної обробки цифрових документів є доцільним використання розглянутих в даній статті спеціалізованих програмних комплексів, що надають необхідний інструментарій для програмної роботи з контентом відповідних файлів, зокрема: Microsoft.Office.Interap.Word.dll [2], DocumentFormat.OpenXml.dll [3], Spire.Doc.dll [4].

Метою роботи є аналіз сучасних існуючих спеціалізованих програмних розширень для автоматизації обробки цифрових документів і визначення найбільш ефективного програмного розширення.

На сучасному етапі для зберігання електронних документів навчальних матеріалів використовується файл з розширенням .DOCX (або створених на його основі .HTML, .PDF тощо). На відміну від файлів .DOC, які зберігають дані документа в одному бінарному файлі, файли .DOCX створюються за допомогою відкритого формату XML, в якому зберігаються документи як колекції окремих файлів і папок в стиснутому пакеті. .DOCX-файли містять .XML-файли і три папки, docProps, Word, і _rels (Рис. 1), які містять властивості документа, його зміст і відношення між файлами, тему та включені файли. .DOCX-файли розроблені, щоб зробити вміст документів доступним і відкритим – так, текстовий документ чи зображення зберігаються як окремі прості файли в такій колекції у складі одного файлу .DOCX.

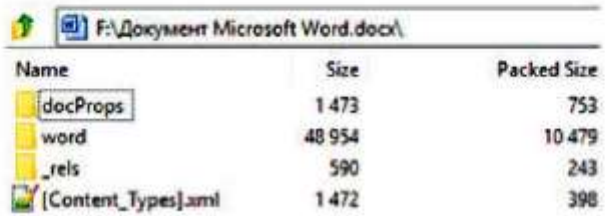


Рисунок 1 – Структура .DOCX-файлу

Office OpenXML є форматом електронних офісних документів, таблиць, презентацій та діаграм, розроблений компанією Microsoft. Даний формат є міжнародним стандартом, який затверджений в Міжнародній організації зі стандартизації (ISO) [5]. Він включає в себе такі первинні мови розмітки:

- WordprocessingML – для обробки текстових документів;
- SpeadsheetXL – електронних таблиць;
- PresentationML – для презентацій;
- DrawingML – для векторної графіки, діаграм.

Хоча Word реалізує свою функціональність через об'єктну модель, збереження даних відбувається у наборі XML-файлів, що ускладнює можливості для автоматизації прямого програмного парсингу [6]. Оскільки файл стилів та текст знаходяться в різних файлах у складі .XML, потрібно зчитувати одразу ряд файлів та на базі їх взаємозв'язків програмно формувати об'єктну модель документа, що є не найкращим рішенням.

Бібліотека ***Microsoft.Office.Interap.Word.dll*** [2] дозволяє керованому коду взаємодіяти з MS Office та об'єктною моделлю на базі COM.

За допомогою процесів автоматизації Office Automation програми, написані мовами, такими як Visual C# .NET, отримують можливість програмно керувати іншими програмами. Автоматизація Microsoft Word (далі Word) дозволяє виконувати відповідні дії, такі як створення нового документу, додавання даних у документ або створення таблиць. З програмами Word та іншими програмами Microsoft Office практично всі дії, які користувач може виконувати вручну за допомогою користувацького інтерфейсу, також можуть бути виконані програмним шляхом за допомогою автоматизації Office Automation.

Word реалізує цю програмну функціональність через об'єктну модель [7]. Об'єктна модель є набором класів і методів, які служать аналогами логічних компонентів Word. Наприклад, існує об'єкт Application, об'єкт Document та об'єкт Paragraph, кожен з яких містить функціональність цих частин Word.

Щоб використовувати функції програми MS Office в проекті, можна використовувати первинний взаємозв'язок PIA (Primary Interop Assembly). PIA дозволяє керованому коду взаємодіяти з MS Office та об'єктною моделлю на базі COM. При створенні нового проекту Office, Visual Studio додає посилання на PIA, необхідний для побудови проекту. При цьому, у деяких сценаріях доводиться додавати посилання на додаткові PIA (наприклад, якщо необхідно використовувати функцію MS Office Word в проекті для MS Office Excel).

DocumentFormat.OpenXml.dll

Бібліотека класів .NET DocumentFormat.OpenXml.dll (версії 2.0 / 2.5 / 2.8.1) [3] дає можливість розробникам програмного забезпечення працювати з пакетом Microsoft Office. Компанія надає дану бібліотеку безкоштовно в повному доступі, її її можна завантажити з офіційного сайту Microsoft або засобами Visual Studio.

Для роботи з файлами формату .DOCX, використовується мова розмітки WordprocessingML [8]. На рисунку 2 зображено приклад структури цієї мови розмітки.

Теги, наведені на рисунку 2, мають такі властивості:

- *document* – є основою частиною документа
- *body* – контейнер для збору структур рівня блоку, які складають основну історію;
- *p* – абзац (paragraph);
- *r* – контейнер який містить в собі текст з однаковими властивостями (run);
- *t* – власне текстовий елемент (text).

Всі інші теги, які можуть використовуватися, є допоміжними до вищенаведених і додають їм стильових властивостей.


```
<?xml version="1.0" encoding="utf-8"?>
<w:document
xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2
006/main">
  <w:body>
    <w:p>
      <w:r>
        <w:t> Текст </w:t>
      </w:r>
    </w:p>
  </w:body>
</w:document
```

Рисунок 2 – Приклад WordprocessingML-розмітки .DOCX-документу

Spire.Doc.dll [4] для .NET є повністю незалежною бібліотекою класів .NET Word, спеціально створеною для розробників, й дозволяє швидко генерувати, відкривати, редагувати та зберігати документи Word різних версій. Конвертація функціональних можливостей дозволяє розробникам здійснювати перетворення Word в інші актуальні формати документів (.PDF, .EPub, .HTML, .RTF, .Image, .XML тощо). Можна створювати та обробляти вже існуючі документи Word динамічно.

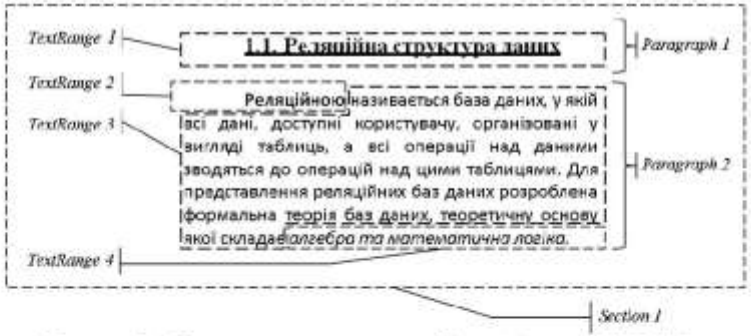


Рисунок 3 – Приклад використання об'єктної моделі MS Office

Бібліотека Spire.Doc.dll забезпечує роботу з майже всіма елементами документа Word, а саме: сторінки, розділи, заголовки, колонтитули, абзаци, переліки, таблиці, текст, виноски, поля, гіперпосилання, закладки, коментарі, зображення, стилі, фонові параметри, функції друку, налаштування документа та захисту. Крім того, підтримуються графічні об'єкти, включаючи форми, текстові поля,

зображення, OLE-об'єкти та елементи керування. Бібліотека Spire.Doc.dll підтримує функцію пошуку та заміни, вирівнювання, перерву сторінки, поле заповнення, об'єднання документів, копіювання документів, друк тощо.

При зчитуванні файлу Spire.Doc.dll перетворює документ в об'єктну модель. Ця модель має структуру, яка починається документом і закінчується об'єктом *TextRange*. Приклад використання такої об'єктної моделі документу MS Office показаний на Рисунку 3.

Одержати відомості про стилі *Paragraph* можна за допомогою методу *GetStyle()* (Рис. 4). Через роботу з відповідними властивостями, можна одержати текст та стилі (Рис. 5), що відносяться до визначеного *TextRange*.

