ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій

Кафедра телекомунікацій, медійних та інтелектуальних технологій

ДИПЛОМНА РОБОТА

Другий (Магістерський)
Освітній рівень

Галузь знань 17 Електроніка, автоматизація та електронні комунікації
Шифр і назва галузі

Спеціальність 172 Електронні комунікації та радіотехніка
Шифр і назва спеціальності

на тему Інфокомунікаційна система екзаменаційно-тренінгового центру іноземних мов

ДРЕКР. 024025.01.03.ПЗ

Виконав: студент 2 курсу, група $ЕКР_{м}$-24-1 _____ М.М. Коржан
підпис Ініціали, прізвище

Керівник: д-р техн. наук, проф. _____ С.К. Підченко
підпис Ініціали, прізвище

До захисту допускаю:

Зав. кафедри: д-р техн. наук, проф. _____ С.К. Підченко
підпис Ініціали, прізвище

8 12 2025 р.

Хмельницький, 2025

Хмельницький національний університет

Факультет *інформаційних технологій*
Кафедра *телекомунікацій, медійних та інтелектуальних технологій*
Освітній рівень *другий (магістерський)*
Галузь знань *17 – Електроніка, автоматизація та електронні комунікації*
Спеціальність *172 – Електронні комунікації та радіотехніка*
Освітня-професійна програма *Електронні інформаційно-комунікаційні системи та мережі*

ЗАТВЕРДЖУЮ
Зав. кафедрою ТМІТ
С.К. Підченко
«1» 09 2025р.

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ

Коржан Микола Миколайович

1 Тема роботи: *Інфокомунікаційна система екзаменаційно-тренінгового центру іноземних мов*
керівник роботи Підченко Сергій Костянтинович, д.т.н., професор
Затверджено наказом по університету від «25» 08 2025р. № 65.
2 Строк подання студентом роботи на кафедру: 1 грудня 2025р.
3 Вихідні дані (характеристика об'єкта, умов дослідження та ін.)
Мета роботи: підвищення надійності програмних компонентів інфокомунікаційної системи екзаменаційно-тренінгового центру іноземних мов.
Об'єкт дослідження: інформаційно-комунікаційні процеси електронної комунікації між учасниками екзаменаційно-тренінгового центру іноземних мов
Предмет дослідження: програмне забезпечення та метод підвищення надійності компонентів інфокомунікаційної системи екзаменаційно-тренінгового центру іноземних мов.
4. Зміст пояснювальної записки (перелік питань, що їх належить розробити):
1. Огляд існуючих засобів забезпечення функціонування інфокомунікаційних систем освіти;
2. Моделі та методи інфокомунікаційної системи екзаменаційно-тренінгового центру іноземних мов;
3. Розробка варіантів використання системи та бази даних;
4. Програмна реалізація та розгортання компонентів системи на обчислювальних вузлах.

Завдання отримав _____
Науковий керівник _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів (розділів) дипломної роботи	Строк виконання етапів дипломної роботи	Примітка
1	*Вибір теми роботи*	до 01.09.25	*обрано*
2	*Аналіз вихідних даних*	01.09.25 – 05.09.24	*виконано*
3	*Постановка технічного*		
	завдання	06.09.25 – 10.09.25	*виконано*
4	*Підготовка вступу*		
	та 1-го розділу роботи	11.09.25 – 20.09.25	*виконано*
5	*Підготовка матеріалів*		
	2-го розділу	21.09.25 – 30.09.25	*виконано*
6	*Оформлення та подання*		
	тез для доповіді	01.10.25 – 03.10.25	*виконано*
7	*Підготовка матеріалів*		
	3-го розділу	04.10.25 – 20.10.25	*виконано*
8	*Підготовка матеріалів*		
	4-го розділу	21.10.25 – 10.11.25	*виконано*
9	*Виправлення недоліків та*		
	робота над зауваженнями	11.11.25 – 15.11.25	*виконано*
10	*Підготовка презентаційних*		
	матеріалів для захисту	16.11.25 – 20.11.25	*виконано*
11	*Перевірка на плагіат*		
	та офіційне опонування	24.11.25	*пройдено*
12	*Подання до захисту*	1.12.25	*подано*

Студент _____ Коржан М.М.
Підпис Ініціали, прізвище

Керівник роботи _____ Підченко О.К.
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема дипломної роботи: Інфокомунікаційна система екзаменаційно-тренінгового центру іноземних мов.

Автор роботи: Коржан Микола Миколайович

Керівник роботи: Підченко Сергій Костянтинович

Пояснювальна записка: 81 сторінка, 43 рисунки, 10 таблиць, 43 джерела посилання, 5 додатків.

Графічна частина: 25 презентаційних слайдів.

КЛЮЧОВІ СЛОВА: ІНФОКОМУНІКАЦІЙНА СИСТЕМА, ЕЛЕКТРОННІ КОМУНІКАЦІЇ, НАДІЙНІСТЬ ПРОГРАМНИХ КОМПОНЕНТІВ.

Мета роботи: підвищення надійності програмних компонентів інфокомунікаційної системи екзаменаційно-тренінгового центру іноземних мов

Об'єкт дослідження: інформаційно-комунікаційні процеси електронної комунікації між учасниками екзаменаційно-тренінгового центру іноземних мов

Предмет дослідження: програмне забезпечення та метод підвищення надійності компонентів інфокомунікаційної системи екзаменаційно-тренінгового центру іноземних мов.

В першому розділі роботи проведено аналіз існуючих ІКС для вивчення іноземних мов та здачі екзаменів для підтвердження рівня володіння іноземною мовою; виділено основні інфокомунікаційні процеси, які є суттєвими для практичної реалізації ПЗ системи, а також визначено основних користувачів; сформульовано специфікацію функціональних та нефункціональним вимог та виконано постановку задачі роботи.

В другому розділі робити була розроблена формальна модель ІКС на основі методики структурного аналізу. Були визначені вхідні, вихідні та керуючі потоки даних системи, а також виконана детальна декомпозиція системи; запропонована модель підвищення надійності системи шляхом резервування програмних компонентів підключення до БД.

В третьому розділі робити була розроблена модель варіантів використання системи з визначенням головних акторів та ініційованих ними прецедентів. На основі моделей, які були представлені в другому розділі, та керуючись технічним завданням і відомостями з предметної області, була розроблена схема БД, яка в свою чергу була імплементована на реляційній СУБД PostgreSQL.

В четверному розділі роботи було описано процес програмної реалізації веб-додатку інфокомунікаційної системи екзаменаційно-тренінгового центру іноземних мов на базі технологій Java EE. Було розроблено мінімально достатній програмний продукт, що дозволяє виконувати базові функції згідно технічного завдання.

ABSTRACT

In the first section of the work, an analysis of existing ICTs for learning foreign languages and passing exams to confirm the level of proficiency in a foreign language was conducted; the main infocommunication processes that are essential for the practical implementation of the system software were identified, and the main users were identified; a specification of functional and non-functional requirements was formulated and the task of the work was stated.

In the second section, a formal model of ICTs was developed based on the structural analysis methodology. The input, output and control data flows of the system were determined, and a detailed decomposition of the system was performed; a model for increasing the reliability of the system by reserving program components for connecting to the database was proposed.

In the third section, a model of system use cases was developed with the definition of the main actors and the precedents initiated by them. Based on the models presented in the second section, and guided by the technical task and information from the subject area, a database schema was developed, which in turn was implemented on the PostgreSQL relational DBMS.

The fourth section of the work describes the process of software implementation of the web application of the infocommunication system of the foreign language examination and training center based on Java EE technologies. A minimally sufficient software product was developed that allows performing basic functions according to the technical specifications.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API	–	Application programming interface
CEFR	–	Common european framework of reference for languages
CRM	–	Customer relationship management
CSS	–	Cascading style sheets
DAO	–	Data access object
EJB	–	Enterprise javabeans
HTTP	–	Hypertext markup language
JDBC	–	Java database connectivity
JEE	–	Java enterprise edition
JMS	–	Java message service
JSF	–	Javaserver Faces
JVM	–	Java virtual machine
LTS	–	Long time support
MVC	–	Model-view-controller
SADT	–	Structured analysis and design technique
SDK	–	Software development kit
SQL	–	Structured query language
БД	–	База даних
ЕКЦ	–	Екзаменаційно-тренінговий центр
ЗО	–	Заклад освіти
ІКС	–	Інфокомунікаційна система
ОС	–	Операційна система
ПЗ	–	Програмний засіб/забезпечення
ПК	–	Персональний комп’ютер
СУБД	–	Система керування базою даних
ТЗ	–	Технічне завдання
ШІ	–	Штучний інтелект

ВСТУП

Технології та засоби електронних комунікацій сприяють автоматизації процесів функціонування систем різного призначення. Зокрема, в освітній діяльності доцільно керувати навчальним процесом через централізоване програмне забезпечення інфокомунікаційної системи. В процесі вивчення іноземних мов та проведення іспитів виникають проблеми, що пов’язані із керуванням взаємодії учасників освітнього процесу, що особливо актуально для дистанційної форма навчання. Крім того, сучасні інформаційні технології та мережа Інтернет надає цілодобовий доступ до інформаційного ресурсу з розкладом, графіками іспитів, домашнім завданим тощо.

Актуальність проблеми. Незважаючи на доволі об’ємну матеріально-технічну базу та методичні напрацювання щодо розробки програмного забезпечення для інфокомунікаційних освітніх систем, актуальною залишається проблема забезпечення надійного та безвідмовного функціонування останніх. Особливо гостро ця проблема ставиться у випадку суто дистанційного (онлайн) навчання, для великої кількості учасників, що одночасно здійснюють запити до системи. В таких випадках відмова одного або декількох ключових програмних компонентів призводять до відмови усієї системи.

Мета роботи: підвищення надійності програмних компонентів інфокомунікаційної системи екзаменаційно-тренінгового центру іноземних мов.

Для досягнення поставленої мети сформульовані та вирішені такі ***задачі дослідження***:

1. Провести аналіз предметної області, актуальних публікацій згідно теми, розглянути переваги та недоліки існуючих програмних систем.

2. Розробка формальної моделі інфокомунікаційної системи екзаменаційно-тренінгового центру іноземних мов з урахуванням головних процесів та учасників. Провести декомпозиція системи у відповідності до структурного підходу з визначенням вхідних та вихідних потоків даних, параметрів, а також керуючих сигналів.

3. Розробка моделі надійності системи на основі резервування найбільш вразливих програмних компонентів.

4. Розробка моделі варіантів використання ІКС на основі функціональних вимог; алгоритмів та моделювання інфокомунікаційних процесів взаємодії акторів та системи; моделі даних та практична реалізація бази даних.

5. Програмна реалізація системи згідно триступеневої архітектури з дотриманням нефункціональних вимог. Реалізація моделі даних та компонентів взаємодії з БД, компонентів автоматизації інфокомунікаційних процесів системи та елементів інтерфейсу.

Об'єкт дослідження: інформаційно-комунікаційні процеси електронної комунікації між учасниками екзаменаційно-тренінгового центру іноземних мов.

Предмет дослідження: програмне забезпечення та метод підвищення надійності компонентів інфокомунікаційної системи екзаменаційно-тренінгового центру іноземних мов.

Наукова новизна та практична значення.

1. Отримав подальшого розвитку метод ковзного резервування програмних компонентів доступу до бази даних серверної частини програмного забезпечення, що дозволяє збільшити середній час напрацювання на відмову до 154 годин (більше 6 днів), що понад у 12 разів більше за відповідний показник для лише одного елементу із базовими показниками надійності, з середнім часом відновлення до 7 хвилин.

2. Розроблено прототип програмного забезпечення інфокомунікаційної системи екзаменаційно-тренінгового центу іноземних мов на базі набору технологій Java EE, що дозволяє виконувати базові функції згідно технічного завдання. Додаток може бути використаний для тестування процесів функціонування відповідних систем з метою оптимізації бізнес-процесів освітньої діяльності.

Апробація результатів: за матеріалами роботи підготовлені тези доповіді для XVII всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук», АПКН–2025.

1 ОГЛЯД ІСНУЮЧИХ ЗАСОБІВ ЗАБЕЗПЕЧЕННЯ ФУНКЦІОНУВАННЯ ІНФОКОМУНІКАЦІЙНИХ СИСТЕМ ОСВІТИ

1.1 Огляд програмно-апаратних засобів

Забезпечення функціонування інфокомунікаційних систем (ІКС), в тому числі освітніх, покладено на всебічне використання програмних, апаратних, мережевих засобів [1-3]. Незалежно від масштабу та форми власності закладу освіти (ЗО), такі технології покращують якість надання послуг здобувачам освіти (студентам) за рахунок автоматизації процесів. Наприклад, для приватних шкіл вивчення іноземної мови типовим є координація студентів, викладачів та менеджерів за допомогою систем взаємодії з клієнтами (Customer Relationship Management, CRM), а ефективна підтримка навчального процесу здійснюється засобами платформ дистанційного навчання (наприклад, Moodle [4]).

Відповідно до досліджень [5], використання ІКС для вивчення іноземних мов сприяє підвищенню мотивації, ефективності взаємодії учасників освітнього процесу, а також дозволяє моделювати реальні комунікаційні ситуації в штучному середовищі.

Такі інструменти особливо корисні для вивчення іноземних мов, де акцент робиться на розмовній практиці, аудіюванні та культурному обміні, а також для атестації знань студентів через, автоматизовану оцінку та персоналізований зворотний зв’язок [6].

Практика свідчить, що застосування ІКС в освіті сприяє розвитку від монотонного заучування до появи інтегральних методів, що суттєво підвищують якість освітніх послуг [7].

Природною є класифікація засобів підтримки ІКС на апаратні та програмні [7, 8].

Апаратні засоби є базою для телекомунікацій та електронних комунікацій, що надають доступ до онлайн-ресурсів та забезпечують відеозв’язок. Так, наприклад, в ЗО вони використовуються для підтримки дистанційного навчання,

особливо в приватних закладах з гнучкими моделями – засоби аудіовізуальної комунікації для підтримки навчального процесу та організації онлайн-тестування з фіксацією відповідей засобами штучного інтелекту (ШІ) [9].

В Таблиці 1.1 подана загальна класифікація апаратних засобів підтримки ІКС, підкріплена прикладами та стислим описом.

Таблиця 1.1 – Апаратні засоби ІКС

Категорія	Опис та приклади	Застосування в освіті
Обчислювальні пристрої	Комп’ютери, ноутбуки, планшети, мейнфрейми (для централізованих систем).	Доступ до онлайн-курсів мов; проведення тестів з оцінкою засобами ШІ.
Комунікаційне обладнання	Камери, мікрофони, гарнітури, мережеві пристрої (маршрутизатори LAN/WAN).	Відеоконференції для розмовної практики мов; запис відповідей для атестації.
Дисплейні та інтерактивні пристрої	Розумні дошки, проектори, гарнітури доповненої реальності, мультимедійні комп’ютери.	Використання прогресивних технологій віртуальної та доповненої реальності; інтерактивні тести на екранах.

З іншого боку, програмні засоби (ПЗ) сфокусовані на комунікації, персоналізації та оцінці. Вони інтегрують технології Web 2.0 для соціальних мереж, що є ефективним для вивчення мов шляхом аутентичної взаємодії (наприклад, спілкування з носіями мови) [5]. Для оцінки знань студентів актуальним є використання ПЗ в реальному часі з оцінкою відповідей засобами ШІ в хмарному середовищі [6].

В Таблиці 1.2 подана узагальнена класифікація ПЗ з прикладами та стислим описом [10–14].

Таблиця 1.2 – Програмні засоби ІКС

Категорія	Опис та приклади	Застосування в освіті
Системи управління навчанням (СКН)	Платформи для організації навчання: Google Classroom, Moodle, Blackboard.	Курси іноземних мов з модулями; онлайн-атестація з контролем прогресу.
Інструменти для комунікації	Відеоконференції: Zoom, Microsoft Teams, Google Meet; синхронні (інтерактивні чати) та асинхронні (e-mail, форуми).	Практика розмовної мови в реальному часі; групові проекти для атестації.
Додатки для вивчення іноземних мов	CALL-програми: Duolingo, Rosetta Stone, Ellis, Dynamic English; мультимедіа з розпізнаванням мови (SpeechViewer).	Самостійне вивчення граматики/словника; інтегративні симуляції для практики.
Інструменти атестації та ШІ	Платформи оцінювання: Quizlet, Edulastic; ШІ-інструменти для зворотного зв’язку та перевірка граматики.	Формативні тести з даними; персоналізована оцінка прогресу студентів.

Акцентуючи увагу саме на програмних засобів для вивчення іноземних мов та закріплення досягнутих результатів шляхом здачі іспитів, проведемо огляд відомих онлайн платформ, що забезпечують відповідні функції.

Онлайн-платформи для вивчення іноземних мов часто поєднують інтерактивні уроки, персоналізоване навчання та інструменти оцінки, такі як тести на рівень володіння мовою, наприклад, відповідно до Common European Framework of Reference for Languages (CEFR) [15], та підготовку до іспитів. Це

дозволяє фіксувати здобуті результати через формальну атестацію, наприклад, сертифікати або офіційні тести.

Серед найбільш популярних серед викладачів та студентів платформ, що реалізують вищеописані вимоги та функції(уроки, вправи, комунікація), а також підтримують оцінку знань (тести, підготовка до іспитів, сертифікація), виділимо такі ІКС:

- Lingoda [16],
- Preply [17],
- Italki [18],
- Duolingo [19],
- Babbel [20].

Наведемо короткий опис вищезазначених ІКС відповідно інформації з офіційних сайтів.

Онлайн-платформа Lingoda [16] призначена для вивчення іноземних мов з акцентом на живі заняття з носіями мови Підтримуються курси англійської, німецької, французької, іспанської та бізнес-англійської. До ПЗ платформи входять інтерактивні віртуальні класи на базі Zoom, де проводяться групові (3-5 студенти) або індивідуальні уроки. Курси структуровані за CEFR (A1-C2), з фокусом на розмовну практику, граматику, письмо та аудіювання. Матеріали включають інтерактивні PDF-презентації, домашні завдання та словники, які доступні для завантаження. Для закріплення результатів Lingoda пропонує тести на визначення рівня володіння мовою перед початком навчання та сертифікати CEFR після завершення кожного рівня (зазвичай 50-60 занять), які визнаються для академічних і професійних цілей. Платформа також підтримує підготовку до офіційних іспитів, таких як Goethe-Zertifikat з французької мови [22] та DELF з німецької мови [23], через проходження спеціалізованих модулів. Вартість залежить від типу підписки: групові заняття коштують від €10 за урок, індивідуальні – від €20 за урок, з гнучкими пакетами (4-40 занять на місяць). Пробний період включає 7 днів із можливістю повернення коштів [16]. Інтерфейс сайту Lingoda показано на Рисунку 1.1.

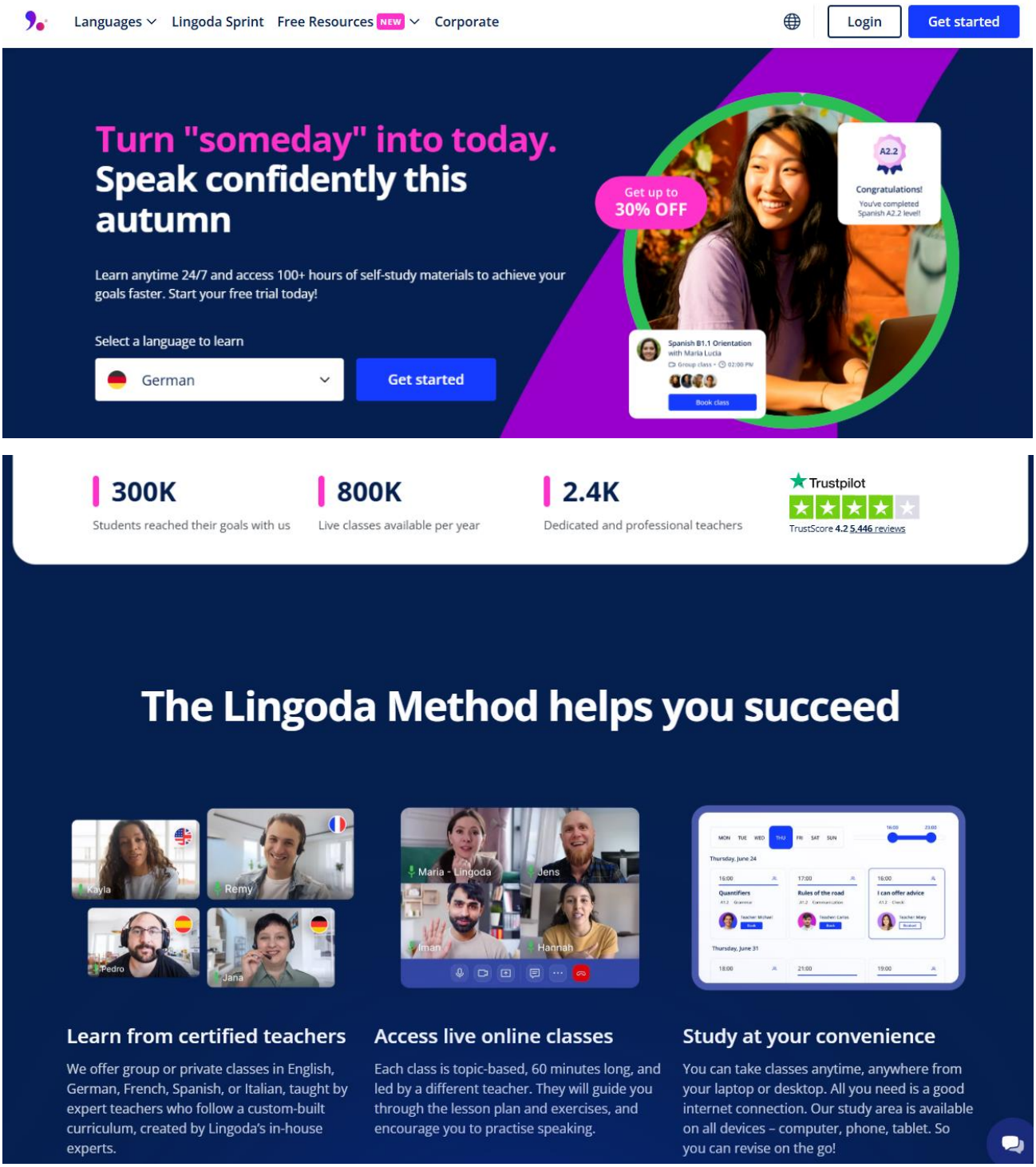


Рисунок 1.1 – Інтерфейс сайту платформи Lingoda

Іншим прикладом ІКС відповідно до предметної області роботи є платформа Preply [17], яка використовується для вивчення понад 90 мов (включаючи англійську, французьку, іспанську та німецьку) через індивідуальні уроки з репетиторами-носіями. Вона включає в себе віртуальні класи з інструментами, такими як інтерактивні дошки, флеш-карти, словники та

інструменти ШІ для нотаток та домашніх завдань. Уроки персоналізовані під конкретні цілі учня з гнучким розкладом. Для закріплення результатів платформа пропонує безкоштовні тести на рівень володіння (для англійської, іспанської, французької, німецької) за CEFR, а також спеціалізовану підготовку до іспитів, як DELF, DALF (французька мова) та TELC (німецька мова) за індивідуальними планами. Вартість уроків становить від $15 до $50 [17]. Інтерфейс сайту Preply показано на Рисунку 1.2.

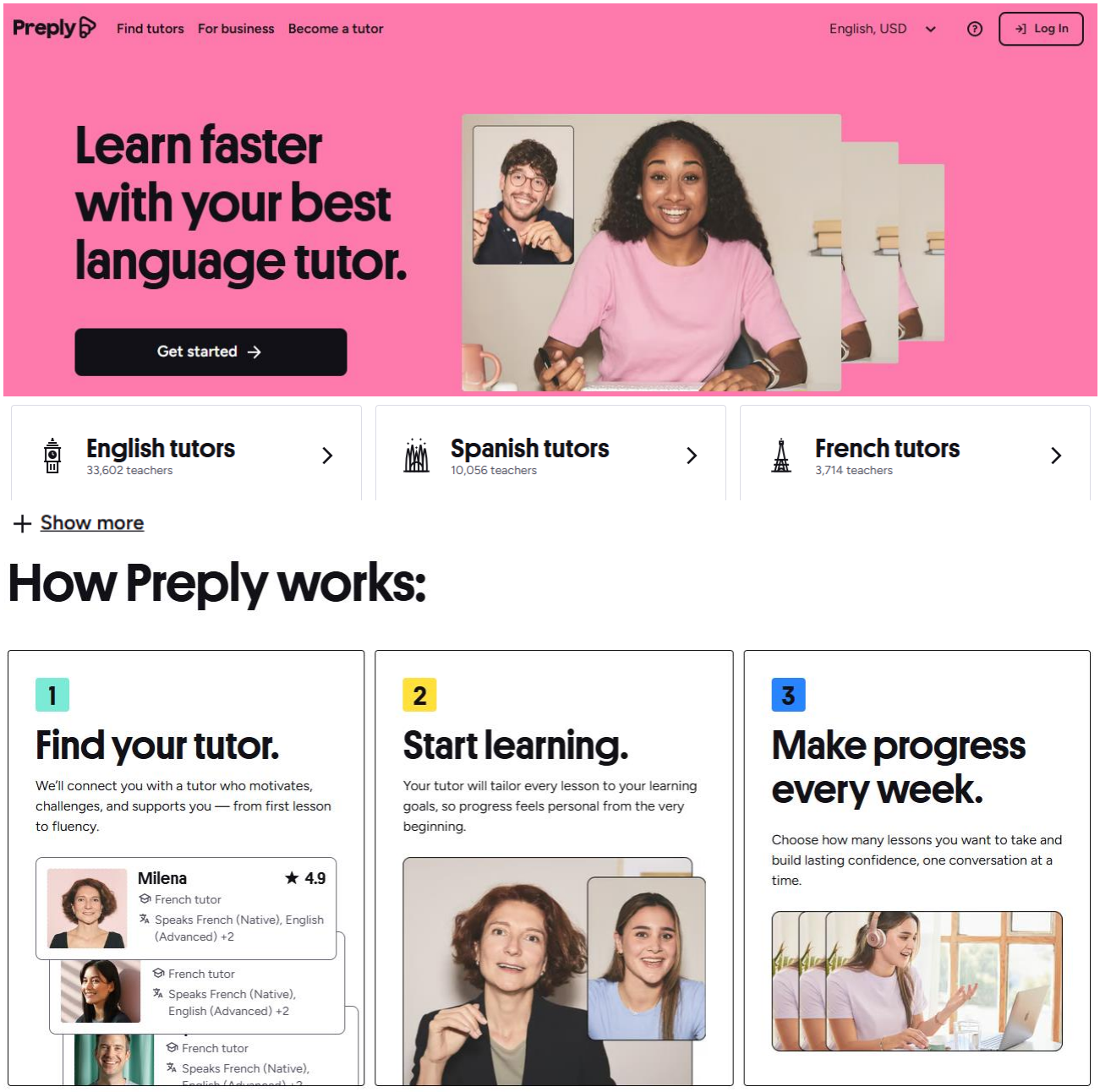


Рисунок 1.2 – Інтерфейс сайту платформи Preply

ІКС Italki [18] підтримує вивчення понад 150 мов (англійська, японська, іспанська, китайська тощо) з сертифікованими викладачами. Студентам надаються персоналізовані уроки, групові онлайн-класи, спільнота для безкоштовної практики та обміну досвідом з іншими студентами з понад 190 країн. Уроки адаптовані під бюджет і розклад, з фокусом на розмовну практику. Для атестації доступний безкоштовний тест на рівень володіння для ключових мов (англійська, іспанська, французька, німецька, японська тощо), що допомагає відстежувати прогрес і закріплювати знання. Платформа підходить для всіх рівнів, з гнучким ціноутворенням залежно від потреб студентів та викладачів. Інтерфейс сайту Italki показано на Рисунку 1.3.

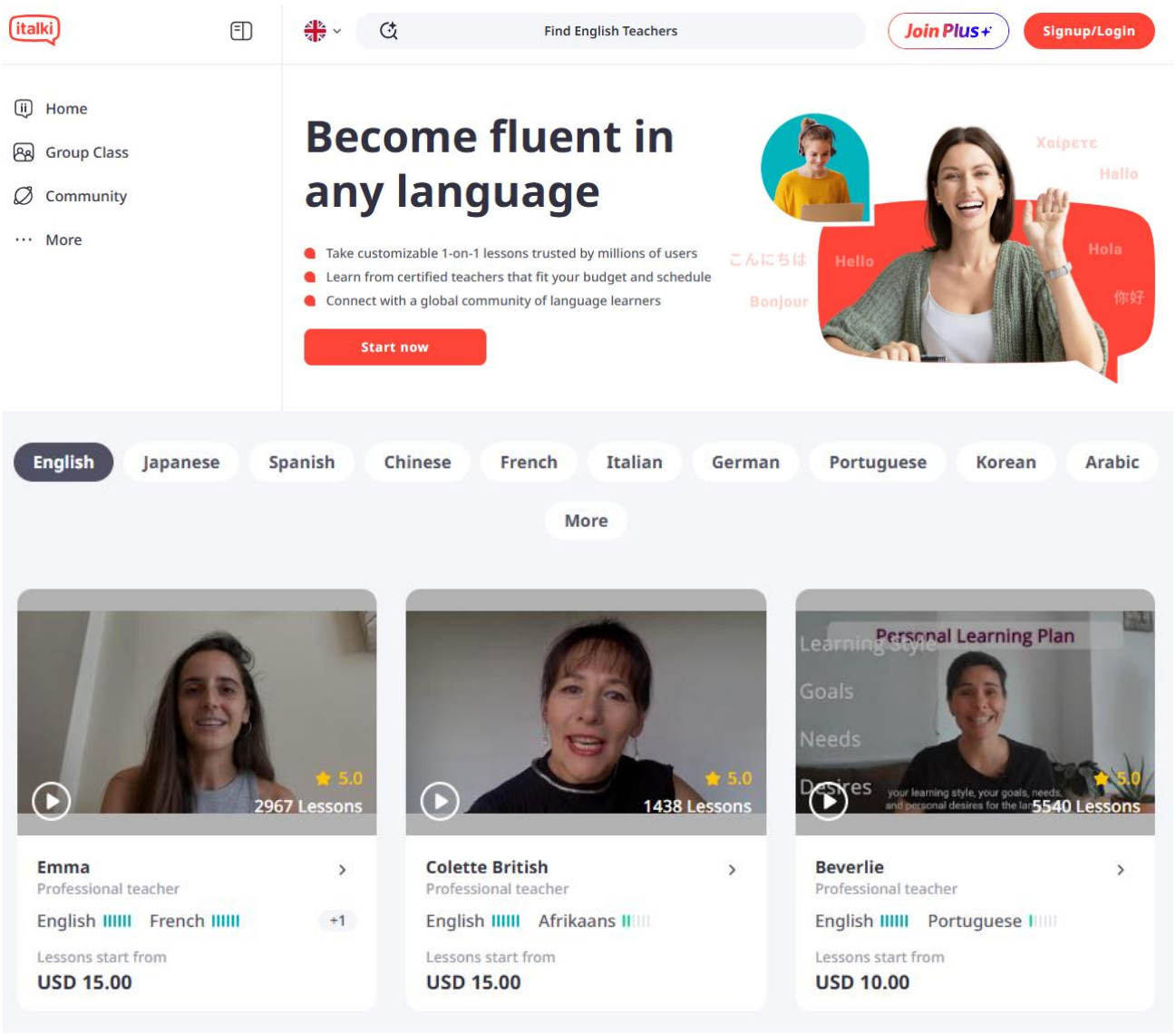


Рисунок 1.3 – Інтерфейс сайту платформи Italki

Платформа Duolingo [19] – ігровий ресурс для вивчення 43 мов (англійська, французька, іспанська, німецька, японська тощо) з акцентом на щоденні короткі уроки. Студентам пропонуються інтерактивні вправи з елементами повторення, аудіо від носіїв, історії та фрази для реальних ситуацій, а також функції ШІ для рольових ігор і пояснень помилок. Програми курсів поділені за CEFR. Для закріплення результатів пропонуються проходження Duolingo English Test (DET) – онлайн-іспиту на знання англійської мови (вартістю $70), що оцінює читання, письмо, усну мову та аудіювання, а також внутрішній Duolingo Score для відстеження прогресу. Базова версія безкоштовна, з можливістю подальшої преміум-підписки для отримання додаткових функцій [19]. Інтерфейс сайту Duolingo показано на Рисунку 1.4.

Рисунок 1.4 – Інтерфейс сайту платформи Duolingo

Онлайн-платформа Babbel – призначення для вивчення 14 мов (іспанська, французька, німецька, італійська, португальська тощо) з фокусом на розмовну практику. Вона пропонує включають короткі уроки (до 15 хв), аудіо від носіїв, розпізнавання мови для тренування вимови, подкасти, бізнес-курси та Babbel Live – групові/приватні онлайн класи з сертифікованими викладачами. Курси адаптовані за темами (подорожі, робота тощо). Для атестації знань пропонується тест для визначення рівня (CEFR A1-C1), Babbel English Test (у партнерстві з Cambridge English, вартістю €39) для сертифікації слухання та читання від A1 до B1, з сертифікатом для резюме чи підготовки до іспитів [20]. Інтерфейс сайту Babbel показано на Рисунку 1.5.

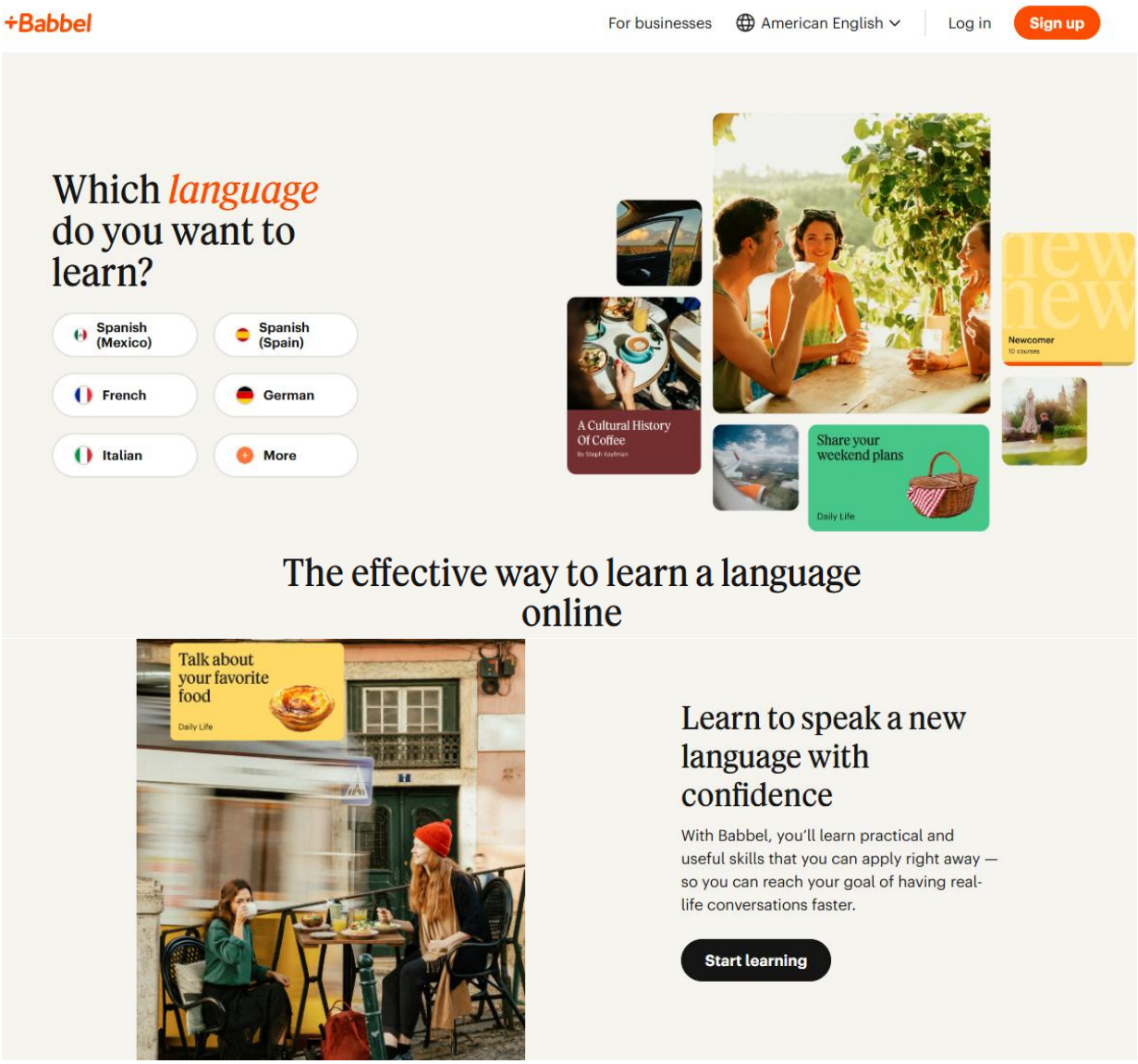


Рисунок 1.5 – Інтерфейс сайту платформи Babbel

Результати порівняльного аналізу вищеописаних онлайн-платформ для вивчення іноземних мов та атестації знань зведено до Таблиці 1.3.

Таблиця 1.3 – Порівняльна таблиця п’яти відомих онлайн платформ для вивчення іноземних мов та атестації знань

Назва	Кількість мов	Програмні засоби	Функції	Атестація знань	Орієнтовна вартість
1	*2*	*3*	*4*	*6*	*7*
Lingoda	5 (англійська, німецька, франц., іспанська, бізнес-англійська)	Віртуальні класи Zoom, інтеракт. PDF, словники, домашні завдання.	Груп. та інд. уроки з носіями, рівні A1-C2, розм. практика, граматика, письмо, аудіювання.	Тест на рівень (CEFR), сертифікати після завершення рівня (50-60 занять), підготовка до іспитів.	Групові: від $11/урок, індивідуальні: від $20/урок, пакети 4-40 занять/міс, 7-денний пробний період.
Preply	Понад 90 (англійська, франц., іспанська, німецька тощо)	Віртуальний клас, дошки, флеш-карти, ШІ-аналіз для нотаток та словників, домашні завдання.	Індивід. уроки з репетиторами-носіями, гнучкий розклад, індивідуал. плани для розмовної практики та граматики.	Безкоштовн. тест рівня мови, підготовка до DELF, TELC, DALF з прикладами завдань іспитів.	Від $15-$50/урок, місячні підписки, пробний урок.
Italki	Понад 150 (англійська, японська, іспанська, китайська тощо)	Віртуальний клас, чат, спільнота для практики, записи уроків.	Індивід. та груп. роки, гнучкий розклад, фокус на розмовну практику, спільнота для обміну знань.	Безкоштовн. тест рівня мови, відстеження прогресу.	Вартість залежить від викладача (від $10/урок), без фіксованої підписки.

Кінець Таблиці 1.3

1	*2*	*3*	*4*	*6*	*7*
Duolingo	43 (англійська, франц., іспанська, німецька, японська тощо	Інтерактивні вправи, аудіо від носіїв, AI-рольові ігри, історії, пояснення помилок,	Ігрові уроки, курси за CEFR, фокус на словник, граматику, аудіювання, розмовні фрази.	Duolingo English Test ($70, слухання/ читання/ говоріння/ письмо, Duolingo Score.	Безкоштовна базова версія, преміум-підписка від $7/міс.
Babbel	14 (іспанська, франц., німецька, італійська тощо)	Інтерактивні уроки, розпізнаван. мови, подкасти, Babbel Live (віртуальний клас), тематичні курси.	Короткі уроки, аудіо від носіїв, Babbel Live для груп. та інд. занять.	Тест на рівень (CEFR A1-C1), Babbel English Test ($40, слухання/читання, A1-B1, сертифікат), підготовка до іспитів.	Від $22/міс, безкоштовний пробний урок для Babbel Live.

1.2 Аналіз інфокомунікаційних процесів та головних вимог

Узагальнивши відомості щодо відомих онлайн платформ, що широко використовуються серед учнів та викладачів іноземних мов, можна виділити такі інфокомунікаційні процеси, що виникають в навчальному процесі:

1) навчання студента на курсі (Рисунок 1.6), що передбачає реєстрацію, попередню оцінку рівня підготовки студента, персональні побажання та графік навчання.

2) атестація знань студента (Рисунок 1.7) з метою підтвердження рівня володіння іноземною мовою з отриманням відповідного сертифікату. Проведення екзаменів відбувається з урахуванням програми навчання на курсах,

однак є незалежним процесом. Таким чином, студенту пропонується спочатку пройти курс навчання, по завершенню якого здати екзамен на підтвердження рівня, або одразу здати іспит на підтвердження рівня.

3) процеси адміністрування та підтримки навчального процесу (внутрішні процеси системи). До них належить підготовка та публікація доступних курсів, розробка графіку проведення іспитів, призначення викладачів та екзаменаторів, реєстрація нових студентів тощо.

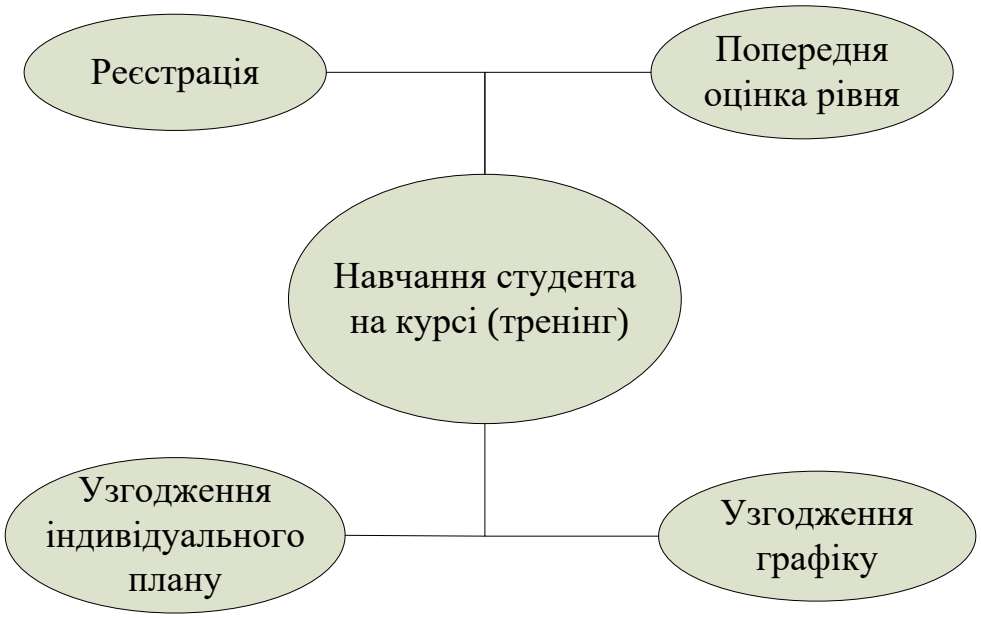


Рисунок 1.6 – Процес «Навчання студента на курсі»

Відповідно до визначених вище головних процесів ІКС, виділимо також відповідних ініціаторів цих процесів – головних користувачів. Для досліджуваної системи це:

1) студент, який реєструється на курси та керує своїми курсами згідно індивідуального плану;

2) екзаменований – студент, що вже пройшов відповідний курс підготовки, або людина, яка володіючи відповідним рівнем знань бажає підтвердити його офіційним сертифікатом;

3) адміністратор – співробітник екзаменаційно-тренінгового центру (ЕКЦ), або група співробітників з розмежуванням відповідальності, що виконує (виконують) функції реєстрації нових студентів та екзаменованих осіб в системі, керуванням розкладами учбових занять та екзаменів.

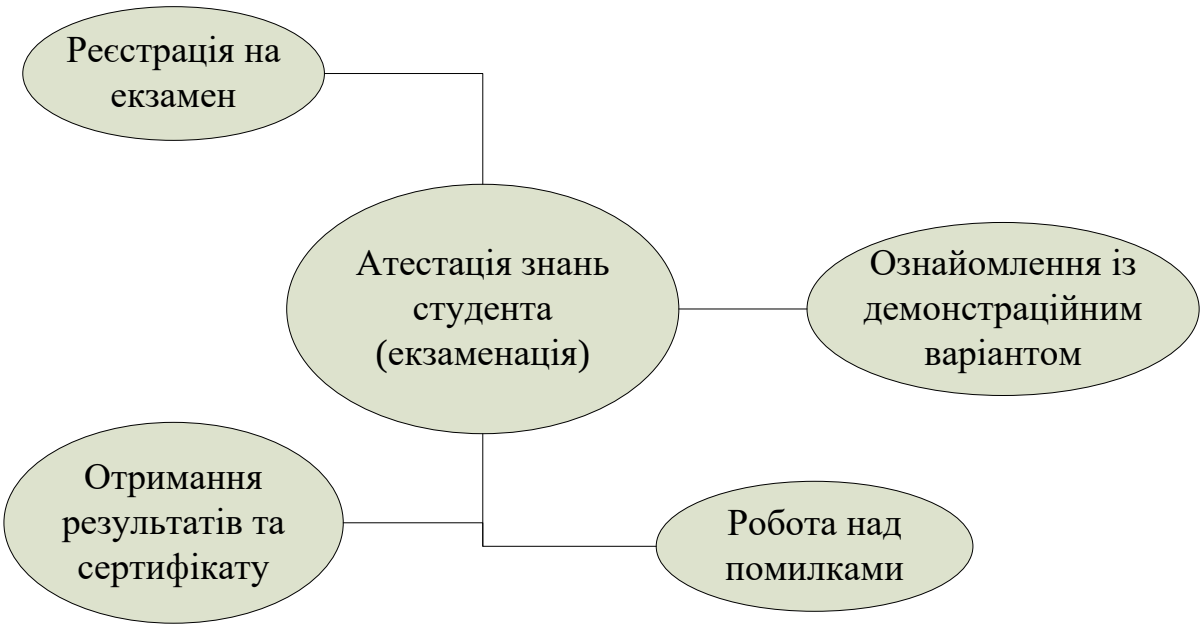


Рисунок 1.7 – Процес «Атестація знань»

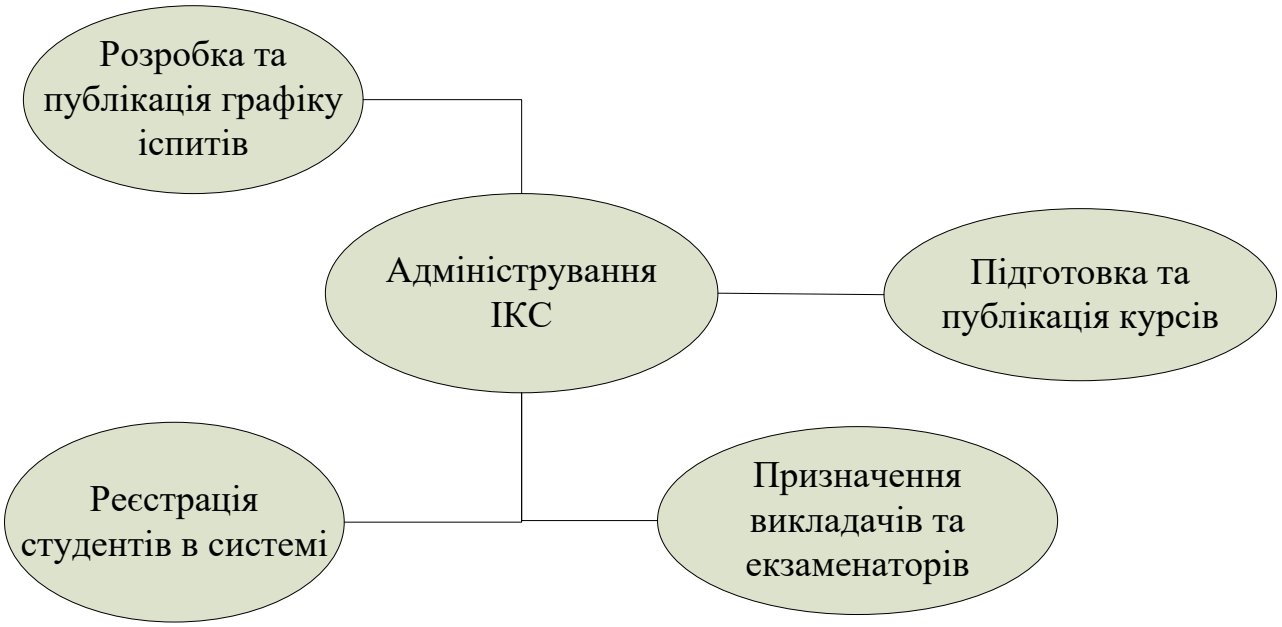


Рисунок 1.8 – Процес «Адміністрування ІКС»

Вимоги (функціональні та нефункціональні) до ІКС екзаменаційно-тренінгового центу іноземних мов наведені в Таблиці 1.4.