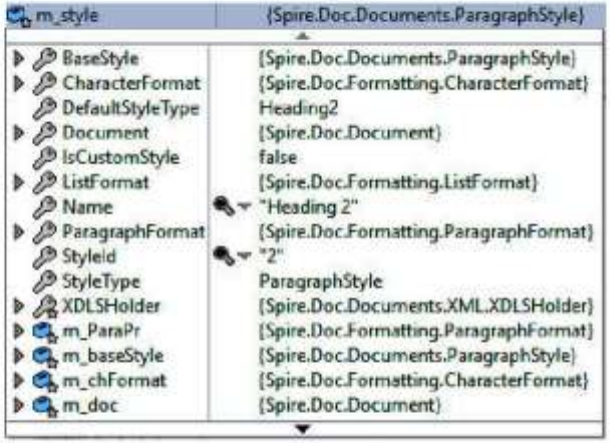


Рисунок 4 – Одержання відомостей про стилі Paragraph

Хоч бібліотека Microsoft.Office.Interap.Word.dll надає доступ до всіх функцій MS Office, так як працює безпосередньо з PIA, для її роботи потрібна ліцензія для MS Office на кожному клієнтському комп'ютері. Крім того, при використанні Automation, MS Office завантажується у фоновому режимі, внаслідок чого займає деякий обсяг оперативної пам'яті та завантажує велику кількість файлів і DLL. Додатки MS Office були розроблені як додатки для користувацького інтерфейсу, і через це Microsoft.Office.Interop.Word.dll працює повільно. Microsoft не рекомендує використовувати Office Automation (або будь-який Office Interop) на сервері.

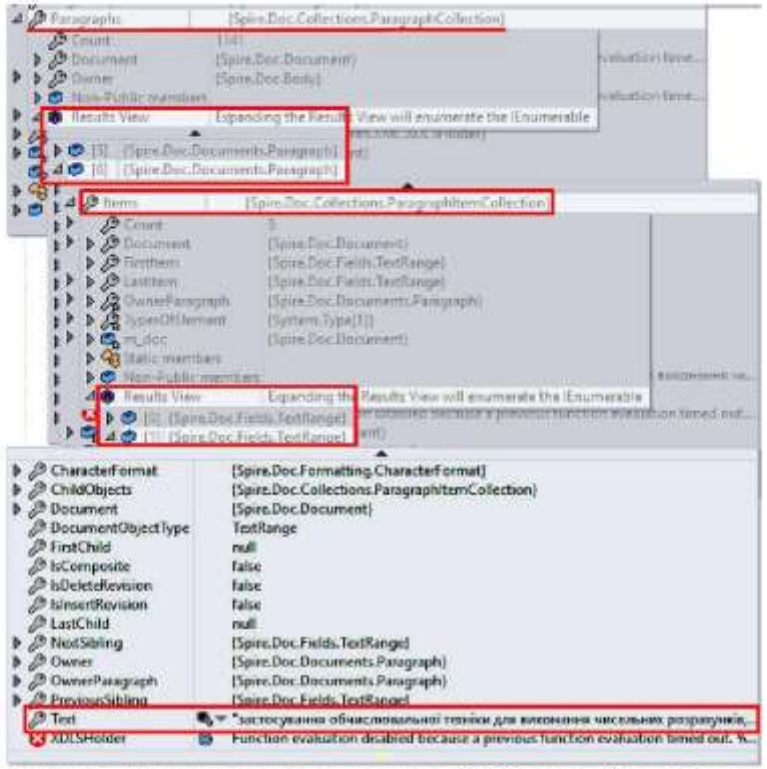


Рисунок 5 – Позиція визначеного тексту в TextRange у об'єктній моделі

На відміну від Microsoft.Office.Interap.Word.dll, DocumentFormat.OpenXml.dll та Spire.Doc.dll не вимагають наявності MS Office на машині користувача та його запуску в фоновому режимі.

При роботі з docx-файлом за допомогою DocumentFormat.OpenXml.dll потрібно дотримуватися тієї ж ієрархії, що і сама структура розмітки тобто: *document-paragraph-run-text*. Також відповідно до розглянутих вище особливостей взаємозв'язків всередині документу, при присвоєнні параграфу деякого стилю, у властивостях параграфа надається лише ідентифікатор (ID) на даний стиль, натомість сам стиль описується окремо у файлі style.xml. Внаслідок необхідності регулярного співставлення ID стиля з контейнером style.xml для одержання характеристик стилів абзаців, зростає складність програмного використання бібліотеки. Перевагою Spire.Doc.dll визначено відсутність

необхідності співставлення ID стиля з контейнером style.xml, оскільки дана функція реалізована на рівні бібліотеки.

Таким чином, в результаті аналізу сучасних існуючих спеціалізованих програмних розширень для автоматизації обробки цифрових документів, було визначено Spire.Doc.dll оптимальним варіантом для використання в автоматизації обробки цифрових документів. Основними перевагами Spire.Doc.dll встановлено відсутність необхідності наявності MS Office на машині користувача та запуску в фоновому режимі, а також реалізацію функцій автоматичного співставлення стилів текстових блоків їх властивостям на рівні розширення.

Перенесення функцій автоматичного співставлення стилів текстових блоків їх властивостям з рівня функціоналу програмного коду застосунка на рівень функціоналу бібліотеки дозволяє спростити як роботу застосунка з цифровим документом, так і процес програмування.

Перелік посилань

1. Мазурець О. В. Використання спеціалізованих програмних розширень для автоматизації роботи з цифровими документами навчальних матеріалів / О. В. Мазурець, О. В. Ковальчук, В. О. Слободзян // Науковий журнал «Вісник Хмельницького національного університету» серія: Технічні науки. Хмельницький, 2018, №1. – С.61-69.
2. Considerations for server-side Automation of Office [Електронний ресурс]. – Режим доступу: https://support.microsoft.com/en-us/help/257757/considerations-for-server-side-automation-of-office/
3. International Organization for Standardization - Open XML file formatsb [Електронний ресурс]. – Режим доступу: https://www.iso.org/search/x/query/Open%2520XML
4. Spire.Doc for .NET [Електронний ресурс]. – Режим доступу: https://www.nuget.org/packages/Spire.Doc/
5. International Organization for Standardization - Open XML file formatsb [Електронний ресурс]. – Режим доступу: https://www.iso.org/search/x/query/Open%2520XML
6. Office Open XML [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Office_Open_XML
7. How to automate Microsoft Excel from Microsoft Visual C#.NET [Електронний ресурс]. – Режим доступу: https://support.microsoft.com/en-us/help/302084/how-to-automate-microsoft-excel-from-microsoft-visual-c--net
8. DocX for creates or modifies Microsoft Word files [Електронний ресурс]. – Режим доступу: https://github.com/xceedsoftware/DocX
9. Free Spire.Doc for .NET [Електронний ресурс]. – Режим доступу: https://www.e-iceblue.com/Introduce/free-doc-component.html

Міністерство освіти і науки України
Хмельницький національний університет

ЗБІРНИК НАУКОВИХ ПРАЦЬ
за матеріалами XII всеукраїнської науково-практичної конференції
«Актуальні проблеми комп'ютерних наук АПКН-2020»

9-10 листопада 2020

Хмельницький 2020

УДК 004:37:001:62

Збірник наукових праць за матеріалами XII всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2020». Хмельницький – 2020. – 365с.

У збірнику наукових праць подані перспективні практичні розробки аспірантів, студентів та здобувачів в області сучасних інформаційних технологій. Розглянуто актуальні проблеми комп'ютерних наук, комп'ютерної інженерії, прикладної математики й інженерії програмного забезпечення, приведено ряд робіт по впровадженню інформаційних технологій у виробництво та управління. Висвітлено перспективні розробки сучасних систем пошуку, обробки й захисту інформації, медійних та комунікаційних системи.

УДК 004:37:001:62

Матеріали конференції відтворені з авторських оригіналів. Оргкомітет конференції висловлює подяку учасникам конференції та сподівається на подальшу співпрацю.

З питань проведення конференції та подальшого обміну інформацією звертатись на e-mail конференції: *apkt.khnu@gmail.com*

АКТУАЛЬНІ ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК - 2020

XII Всеукраїнська науково-практична конференція

Метою конференції є висвітлення актуальних проблем комп'ютерних наук, інформатики та інформаційних технологій.

СЕКЦІЇ КОНФЕРЕНЦІЇ:

1. Комп'ютерні науки та прикладні інформаційні технології.
2. Комп'ютерна інженерія та системи захисту інформації.
3. Математичне моделювання та інженерія програмного забезпечення
4. Телерадіокомунікації, медійні та комунікаційні системи.
5. Проблеми впровадження інформаційних технологій у виробництво та управління.

Робочі мови конференції: українська, англійська

ОРГКОМІТЕТ:

СИНЮК О. М. голова оргкомітету, проректор Хмельницького національного університету з наукової роботи, доктор технічних наук, професор

СОРОКАТИЙ Р. В. заступник голови оргкомітету, завідувач кафедри Комп'ютерних наук та інформаційних технологій ХНУ, доктор технічних наук, професор

БАРМАК О. В. заступник голови оргкомітету, доктор технічних наук, професор

САВЕНКО О. С. декан Факультету програмування та комп'ютерних і телекомунікаційних систем ХНУ, доктор технічних наук, професор

ВИСОЦЬКА О. В. доктор технічних наук, завідувач кафедри радіоелектронних та біомедичних комп'ютеризованих засобів і технологій Національного аерокосмічного університету ім. М. Є. Жуковського «Харківський авіаційний інститут», професор

ЛАВРОВ Є. А. доктор технічних наук, професор (Сумський державний університет)

ТІМОФЄЄВА Л. В. відповідальна за студентську науково-дослідну роботу ХНУ

МАЗУРЕЦЬ О. В. секретар конференції, старший викладач кафедри комп'ютерних наук та інформаційних технологій ХНУ

КОНТАКТНА ІНФОРМАЦІЯ:

e-mail для листування: *apkt.khnu@gmail.com*

ЗМІСТ

УДК 004.4

Білоус Г. А., Мазурець О. В.

Хмельницький національний університет

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АДАПТИВНОГО ТЕСТУВАННЯ РІВНЯ ЗНАНЬ

Розглянуто кроки розробленої інформаційної технології адаптивного тестування знань з використанням тестових завдань сформованих на основі ключових термінів інформаційного навчального матеріалу. Тип адаптивного тестування та поточний елемент семантичної структури обумовлюють формування множини тестових завдань з якої за критеріями обирається тестове завдання для перевірки поточного ключового терміну. Далі залежно від оцінки відповіді на поточне тестове завдання обирається наступне та при завершенні тестування обраховується загальний результат.

There are considered the steps of the developed information technology of adaptive testing of knowledge with the use of test tasks formed based on key terms of information educational material. The type of adaptive testing and the current element of the semantic structure specifies the formation of the test tasks set, from which according to the criteria a test task is selected to check the current key term. Then, depending on the estimation of the answer to the current test task, the following is selected, and at the end of testing the total result is calculated.

Розвиток сучасного суспільства і системи освіти пред'являють все більш високі вимоги до якості підготовки студентів. Особливу роль в підвищенні якості освіти відіграє його комп'ютеризація, яка розуміється не як просте передавання навчальної інформації в цифровому вигляді, а як створення спроектованого з психолого-педагогічної та методичної точок зору інформаційно-освітнього навчального середовища [1]. Використання інформаційних технологій у навчанні забезпечує персоналізований процес навчання та надає можливості для ефективного самоконтролю та контролю з діагностикою помилок і зворотним зв'язком. Одним із основних інструментів вимірювання рівня знань у системі освіти залишається тестування, за допомогою якого можна не тільки виявити якість навчання, але і ефективно управляти навчальним процесом [2].

Комп'ютерне тестування може проводитися в різних формах, що розрізняються за технологією об'єднання завдань в тест. Однією з перспективних форм тестування є адаптивне. Адаптивне тестування [3] полягає в тому, що запропоновані студенту поточні завдання залежать від результатів його відповідей на попередні завдання, що дозволяє значно знизити трудомісткість і час тестування, що на практиці буває дуже важливо. Існує декілька підходів до формування

тестових завдань для адаптивного тестування та власне його проведення [4, 5, 6].

У тестології існує проблема рівномірного покриття тестовим навчальним матеріалом семантики контенту інформаційного навчального матеріалу (ІНМ). Повнота покриття тестами ключових положень навчального матеріалу може визначатися через аналіз використання у тестових завданнях ключових термінів із навчальних матеріалів. Рівномірність покриття тестами структури ІНМ визначається через аналіз прив'язки тестових завдань до ієрархії структурних елементів навчального матеріалу. Така прив'язка може бути забезпечена аналізом контенту тестових завдань на предмет наявності у них ключових термінів із навчальних матеріалів, за умови прив'язки ключових термінів до ієрархії структурних елементів навчального матеріалу, що є також важливим аспектом при формуванні наборів тестових завдань [7].

В рамках розробленої інформаційної технології запропоновано використання тестових завдань для адаптивного тестування, сформованих на основі ключових термінів, вжитих у ІНМ. Також, базуючись на важливості певного ключового терміну в межах заголовку ІНМ, здійснюється адаптивне тестування рівня знань.

Кроки інформаційної технології адаптивного тестування рівня знань зображено на рис. 1. *Вхідними даними* методу адаптивного тестування є поточний елемент семантичної структури m'_H та тип адаптивного тестування t. Тип адаптивного тестування обумовлює семантичну важливість початкового ключового терміну (найбільш важливий, найменш важливий, середньої важливості), що була визначена методом дисперсійного оцінювання [8].

На *Кроці 1* формується множина тестових завдань $M''_{\text{ТЗ}}$ для перевірки поточного ключового терміну $K^*[\text{n}]$. Для цього на *Кроці 1.1* визначається кількість k^{max} ключових термінів відповідних актуальному елементу семантичної структури m'_H. Далі на *Кроці 1.2* утворюється множина тестових завдань $M'_{\text{ТЗ}}$ відповідних актуальному елементові семантичної структури m'_H з загальної множини тестових завдань $M_{\text{ТЗ}}$ ($M_{\text{ТЗ}} => M'_{\text{ТЗ}}$) та на *Кроці 1.3* визначається поточний ключовий термін $K^*[\text{n}]$ для перевірки згідно типу адаптивного тестування t, де $\text{n} \in [1..k^{max}]$. Завершальним є *Крок 1.4* на якому формується множина тестових завдань $M''_{\text{ТЗ}}$ ($M'_{\text{ТЗ}} => M''_{\text{ТЗ}}$, $M''_{\text{ТЗ}}$: $m''_{\text{ТЗ},k_z}$, де $k_z \in [k^1..k^*]$).

На *Кроці 2* інформаційної технології відбувається вибір тестового завдання $m^*_{\text{ТЗ}}$ для перевірки поточного ключового терміну K^*. Множина тестових завдань $M''_{\text{ТЗ}}$ ранжується на *Кроці 2.1* за множиною тестових завдань $m_{\text{ТЗ}} \in M''_{\text{ТЗ}}$, які вже було використано, що формує множину $M'''_{\text{ТЗ}}$ ($M''_{\text{ТЗ}} => M'''_{\text{ТЗ}}$). Тестові завдання із множини $M'''_{\text{ТЗ}}$ в свою чергу на *Кроці 2.2* сортуються за збігом $m_{\text{ТЗ}} \in M'''_{\text{ТЗ}}$ із множиною використаних речень $M_{\text{реч}}$ таким чином утворюючи множину $M''''_{\text{ТЗ}}$ ($M'''_{\text{ТЗ}} => M''''_{\text{ТЗ}}$).