Таблиця 1.4 – Головні вимоги до системи

<table>
<tr><th>Користувач (User)</th><th>Вимоги функціональні (System requirements)</th><th>Вимоги нефункціональні (General requirements)</th></tr>
<tr><td rowspan="4">Студент (Student)</td><td>1. Переглянути список доступних курсів</td><td rowspan="13">15. Обробка запитів з часом відгуку не більше 2 секунд з навантаження до 1000 одночасних користувачів.
16. Використання HTTPS-протокол для всіх сесій, шифрування персональних даних за стандартом AES-256.
17. Система повинна бути сумісною з основними браузерами (Chrome, Firefox, Safari, Edge) версій не старше 2 років, та операційними системами (Windows, macOS, iOS, Android), забезпечуючи коректне відтворення мультимедіа для мовних вправ (аудіо/відео).
18. Система повинна мати високий рівень доступності з резервним копіюванням даних та можливістю відновлення сесії тесту у разі збою мережі.</td></tr>
<tr><td>2. Записатись на курс</td></tr>
<tr><td>3. Переглянути список своїх курсів</td></tr>
<tr><td>4. Припинити навчання на курсі</td></tr>
<tr><td rowspan="4">Екзаменований (Exam Taker)</td><td>5. Переглянути список доступних екзаменів</td></tr>
<tr><td>6. Зареєструватись на екзамен</td></tr>
<tr><td>7. Переглянути список своїх екзаменів</td></tr>
<tr><td>8. Скасувати або перенести екзамен</td></tr>
<tr><td rowspan="6">Адміністратор (Admin)</td><td>9. Додати нового користувача (студента або екзаменованого)</td></tr>
<tr><td>10. Редагувати профіль користувача</td></tr>
<tr><td>11. Додати новий курс</td></tr>
<tr><td>12. Редагувати курс</td></tr>
<tr><td>13. Запланувати екзамен</td></tr>
<tr><td>14. Змінити деталі екзамену (дату, час, екзаменатор тощо)</td></tr>
</table>

1.3 Постановка задачі

Відповідно до проведеного аналізу предметної області, відомих ІКС згідно обраної тематики, а також визначених функціональних та нефункціональних вимог, сформулюємо основні задачі, які необхідно вирішити в рамках дипломної роботи магістра:

1. Розробка формальної моделі ІКС екзаменаційно-тренінгового центру іноземних мов з урахуванням головних процесів та учасників.
2. Декомпозиція системи у відповідності до структурного підходу з визначенням вхідних та вихідних потоків даних, параметрів, а також керуючих сигналів.
3. Розробка моделі надійності системи на основі резервування найбільш вразливих програмних компонентів.
4. Розробка моделі варіантів використання ІКС на основі функціональних вимог Таблиці 1.4.
5. Розробка алгоритмів та моделювання інфокомунікаційних процесів взаємодії акторів та системи.
6. Розробка моделі даних та практична реалізація бази даних (БД).
7. Обгрунтування вибору набору технологій до практичної реалізації ІКС.
8. Програмна реалізація системи згідно триступеневої архітектури з дотриманням нефункціональних вимог Таблиці 1.4. Реалізація моделі даних та компонентів взаємодії з БД, компонентів автоматизації інфокомунікаційних процесів системи та елементів інтерфейсу.

1.4 Висновки до першого розділу

В ході роботи над першим розділом дипломної роботи магістра були виконані такі підготовчі етапи для подальшої розробки ПЗ для ІКС екзаменаційно-тренінгового центру іноземних мов:

1. Проведено аналіз існуючих ІКС для вивчення іноземних мов та здачі екзаменів для підтвердження рівня володіння іноземною мовою.

2. Виділено основні інфокомунікаційні процеси, які є суттєвими для практичної реалізації ПЗ системи, а також визначено основних користувачів.

3. Сформульовано специфікацію функціональних та нефункціональним вимог та виконано постановку задачі роботи.

2 МОДЕЛІ ТА МЕТОДИ ІНФОКОМУНІКАЦІЙНОЇ СИСТЕМИ ЕКЗАМЕНАЦІЙНО-ТРЕНІНГОВОГО ЦЕНТРУ ІНОЗЕМНИХ МОВ

2.1 Формальна модель ІКС екзаменаційно-тренінгового центру іноземних мов

Проведемо формалізацію системи згідно методології структурного аналізу та дизайну (Structured analysis and design technique, SADT) [23] та загальних принципів моделювання систем, зокрема, з використанням методу «чорного ящика» [24].

Представимо ІКС I_S у вигляді трійки:

$$I_S = \{X, Y, Z\}, \tag{2.1}$$

де X – множина вхідних потоків даних (Inputs);

Y – множина вихідних потоків (Outputs);

Z – параметри системи, що в свою чергу згідно методології SADT поділяють на сигнали контролю (Controls – C) та механізми виконання (Mechanisms – M):

$$Z = \{C, M\}. \tag{2.2}$$

Схематично система може бути представлена у спрощеному вигляді, так як це показано на Рисунку 2.1.

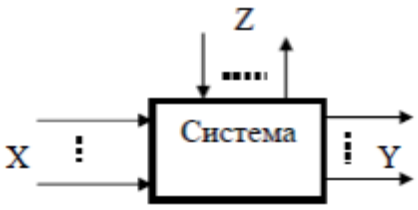


Рисунок 2.1 – Модель системи за методом «чорного ящика»

На основі базової моделі «чорного ящика» складемо контекстну діаграму (A-0), яка описуватиме всю систему у вигляді єдиного блоку активності Activity A-0: «Керування реєстрацією на курси іноземних мов та екзамени».

Ця активність охоплює всі взаємодії з користувачами, обробку даних та адміністративні функції через веб-додаток на основі триступеневої архітектури (Model-View-Controller, MVC), БД (доступ до якої здійснюється через драйвер з пулом підключень) та асинхронні обробники (для паралельного виконання).

Виділимо входи (Inputs, X), виходи (Outputs, Y), контрольні сигнали (Controls, C) та механізми виконання (Mechanisms, M) згідно виразів (2.1) і (2.2). Назва потоків, приклад даних та опис для входів, виходів, сигналів контролю та механічнім керування подано в Таблицях 2.1 – 2.4.

Вхідні потоки даних є ініційованими користувачами або зовнішніми джерелами, які обробляє система.

Таблиця 2.1 – Входи (Inputs) контекстної моделі A-0

Назва	Приклад даних	Опис
1	*2*	*3*
Облікові дані автентифікації користувача	Ім’я користувача, пароль	Дані для входу та початку сеансу.
Запити на реєстрацію	Ідентифікатори предмета та студента	Надходять від студентів, які реєструються на предмети.
Запити на реєстрацію на екзамен	Ідентифікатори екзамену та екзаменованого, бажана дата.	Надходить від екзаменованого.
Запити на відрахування або скасування	Ідентифікатор предмета або ідентифікатор іспиту для видалення.	З обгрунтуванням причини.

Кінець Таблиці 2.1

1	*2*	*3*
Запити на перепланування	Ідентифікатор іспиту, нова дата та час проведення.	Через відповідний модуль інтерфейсу користувача
Запити на предмети та екзамени	Фільтри для перегляду предметів, іспитів або особистих записів	
Вхідні дані для створення/редагування адміністратором	Дані нового користувача, назва опис предмета, деталі розкладу іспиту	

Вихідні потоки даних є відповідями або оновлені стани, що були генеровані системою.

Таблиця 2.2 – Виходи (Outputs) контекстної моделі A-0

Назва	Приклад даних	Опис
1	*2*	*3*
Підтвердження реєстрації	Повідомлення про успішне завершення, оновлений список предметів для користувача	Передача користувачу облікових даних через електронну пошту.
Підтвердження реєстрації/перенесення/ скасування іспиту	Оновлений статус іспиту, новий розклад	Із внесенням змін до переліку задач в календарі.
Результати запитів	Список предметів, іспитів, зареєстрованих предметів або зареєстрованих іспитів	Виведення у вигляді таблиць.

Кінець Таблиці 2.2

1	*2*	*3*
Оновлення профілю користувача	Відредаговані дані для користувачів, предметів або іспитів.	Переправлення на головну сторінку користувача
Сповіщення про помилки	Недійсний запит, конфлікти, такі як дублювання реєстрації	Виведення назви та опису помилки на екран.
Журнали або звіти моніторингу	Записи дій адміністрування, наприклад, створення та редагування розкладів.	Збереження даних за лімітований період часу.

Елементи керування регулюють виконання дії, включаючи бізнес-правила, параметри та зовнішні обмеження.

Таблиця 2.3 – Виходи (Outputs) контекстної моделі A-0

Назва	Приклад даних	Опис
1	*2*	*3*
Політики автентифікації	Доступ на основі ролей: користувач/студент може лише реєструватися, Екзаменований – для іспитів, адміністратор – для управління.	На розсуд адміністратора інфраструктури.
Правила перевірки	Передумови для предметів, критерії відповідності іспитам, перевірки дати і часу.	Погоджується із навчально-методичним відділом.

Кінець Таблиці 2.3

1	*2*	*3*
Системні параметри	Максимальна кількість реєстрацій на предмет, місткість іспитів, вікна перепланування, наприклад, «протягом 7 днів»).	Погоджується із навчально-методичним відділом
Елементи керування безпекою	Тайм-аути сеансів, вимоги до шифрування для передачі даних.	Відповідно до технічного завдання
Обмеження бази даних	Унікальні ключі для предметів/іспитів, цілісність посилань.	
Правила асинхронної обробки	обмеження пулу виконавців, політики повторних спроб для невдалих операцій з базою даних).	

Механізми складаються із засобі для виконання активності. (не завжди показано в A-0, але приймаються до уваги).

Таблиця 2.4 – Виходи (Outputs) контекстної моделі A-0

Назва	Приклад даних	Опис
1	*2*	*3*
Веб-додаток	Компоненти логіки даних, інтерфейсу користувача, контролери для обробки запитів.	На розсуд адміністратора інфраструктури.

Кінець Таблиці 2.4

1	*2*	*3*
Сервер бази даних	З драйвером для підключень.	Погоджується із навчально-методичним відділом.
Пул підключень	Для ефективного повторного використання доступу до бази даних.	Залежить від реалізації
Пул обробників	Для асинхронних завдань, таких як фонові оновлення або сповіщення.	
Користувачі/Актори	Браузери або клієнти, що використовуються Користувачем, Студентом, Екзаменованим, Адміністратором	Дані предметної області

Таким чином, маємо змодельовану згідно SADT систему, контекстна модель А-0 якої показана на Рисунку 2.2. В якості цілі практичної реалізації розглядається веб-додаток, який використовується студентами та тими, хто складає екзамени (екзаменованими), для реєстрації на курси іноземних мов та реєстрації на іспити відповідно. Додаток доцільно побудувати за шаблоном MVC, з використанням пулу підключень до БД з метою забезпечення надійності даних. Асинхронне виконання програми забезпечується пулом паралельних обробників. Основними користувачами системи є: Користувач, Студент, Екзаменований.

Рисунок 2.2 – Контекстна діаграма моделі системи

2.2 Декомпозиція моделі системи

Виконаємо декомпозицію моделі системи (Рисунок 2.2) шляхом виділення дочірніх підсистем в ієрархічній послідовності. Відповідно до визначених ролей користувачів, сформулюємо активності нижніх рівнів, згруповані за ролями: Студент/Користувач, Екзаменований/Користувач, Адміністратор.

На Рисунку 2.3 в якості прикладу наведено підсистему «Перегляді усіх іспитів». Решта активностей, що були отримані в результаті декомпозиції системи винесено в Додаток А. Головні потоки даних активності для вищезазначений ролей зведені до Таблиць А.1 – А.3. Дочірні активності користувачів показані на Рисунках А.1 – А.7.

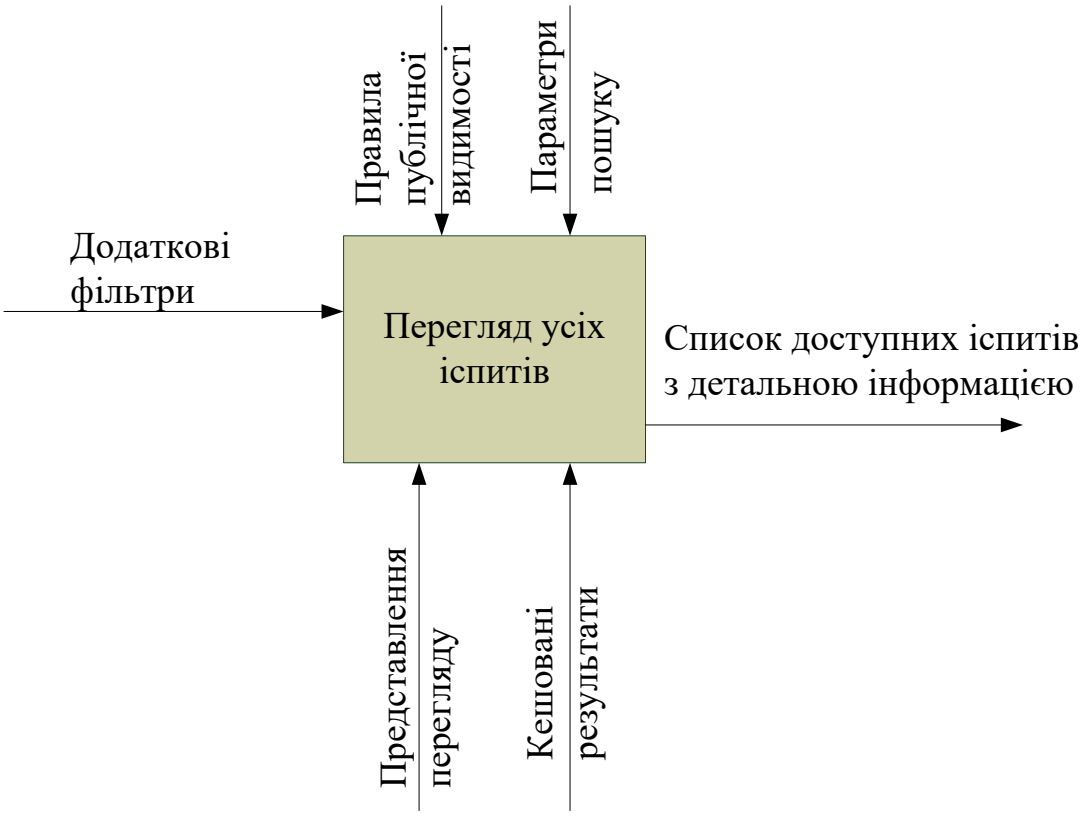


Рисунок 2.3 – Активність «Перегляд усіх іспитів»

Особливо варто виділити активності, які пов’язані із адмініструванням системи. Дочірні активності адміністрування зображені на Рисунках А.1 -А.7. Це

усі службові функції, що виконуються в рамках забезпечення функціонування інформаційно-комунікаційних процесів ІКС.

В рамках описаної системи, усі додаткові функції винесені в єдиних модуль адміністрування, однак для реальної імплементації ПЗ системи, доцільним є подальша декомпозиція даного модуля із окремим виділення таких модулі, як «Навчальний відділ», «Бухгалтерія», власне «Адміністрування» тощо.

Функції реєстрації користувачів та редагування профілів, наприклад, можна покласти на операторів методичних відділів, в той час як редагування списку доступних предметів, а також розклад іспитів доцільно пов’язати із додатковим програмним модулем, що вдало вписується в дану архітектуру та відповідає предметній області, – відділ Викладачів та екзаменаторів.

Однак, в межах поставленої задачі, обмежимось модулями системи, які були описані в даному розділі.

2.3 Розробка моделі надійності програмних компонентів

В результаті аналізу моделі системи, яка була розроблена в попередньому розділі, можна побачити, що істотна кількість потоків інформації пов’язана із зверненням до сховища даних. В більш конкретній імплементації системи, що пов’язана із розробкою конкретного програмного продукту, ці потоки даних зводяться до звернення до БД шляхом створення і підтримки екземплярів підключення до БД.

З огляду на важливість забезпечення безвідмовного доступу ПЗ до БД, розглянемо моделі резервування програмних компонентів для підключення програного додатку до БД.

Відома модель мікросервісів [25], згідно якої архітектура ПЗ розділяється на множину окремих компонентів (сервісів), що функціонують паралельно в єдиній екосистемі. Використання таких сервісів сприяє підвищенню надійності

системи та спрощення структури окремих сервісів, які виконують кожен свою просту функцію (спрощено).

Представимо механізм доступу до БД у вигляді програмного модуля з багатосерсвісною архітектурою, що функціонує з метою, наприклад, використання одним додатком декількох БД, розташованих фізично на різних серверних платформах.

Об’єктно-орієнтована модель ПЗ дозволяє абстрагуватись від конкретної системи управління базою даних (СУБД), її драйвера та параметрів підключення, інкапсулюючи ці дані у вигляді об’єкта підключення.

На Рисунку 2.4 показана структура вищеназваних програмних компонентів, де головним елементом є пул підключень до БД, який складається із N об’єктів підключень та K резервних об’єктів. Метод резервування в загальному випадку може бути довільним, на розсуд архітектора або розробника ПЗ, однак, в рамках даної роботи обмежимось доволі простим випадком відновлювальної резервованої системи з резервом кратності 1/2. Така система складається із трьох елементів різної надійності.

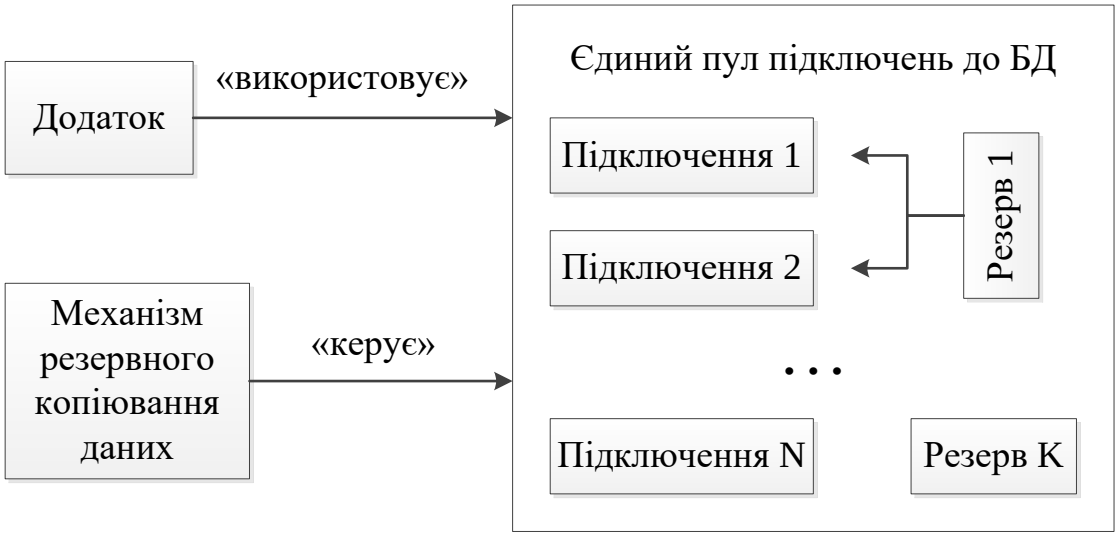


Рисунок 2.4 – Структура програмного модуля із резервуванням підключень до БД

Як видно із Рисунку 2.4, концепція використання пулу підключень до БД полягає у постійному використання підключень для нормальної роботи програмного додатку, а також у забезпеченні резервного копіювання даних між основними та резервними сховищами для можливості найшвидшого відновлення у випадку відмови.

Виконаємо моделювання такої системи в умовах високого навантаження невеликим часом напрацювання відмови програмних компонентів. В таблиці 2.5 подані відповідні параметри надійності.

Таблиця 2.5 – Параметри надійності елементів системи

Параметр / № Елемента	1	2	3
Назва	Підключ. 1	Підключ. 1	Резерв 1
Середній час напрацювання на відмові (Mean Time Between Failures, MTBF), години	12	8	8
Середній час відновлення (Mean Time To Recovery, MTTR), хвилини	10	10	5
Середня інтенсивність відмови λ_i, год$^{-1}$	0,083	0,125	0,125
Середня інтенсивність відновлення μ_i, год$^{-1}$	6	6	12

Функціональний граф станів системи показано на Рисунку 2.5. Стани системи показані вершинами графу 0-11. Над вершинами графу 6-11 позначені номери елементів, які відмовили. Так, до прикладу, «1» означає випадок відмови першого елементу за роботи решти двох. Після відновлення неробочого елементу система переходить у стан справності усіх елементів, однак штатно працюватимуть перший та третій елементи.

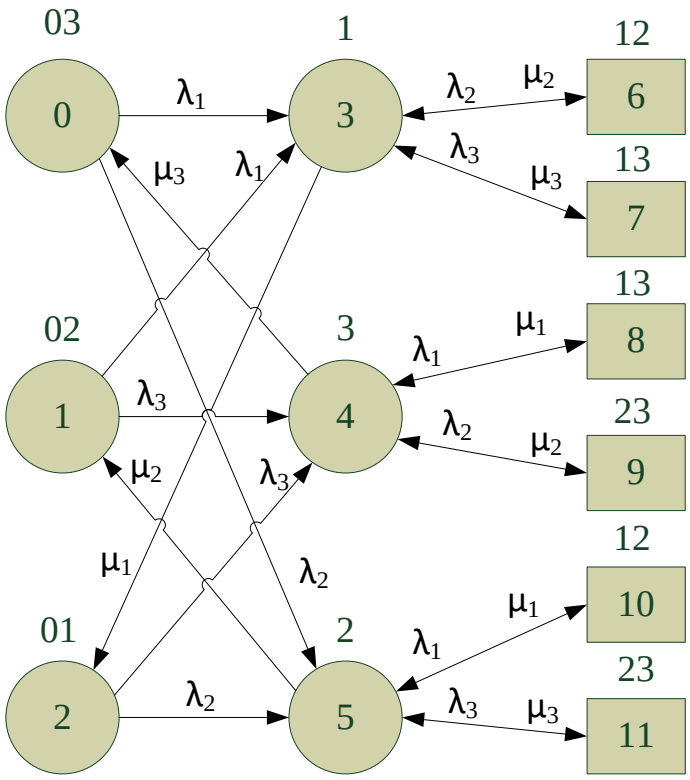


Рисунок 2.5 – Граф станів системи

На Рисунку 2.5 стани, що відповідають справності усіх елементів розміщені скраю зліва рисунку, наприклад «01» означає, що перший елемент знаходиться в резерві, а працюють другий і третій елементи і т.д. Стани, які позначені прямокутниками, відповідають відмовам.

$$\boldsymbol{AP} = \boldsymbol{B}, \tag{2.3}$$

де $\boldsymbol{A}$ – матриця коефіцієнтів системи розміром (m×n), що визначається параметрами λ_i та μ_i (див. Таблицю 2.5);

$\boldsymbol{P}$ – вектор довжиною n ймовірностей перебування системи в кожному з можливий станів;

$\boldsymbol{B}$ – вектор довжиною m вільних членів правої частини лінійного рівняння.

Оскільки ймовірності $p_k \in \boldsymbol{P}$, $k = \overline{1 \dots n}$ становлять повну групу, їх сума повинна дорівнювати одиниці:

$$\sum_{k=1}^{n} p_k = 1. \tag{2.4}$$

Коефіцієнти матриці **A** параметрів системи (2.3) наведені в таблиці 2.6.

Таблиця 2.6 – Коефіцієнти матриці **A** рівняння (2.3)

m \ n	1	2	3	4	5	6	7	8
1	$-(\lambda_1+\lambda_2)$	0	0	0	μ_3	0	0	0
2	0	$-(\lambda_1+\lambda_3)$	0	0	0	μ_2	0	0
3	0	0	$-(\lambda_2+\lambda_3)$	μ_1	0	0	0	0
4	λ_1	λ_1	0	$-(\mu_1+\lambda_2+\lambda_3)$	0	0	μ_2	0
5	0	λ_3	λ_3	0	$-(\mu_3+\lambda_1+\lambda_2)$	0	0	μ_1
6	λ_2	0	λ_2	0	0	$-(\mu_3+\lambda_1+\lambda_3)$	0	0
7	0	0	0	λ_2	0	0	$-\mu_2$	0
8	0	0	0	λ_3	0	0	0	$-\mu_3$
9	0	0	0	0	0	λ_1	0	0
10	0	0	0	0	λ_2	0	0	0
11	0	0	0	0	0	λ_1	0	0
12	0	0	0	0	0	0	λ_3	0

m \ n	9	10	11	12
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0
9	$-\mu_1$	0	0	0
10	0	$-\mu_2$	0	0
11	0	0	$-\mu_1$	0
12	0	0	0	$-\mu_3$

Розв’язавши рівняння (2.3) з урахуванням (2.4), отримуємо вектор стаціонарних ймовірностей **P**, значення якого наведені на Рисунку 2.6.