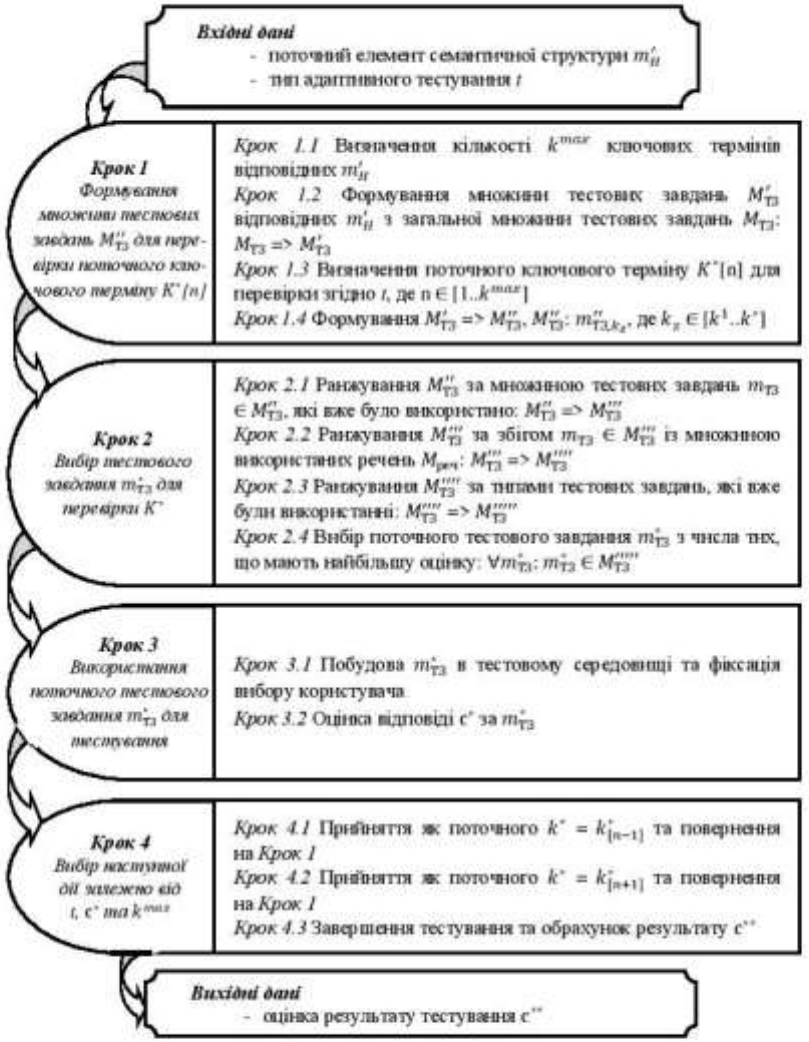


Рисунок 1 – Кроки інформаційної технології адаптивного тестування рівня знань

Далі на *Кроці 2.3* множина тестових завдань M''''_{T3} ранжується за типами тестових завдань, які вже були використанні, що зумовлює формування множини

M'''''_{T3} (M''''_{T3} => M'''''_{T3}). На *Кроці 2.4* відбувається вибір поточного тестового завдання m^*_{T3} з числа тих, що мають найбільшу оцінку $\forall m^*_{T3}$: $m^*_{T3} \in M'''''_{T3}$.

Використання поточного тестового завдання m^*_{T3} для тестування здійснюється на *Кроці 3*. При цьому відбувається формування m^*_{T3} в тестовому середовищі, фіксується вибір (відповідь) користувача на *Кроці 3.1* та відбувається оцінка відповіді c^* за поточне тестове завдання m^*_{T3} на *Кроці 3.2*.

На *Кроці 4* обирається подальша дія залежно від типу адаптивного тестування t, оцінки відповіді c^* та загальної кількості ключових термінів k^{max}: поточний ключовий термін приймається як $k^* = k^*_{[n-1]}$ на *Кроці 4.1* чи $k^* = k^*_{[n+1]}$ на *Кроці 4.2* та відбувається повернення на *Крок 1*. Або на *Кроці 4.3* тестування завершується та обраховується результат c^{**}.

Вихідними даними методу адаптивного тестування є оцінка результату тестування c^{**}, що вираховується залежно від відповідей опитуваного впродовж тестування.

Для дослідження ефективності використання тестових завдань за ІТ адаптивного тестування рівня знань було використано призначену для проведення адаптивного тестування тестову програмну систему (рис. 2). Використовуючи ідентичну множину тестових завдань, було запропоновано піддослідним (студентам за власним бажанням) пройти тестування з використанням базового алгоритму проходження тесту середовища Moodle та з використанням адаптивного алгоритму проходження тесту. Достатня кількість тестових завдань за традиційного алгоритму визначається шляхом їх випадкового додавання до тесту до вичерпання вказаного граничного часу. Достатня кількість тестових завдань за адаптивного алгоритму визначається шляхом одержання кінцевого результату по рівню знань за кожним із заголовків ІНМ.

Дослідження ефекту від використання множин тестових завдань для адаптивного тестування виконано шляхом оцінки різниці часу на проходження тесту та кількості тестових завдань, використаних для тестування, за наведеними двома алгоритмами, як для кожної категорії результату (за національною шкалою), так і в середньому.

Ефект від використання адаптивного тестування за різницею в часі тестування обраховується за формулою:

$$\Delta T_i = \frac{TC_i - TA_i}{TC_i} \cdot 100, \ \overline{\Delta T_i} = \frac{\sum_{j=1}^{n} \Delta T_{i,j}}{n}, \qquad (1)$$

де TC_i – час, затрачений на тестування за традиційного алгоритму i-м досліджуваним; TA_i – час, затрачений на тестування за адаптивного алгоритму i-м досліджуваним; $\overline{\Delta T_i}$ – середнє значення для вибірки з n проходжень тесту двома алгоритмами.

Рисунок 2 – Процес тестування в інформаційній системі розробленій за ІТ адаптивного тестування рівня знань

Ефект від використання адаптивного тестування за різницею в кількості одержаних на тестуванні тестових завдань обраховується так:

$$\Delta N_i = \frac{NC_i - NA_i}{NC_i} \cdot 100, \quad \overline{\Delta N_i} = \frac{\sum_{j=1}^{n} \Delta N_{i,j}}{n}, \tag{2}$$

де NC_i – кількість завдань, одержана на тестування за традиційного алгоритму i-м досліджуваним; NA_i – кількість завдань, одержана на тестування за адаптивного алгоритму i-м досліджуваним; $\overline{\Delta N_i}$ – середнє значення для вибірки з n проходжень тесту двома алгоритмами.

Використана вибірка тесових завдань містить 154 тестових завдання, з яких логічного типу – 41, одиничного вибору – 44, множинного вибору – 33, із введенням тексту – 36 завдань. Використовуючи один і той самий набір тестових завдань, студенти проходили тестування з використанням базового алгоритму середовища Moodle та з використанням адаптивного алгоритму.

Середній час проходження тестів для 55 студентів за категоріями згідно одержаної за національною шкалою оцінки з використанням базового алгоритму середовища Moodle склав: для оцінки «незадовільно» $\overline{TC_2}$ = 25,48 хв., для оцінки «задовільно» $\overline{TC_3}$ = 27,33 хв., для оцінки «добре» $\overline{TC_4}$ = 29,02 хв., для оцінки «відмінно» $\overline{TC_5}$ = 28,36 хв., середній для всіх категорій $\overline{TC_5}$ = 27,62 хв.. Середній час проходження за аналогічних умов з використанням адаптивного алгоритму

проходження тесту склав: для оцінки «незадовільно» $\overline{TA_2}$ = 13,27 хв., для оцінки «задовільно» $\overline{TA_3}$ = 15,58 хв., для оцінки «добре» $\overline{TA_4}$ = 24,17 хв., для оцінки «відмінно» $\overline{TA_5}$ = 27,54 хв., середній для всіх категорій $\overline{TA_5}$ = 21,95 хв.. Візуальне подання результатів проходження тестів за середнім часом зображено на рис. 3.

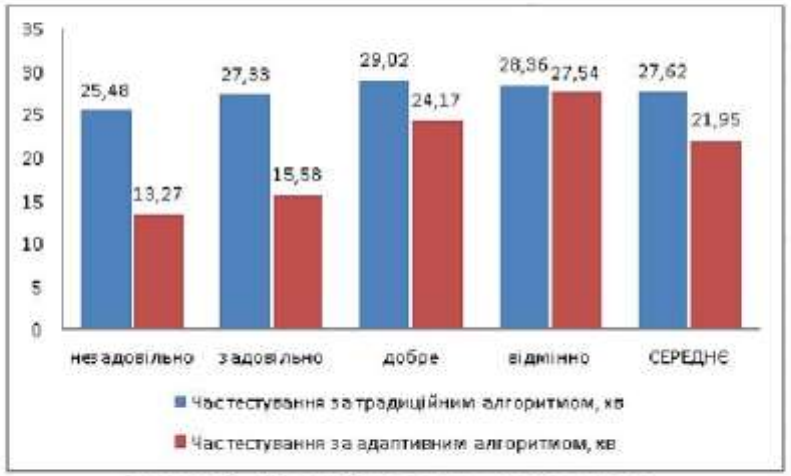


Рисунок 3 – Середній час проходження тестів, хв.

Одержані результати тестування дозволили за (1) визначити ефект від використання адаптивного тестування за різницею в часі тестування: для оцінки «незадовільно» $\overline{\Delta T_2}$ = 47,92 %, для оцінки «задовільно» $\overline{\Delta T_3}$ = 42,99 %, для оцінки «добре» $\overline{\Delta T_4}$ = 16,71 %, для оцінки «відмінно» $\overline{\Delta T_5}$ = 2,89 %, середнє значення за всіма категоріями $\overline{\Delta Ts}$ = 20,53 %.