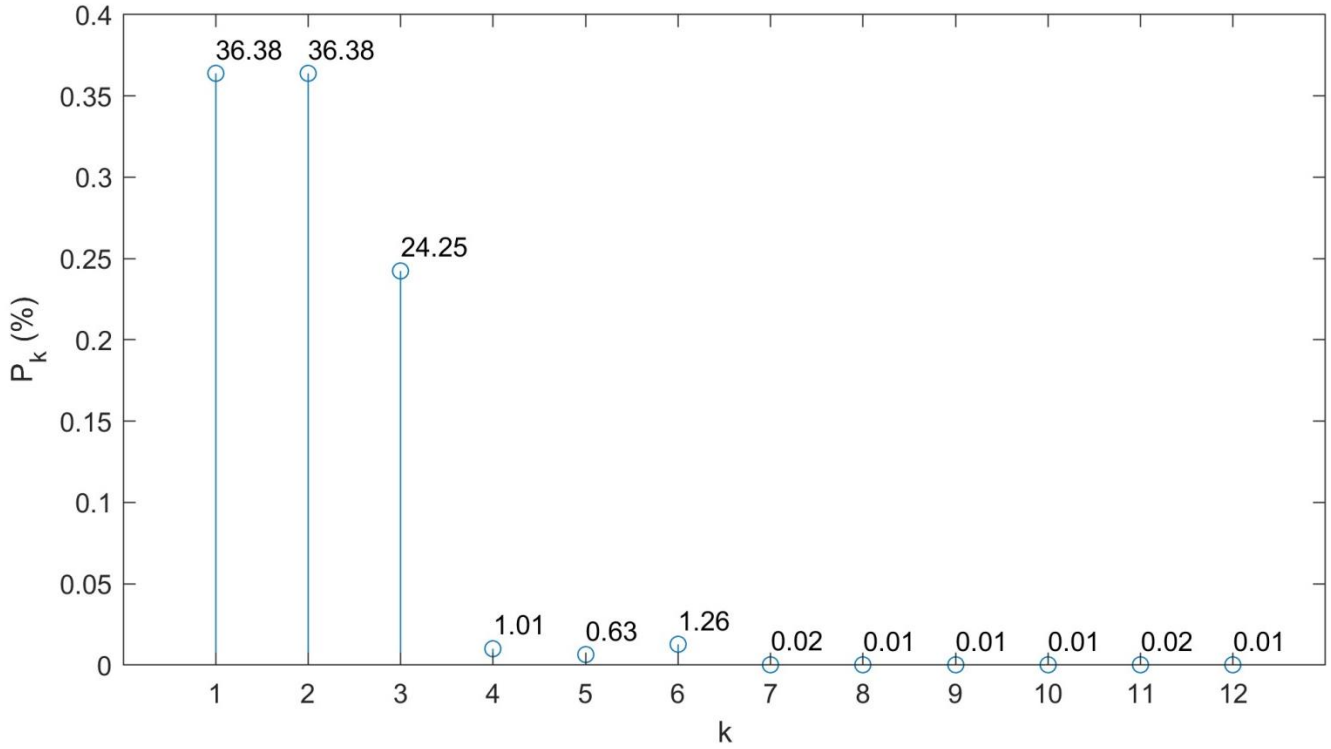


Рисунок 2.6 – Розраховані стаціонарні ймовірності перебування системи в k-му стані

Відповідно до моделі на Рисунку 2.5, робочими є стани 0-5, отже, коефіцієнт готовності становить суму відповідний ймовірностей:

$$K_s = \sum_{k=0}^{5} P_k = 99{,}92\ \%. \quad (2.5)$$

Характеристика потоку відмов:

$$\omega = (\lambda_2 + \lambda_3)p_4 + (\lambda_1 + \lambda_2)p_5 + (\lambda_1 + \lambda_3)p_6 = 0{,}0893\ (\text{год.}^{-1}). \quad (2.6)$$

Час напрацювання на відмову:

$$T_f = \frac{K_r}{\omega} = 154\ \text{год. 20 хв.} \quad (2.7)$$

Середній час відновлення:

$$T_r = \frac{1-K_r}{\omega} = 7\ \text{хв. 48 сек.} \quad (2.8)$$

Отже, запропонований метод дозволяє збільшити середній час напрацювання на відмову до 154 годин (більше 6 днів), що понад у 12 разів більше за відповідний показник для лише одного елементу із базовими показниками надійності, з середнім часом відновлення до 7 хвилин.

2.4 Висновки до другого розділу

В другому розділі робити була розроблена формальна модель ІКС на основі методики структурного аналізу. Були визначені вхідні, вихідні та керуючі потоки даних системи, а також виконана детальна декомпозиція системи.

Запропонована модель підвищення надійності системи шляхом резервування програмних компонентів підключення до БД.

3 РОЗРОБКА ВАРІАНТІВ ВИКОРИСТАННЯ ТА БАЗИ ДАНИХ

3.1 Розробка моделі варіантів використання системи

Відправним етам розробки ПЗ для ІКС є побудова моделі варіантів використання системи. Доцільність такого підходу обумовлена необхідністю чіткого формулювання технічного завдання (ТЗ) та затвердженням мінімальним функціональним вимогам до кінцевого програмного продукту.

Модель варіантів використання складається із дійових осіб – акторів системи та прецедентів (варіантів використання) – тих дій, які виконуються за запитом відповідних акторів. Метою моделі є відповідь на два головних запитання: «Хто користується системою (програмним забезпеченням)?» та «Що він/вони можуть роботи за допомогою цієї програми?».

На Рисунку 3.1 Показана діаграма варіантів використання актора Студент (Student), який може ініціювати в системі наступні прецеденти:

- «See My Subjects» – переглянути мої дисципліни;
- «See All Subjects» – переглянути усі доступні до вивчення дисципліни;
- «Enroll subject» – записатись на вивчення даної дисципліни;
- «Enenroll subject» – скасувати вивчення (відписатись від даної дисципліни).

На Рисунку 3.2 показана діаграма варіантів використання актора Екзаменований (ExamTaker), який може ініціювати в системі такі прецеденти:

- «See All Exams» – перегляд усіх доступних екзаменів;
- «Register Exams» – записатись на екзамен;
- «See My Exams» – переглянути список своїх екзаменів;
- «Reschedule Exam» –перенести екзамен на інший час або дату;
- «Cancel Exam» – скасувати свою реєстрацію на екзамен.

На Рисунку 3.3 показана діаграма варіантів використання актора Адміністратор (Admin), який може ініціювати в системі такі прецеденти:

- «Add New User» – додати нового користувача;
- «Edit Student Profile» – редагувати дані користувача;
- «Add New Subject» – додати новий предмет (навчальну дисципліну);
- «Edit Subject Details» – редагувати параметри дисципліни;
- «Schedule New Exam» – запланувати новий екзамен;
- «Edit Schedule Exam» – редагувати запланований екзамен.

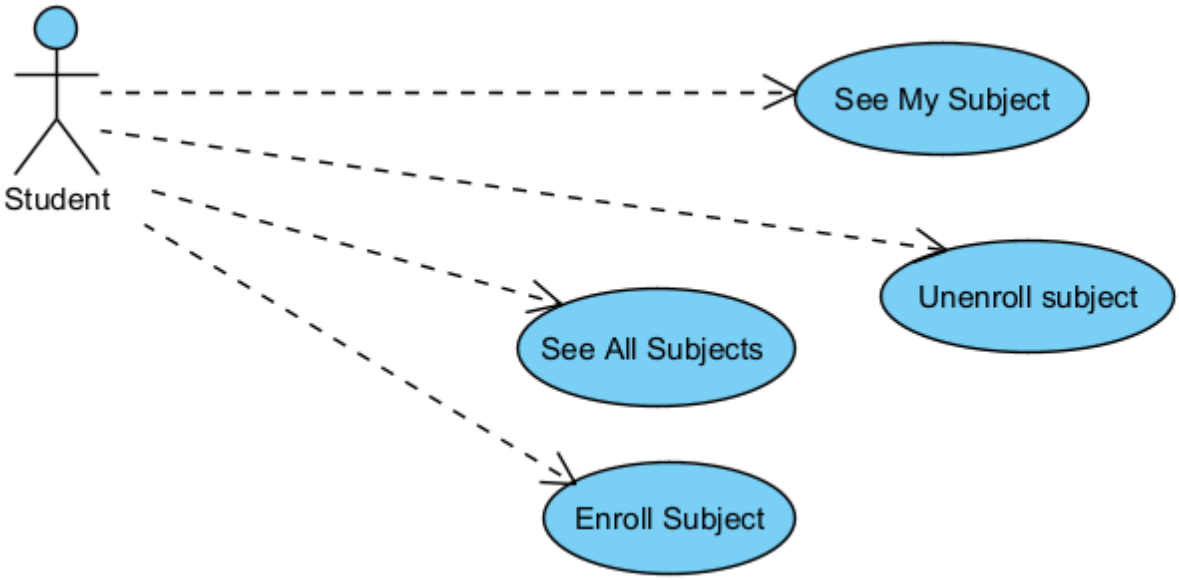


Рисунок 3.1 – Діаграма варіантів використання актора Student

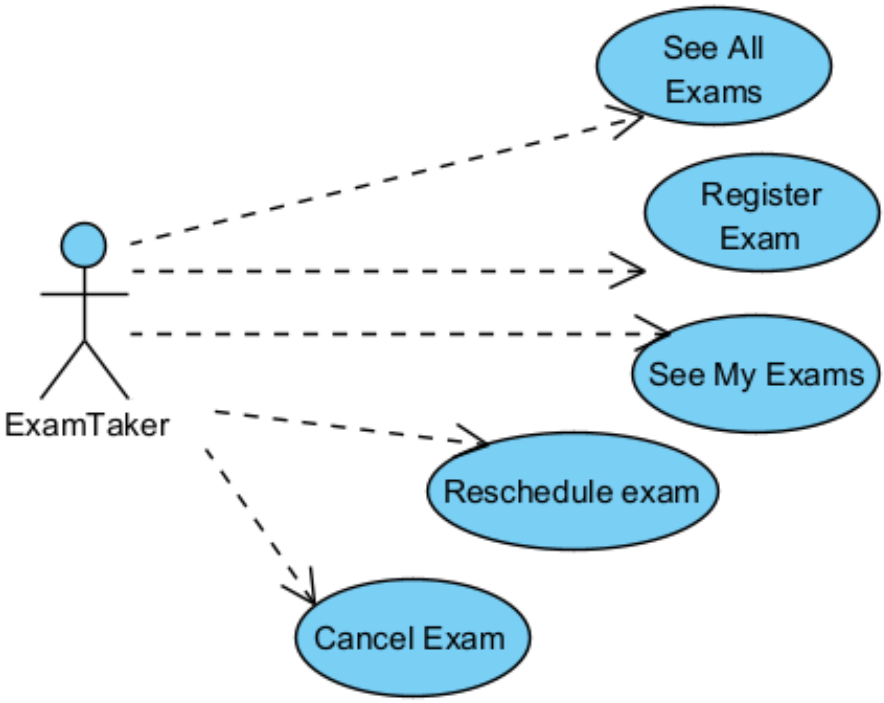


Рисунок 3.2 – Діаграма варіантів використання актора ExamTaker

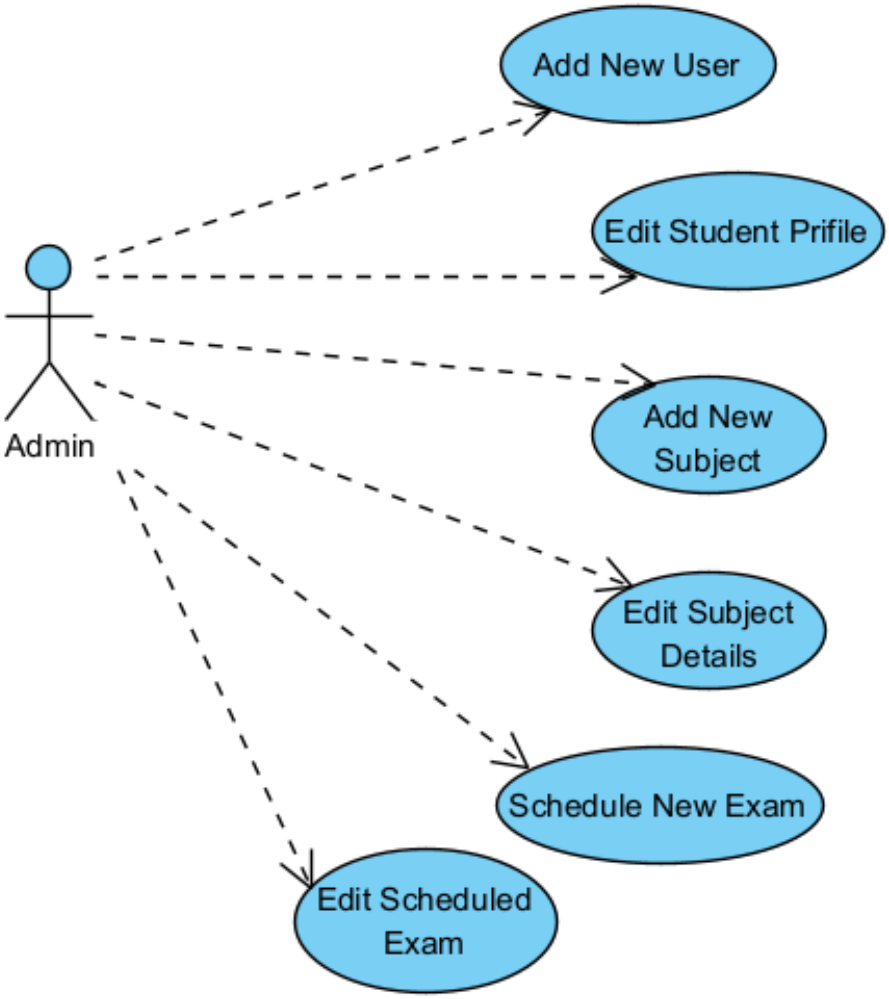


Рисунок 3.3 – Діаграма варіантів використання актора Admin

3.2 Розробка схеми даних

Відповідно до трирівневої моделі додатку, модель та сховище даних виконані на базі схеми БД, яка є сумісною із більшістю сучасних реляційних СУБД.

Повна схема даних ІКС подана в Додатку Б, звідки видно, що модель складається із 9 таблиць, що зв’язані між собою відповідно до контексту даних.

Таблиці схеми даних з Додатку Б можна умовно поділити на три групи відповідно до підмножини запитів, що виконуються за викликом відповідних програмних модулів:

1) група «Студенти», до складу якої входять таблиці «students», «student_subjects», «subjects»;

2) група «Екзамени», що в свою чергу містить таблиці «exams», «student_exams», «subject_exam»;

3) група «Користувачі та Ролі», що складається із таблиці «users», «user_roles», «roles».

Проведемо детальний опис таблиць кожної з вищезазначених груп, взаємозв'язки між таблиця в межах групи, а також зв'язки між таблицями різних груп.

Керування даними, що пов'язані із відомостями про студентів ІКС та дисциплінами (предметами), що вивчаються ними, покладено на групу «Студенти».

Таблиця «students» містить записи про студентів, а саме:

- id – числовий ідентифікатор студента, унікальний в системі;
- name – повне ім'я користувача;
- program – назва програми підготовки, наприклад «Електронні комунікації», «Радіотехніка» тощо;
- record_number – персональний номер студента (за аналогією до номеру залікової книжки у вітчизняних університетах);
- user_id – ідентифікатор користувача, у тому випадку, коли даний студент є зареєстрованим в системі з відповідною роллю.

Наступною важливою таблицею групи «Студенти» є таблиця «subjects», що акумулює записи про навчальні дисципліни:

- id – числовий ідентифікатор дисципліни, унікальної в системі;
- title – заголовок назви дисципліни – офіційна назва предмету, наприклад, в навчальному плані;
- lecturer – ім'я викладача;
- semester – номер семестру, протягом якого проводиться навчання даної дисципліни;
- credits – кількість кредитів ЄКТС [26].

В середині групи «Студенти» таблиця «students» пов'язана із таблицею «subjects» за допомогою проміжної таблиці «student_subjects», яка розв'язує

колізію відношення «багато-до-багато», яка природно виникає з міркувань, щодо яких один студент може вивчати декілька дисциплін, а одну окрему дисципліну слухає група студентів. Таблиця «student_subjects» містить посилання на відповідні ідентифікатори у якості зовнішніх ключів.

Наступна група таблиць – «Екзамени», вона включає в себе такі таблиці: «exams», «student_exams», «subject_exam». Як видно із назв таблиці, тут зосереджені запити на керування даними екзаменів із дисциплін, а також керування записами студентів на екзамени. Головною таблицею в групі є таблиця екзаменів «exams», яка містить такі записи:

- id – числовий ідентифікатор екзамену;
- place – запис про місце проведення екзамену (адреса);
- room – назва аудиторії;
- time – дата та час проведення екзамену.

Таблиця «exams» пов’язана із таблицями «students» та «subjects» попередньої групи таблиць за допомогою відповідних допоміжних таблиць «student_exams» та «subject_exam» відповідно. Таблиця «student_exams» пов’язує таблицю «students» із таблицею «exams», розв’язуючи колізію «багато-до-багато» керуючись міркуваннями, щодо яких один окремий студент може здавати екзамени із різних дисциплін, а на одному екзамені можуть одночасно бути присутніми декілька студентів. Також студенти мають право на перездачу екзамену. Додатковим записом в таблиці «student_exams» є відомість про кількість набраних балів (marks), що заповнюється після перевірки роботи студента.

Взаємозв’язок між таблицями «exams» та «subjects» пов’язує екзамени, які заплановані навчальним відділом з дисциплін. Для реалізації гнучкої системи екзаменів, наприклад, для можливості перездачі студентом екзамену, а також проведення суміжних іспитів в один день з метою економії часу, тут також має місце колізія «багато-до-багато», що розв’язується за допомогою проміжної таблиці «exam_subject», яка додатково містить запис про уповноваженого викладача, який приймає даний екзамен (examiner).

Остання група таблиць – «Користувачі та Ролі». Вона має службове призначення і складається із таких таблиць: «users» для користувачів системи, «roles» - ролі користувачів в системі, а також проміжної таблиці «user_roles» для однозначного взаємозв'язку між ними. Відповідно до обраної стратегії функціонування, виділено такі наперед задані ролі користувачів: «адміністратор», «студент», «екзаменований». Розширення списку ролей в рамках даної предметної області не передбачається, однак кількість різних користувачів, в тому числі із правами адміністрування, може бути довільною. Цим і аргументується доцільність проміжної таблиці. Крім того, таблиця користувачів «users» пов'язана зав'язком «один-до-одного» із таблицею студентів «students», при чому головною таблицею є перша з них. Таке рішення пояснюється фактом можливості студента не бути зареєстрованим у системі на момент початку навчання, що є доволі типовою ситуацією.

Записи таблиці «users»:

- id – числовий ідентифікатор користувача, що є унікальним в системі;

- username – ім'я користувача;

- e_mail – адреса електронної пошти;

- passwd_hash – результат хешування пароля користувача (збереження пароля в явному вигляді в БД суперечить правилам інформаційної безпеки).

Записи таблиці «roles»:

- id – числовий ідентифікатор ролі. В системі передбачені такі ролі: 1 – «Адміністратор», 2 – «Студент», 3 – «Екзаменований»;

- name – коротка назва ролі;

- title – розширена назва ролі (заголовок);

- roleMessage – текстове повідомлення користувачам з даною роллю з появленнями наданими їм функціями.

Проміжна таблиця «user_roles» пов'язує вищеописані таблиці, розв'язуючи колізію «багато-до-багато», користуючись міркуваннями, згідно який для вказаної ролі може існувати довільна кількість користувачів, один користувач

може мати декілька ролей в системі (наприклад, одночасно студент та екзаменований).

3.3 Практична реалізації бази даних в PostgreSQL

3.3.1 Загальний опис PostgreSQL

В якості СУБД було обрану PostgreSQL [27]. Дамо стислу характеристику даного ПЗ з позиції обгрунтування його вибору.

PostgreSQL є відкритою об’єктно-реляційною СУБД, що була розроблена в 1986 року в Каліфорнійському Університеті в місті Берклі (США). Наразі підтримується глобальною спільнотою розробників. PostgreSQL є безкоштовною з відкритим вихідним кодом під ліцензією, подібною до BSD. Вона характеризується високою надійністю, розширюваністю та суворим дотриманням стандартів SQL, таких як ANSI-SQL:2008 та новіших версій. Станом на вересень 2025 року актуальною версією є PostgreSQL 17, що має покращені характеристики роботи, безпеки та інтеграції з сучасними технологіями [27].

До основних можливостей PostgreSQL відноситься підтримка реляційної моделі з об’єктними розширеннями, включаючи таблиці, ключі, обмеження, тригери, представлення та матеріалізовані представлення, а також можливість успадкування таблиць, подібно до принципів об’єктно-орієнтованого програмування (ООП). Система дозволяє створювати власні типи даних, такі як композитні дані, масиви або перелічення, функції на різних мовах, включаючи SQL, PL/pgSQL, Python або JavaScript, а також оператори, індекси та розширення, наприклад, PostGIS для геопросторових даних, pg_trgm для повнотекстового пошуку або TimescaleDB для роботи з часовими рядами. Крім того, PostgreSQL вдало інтегрується з нереляційними даними, маючи вбудовану підтримку JSON/JSONB для NoSQL-подібних запитів, XML та HStore для ключ-значення, що дозволяє використовувати гібридні схеми без втрати ефективності [28]. Що стосується транзакцій та надійності, PostgreSQL

забезпечує повну сумісність з ACID-принципами, включаючи атомарність, узгодженість, ізоляцію та стійкість, завдяки багатоверсійному керуванню паралелізмом, що мінімізує блокування. Система підтримує точкові відновлення та журналювання Write-Ahead Logging для резервного копіювання. Безпека СУБД реалізована через рольову модель доступу, шифрування даних, автентифікацію та аудит. Інтеграція включає обгортки даних для з'єднання з іншими СУБД та інтерфейсами для багатьох мов програмування, з інструментами адміністрування [29].

Порівняно з іншими СУБД, PostgreSQL відрізняється балансом між функціональністю, вартістю та гнучкістю, з широкими функціональними можливостями як для відкритої системи. У порівнянні з MySQL або MariaDB, Postgres пропонує кращу підтримку складних запитів, ширший набір типів даних, вищий рівень ACID-сумісність та багатоверсійне керування без блокувань, що робить її ефективнішою для навантажень з інтенсивними записами, тоді як MySQL може бути швидшим для простих читань, але менш гнучким для аналітики та enterprise-систем. Порівняно з Microsoft SQL Server, PostgreSQL є безкоштовною альтернативою з подібними функціями, але з більшою гнучкістю в розширюваності та крос-платформенністю, хоча SQL Server краще інтегрується з екосистемою Microsoft, але потребує придбання ліцензії для комерційного використання. Порівнюючи систему з СУБД Oracle, Postgres надає enterprise-функції без ліцензійних витрат, з подібною продуктивністю, простішим налаштуванням та активнішою спільнотою розробників. Загалом, переваги PostgreSQL включають високу стабільність, активну спільноту розробників, низьку вартість підтримки, а також підтримку гібридних даних, що робить оптимальною для сучасних додатків [31-32].

3.3.2 Створення таблиць та зв'язків у редакторі pgAdmin

Для візуального створення та редагування БД інсталяційний пакет PostgreSQL поставляється із інструментом, що називається pgAdmin. На Рисунку 3.4 показано головне меню цієї програми.