Середня кількість одержаних тестових завдань за використання базового алгоритму середовища Moodle склала: для оцінки «незадовільно» $\overline{NC_2}$ = 15,31 од., для оцінки «задовільно» $\overline{NC_3}$ = 14,83 од., для оцінки «добре» $\overline{NC_4}$ = 15,11 од., для оцінки «відмінно» $\overline{NC_5}$ = 15,07 од., середній для всіх категорій $\overline{NC_5}$ = 15,13 од.. Середня кількість одержаних тестових завдань за використання адаптивного алгоритму проходження тесту склав: для оцінки «незадовільно» $\overline{NA_2}$ = 6,49 од.,

для оцінки «задовільно» $\overline{NA_3}$ = 8,67 од., для оцінки «добре» $\overline{NA_4}$ = 12,25 од., для оцінки «відмінно» $\overline{NA_5}$ = 16,12 од., середній для всіх категорій $\overline{NA_S}$ = 11,83 од. (рис. 4).

Одержані результати тестування дозволили за (2) визначити ефект від використання адаптивного тестування за кількістю одержаних тестових завдань: для оцінки «незадовільно» $\overline{\Delta N_2}$ = 57,61 %, для оцінки «задовільно» $\overline{\Delta N_3}$ = 41,54 %, для оцінки «добре» $\overline{\Delta N_4}$ = 18,93 %, для оцінки «відмінно» $\overline{\Delta N_5}$ = -6,97 %, середнє значення за всіма категоріями $\overline{\Delta Ns}$ = 17,28 %.

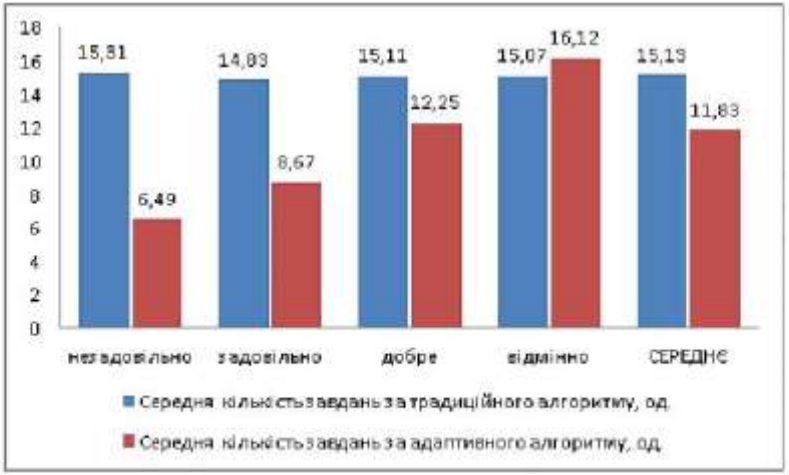


Рисунок 4 – Середня кількість одержаних тестових завдань, од.

Результати дослідження підтвердили можливість використання створених за розробленою інформаційною технологією множин тестових завдань не тільки для проведення тестування за традиційним алгоритмом тестування, а й для адаптивного тестування.

В середньому для визначення рівня знань за адаптивного тестування знадобилося використання на $\overline{\Delta Ns}$ = 17,28 % меншої кількості завдань. При цьому чим вищим виявився рівень знань піддослідного, тим більша кількість завдань знадобилася для визначення рівня знань за адаптивного тестування, від $\overline{NA_2}$ = 6,49 од. для оцінки «незадовільно» до $\overline{NA_5}$ = 16,12 од. для оцінки «відмінно». Причому,

якщо для оцінки «незадовільно» ця кількість була суттєво меншою за показник традиційного алгоритму ($\overline{\Delta N_2}$ = 57,61 %), то для оцінки «відмінно» кількість одержаних тестових завдань виявилась навіть дещо більшою ($\overline{\Delta N_5}$ = -6,97 %), що було пов'язано з необхідністю максимального заглиблення перевірки в кожен елемент структури ІНМ.

Загалом, за $\overline{TA_S} < \overline{TC_S}$, алгоритм адаптивного тестування в середовищі Moodle забезпечив більш швидке на $\overline{\Delta Ts}$ = 20,53 % проходження тесту за базовий алгоритм проходження тесту середовища Moodle. Чим вищим виявився рівень знань студента, тим більша кількість завдань знадобилася для визначення рівня знань за адаптивного тестування, й відповідно тим більше часу було витрачено на їх вирішення – якщо для оцінки «незадовільно» $\overline{\Delta T_2}$ = 47,92 %, то для оцінки «відмінно» $\overline{\Delta T_5}$ = 2,89 %. В той час як за традиційного алгоритму ці показники відрізняються несуттєво.

Хоча $\overline{\Delta N_5} < 0$, одержано $\overline{\Delta T_5} > 0$. Це можна пояснити тим, що за адаптивного тестування тестові завдання подавались в логічно послідовному порядку, що дало можливість студентам більш зосереджено їх опрацьовувати і зменшило час на «переключення» між різними рубриками.

Загалом, алгоритм адаптивного тестування в середовищі Moodle забезпечив більш швидке на $\overline{\Delta Ts}$ = 20,53 % проходження тесту, при цьому для визначення рівня знань знадобилося використання на $\overline{\Delta Ns}$ = 17,28 % меншої кількості завдань.

На об'єктивність результатів суттєво впливає коректність сформованих тестових завдань. Адже для створення коректного тестового набору, тестові завдання формуються так, щоб вони рівномірно покривали ІНМ. При коректно сформованому наборі тестових запитань, запропонований підхід дозволяє більш точно визначити рівномірність рівня отриманих знань та виявити прогалини в розумінні вивченого матеріалу.

Отже, на сучасному етапі тести є ефективним методом контролю рівня одержаних знань. В процесі тестування слід виконувати перевірку розуміння студентом складових семантичної структури навчального матеріалу у вигляді системи заголовків та відповідних множин ключових термінів.

Адаптивне тестування є актуатуальним та ефективним видом комп'ютерного тестування, що забезпечує належну об'єктивність процесу контролю рівня знань. А запропоновані кроки інформаційної системи адаптивного тестування рівня знань дозволяють здійснити контроль розуміння ключових

термінів в межах інформаційного навчального матеріалу залежно від їх семантичної важливості.

Перелік посилань

1. Комп'ютерне тестування як одна з форм діагностики та контролю якості знань студентів – [Електронний ресурс]. – Режим доступу: https://naurok.com.ua/stattya-komp-yuterne-testuvannya-yak-odna-z-form-diagnostiki-ta-kontrolyu-yakosti-znan-studentiv-28883.html
2. Концептуальні основи оцінювання якості шкільної освіти – [Електронний ресурс]. – Режим доступу: http://www.ippobuk.cv.ua/images/Шепенюк_7.pdf
3. Сергієнко В.П. Комп'ютерні технології в тестуванні: навч. посіб. / В. П. Сергієнко, М. П. Малежик, Т. В. Сіткар. – Луцк: «Волиньполіграф», 2012. – 290 с.
4. Нехаев И. Н. Постановка задачи эффективного адаптивного тестирования уровня знаний / И. Н. Нехаев // Вестник Московского городского педагогического университета, серия «Информатика и информатизация образования». – 2008. – № 15. – С. 124–127.
5. Іванова О. Н. Адаптивне тестування на практиці діагностики вмінь та знань / О. Н. Іванова, О. Н. Кононов [Електронний ресурс]. – Режим доступу: http://www.ht.bitnet.ru/press/articles/?view=art129.
6. Гайтан О. М. Елементи технології реалізації автоматизованого адаптивного контролю знань студентів в комп'ютерних системах навчання / О. М. Гайтан // Радіоелектронні і комп'ютерні системи. – 2014. – № 4. – С. 97-105.
7. Barmak O. Information technology for creation of semantic structure of educational materials / O. Barmak, O. Mazurets, I. Krak, A. Kulias, A. Smolarz, L. Azarova, K. Gromaszek, S. Smailova // Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments. – 2019. – Vol. 11176. [Електронний ресурс]. – Режим доступу: https://doi.org/10.1117/12.2537064
8. Мазурець О. В. Метод формального опису елементів моделей автоматизованого формування тестових завдань / О. В. Мазурець, О. В. Ковальчук, В. О. Слободзян, Г. А. Білоус // Збірник наукових праць за матеріалами Всеукраїнської науково-практичної конференції «Інтелектуальний потенціал – 2018». Хмельницький, 2018, Ч.1. – С.51-56.