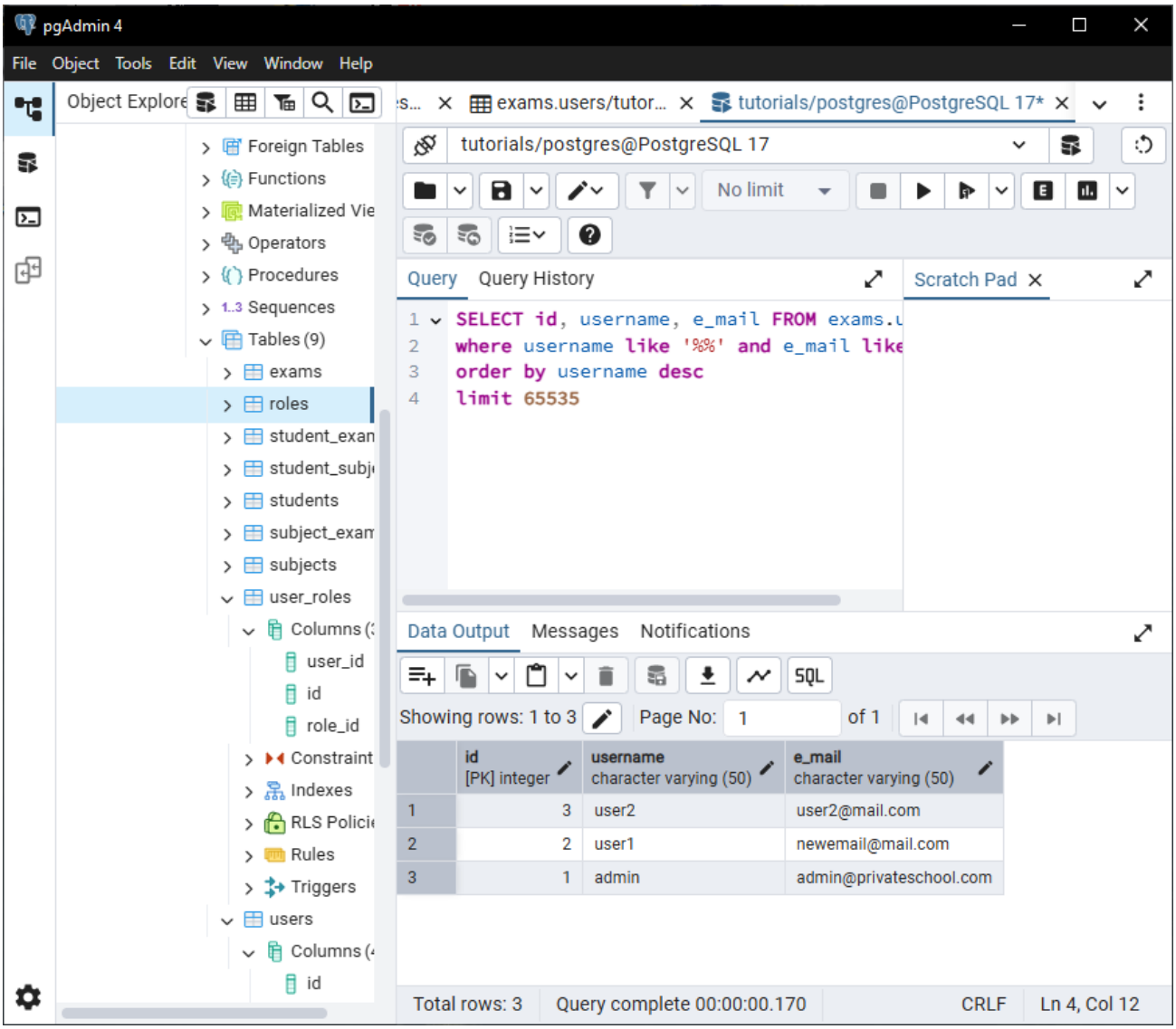


Рисунок 3.4 – Інтерфейс програми pgAdmin

Головне вікно програми pgAdmin складається із трьох основних компонентів: меню панелі інструментів, бокова панель навігації існуючими БД та їх таблицями, а також головна панель для роботи із таблицями БД, їх властивостями, SQL-запитами, як в графічному режимі, так і в режимі конструктора.

Усі спроектовані в попередньому підрозділі роботи таблиці БД були реалізовані в PostgreSQL засобами pgAdmin. Відповідні стовпці таблиць налаштовані згідно формату даних, зв’язки між таблицями виставлені шляхом налаштування зовнішніх ключів згідно схеми БД, що наведена в Додатку Б.

На Рисунку 3.5 зображено фрагмент бокової панелі навігації у створеній БД для ІКС, де в розгорнутому вигляді перелічені усі таблиці.

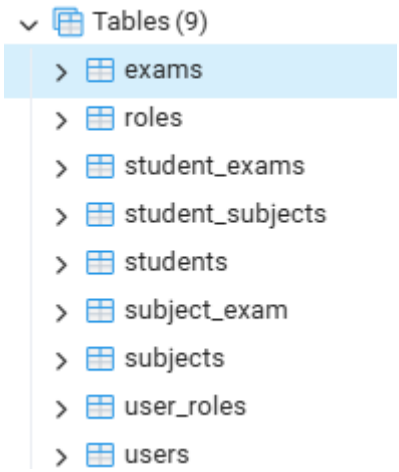
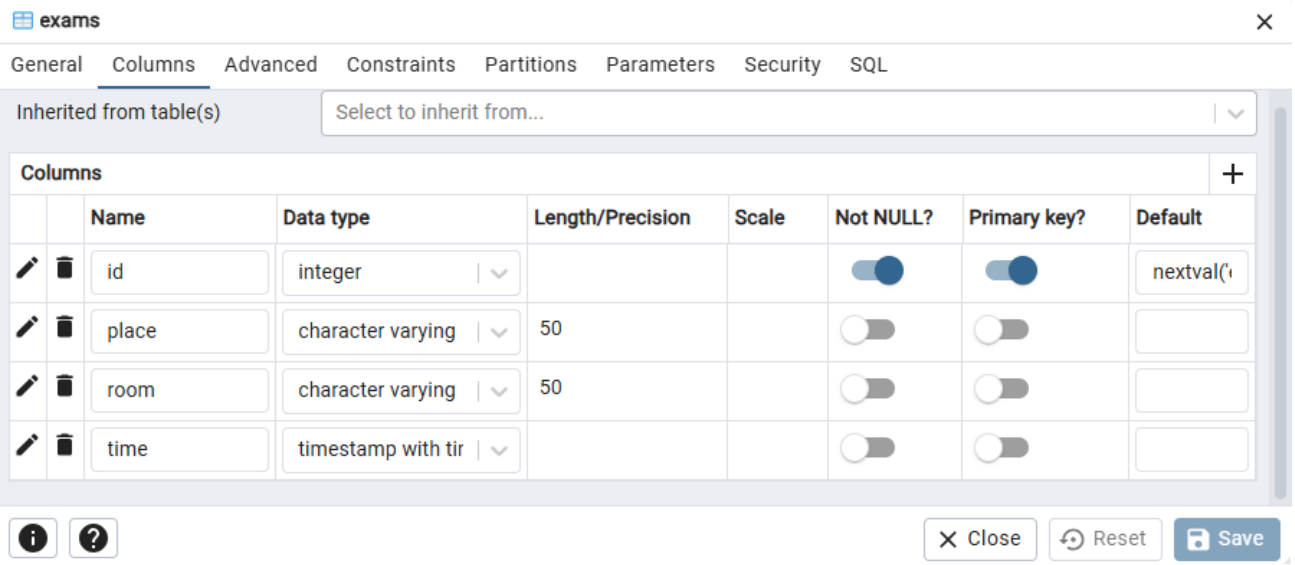


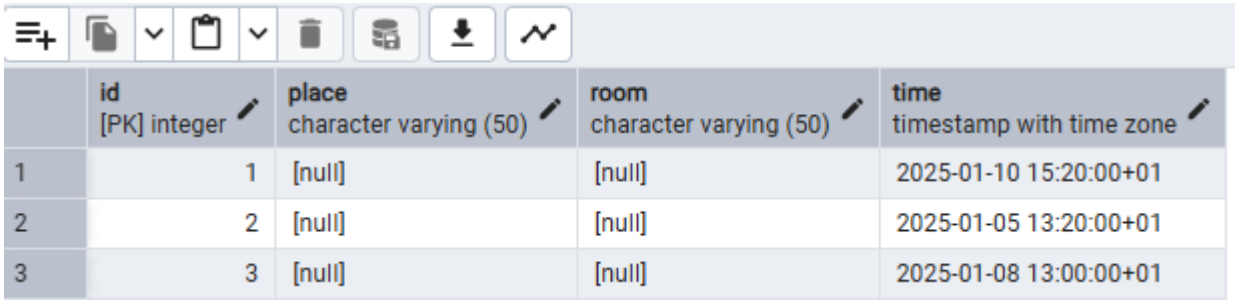
Рисунок 3.5 – Таблиці БД в PostgreSQL

На Рисунку 3.6,а показано панель для створення таблиці «exams» із зазначенням відповідних полів та типів даних. Для ідентифікатора «id» виставлено цілочисловий тип Integer з автоматичним інкрементом для нових записів, так-як дане поле є первинним ключем таблиці. Стовпці «place» та «room» мають символьний тип з обмеженням до 50 символів. Для збереження дати та часу, тип даних поля «time» виставлено у форматі Datatime. Для тестування ПЗ в таблиці «exams» створено три тестові записи без зазначення місця проведення іспиту та екзаменатора (Рисунок 3.6,б), оскільки дані поля дозволяють зберігати пусті значення типу null.

Створення таблиці «roles» виконано аналогічним чином, а відповідне вікно з параметрами стовпців показано на Рисунку 3.7,а. Числовий ідентифікатор «id» з автоінкрементом виконує функцію первинного ключа, решта символьних полів – «name», «title» та «roleMessage» мають тип Character з обмеженням на довжину символі рівну 50, 50 та 255 відповідно. Як було зазначено в попередньому підрозділі, в додатку визначено всього три ролі користувачів, які й були записані в дану таблиці (Рисунок 3.7,б).



а)

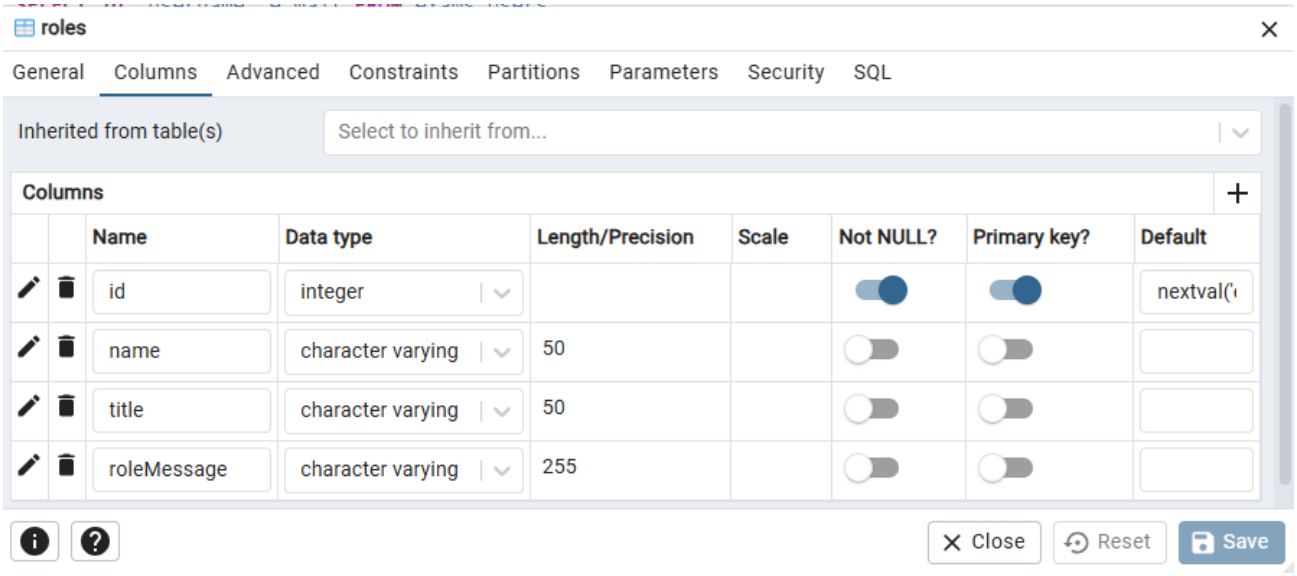


	id [PK] integer	place character varying (50)	room character varying (50)	time timestamp with time zone
1	1	[null]	[null]	2025-01-10 15:20:00+01
2	2	[null]	[null]	2025-01-05 13:20:00+01
3	3	[null]	[null]	2025-01-08 13:00:00+01

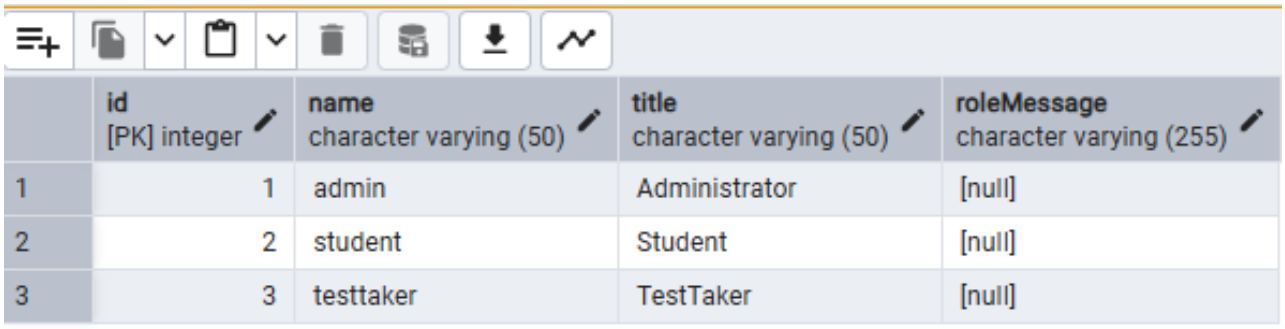
б)

Рисунок 3.6 – Створення таблиці «exams» (а)
та її заповнення тестовими даними (б)

Допоміжні таблиці таблиць «student_exams» та «student_subject» створені та заповнені у відповідності до схеми даних і використовують в якості первинного ключа по два відповідних поля, своїх головних таблиць (див. Додаток Б). Це поля «student_id», «exam_id» та «subject_id», що посилаються на первинні ключі своїх головних таблиць, а отже, до них не застосовується функція актоінкременту. Як і для попередніх таблиці, з метою наявності повнофункціональної БД на етапі тестування, ці таблиці були заповнені тестовими даними (Рисунок 3.8).



а)



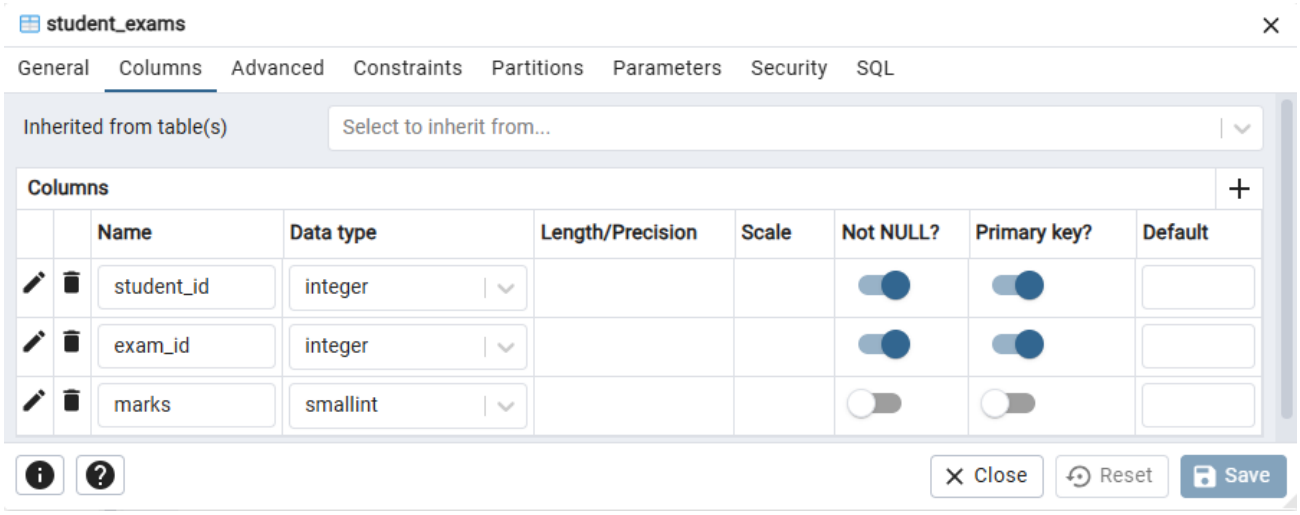
	id [PK] integer	name character varying (50)	title character varying (50)	roleMessage character varying (255)
1	1	admin	Administrator	[null]
2	2	student	Student	[null]
3	3	testtaker	TestTaker	[null]

б)

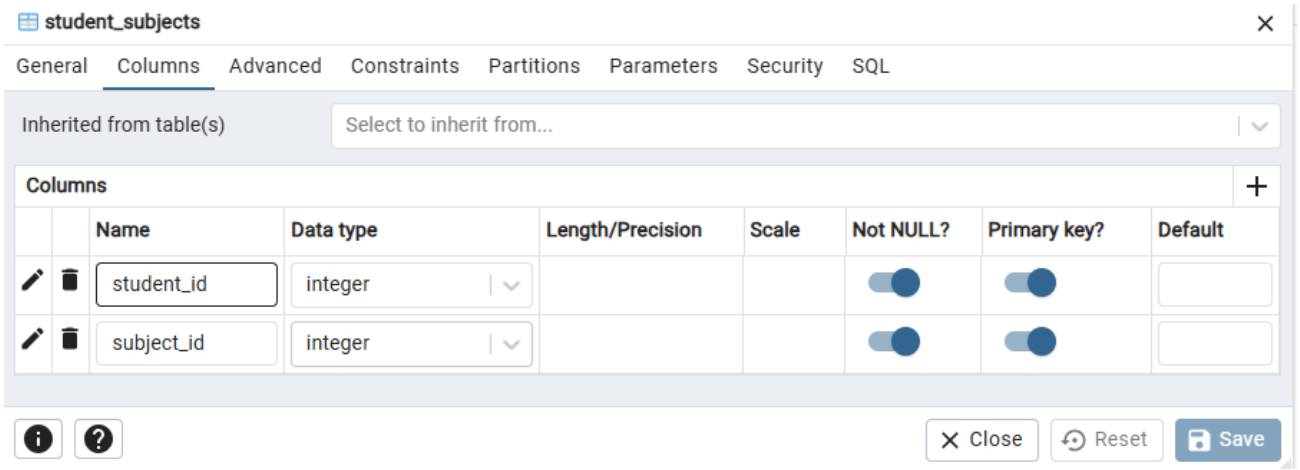
Рисунок 3.7 – Створення таблиці «roles» (а)
та її заповнення тестовими даними (б)

Решта таблиць були створені та заповнені тестовими даними аналогічним чином у повній відповідності до схеми БД в Додатку Б.

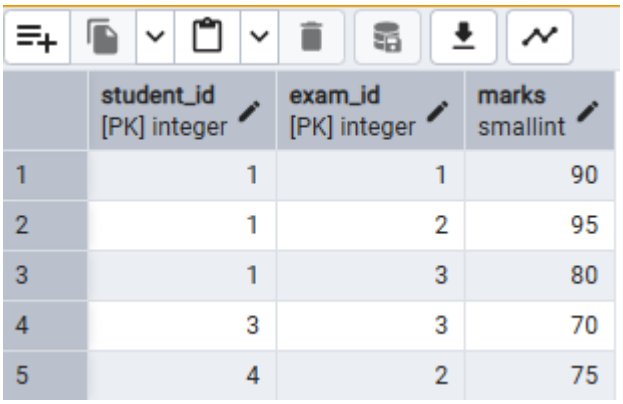
Відповідні фрагменти інтерфейсу програми pgAdmin із налаштуванням стовпців таблиць та відображення даних для решти таблиць зведено в Додаток В та показані на Рисунках В.1 – В.5.



а)

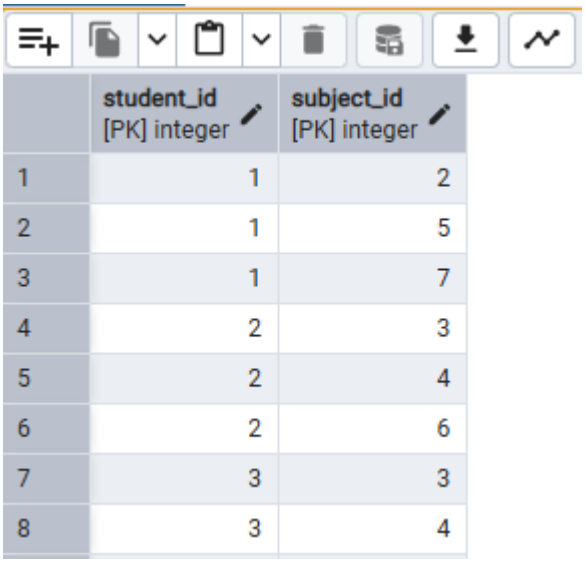


б)



	student_id [PK] integer	exam_id [PK] integer	marks smallint
1	1	1	90
2	1	2	95
3	1	3	80
4	3	3	70
5	4	2	75

в)



	student_id [PK] integer	subject_id [PK] integer
1	1	2
2	1	5
3	1	7
4	2	3
5	2	4
6	2	6
7	3	3
8	3	4

г)

Рисунок 3.8 – Реалізація допоміжних таблиць «student_exams» (а), «student_subject» (б) та їх заповнення даними (в) та (г)

3.4 Висновки до третього розділу

В третьому розділі робити була розроблена модель варіантів використання системи з визначенням головних акторів та ініційованих ними прецедентів. На основі моделей, які були представлені в другому розділі, та керуючись технічним завданням і відомостями з предметної області, була розроблена схема БД, яка в свою чергу була імплементована на реляційній СУБД PostgreSQL.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА РОЗГОРТАННЯ КОМПОНЕНТІВ СИСТЕМИ НА ОБЧИСЛЮВАЛЬНИХ ВУЗЛАХ

4.1 Опис набору технологій для розробки ПЗ

Для розробки ПЗ було обрано мову програмування високого рівня Java [32]. Дані приведемо стислий опис цієї мови програмування та опишемо сфери її застосування.

Java є універсальною об’єктно-орієнтованою мовою програмування, яка була розроблена для підтримки портативності та сумісності з багатьма платформами. Мова Java характеризується простотою, надійністю та безпекою, що робить її придатною для широкого кола додатків, від веб- та мобільної розробки до корпоративних систем та вбудованих пристроїв. Згідно філософії Java, що формулюється фразою «пиши один раз, запускай будь-де», що реалізовано завдяки віртуальній машині Java (JVM), скомпільований код може виконуватись на будь-якій платформі із сумісною JVM. Мова підтримує багатопоточність, автоматичне керування пам’яттю за допомогою вбудованого механізму збирання сміття (звільнення пам’яті від невикористаних змінних) та доволі обширну стандартну бібліотеку, що сприяє ефективній та масштабованій розробці. Підтримка статичної системи типів та механізмів обробки винятків підвищують надійність та зручність обслуговування коду [33, 34].

Механізм обробки виключних ситуацій Java покращує керування помилками, а її велика стандартна бібліотека надає інструменти для роботи в мережі, обробки файлів та графічних інтерфейсів користувача. Мова надає пріоритет безпеці за допомогою таких функцій, як «пісочниця» для ненадійного коду, та підтримує модульність за допомогою пакетів та модулів. Версії мови Java відповідають структурованій моделі випусків. Остання версія з довгостроковою підтримкою (LTS) станом на жовтень 2025 року – це Java 21, при цьому Java 22 та 23 як випуски без LTS впроваджують додаткові функції. Java 17, ще одна версія LTS, залишається широко використовуваною завдяки

своїй стабільності в корпоративних середовищах. Новіші версії зосереджені на покращенні продуктивності. Типове використання охоплює корпоративні додатки, де фреймворки, такі як Spring та Hibernate, забезпечують роботу великомасштабних систем, і веб-розробку з такими технологіями, як JavaServer Faces. Java поширена в розробці додатків для Android, використовуючи свій SDK для мобільних додатків. Вона також підтримує настільні додатки, наукові обчислення та вбудовані системи завдяки своїй універсальності [35, 36].

Java Enterprise Edition (Java EE), тепер відома як Jakarta EE, – це платформа, побудована на базі Java Standard Edition (Java SE), призначена для розробки масштабних, розподілених та надійних корпоративних застосунків в різних сферах життєдіяльності, фінансової та банківської, охорони здоров'я, телекомунікації тощо. Платформа Java EE надає комплексний набір API та специфікацій для створення масштабованих, безпечних транзакційних систем, головним чином для веб-середовищ та корпоративних середовищ. Java EE спрощує складні завдання, такі як керування паралельним виконанням, безпекою та збереженням даних, пропонуючи стандартизовані рішення для потреб підприємства. Вона включає такі технології, як Enterprise JavaBeans (EJB) для бізнес-логіки, Java Message Service (JMS) для обміну повідомленнями та JavaServer Faces (JSF) для веб-інтерфейсів, що робить її ідеальною для застосунків, що вимагають високої доступності та інтеграції з базами даних або застарілими системами. На базовому рівні для роботи з протоколом HTTP розроблено бібліотеку сервлетів. Сервлети – це класи Java, що розширюють можливості серверів, в основному використовуються для обробки HTTP-запитів та відповідей у веб-додатках. Працюючи в веб-контейнери, сумісними з Java EE (наприклад, Apache Tomcat або Jetty), сервлети обробляють запити клієнтів, генерують динамічний контент (наприклад, HTML-сторінки) та керують сеансами. Вони є доволі гнучкими, підтримують багатопоточність для одночасної обробки кількох запитів і часто використовуються разом із

фреймворками, такими як Spring MVC, для створення RESTful API або динамічних веб-сайтів. Сервлети забезпечують легкий та ефективний спосіб створення логіки на стороні сервера, формуючи основу багатьох веб-додатків на базі Java. Для керування доступом до БД використовується відповідна бібліотека: Java Database Connectivity (JDBC) – це набір інтерфейсів та функцій, що дозволяє Java-додаткам взаємодіяти з реляційними БД. Вони надають стандартний інтерфейс для підключення до БД, виконання SQL-запитів та отримання результатів, абстрагуючи складнощі протоколів БД, специфічних для певних постачальників. JDBC підтримує такі операції, як запити, оновлення та управління транзакціями, що робить його важливим для додатків, що потребують постійного зберігання даних. Він легко інтегрується з Java EE, дозволяючи розробникам підключатися до баз даних, таких як MySQL, PostgreSQL або Oracle, використовуючи драйвери, що надаються постачальниками баз даних.