Додаток И

Презентаційний матеріал

3 Аналіз відомих алгоритмів адаптивного тестування рівня знань

Назва	Опис
Алгоритм адаптивного тестування, при якому на початку тестування даються завдання середнього рівня складності	При використанні даного алгоритму тестування розпочинається із завдання середнього рівня складності й далі, залежно від відповідей, визначається складнішими чи легшими будуть подальші завдання. Недоліком даного алгоритму є застосування більшої кількості завдань у процесі тестування, аніж при алгоритмах, що починаються із перевірки найважливіших або найменш важливих термінів.
Алгоритм адаптивного тестування, що починається із перевірки найважливіших термінів	В ході тестування у випадку, коли студент відповідає правильно, складність тестових завдань підвищується шляхом перевірки менш важливих термінів. Недоліком є те, що коли студент має хороші знання, то він отримує більшу кількість завдань, аніж студент, що має задовільний рівень знань.
Алгоритм адаптивного тестування, при якому існує база знань, що містить у собі розподілені за рівнем складності тестові завдання	При використанні даного алгоритму за умови, що студент дав неправильну відповідь, наступне завдання обирається із вибірки більш легших завдань, при правильній – з більш складних. Недоліком алгоритму є те, що при визначеності складності завдань не враховується семантична структура ІНМ, а, отже, неможливо повноцінно перевірити володіння навчальним матеріалом.
Алгоритм адаптивного тестування, що базується на обході за структурою НК зверху вниз	За умови застосування даного алгоритму, у випадку неправильної відповіді на найлегші тестові завдання, тестування у частині структури, що розташовується нижче – не проводиться. Недолік – при проходженні тестування студентами з рівнем знань відмінно, не перевірятиметься повноцінне володіння матеріалом НК.
Алгоритм адаптивного тестування, що передбачає, що контроль знань починається з будь-якого рівня складності тестових завдань	Даний алгоритм передбачає, що контроль знань починається з будь-якого рівня складності тестових завдань й далі ітераційним методом наближається до рівня складності, що є відповідним реальному рівню знань студента. Недоліком методу є те, що тестування може розпочинатися із різних рівнів складності для різних студентів, що може вплинути на результат тестування.

4 Постановка задачі

Мета роботи полягає у розробці інформаційної технології для адаптивного тестування знань, що використовує показники семантичної важливості ключових термінів інформаційного навчального матеріалу для адаптивного вибору тестових завдань у процесі тестування.

Для досягнення мети необхідно розв'язати наступні ***задачі***:

- дослідити відомі підходи до адаптивного тестування рівня знань;
- вдосконалити інформаційну модель адаптивного тесту для формального подання елементів термінооріентованого адаптивного тестування рівня знань;
- вдосконалити метод адаптивного вибору тестових запитань для динамічного вибору тестових завдань у процесі тестування;
- розробити інформаційну технологію адаптивного тестування рівня знань, що використовує вдосконалені модель і метод;
- розробити інформаційну систему адаптивного тестування рівня знань на основі розробленої інформаційної технології;
- дослідити практичну ефективність інформаційної технології адаптивного тестування рівня знань шляхом аналізу функціональності й ефективності застосування відповідної інформаційної системи.

6

Інформаційна модель адаптивного тесту

Подання інформаційної моделі адаптивного тесту (АТ) має наступний вигляд:

$$\{M_{IEM} \cup M_H \cup M_P \cup M_{Pt} \cup M_S \cup M_{St} \cup M_{Term} \cup M_{KT} \cup M_{TS} \cup M_{TTasks} \cup M_{TTaskst} \cup \cup M_A \cup M_{STTasks} \cup M_{TTasksS} \cup M_{TA} \cup M_{Tt} \cup M_{R1TTasks} \cup M_U \cup M_{Ut}\} \subset AT,$$

де M_{IEM} – множина інформаційних навчальних матеріалів, M_H – множина заголовків ІНМ; M_P – множина абзаців, M_{Pt} – множина типів абзаців, M_S – множина речень, M_{St} – множина типів речень, M_{Term} – множина термінів, M_{KT} – множина ключових термінів, M_{TS} – множина появи терміну у реченнях, M_{TTasks} – множина тестових завдань, $M_{TTaskst}$ – множина типів тестових завдань, M_A – множина відповідей, $M_{STTasks}$ – множина вибірок тестових завдань, $M_{TTasksS}$ – множина тестових завдань у вибірках, M_{TA} – множина спроб тестування, M_{Tt} – множина типів тестування, $M_{R1TTask}$ – множина результатів проходження одного тестового завдання, M_U – множина користувачів, M_{Ut} – множина типів користувачів.

7

Метод адаптивного вибору тестових запитань

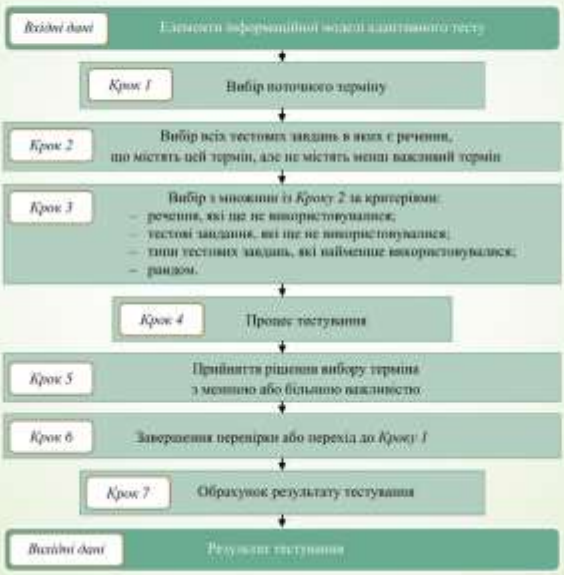
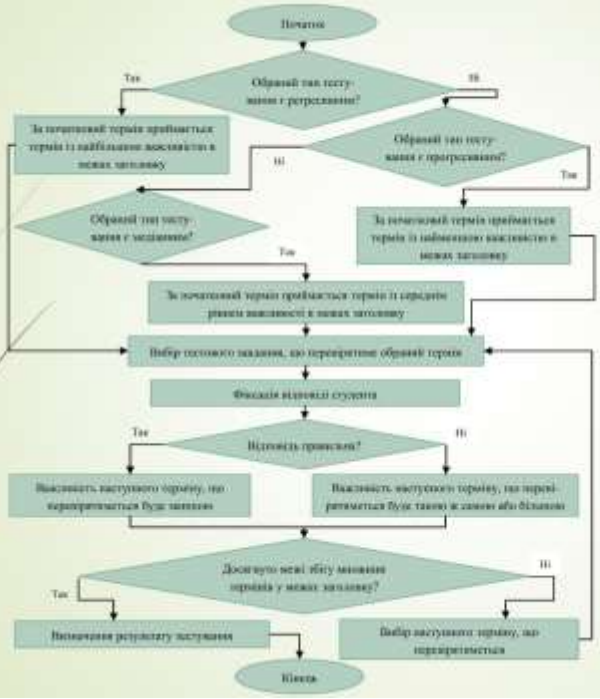


Схема методу

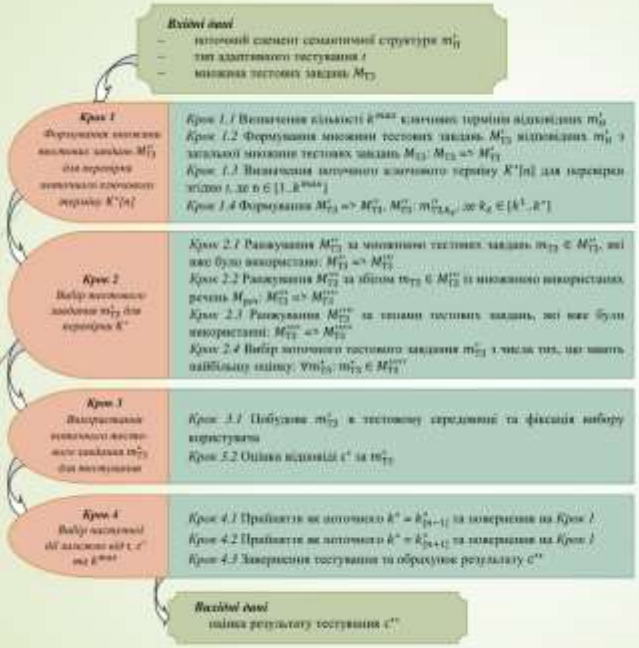
8



Метод адаптивного вибору тестових запитань

Схема адаптивного алгоритму вибору тестових запитань

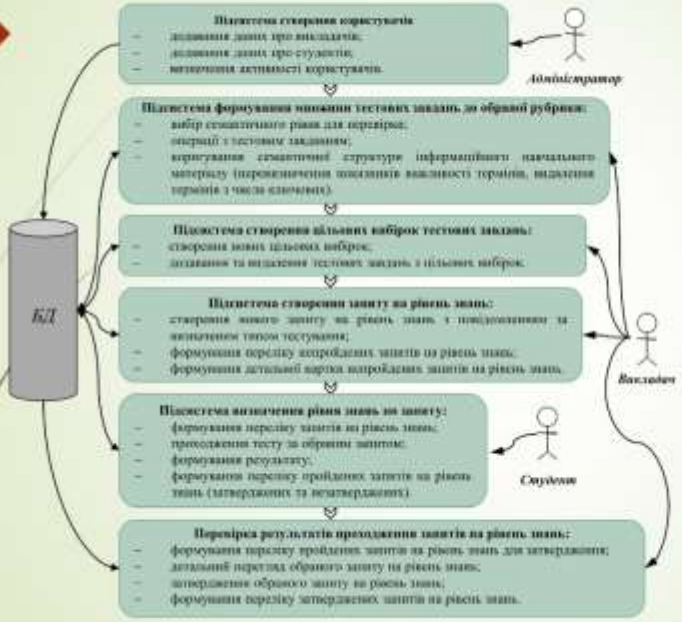
9



Інформаційна технологія адаптивного тестування рівня знань

Схема та кроки інформаційної технології

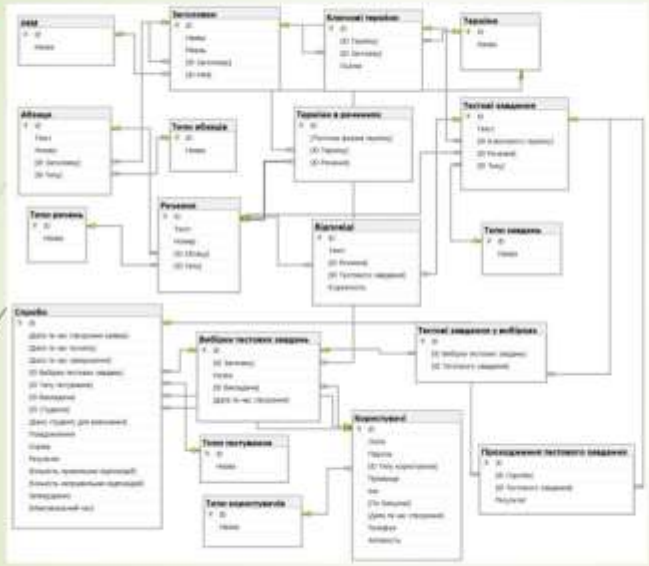
10



Інформаційна система адаптивного тестування рівня знань

Схема інформаційної системи

11



Інформаційна система адаптивного тестування рівня знань

Діаграма бази даних інформаційної системи

12

Інформаційна система адаптивного тестування рівня знань

Структура компонентів інформаційної системи

13

Практичне значення одержаних результатів

На основі розробленої інформаційної технології адаптивного тестування рівня знань було створено відповідну **інформаційну систему** адаптивного тестування рівня знань.

Проведення прикладного дослідження практичної ефективності інформаційної технології адаптивного тестування рівня знань шляхом аналізу функціональності й ефективності застосування відповідної інформаційної системи дозволило встановити, що запропоновані й реалізовані алгоритми адаптивного тестування забезпечують в середньому на 22,06 %. більш швидке проведення процесу тестування, а для визначення рівня знань студентів потрібно в середньому на 29,26 % менше тестових завдань.

14

Дослідження ефективності за часом, що витрачається під час адаптивного тестування

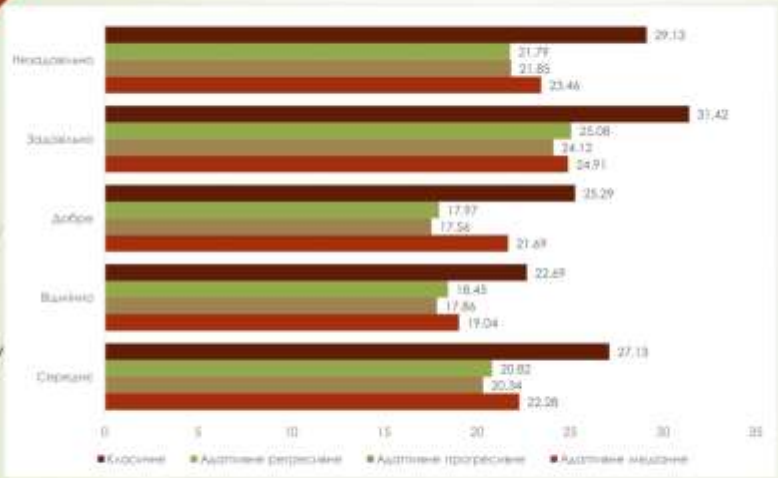
$$\Delta T1_i = \frac{TK_i - TA1_i}{TK_i} \cdot 100, \qquad \overline{\Delta T1_i} = \frac{\sum_{j=1}^{n} \Delta T1_{i,j}}{n},$$

$$\Delta T2_i = \frac{TK_i - TA2}{TK_i} \cdot 100, \qquad \overline{\Delta T2_i} = \frac{\sum_{j=1}^{n} \Delta T2_{i,j}}{n},$$

$$\Delta T3_i = \frac{TK_i - TA3_i}{TK_i} \cdot 100, \qquad \overline{\Delta T3_i} = \frac{\sum_{j=1}^{n} \Delta T3_{i,j}}{n},$$

де TK_i – час, що витрачається за класичного алгоритму тестування i-м студентом; $TA1_i$ – витрачений час на тестування за регресивного адаптивного алгоритму i-м студентом; $TA2_i$ – витрачений час на тестування за прогресивного адаптивного алгоритму i-м студентом; $TA3_i$ – витрачений час на тестування за медіанним адаптивним алгоритмом i-м студентом; $\overline{\Delta T1_i}$ – середнє значення для вибірки з n проходжень тесту за класичним та регресивним адаптивним алгоритмами; $\overline{\Delta T2_i}$ – середнє значення для вибірки з n проходжень тесту за класичним та прогресивним адаптивним алгоритмами; $\overline{\Delta T3_i}$ – середнє значення для вибірки з n проходжень тесту за класичним та медіанним адаптивним алгоритмами.

15



Середній час тестування, хв.

Дослідження ефективності за часом, що витрачається під час адаптивного тестування

16

Дослідження ефективності за кількістю одержаних тестових завдань під час адаптивного тестування

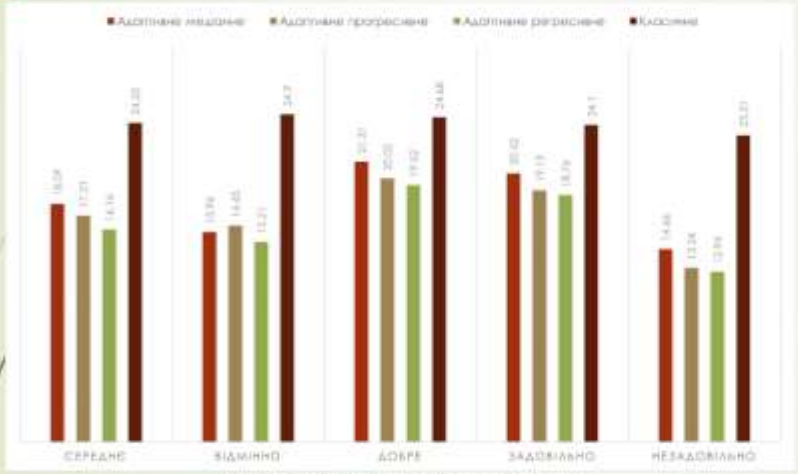
$$\Delta N1_i = \frac{NK_i - NA1_i}{NK_i} \cdot 100, \qquad \overline{\Delta N1_i} = \frac{\sum_{j=1}^{n} \Delta N1_{i,j}}{n},$$

$$\Delta N2_i = \frac{NK_i - NA2_i}{NK_i} \cdot 100, \qquad \overline{\Delta N2_i} = \frac{\sum_{j=1}^{n} \Delta N2_{i,j}}{n},$$

$$\Delta N3_i = \frac{NK_i - NA3_i}{NK_i} \cdot 100, \qquad \overline{\Delta N3_i} = \frac{\sum_{j=1}^{n} \Delta N3_{i,j}}{n},$$

де NK_i – кількість завдань виконаних при тестуванні за класичним алгоритмом i-м студентом; $NA1_i$ – кількість завдань, одержана під час тестування за регресивного адаптивного алгоритму i-м студентом; $NA2_i$ – кількість завдань, одержана під час тестування за прогресійного адаптивного алгоритму i-м студентом; $NA3_i$ – кількість завдань, одержана під час тестування за медіанного адаптивного алгоритму i-м студентом; $\overline{\Delta N1_i}$ – середнє значення для вибірки з n проходжень тесту за класичним та регресивним адаптивним алгоритмами; $\overline{\Delta N2_i}$ – середнє значення для вибірки з n проходжень тесту за класичним та прогресійним адаптивним алгоритмами; $\overline{\Delta N3_i}$ – середнє значення для вибірки з n проходжень тесту за класичним та медіанним адаптивним алгоритмами.