Ці технології широко використовуються в корпоративних системах, таких як банківські платформи, додатки електронної комерції та системи управління контентом, де масштабованість, надійність та інтеграція даних є критично важливими [37-39].

Для гнучкого керування підключеннями до БД також використовується бібліотека HikariCP. Hikari – це високопродуктивна, гнучка бібліотека пулу JDBC-підключень для Java-додатків, розроблена для ефективного керування підключеннями до БД шляхом підтримки пулу повторно використовуваних підключень, а не їх створення та закриття для кожного запиту. Такий підхід мінімізує накладні витрати, покращує масштабованість та зменшує затримку в середовищах з інтенсивним використанням БД. Ключові функції включають налаштовуване визначення розміру пулу для неактивних та максимальних підключень, автоматичну перевірку підключень та виявлення витоків, обробку тайм-ауту для отримання даних та витоків, а також підтримку швидкого відновлення після перезапуску БД або проблем з мережею. Вона бездоганно

інтегрується з такими фреймворками, як Spring Boot, де служить пулом за замовчуванням, і працює з драйверами JDBC для баз даних, таких як MySQL, PostgreSQL та Oracle. Розробники зазвичай налаштовують її за допомогою таких властивостей, як maximumPoolSize (за замовчуванням 10), minimumIdle, connectionTimeout та idleTimeout, щоб збалансувати продуктивність та використання ресурсів [40, 41].

Для підтримки асинхронного програмування використано вбудовані класи CompletableFuture. Клас CompletableFuture в Java, що вперше представлений у Java 8, є потужним інструментом для асинхронного програмування, що дозволяє неблокуюче, одночасне виконання завдань. Будучи частиною пакета java.util.concurrent, він розширює інтерфейс Future та надає гнучкий, вільний API для обробки асинхронних обчислень. На відміну від традиційного Future попередніх версій мови Java, який вимагає ручного опитування результатів, CompletableFuture дозволяє розробникам виконувати операції асинхронного, обробляти результати або винятки та декларативно складати кілька асинхронних завдань. Ключові функції включають можливість асинхронного виконання завдань за допомогою таких методів, як supplyAsync (для завдань, що повертають значення) або runAsync (для завдань без повертання значення), зазвичай використовуючи Executor або ForkJoinPool за замовчуванням. Він підтримує ланцюжок операцій з такими методами, як thenApply (для перетворення результатів), thenAccept (для споживання результатів) та thenRun (для виконання подальших дій). Для об’єднання кількох екземплярів CompletableFuture доступні методи, такі як thenCompose (для залежних завдань) та allOf/anyOf (для обробки кількох об’єктів). Обробка винятків спрощена за допомогою exceptional або handle, що дозволяє коректне відновлення після збоїв. CompletableFuture широко використовується в сценаріях, що вимагають високого рівня паралельності, таких як обробка викликів API, запитів до БД або операцій вводу-виводу у веб-додатках, мікросервісах або реактивних системах. Він спрощує складні асинхронні робочі процеси, покращуючи продуктивність та швидкість реагування, зберігаючи при цьому читабельність коду [42-43].

4.2 Програмна реалізація системи згідно триступеневої архітектури

4.2.1 Реалізація компонентів взаємодії з БД

Для взаємодії з БД в програмі необхідно передбачити гнучкий механізм отримання об’єкту підключення Connection, що є частиною інтерфейсу (API) специфікації JDBC. Цей об’єкт підключення інкапсулює дані про підключення до БД, драйвер для роботи з БД та інші параметри, наприклад облікові дані для авторизованого доступу до даних БД. На Рисунку 4.1 показано ієрархію викликів (звернень до БД) засобами JDBC для платформи Java. Верхній рівень архітектури представлений прикладним ПЗ (Application), що виконується на цільовій платформі, наприклад в контейнері сервлетів (Servlets), або іншому середовищі. Методи та інтерфейси JDBC виконують роль проміжного середовища між ПЗ та цільовою СУБД, до них відносяться інтерфейс API та драйвер JDBC. Нарешті конкретна СУБД, наприклад Oracle, MS SQL Server, PostgreSQL тощо, знаходиться на найнижчому рівні ієрархії.

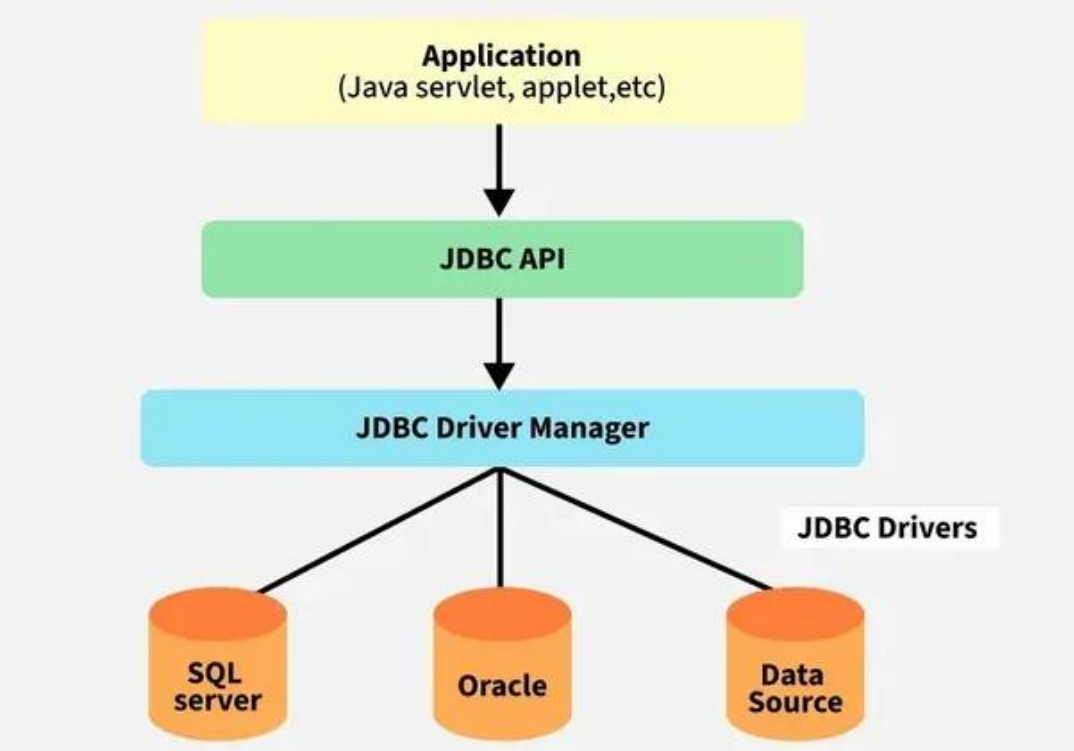


Рисунок 4.1 – Архітектура компонентів взаємодії з БД

Для розширення функціональних можливостей програмних компонентів роботи з БД додатково використовується пул підключень. На Рисунку 4.2 показано схему взаємодії елементі прикладного коду додатку (App) через деякий об’єкт доступу до даних (Data Access Object, DAO) з СУБД (DBMS) через проміжний пул підключення (Connection Pool). Згідно цієї архітектури код прикладного ПЗ отримає об’єкт підключення Connection шляхом виклику методу getConnection, який в свою чергу повертає один із зарезервованих підключень до ціпової СУБД, що керується бібліотекою пулу.

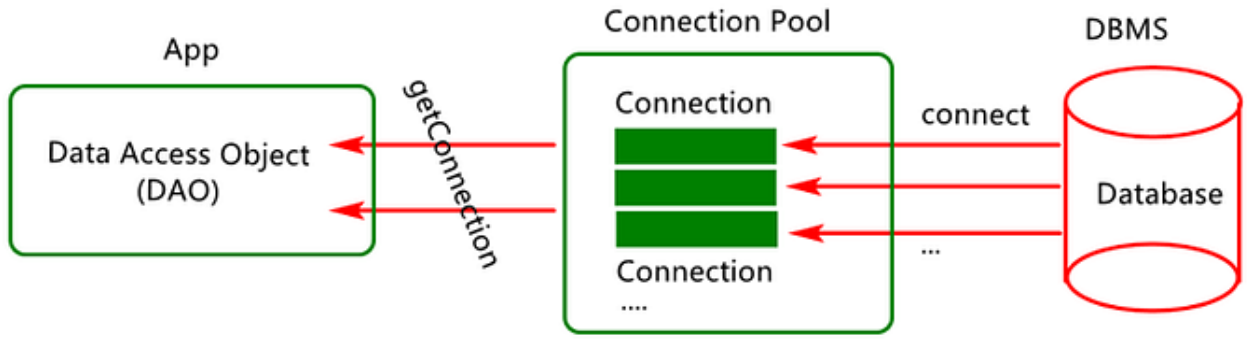


Рисунок 4.2 – Взаємодія з СУБД через пул підключень

Програмна реалізації вищеописаної архітектури наведена в Лістингу 4.1. Пул підключень реалізовано за допомогою бібліотеки Hikari. Додатково стандартними засобами мови Java забезпечено завантаження параметрів з конфігураційного файлу db.properties.

Лістинг 4.1 – Програмна реалізації джерела підключень до БД через пул підключень Hikari

```
public class DbConnection {
     private static final HikariDataSource ds = new
HikariDataSource(loadConfig());

     private static HikariConfig loadConfig() {
          Properties p = new Properties();
          HikariConfig config = new HikariConfig();
```

```
          try (InputStream input =
DbConnection.class.getClassLoader().getResourceAsStream("db.propert
ies")) {
               p.load(input);
               config.setJdbcUrl(p.getProperty("connection_url"));

     config.setDriverClassName(p.getProperty("db_driver"));
             config.setUsername(p.getProperty("username"));
             config.setPassword(p.getProperty("password"));

config.setMaximumPoolSize(Integer.parseInt(p.getProperty("maximumPo
olSize")));

config.setIdleTimeout(Integer.parseInt(p.getProperty("idleTimeout")
));

config.setMinimumIdle(Integer.parseInt(p.getProperty("minimumIdle")
));

config.setConnectionTimeout(Integer.parseInt(p.getProperty("connect
ionTimeout")));
             config.addDataSourceProperty("cachePrepStmts",
p.getProperty("cachePrepStmts"));
             config.addDataSourceProperty("prepStmtCacheSize",
p.getProperty("prepStmtCacheSize"));
             config.addDataSourceProperty("prepStmtCacheSqlLimit",
p.getProperty("prepStmtCacheSqlLimit"));

     public static Connection getConnection() throws SQLException {
          return ds.getConnection();
     }

}
```

Як видно із Лістингу 4.1, компонент доступу до БД реалізовано у вигляді окремого класу Java, що інкапсулює об’єкт HikariDataSource як статичне поле, що ініціалізується на етапі завантаження за допомогою додаткового методу loadConfig. Інтерфейс класу для подальшого використання представлений єдиним публічним методом getConnection, що повертає об’єкт Connection для підключення до БД.

4.2.2 Реалізація програмних модулів

Серверні компоненти ПЗ для ІКС екзаменаційно-тренінгового центру іноземних мов написані на мові Java і складаються із об’єктів моделі User, Role,

Student, Exam, Subject, а також касів DAO для доступу до даних (інкапсулюють JDBC): ExamDao, StudentDao, SubjectDao. Лістинги відповідних компонентів наведені в Додатку Д.

Клас User (див. Лістинг Д.1) представляє користувача в системі є однією з ключових складових моделі даних додатку. Цей клас відповідає за збереження основної інформації про користувача, включаючи унікальний ідентифікатор, ім’я користувача та адресу електронної пошти, а також зв’язок із ролями, які визначають права доступу користувача в системі. Для зберігання ролей використовується колекція типу ArrayList, що дозволяє асоціювати користувача з кількома ролями одночасно. Конструктори класу дають змогу створювати об’єкти користувача або з вказанням ідентифікатора, імені та електронної пошти, або лише з ім’ям і електронною поштою, що забезпечує гнучкість при ініціалізації об’єктів. Для доступу до полів класу передбачені методи отримання та встановлення значень ідентифікатора, імені користувача та електронної пошти, що забезпечує інкапсуляцію даних. Окремо реалізований метод для отримання масиву ролей, який повертає копію списку ролей у вигляді масиву, та метод для додавання нової ролі до списку, що дозволяє динамічно змінювати набір ролей користувача. Метод toString перевизначений для створення текстового представлення об’єкта, яке включає ідентифікатор, ім’я, електронну пошту та перелік назв ролей, якщо такі є, або повідомлення про їх відсутність. Така структура класу User забезпечує зручне управління даними користувача та їх взаємозв’язок із ролями в системі, що є важливим для реалізації функціоналу авторизації та розмежування доступу в додатку.

Клас Role (див. Лістинг Д.2) відповідає за представлення ролі користувача, яка визначає його права та функції в додатку. Цей клас містить чотири поля: унікальний ідентифікатор, назву ролі, заголовок та повідомлення, пов’язане з роллю. Назва ролі слугує для ідентифікації ролі в системі, заголовок може використовуватися для відображення в інтерфейсі, а повідомлення містить додаткову інформацію та опис ролі. Конструктор класу дозволяє створювати об’єкт ролі з ініціалізацією всіх полів, забезпечуючи повне заповнення даних

при створенні. Для доступу до полів передбачені методи отримання та встановлення значень ідентифікатора, назви, заголовка та повідомлення, що забезпечує інкапсуляцію даних і можливість їхньої зміни. Метод toString перевизначений для створення текстового представлення об’єкта, яке включає всі поля ролі, що зручно для налагодження або налагодження. Клас Role тісно пов’язаний із класом User, оскільки користувач може мати кілька ролей, які додаються до його списку через відповідний метод у класі User. Ця структура забезпечує гнучке управління ролями в системі, дозволяючи визначати різні рівні доступу та функціональні можливості для користувачів.

Клас Student (див. Лістинг Д.3) відповідає за представлення студента. Цей клас містить поля для зберігання унікального ідентифікатора, імені студента, назви навчальної програми, номера залікової книжки, посилання на об’єкт користувача (User) та списку екзаменів, які студент складав або планує складати. Список екзаменів реалізовано за допомогою колекції ArrayList, що дозволяє динамічно додавати та зберігати інформацію про екзамени. Клас має три конструктори, які забезпечують гнучкість при створенні об’єктів: перший приймає ідентифікатор, ім’я, програму та номер залікової книжки; другий додатково включає посилання на об’єкт користувача; третій ініціалізує об’єкт без ідентифікатора, що може бути корисно при створенні нового студента до збереження в БД. Для доступу до полів передбачені методи отримання та встановлення значень ідентифікатора, імені, програми, номера залікової книжки та пов’язаного користувача, що забезпечує інкапсуляцію даних. Також є методи для додавання екзамену до списку та отримання масиву екзаменів, що дозволяє керувати зв’язком студента з екзаменами. Метод toString перевизначений для створення текстового представлення об’єкта, яке включає всі поля, зокрема список екзаменів і пов’язаного користувача, що зручно для відображення інформації. Методи hashCode та equals перевизначені для порівняння об’єктів за ідентифікатором, ім’ям, програмою та номером залікової книжки, що забезпечує коректну роботу при використанні об’єктів у колекціях. Клас Student пов’язує

дані користувача (через об’єкт User) та екзамени, що є важливим для відстеження навчального прогресу студента в системі.

Клас Exam (Лістинг Д.4) відповідає за представлення екзамену. Цей клас містить поля для зберігання унікального ідентифікатора, місця проведення екзамену, номера аудиторії, дати та часу проведення, а також посилання на об’єкт предмета (Subject), з якого проводиться екзамен. Крім того, клас включає колекцію студентів, які беруть участь в екзамені, реалізовану через ArrayList, та словник для зберігання оцінок студентів, де ключем є об’єкт студента, а значенням – його оцінка. Для створення об’єктів екзамену передбачено два конструктори: один ініціалізує всі основні поля, включаючи предмет, а другий дозволяє створювати екзамен без вказання предмета, що може бути корисно на етапі підготовки даних. Для доступу до полів класу реалізовані методи отримання та встановлення значень ідентифікатора, місця, аудиторії, дати, часу та предмета, що забезпечує інкапсуляцію даних. Методи для роботи зі списком студентів дозволяють додавати студента до екзамену та отримувати масив зареєстрованих студентів. Окремо передбачені методи для встановлення та отримання оцінки для конкретного студента, що забезпечує управління результатами екзамену. Метод toString перевизначений для створення текстового представлення об’єкта, яке включає ідентифікатор, місце, аудиторію, дату, час і предмет, що зручно для відображення інформації. Методи hashCode та equals перевизначені для порівняння екзаменів за їхніми ключовими полями, що забезпечує коректну роботу з об’єктами в колекціях. Клас Exam відіграє важливу роль у системі, пов’язуючи студентів, предмети та результати екзаменів, що дозволяє відстежувати навчальний процес і оцінювання.

Клас Subject (Лістинг Д.5) відповідає за представлення навчального предмета, який є важливою частиною структури додатку. Цей клас містить поля для зберігання унікального ідентифікатора, назви предмета, імені викладача, номера семестру, в якому викладається предмет, та кількості кредитів, що присвоюються за його успішне завершення. Крім того, клас включає дві колекції типу ArrayList: одну для зберігання списку студентів, які вивчають предмет, і

другу для списку екзаменів, пов’язаних із предметом. Конструктор класу ініціалізує об’єкт предмета з указанням ідентифікатора, назви, викладача, семестру та кредитів, забезпечуючи повне заповнення даних. Для доступу до полів передбачені методи отримання та встановлення значень ідентифікатора, назви, викладача, семестру та кредитів, що забезпечує інкапсуляцію даних. Методи для роботи зі списками дозволяють додавати студента або екзамен до відповідних колекцій, а також отримувати масиви студентів і екзаменів, що забезпечує управління зв’язками між предметом, студентами та екзаменами. Метод toString перевизначений для створення текстового представлення об’єкта, яке включає всі поля, а також списки студентів і екзаменів, що зручно для відображення інформації. Методи hashCode та equals перевизначені для порівняння об’єктів предметів за їхніми ключовими полями, що забезпечує коректну роботу в колекціях. Клас Subject відіграє ключову роль у системі, пов’язуючи навчальні програми, студентів і екзамени, що дозволяє організувати процес навчання та оцінювання.

Клас UserDao (Лістинг Д.6) відповідає за управління даними користувачів і їхніх ролей, забезпечуючи взаємодію з БД через JDBC. Цей клас використовує пул із десяти потоків для асинхронного виконання операцій, що дозволяє ефективно обробляти запити до бази даних. Основна функціональність класу охоплює отримання даних про користувачів за ідентифікатором або ім’ям, пошук списків користувачів із можливістю фільтрації за ім’ям і електронною поштою, а також із обмеженням кількості записів. Для кожної з цих операцій передбачені асинхронні методи, які повертають об’єкти CompletableFuture, забезпечуючи неблокуюче виконання. Наприклад, метод для отримання користувача за ідентифікатором формує SQL-запит до таблиці users, отримує ім’я та електронну пошту, створює об’єкт користувача і повертає його, або null у разі помилки. Аналогічно працює метод пошуку за ім’ям користувача. Для отримання списків користувачів реалізовані перевантажені методи, які дозволяють задавати критерії пошуку та обмеження кількості записів, використовуючи параметризовані запити для безпеки та гнучкості. Клас також

підтримує операції оновлення профілю користувача, зокрема зміну імені та електронної пошти, а також оновлення хешу пароля. Для автентифікації користувача передбачений метод, який перевіряє відповідність імені та хешу пароля запису в базі даних. Операція вставки нового користувача додає дані в таблицю users, зберігає хеш пароля та оновлює ідентифікатор створеного користувача. Видалення користувача реалізовано через SQL-запит із перевіркою кількості змінених записів. Управління ролями користувача включає методи для завантаження ролей із бази даних і їх оновлення. Завантаження ролей виконується через об’єднання таблиць roles і user_roles, додаючи знайдені ролі до об’єкта користувача. Оновлення ролей передбачає видалення існуючих зв’язків і вставку нових у транзакції, що гарантує цілісність даних. Метод для отримання ролі за назвою повертає об’єкт ролі з її ідентифікатором, заголовком і повідомленням. Усі методи обробляють винятки SQL, виводячи повідомлення про помилки в консоль, що полегшує діагностику. Метод завершення роботи закриває пул потоків, забезпечуючи коректне звільнення ресурсів. Клас UserDao забезпечує повноцінне управління користувачами та їхніми ролями, інкапсулюючи логіку доступу до даних і надаючи асинхронний інтерфейс для ефективної роботи системи.

Клас StudentDao (Лістинг Д.7) відповідає за управління даними студентів, забезпечуючи взаємодію з БД. Він реалізує методи для виконання основних операцій із даними студентів, таких як пошук, створення, оновлення та видалення записів. Метод пошуку студента за ідентифікатором виконує SQL-запит до таблиці students, отримуючи ім’я, програму та номер залікової книжки, після чого створює об’єкт студента з цими даними. У разі помилки повертається null, а повідомлення про виняток виводиться в консоль. Аналогічний підхід застосовується в методі пошуку студента за номером залікової книжки, який повертає об’єкт студента з відповідним ідентифікатором, ім’ям і програмою. Для отримання списку студентів передбачений метод, який дозволяє фільтрувати записи за ім’ям і програмою з використанням Optional для гнучкої обробки

параметрів. Цей метод формує параметризований запит із сортуванням за ім’ям, повертаючи масив об’єктів студентів або порожній масив у разі помилки.

Клас також містить методи для отримання списків студентів, які пов’язані з певним предметом або екзаменом. Метод для предмета виконує запит із об’єднанням таблиць students, student_subjects і subjects, щоб знайти всіх студентів, записаних на вказаний предмет, з результатами, відсортованими за ім’ям. Аналогічно, метод для екзамену об’єднує таблиці students, student_exams і exams, щоб повернути студентів, які беруть участь у конкретному екзамені. Обидва методи повертають масиви студентів або порожній масив у разі помилки. Операція оновлення даних студента дозволяє змінювати ім’я, програму, номер залікової книжки та пов’язаного користувача, використовуючи SQL-запит із перевіркою на наявність пов’язаного користувача для коректного встановлення null у полі user_id. Метод створення студента додає новий запис у таблицю students із вказаними ім’ям, програмою та номером залікової книжки, повертаючи true у разі успіху. Видалення студента виконується через запит, який видаляє запис за ідентифікатором, із перевіркою кількості змінених рядків. Усі методи обробляють винятки SQL, виводячи повідомлення про помилки в консоль для спрощення діагностики.

Клас ExamDao (див. Лістинг Д.8) відповідає за управління даними екзаменів, забезпечуючи взаємодію з БД через JDBC. Цей клас містить методи для роботи з екзаменами, зокрема для отримання списку екзаменів, пов’язаних із конкретним студентом. Метод для отримання екзаменів студента виконує SQL-запит, який об’єднує таблиці exams, student_exams і students, щоб знайти всі екзамени, на які зареєстрований вказаний студент. Запит повертає ідентифікатор екзамену, місце проведення, номер аудиторії та дату, які використовуються для створення об’єктів екзаменів. Ці об’єкти додаються до колекції, яка потім перетворюється на масив. У разі помилки SQL-запиту виводиться повідомлення про помилку, а метод повертає порожній масив.

Клас також містить метод для отримання всіх екзаменів, але в поточній реалізації він повертає лише порожній масив, що може свідчити про

незавершену функціональність або заглушку для подальшої розробки. Усі методи обробляють винятки SQL, виводячи повідомлення про помилки в консоль для полегшення діагностики. Клас ExamDao забезпечує базову функціональність для управління екзаменами, зокрема зв’язок між студентами та їхніми екзаменами, що є важливим для організації навчального процесу.

Клас SubjectDao (див. Лістинг Д.9) відповідає за управління даними навчальних предметів, забезпечуючи взаємодію з базою даних через JDBC. Цей клас реалізує методи для виконання основних операцій із предметами, таких як пошук окремого предмета, отримання списку предметів, пов’язаних із певним студентом, та отримання всіх предметів у системі. Метод пошуку предмета за ідентифікатором формує параметризований SQL-запит до таблиці subjects, отримуючи назву, викладача, семестр і кількість кредитів. За наявності результату створюється об’єкт предмета з цими даними, інакше повертається null. У разі виникнення помилки SQL виводиться повідомлення в консоль для полегшення діагностики. Метод для отримання предметів, пов’язаних зі студентом, виконує SQL-запит із об’єднанням таблиць subjects, student_subjects і students, щоб знайти всі предмети, на які записаний студент за його ідентифікатором. Результати запиту обробляються для створення колекції об’єктів предметів, яка потім перетворюється на масив. Якщо запит не виконується через помилку, повертається порожній масив, а повідомлення про виняток виводиться в консоль. Метод для отримання всіх предметів використовує простий SQL-запит до таблиці subjects із сортуванням за назвою предмета, створюючи колекцію об’єктів із отриманих даних, яку потім перетворює на масив. У разі помилки також повертається порожній масив. Усі методи застосовують параметризовані запити або прості запити через Statement, що забезпечує безпеку та захист від SQL-ін’єкцій. Обробка винятків SQL у всіх методах здійснюється з виводом повідомлень про помилки, що сприяє зручному відстеженню проблем. Клас SubjectDao відіграє важливу роль у системі, забезпечуючи управління даними предметів і їхніми зв’язками зі студентами, що необхідно для організації навчального процесу та планування екзаменів.

4.2.3 Реалізація елементів інтерфейсу користувача

Інтерфейс користувача виконано на базі технологій Hypertext Markup Language (HTML) та Cascading Style Sheets (CSS). Відповідно до стандарту HTML5, верстання шаблону сторінок виконано з використанням семантичних елементів: main, nav, article, section, aside, footer. На Рисунку 4.3. показано схематично макет шаблону сторінок сайту додатку.

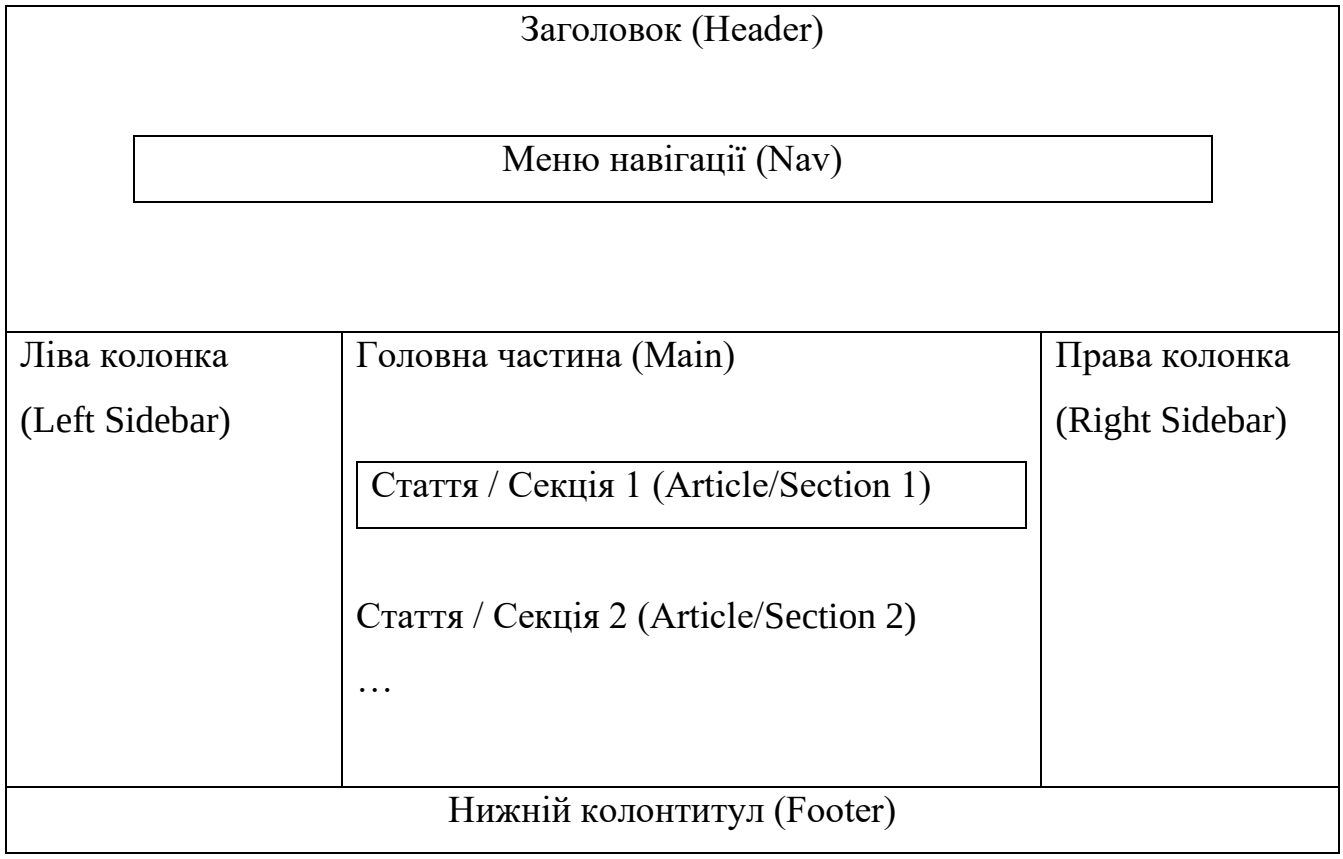


Рисунок 4.3 – Структура шаблону сторінок сайту

Реалізація шаблону сторінки сайту додатку реалізує структуру типової веб-сторінки з використанням JavaServer Pages (JSP) та стандартних бібліотек JSTL. Директива сторінки встановлює мову програмування Java, тип вмісту HTML із кодуванням UTF-8 та кодування сторінки UTF-8, що забезпечує коректне відображення тексту українською та іншими мовами. Директива taglib підключає

бібліотеку JSTL core з префіксом «c», що дозволяє використовувати теги для обробки умов, циклів та інших логічних конструкцій у JSP без вбудовування Java-коду безпосередньо в HTML.

Документ HTML починається з оголошення DOCTYPE та кореневого елемента html. У секції head встановлюється мета-тег charset для UTF-8, підключається таблиця стилів styles.css через тег link із використанням тегу c:url, що забезпечує коректне формування абсолютних шляхів до ресурсів з урахуванням контексту веб-додатку. Заголовок сторінки встановлюється як «Private School Website», що відображає призначення системи. Тіло документа body містить основну структуру сторінки, побудовану на основі включення окремих JSP-фрагме02.01.2026нтів через директиви jsp:include. Шапка сайту включається з файлу header.jsp, розташованого в директорії WEB-INF/views, що забезпечує ізоляцію представлення від прямого доступу користувача через URL. Головна частина сторінки, позначена класом main, складається з двох паралельних включень: sidebars.jsp для бічної панелі навігації та content.jsp для основного вмісту. Така структура дозволяє модульно керувати компонентами інтерфейсу, замінюючи вміст бічної панелі та основної області залежно від логіки додатку та прав доступу користувача. Нижня частина сторінки завершується включенням footer.jsp, що містить підвал сайту з загальною інформацією, копірайтом чи додатковими посиланнями.

Таким чином, даний шаблон забезпечує загальну структуру всіх сторінок додатку, полегшуючи підтримку та модифікацію інтерфейсу. Використання jsp:include дозволяє динамічно генерувати різні сторінки шляхом заміни вмісту content.jsp та sidebars.jsp, зберігаючи спільні елементи шапки та підвалу. Розташування файлів у WEB-INF/views гарантує їхню безпеку, оскільки ця директорія недоступна для прямого запиту через браузер. Підключення стилів через c:url забезпечує коректну роботу з різними серверами та контекстами розгортання, автоматично формуючи правильні шляхи до статичних ресурсів. Такий підхід відповідає принципам MVC-архітектури, де JSP відповідає за представлення, а логіка обробки запитів реалізується в сервлетах або

контролерах. Код файлів index.jsp, header.jsp, sidebars.jsp, content.jsp та footer.jsp показано в Лістингах Д.10 – Д.14. На Рисунку 4.4 показано приклад демонстраційної сторінки сайту додатку.

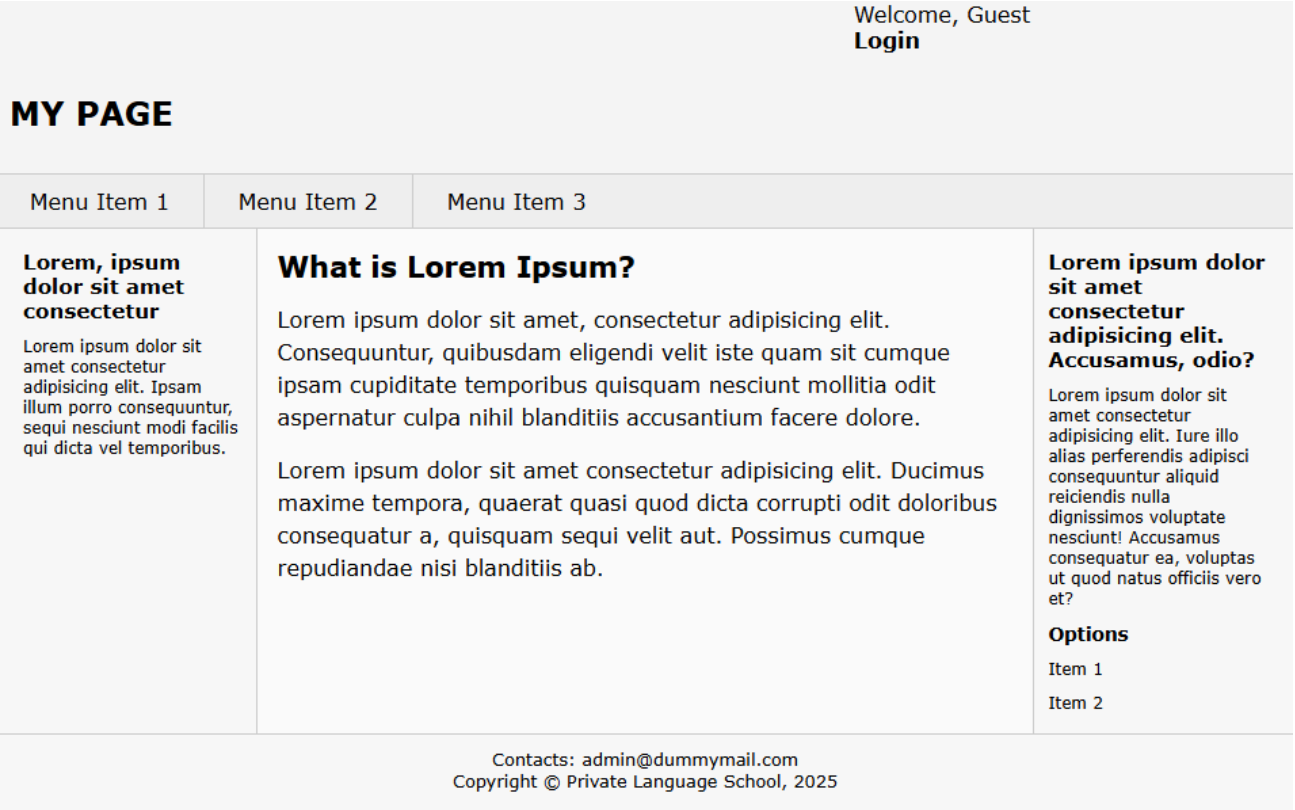


Рисунок 4.4 – Шаблон сторінки сайту системи

Після програмування функціональні елементи сторінки інтерфейсу користувача керуються програмно, шляхом заповнення їх вмісту відповідним серверним компонентом Java-програми.

Першою функцією, яка викликається користувачам після відкриття додатку є авторизація користувача (Рисунок 4.5). Без авторизації користувач бачить фрагменти інтерфейсу із вмістом, що призначений для гостьового перегляду.

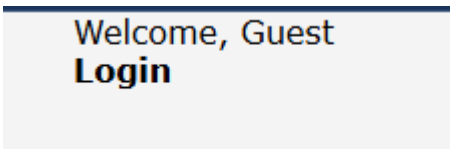


Рисунок 4.5 – Вітання гостя системи із запрошенням авторизації

Після натискання посилання Login (див. Рисунок 4.5) відбувається переплавлення на допоміжну сторінку із формою для введення облікових даних користувача системи (Рисунок 4.6). Як видно із рисунку, форма складається із полів для вводу 1) імені користувача, 2) паролю, а також 3) прапору збереження інформації для входу у файлах кукі.

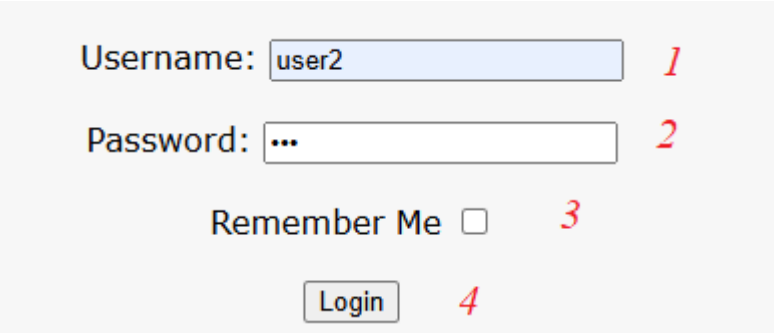


Рисунок 4.6 – Форма авторизації користувача

Після успішного введення імені користувача та пароля відбувається переправлення на головну сторінку, на якій відображається вміст сторінки, що асоціюється із даним користувачем, зокрема, замість запрошення до авторизації відображається вітання користувача та кнопка виходу із системи Logout (Рисунок 4.7).

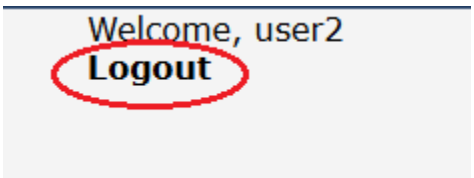


Рисунок 4.7 – Вітання авторизованого користувача та кнопка виходу із системи

Відповідно до структури інтерфейсу, лава бічна панель заповнюється даними щодо користувача системи та усіх його ролей в системі. Також наддається довідкова інформації про доступні функціональні можливості користувача. На Рисунку 4.8 показано приклад такої панелі для авторизованого користувача, який має дві ролі: екзаменований (Test Taker) та студент (Student).

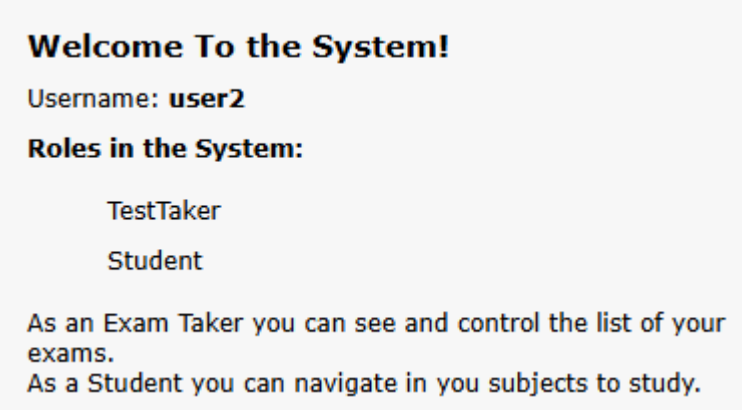


Рисунок 4.8 – Ліва бічна панель із інформацією про користувача та ролі

Аналогічним чином заповнюється даними права бічна панель (Рисунок 4.9), з відмінністю у тому, що тут дані стосуються безпосередньо студента – повне ім’я (Name), освітня програма (Program), та номер залікової книжки (Record).

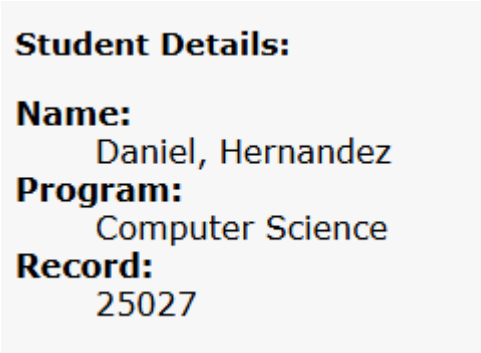


Рисунок 4.9 – Права бічна панель

Після авторизації також стає доступним головне меню користувача із зазначенням пунктів згідно кожної із ролей користувача в системі. Так, наприклад, на Рисунку 4.10 показано фрагмент меню для користувача із усіма доступними ролями: панель адміністрування системи (Admin Menu), дисципліни (My Subjects) та екзамени (My Exams).

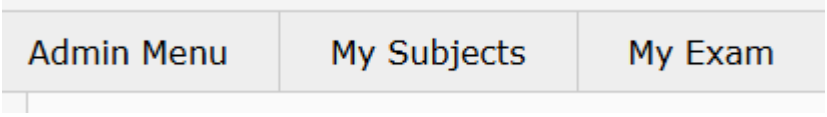


Рисунок 4.10 – Головне меню

Згідно технічного завдання головною функцією авторизованого користувача в системі є реєстрація на вивчення навчальних дисциплін та запис на екзамени. На Рисунку 4.11 показана форма реєстраційної форми, де користувач може обрати доступні до вивчення дисципліни та екзамени. Для цього необхідно вказати ім’я користувача (студента) та його ідентифікатор. Заявка (Application) обробляється адміністратором системи.

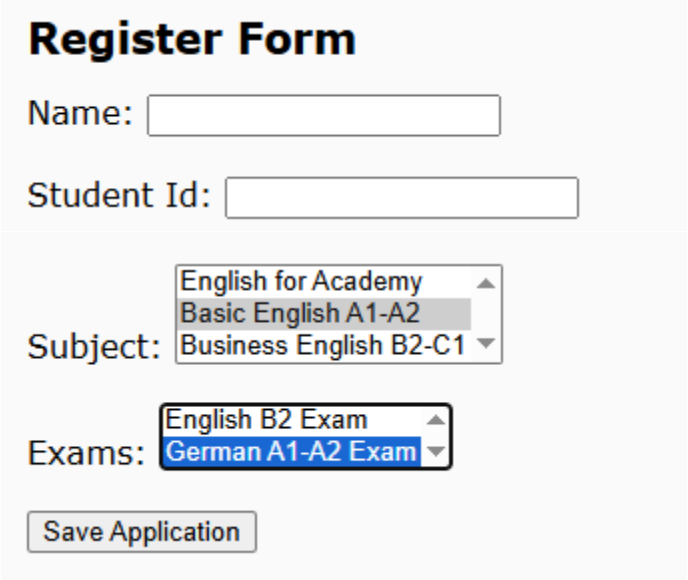


Рисунок 4.11 – Форма реєстрації на курси та екзамени

На Рисунку 4.12, а-б показано списки дисципліни та екзаменів, які можна побачити перейшовши за відповідним пунктом меню користувача, що показано на Рисунку 4.10. Інформація про навчальну дисципліну включає повну назву дисципліни, відомості про викладача, порядковий номер семестру, протягом якого вона вивчається, а також кількість кредитів.

Щодо списку екзаменів, то тут зібрані такі дані: назва дисципліни, з якої проводиться даний екзамен, дата та час проведення, а також адреса місця проведення.

Для редагування даних, в тому числі скасування реєстрації на іспит або самостійне відрахування з дисципліни користувачу необхідно виділити відповідний елемент меню та обрати потрібну дію контекстного меню.

My Subjects

Subject	Lecturer	Semester	Credits
English for Academy	Assoc. Prof. E. J. Kim, PhD	1	40
Basic English A1-A2	Assoc. Prof. E. J. Kim, PhD	1	40
Business English B2-C1	Prof. R. S. Patel, PhD	1	40

а)

My Exams

Exam Title	Date	Place
English B2	25-10-2015 13:00	Khmelnytskyi, Instytutska 11, 4-209
German A1-A2	25-11-2025 09:30	Khmelnytskyi, Instytutska 11, 4-209
Business English B2-C1	25-12-2025 12:30	Khmelnytskyi, Instytutska 11, 4-209

б)

Рисунок 4.12 – Список навчальних дисциплін (а) та екзаменів (б)

4.3 Розгортання програмних компонентів на обчислювальних вузлах системи

Базою для розгортання розробленого ПЗ є операційна система (ОС) на базі ядра Linux. В якості дистрибутиву обрано Ubuntu Linux версії 24.04 LTS, яка характеризується довгим терміном підтримки та є найновішою доступною серверною версією станом на жовтень 2025 року. Ця ОС потребує мінімальну конфігурацію обладнання для її запуску, що відповідає 1 ГГц частоти процесора, 1 Гб оперативної пам’яті та 5 Гб вмісту накопичувача. Таким чином система може бути встановлена на мініатюрний персональний комп’ютер (ПК), портативний сервер, або на віртуальну машину.

В якості середовища виконання ОС було використано програмний засіб віртуалізації Oracle Virtual Box. На Рисунку 4.13 показано головні вікна із налаштуваннями віртуальної машини для задання конфігурації.

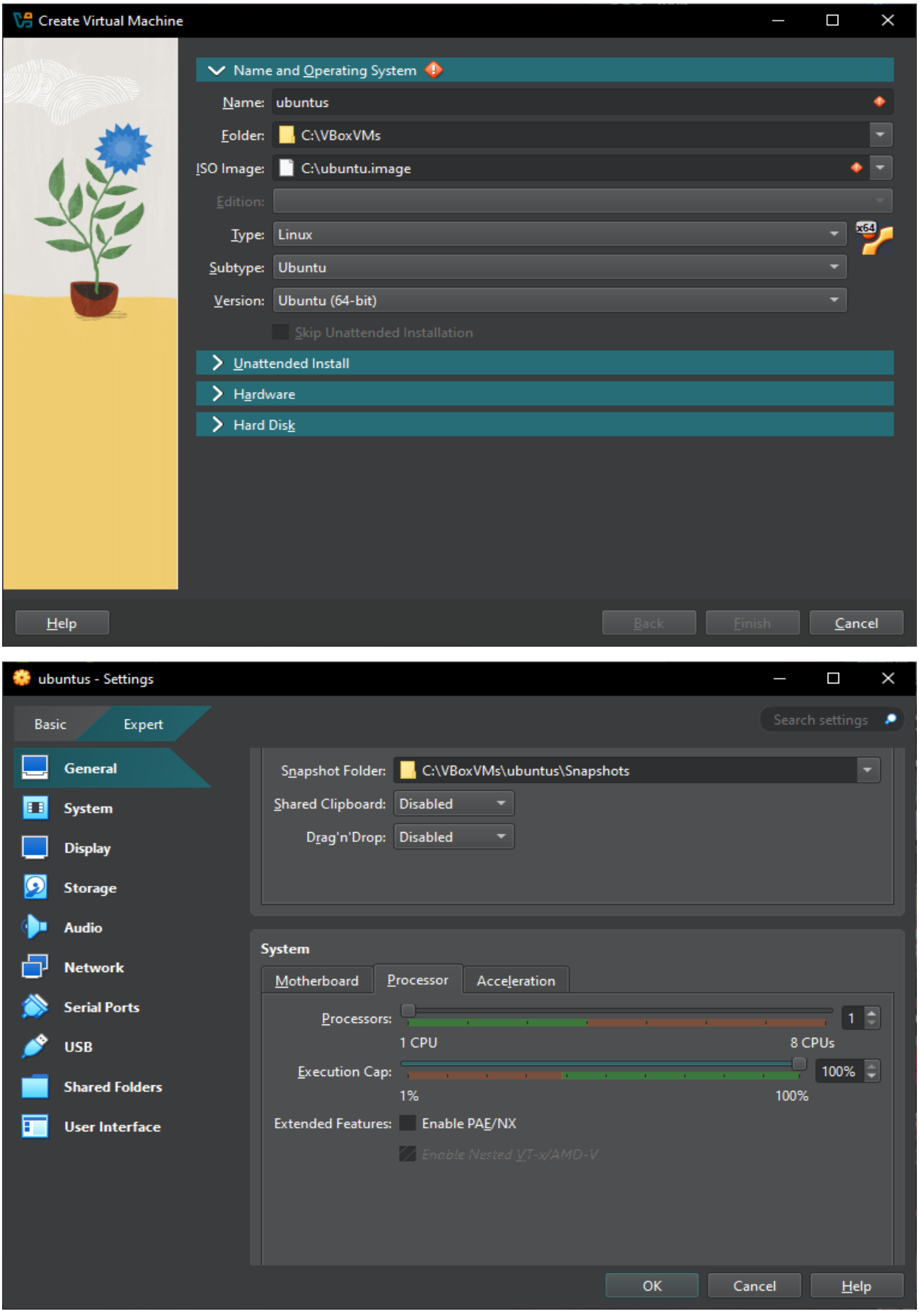


Рисунок 4.13 – Створення та налаштування віртуальної машини у VirtualBox

Для взаємодії із користувачем в якості набору елементі керування робочого столу було обрано систему LXQT, що характеризується низьким споживанням обчислювальних ресурсів. На Рисунку 4.14 показано інтерфейс системи після завантаження.