17



Середня кількість одержаних тестових завдань, од.

Дослідження ефективності за кількістю одержаних тестових завдань під час адаптивного тестування

18

Наукова новизна одержаних результатів

- ***Вдосконалено інформаційну модель адаптивного тесту***, яка відрізняється тим, що містить формальне подання складових предметної області термінооріентованого адаптивного тестування рівня знань.
- ***Вдосконалено метод адаптивного вибору тестових запитань***, який відрізняється тим, що використовує показники семантичної важливості ключових термінів навчального матеріалу для динамічного вибору тестових завдань у процесі тестування, використовуючи один із розроблених алгоритмів адаптивного вибору тестових завдань: регресивний, прогресивний та медіанний.
- ***Розроблено нову інформаційну технологію адаптивного тестування рівня знань***, що використовує розроблені інформаційну модель і метод адаптивного вибору тестових запитань й дозволяє за вхідними даними у вигляді пов'язаних із семантичною структурою навчального курсу тестових завдань одержувати в результаті проходження тесту вихідні дані у вигляді оцінки результату тестування, що вираховується залежно від відповідей опитуваного впродовж тестування.
- ***Розроблено нову інформаційну систему адаптивного тестування рівня знань*** за розробленою інформаційною технологією, що дозволяє працювати з тестовим матеріалом, створювати цільові вибірки тестових завдань, формувати індивідуальні запити на рівень знань, проводити адаптивне тестування за одним із доступних алгоритмів та обраховувати результати проходження запитів на рівень знань.

19

Висновки

Робота розв'язує науково-технічну задачу створення інформаційної системи для забезпечення адаптивного тестування знань. *Результатом дипломної роботи магістра* є відповідна інформаційна технологія та розроблена автоматизована система адаптивного тестування рівня знань. В ході виконання дипломної роботи магістра було вирішено наступні завдання:

- досліджено відомі підходи до адаптивного тестування рівня знань;
- вдосконалено інформаційну модель адаптивного тесту для формального подання елементів термінооріентованого адаптивного тестування рівня знань;
- вдосконалено метод адаптивного вибору тестових запитань для динамічного вибору тестових завдань у процесі тестування;
- розроблено інформаційну технологію адаптивного тестування рівня знань, що використовує розроблені модель і метод;
- розроблено інформаційну систему адаптивного тестування рівня знань на основі розробленої інформаційної технології;
- досліджено практичну ефективність інформаційної технології адаптивного тестування рівня знань шляхом аналізу функціональності й ефективності застосування відповідної інформаційної системи.

20

Апробація результатів дипломної роботи магістра та публікації

За темою дипломної роботи магістра виконано **чотири наукових публікації**.

Основні наукові та практичні результати доповідалися на **конференціях**:

- доповідь на тему «Параметризація моделі тестового завдання при автоматизованому формуванні тестів» на VII Міжнародної науково-практичної конференції «Інформаційні управляючі системи та технології ICST-ODESSA-2018» (Одеса, 17-18 вересня 2018 року, Одеський національний політехнічний університет);
- доповідь на тему «Метод формального опису елементів моделей автоматизованого формування тестових завдань» на Всеукраїнській науково-практичній конференції «Інтелектуальний потенціал – 2018» (Хмельницький, 14-16 листопада 2018 року, Хмельницький національний університет);
- доповідь на тему «Використання програмного розширення spire.doc для автоматизації роботи з цифровими документами» на XI Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук АПКН-2019» (Хмельницький, 14-15 листопада 2019 року, Хмельницький національний університет);
- доповідь на тему «Інформаційна технологія адаптивного тестування рівня знань» на XII Всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук АПКН-2020» (Хмельницький, 9-10 листопада 2020 року, Хмельницький національний університет).

21

Відео-презентація інформаційної системи

Wed Dec 02 11:28:10 EET 2020, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 11%**

	Документ		Сумарний збіг по Базі Даних	
ID: 82011 Назва: Інформаційна технологія адаптивного тестування рівня знань Додано в БД: 2020-12-02 Автора: Білоус Ганна Анатоліївна Керівники: Мазурець О.В. Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	142650	975	4257 (3%)	44 (5%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

РІШЕННЯ КАФЕДРИ

КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: **Інформаційна технологія адаптивного тестування рівня знань**

Автор: **Білоус Ганна Анатоліївна**

Спеціальність: **122 Комп'ютерні науки**

Науковий керівник: **ст. викладач Мазурець Олександр Вікторович**

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних). Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	-
3	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	-
4	Інше:	-

Підтвердження: Показник збігів є незначним і складає 8,68% (найбільша схожість 1.74% з одним джерелом). Виявлені в роботі запозичення є законними і не є плагіатом, оскільки стосуються огляду існуючих рішень, мають відповідні посилання на джерела у Переліку посилань і розміщені в розділах, які не описують безпосередньо авторське дослідження. Випадки віднесення до плагіату елементів бібліографічного опису посилань у Переліку посилань є природніми. Робота приймається до захисту.

02.12.2020

Дата

Підпис керівника

Підпис завідувача кафедри

Хмельницький національний університет

Кафедра комп'ютерних наук та інформаційних технологій

ВІДГУК ОПОНЕНТА

на дипломну роботу магістра

Магістра *гр. КНм-19-1 Білоус Ганни Анатоліївни*

На тему: *Інформаційна технологія адаптивного тестування рівня знань.*

1. Актуальність і значення теми

Контроль рівня знань повинен базуватися на змісті та матеріалах відповідних навчальних курсів. На сучасному етапі найбільш об'єктивним засобом оцінки рівня знань є тестування, що дозволяє неупереджено оцінити навчальні досягнення студентів. Комп'ютерне тестування за класичною формою має наступні проблеми: вибірка тестових завдань може не повністю покривати семантичну структуру навчального матеріалу (що потребує велику кількість тестових завдань); при оцінюванні не враховується зростання рівня складності тестових завдань через використання в контенті різних за рівнем семантичної значущості термінів; використання фіксованого обсягу тестових завдань незалежно від успішності процесу тестування. Тому є доцільною розробка нової інформаційної технології, яка у процесі тестування адаптивно перевірятиме саме розуміння складових семантичної структури навчального матеріалу.

2.Оцінка якості та достовірності проведених досліджень.

У роботі коректно проведено достатню кількість досліджень, які підтверджують достовірність одержаних результатів. Одержані результати добре корелюють з результатами наукових робіт за подібною тематикою.

3. Оцінка запропонованих заходів та пропозицій, практичної цінності та ефективності.

Згідно проведених досліджень, розроблені алгоритми адаптивного тестування забезпечують в середньому на 22,06 % більш швидке проведення процесу тестування й при цьому потребують в середньому на 29,26 % меншу кількість тестових завдань. Це доводить, що розроблену інформаційну технологію можна ефективно використовувати з метою автоматизації процесу адаптивного тестування рівня знань студентів.

4. Загальний висновок та оцінка

Всі поставлені завдання виконані повністю і на високому рівні. За своєю структурою, практичними цінностями, поставленій меті та вирішеними завданнями робота відповідає вимогам вищої школи і вимогам, що пред'являються до освітньо-кваліфікаційного рівня «магістр», а її автор Білоус Г.А. заслуговує присвоєння кваліфікації магістра з комп'ютерних наук.

Робота заслуговує на оцінку *«відмінно».*

Опонент *Марчишин В.В., к.т.н., проф.*