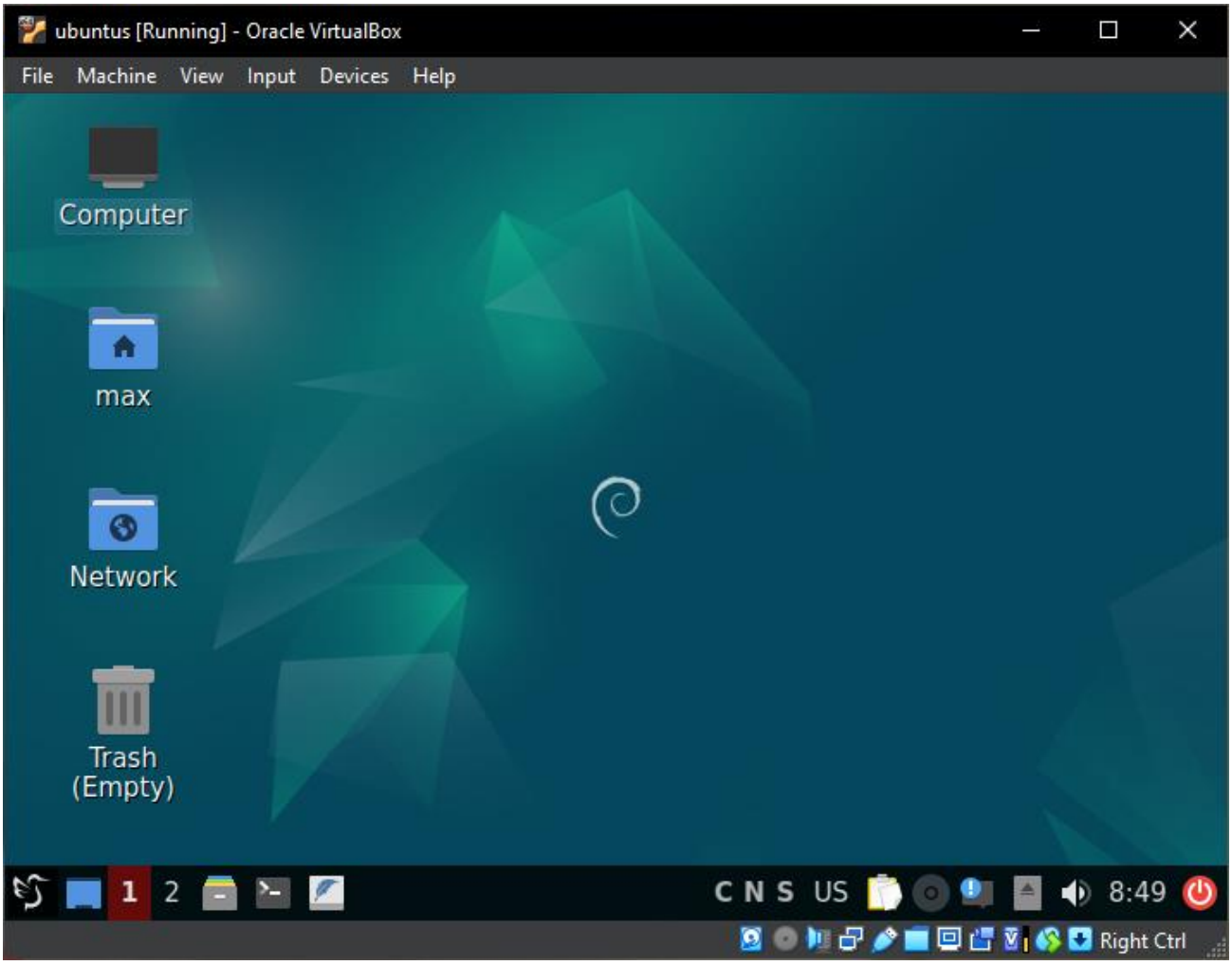


Рисунок 4.14 – Інтерфейс Ubuntu з робочим столом LXQT

Для коректної роботи ПЗ необхідно встановити Apache Tomcat, що є веб-сервером та контейнером сервлетів Java. Дане ПЗ реалізує стандарт Java EE та Jakarta EE.

Під час підготовки проекту було використано Apache Tomcat 10, що функціонує на ОС Ubuntu 24.04. Для коректної та ефективної роботи було встановлено власне сам сервер Tomcat, налаштовано ролі користувачів, а також інтерфейс адміністратора.

Після успішного встановлення ОС необхідно виконати базові налаштування для підвищення безпеки та зручності у подальшому використанні.

Для віддаленого підключення, в загальному випадку необхідно знати IP-адресу сервера, однак у випадку тестового середовища, це адреса в локальній мережі, що керується домашнім маршрутизатором. Безпечну роботу забезпечує протокол Security Shell (SSH). В більшості Linux ОС передбачено користувача із необмеженими правами – це суперкористовач root. З огляду безпеки, рекомендується не використовувати цього користувача для щоденної роботи, а створити окремого користувача.

Для встановлення Tomcat спочатку потрібно завантажити останню версію та налаштувати окремого користувача та дозволи для нього. Також необхідно встановити Java Development Kit (JDK).

З міркувань безпеки Tomcat також має працювати від імені окремого непривілейованого користувача. Для створення такого користовача виконаємо команду:

sudo useradd -m -d /opt/tomcat -U -s /bin/false tomcat

Для встановлення JDK необхідно скористатись менеджером пакетів APT виконавши команду оновлення пакетів та безпосереднього встановлення JDK:

sudo apt update

sudo apt install default-jdk

Перевірити версію встановленої JDK можна за допомогою команди java -version. На Рисунку 4.15 показано результат виведення цієї команди після встановлення JDK версії 11.

```
Output
openjdk version "11.0.14" 2022-01-18
OpenJDK Runtime Environment (build 11.0.14+9-Ubuntu-0ubuntu2.20.04)
OpenJDK 64-Bit Server VM (build 11.0.14+9-Ubuntu-0ubuntu2.20.04, mixed mode, sharing)
```

Рисунок 4.15 – Перевірка версії JDK

Завантажити пакет для встановлення Tomcat можна з репозитарії dlcdn.apache.org засобами програми wget.

wget https://dlcdn.apache.org/tomcat/tomcat-10/v10.0.20/bin/apache-tomcat-10.0.20.tar.gz

Відкрити вміст архіву засобами утиліти tar можна за допомогою команди:

sudo tar xzvf apache-tomcat-10*tar.gz -C /opt/tomcat --strip-components=1

І нарешті, для створеного користувача tomcat налаштуєм права доступу до каталогу, в який був скопійований вміст архіву із файлами програми:

sudo chown -R tomcat:tomcat /opt/tomcat/

sudo chmod -R u+x /opt/tomcat/bin

На рисунку 4.16 показані відповідні зміни конфігураційного файлу, згідно яких визначаються дві ролі користувачів, manager-gui та admin-gui, які надають доступ до сторінок Manager та Host Manager відповідно. Також визначаються два користувача, manager та admin, з відповідними ролями. За замовчуванням Tomcat налаштовано на обмеження доступу до сторінок адміністрування, якщо з'єднання не здійснюється із самого сервера. Щоб отримати доступ до цих сторінок для користувачів, що були щойно визначили, потрібно відредагувати файли конфігурації для цих сторінок.

```
<role rolename="manager-gui" />
<user username="manager" password="manager_password" roles="manager-gui" />

<role rolename="admin-gui" />
<user username="admin" password="admin_password" roles="manager-gui,admin-gui" />
```

Рисунок 4.16 – Зміни конфігурації файлу tomcat-users.xml

Щоб зняти обмеження для сторінки менеджера, відкриємо її конфігураційний файл для редагування:

sudo nano /opt/tomcat/webapps/manager/META-INF/context.xml

В цьому файлі слід закоментувати рядок Valve, як це показано на Рисунку 4.17.

```
...
<Context antiResourceLocking="false" privileged="true" >
  <CookieProcessor className="org.apache.tomcat.util.http.Rfc6265CookieProcessor"
                   sameSiteCookies="strict" />
<!-  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" /> -->
  <Manager sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|Long|Number|S
</Context>
```

Рисунок 4.17 – Конфігураційний файл context.xml

З метою забезпечення роботи Tomcat у фоновому режимі необхідно скористатись службою systemd. Також дана служба автоматично перезапустить Tomcat у разі помилки або збою.

Tomcat, будучи додатком Java, вимагає наявності середовища виконання Java, яке ви встановили разом з JDK. Налаштування Tomcat зберігаються у файлі з назвою tomcat.service, що знаходиться в /etc/systemd/system. Відкриємо новий файл для редагування за допомогою редактора nano:

sudo nano /etc/systemd/system/tomcat.service

Вміст конфігураційного файлу показаний на Рисунку 4.18. У цьому файлі слід замінити виділене значення JAVA_HOME за необхідністю.

Також у файлі визначається служба, яка запускатиме Tomcat, виконуючи надані нею сценарії запуску та завершення роботи; встановлюються кілька змінних середовища, щоб визначити її домашній каталог (/opt/tomcat) та обмеження обсягу пам’яті, що може виділити віртуальна машина Java (у CATALINA_OPTS). У разі помилки служба Tomcat автоматично перезапуститься. Зміни будуть застосовані після перезавантаження служби.

Після запуску служби Tomcat, необхідно налаштувати брандмауер, щоб дозволити підключення до Tomcat. Після цього стає можливим звернутись до його веб-інтерфейсу (Рисунок 4.19). Tomcat використовує порт 8080 для прийняття HTTP-запитів.

Переконавшись у працездатності додатку, перейдемо в меню менеджера додатків, натиснувши кнопку Management App. На цьому кроці буде запропоновано ввести облікові дані. Сторінка менеджера додатків зображена на Рисунку 4.20.

```
[Unit]
Description=Tomcat
After=network.target

[Service]
Type=forking

User=tomcat
Group=tomcat

Environment="JAVA_HOME= /usr/lib/jvm/java-1.11.0-openjdk-amd64"
Environment="JAVA_OPTS=-Djava.security.egd=file:///dev/urandom"
Environment="CATALINA_BASE=/opt/tomcat"
Environment="CATALINA_HOME=/opt/tomcat"
Environment="CATALINA_PID=/opt/tomcat/temp/tomcat.pid"
Environment="CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC"

ExecStart=/opt/tomcat/bin/startup.sh
ExecStop=/opt/tomcat/bin/shutdown.sh

RestartSec=10
Restart=always

[Install]
WantedBy=multi-user.target
```

Рисунок 4.18 – Конфігураційний файл tomcat.service

Менеджер веб-додатків використовується для керування Java-програмами. Користувач може запускати, зупиняти, перезавантажувати, розгортати та скасовувати їх розгортання. Також надаються можливості щодо діагностики своїх додатків (наприклад, для виявлення витоків пам’яті). Інформація про сервер доступна внизу сторінки.

На Рисунку 4.21 показано сторінку менеджера хостів, доступ до якої можна отримати, натиснувши відповідну кнопку на головній сторінці. На цій

сторінці користувач може створювати віртуальні хости для обслуговування програм.

Отже, в останньому розділі роботи було описано процес налаштування середовища для розгортання веб-додатку в екосистемі Java на ОС Linux (Ubuntu). Було встановлено контейнер Tomcat 10 на сервері Ubuntu 20.04.

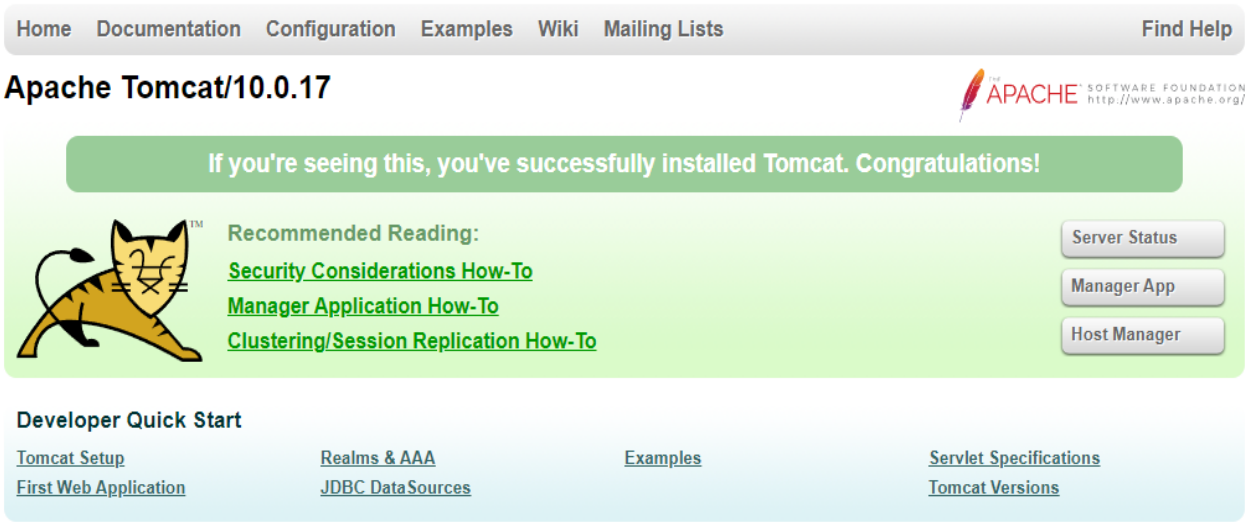


Рисунок 4.19 – Інтерфейс Tomcat 10

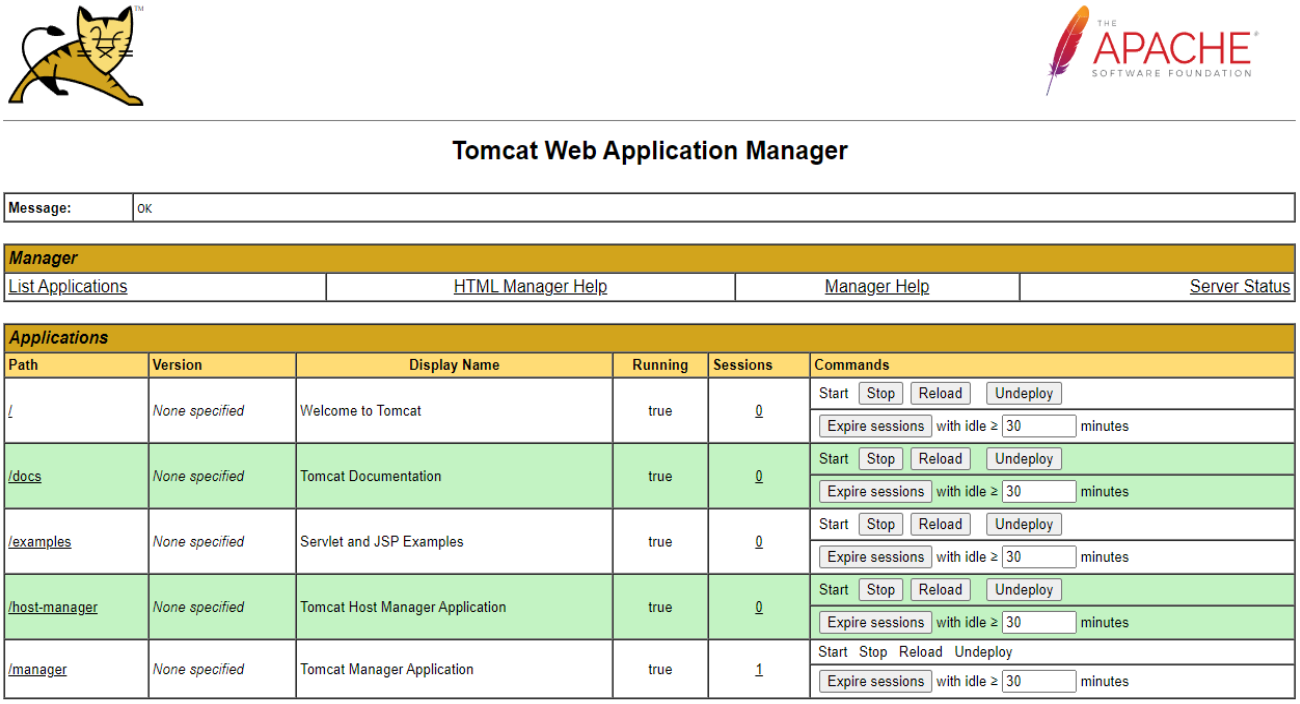


Рисунок 4.20 – Сторінка менеджера додатків

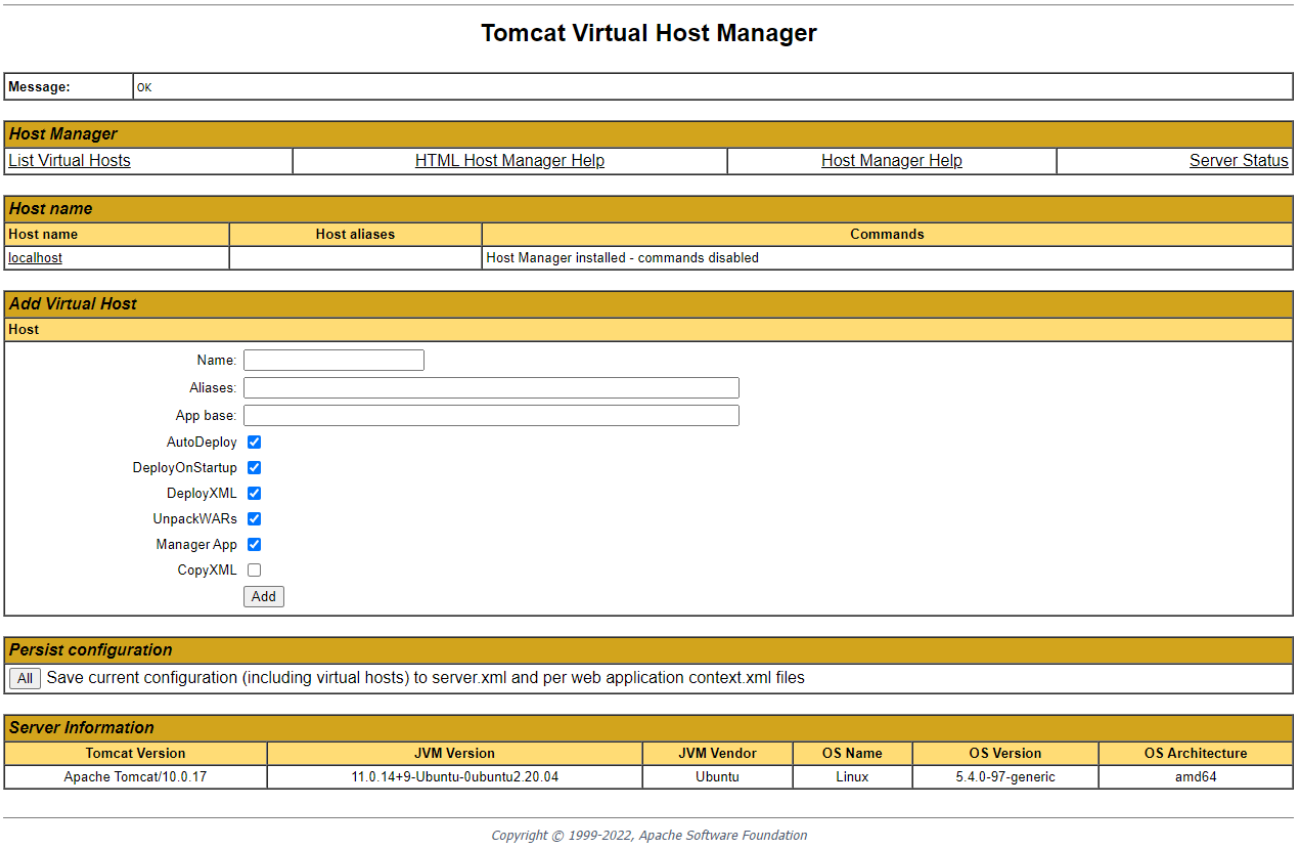


Рисунок 4.21 – Сторінка менеджера хостів

4.4 Висновки до четвертого розділу

В останньому розділі магістерської роботи було описано процес програмної реалізації веб-додатку інфокомунікаційної системи екзаменаційно-тренінгового центру іноземних мов на базі технологій Java EE. Було розроблено мінімально достатній програмний продукт, що дозволяє виконувати базові функції згідно технічного завдання. Додаток може бути використаний для тестування процесів функціонування відповідних систем з метою оптимізації бізнес-процесів освітньої діяльності. Подальше вдосконалення та ускладнення програмного забезпечення можливе в рамках життєвого цикну такого виду програмного забезпечення, однак в рамках даної роботи, виходить за її межі.

Крім того, в розділі наводиться опис процесу конфігурування серверних вузлів для розгортання програмного забезпечення на базі ОС Ubuntu та середовища віртуалізації Virtual Box.

ВИСНОВКИ

1. Кваліфікаційна робота магістра за спеціальність 172 – Електронні комунікації та радіотехніка виконана у повному обсязі у відповідності до поставлених завдань. Передумовами для роботи є ретельний аналіз предметної області, а саме освітній процесів, що виникають під час комунікації учасників та користувачів інфокомунікаційними системами освіти на прикладі екзаменаційно-тренінгового центу іноземних мов. В результаті огляду існуючих програмних систем даної тематики та аналізу літературних джерел було сформульовано технічне завдання та науково-технічні проблеми, що були вирішені в ході виконання.

2. В якості предмету дослідження в роботі було зафіксовано програмне забезпечення інфокомунікаційної системи екзаменаційно-тренінгового центру іноземних мов. В процесі реалізації останнього була вирішена актуальна науково-практична проблема – підвищено надійність критично важливий програмних компонентів системи, а саме модулів доступу до бази даних шляхом введення прийнятного рівня надлишковості та задіяння механізму резервного копіювання даних.

3. Розроблена модель інфокомунікаційних процесів з детальною декомпозицією структурних елементів, що згодом слугувала вихідним джерелом вимог для програмування компонентів системи.

4. Програмна реалізація виконана у вигляді веб-додатку засобами мови програмування високого рівня Java та набору специфікацій корпоративного програмного забезпечення Java EE. Реалізовано схему реляційної бази даних, алгоритми бізнес-логіки додатку та елементи інтерфейсу користувачів системи. В результаті було отримано повнофункціональну версію програмного забезпечення, яка може бути проваджена в освітній процес екзаменаційно-тренінгового центу іноземних мов для безпосереднього використання, а також для глибокого тестування, усунення можливих помилок та розширення функціоналу системи в рамках життєвого циклу програмного забезпечення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Довбиш А. С. Вступ до інформаційного аналізу і синтезу інфокомунікаційних систем : [монографія] / А. С. Довбиш. – Київ, 2016.

2. Осадча Ю. В. Аналіз програмного та інфокомунікаційного забезпечення освітнього процесу : [монографія] / Ю. В. Осадча. – Київ, 2022.

3. Чирвон А. О. Аналіз застосування протоколу IPSec в мережах інфокомунікацій : [монографія] / А. О. Чирвон. – Київ, 2024.

4. Moodle [Електронний ресурс]. – Режим доступу: https://moodle.org/. – Дата звернення: 26.10.2025.

5. Corradini E. Information and communication technologies in foreign language learning / E. Corradini, K. Borthwick, A. Dickens // Centre for Languages, Linguistics and Area Studies, University of Southampton. – Southampton, 2010. – [Електронний ресурс]. – Режим доступу: https://eprints.soton.ac.uk/192773/. – Дата звернення: 26.10.2025.

6. The Role of Technology in Language Learning [Електронний ресурс] / American Council on the Teaching of Foreign Languages. – Режим доступу: https://www.actfl.org/news/the-role-of-technology-in-language-learning. – Дата звернення: 26.10.2025.

7. Warschauer M. Computers and language learning: An overview / M. Warschauer, D. Healey // Language Teaching. – 1998. – Vol. 31. – P. 57–71.

8. Seiz Ortiz R. Classification of Technology-based Language Learning Resources according to Pedagogical and Functional Criteria / R. Seiz Ortiz, M. L. Carrió Pastor // Revista Electrónica de Lingüística Aplicada. – 2025. – Vol. 23, iss. 1. – [Електронний ресурс]. – Режим доступу: https://doi.org/10.58859/rael.v23i1.676. – Дата звернення: 26.10.2025.

9. Hardware and Software You'll Need for eLearning [Електронний ресурс] / eLearners. – Режим доступу: https://www.elearners.com/education-resources/online-learning/hardware-and-software-youll-need-for-elearning/. – Дата звернення: 26.10.2025.

10. Best Education Software: Key Features of Top Education Software [Електронний ресурс] / Research.com. – Режим доступу: https://research.com/software/best-education-software-key-features-of-top-education-software. – Дата звернення: 26.10.2025.

11. Hardware, Software, and Applications Choices for HyFlex Teaching and Learning [Електронний ресурс] / EdTech Books. – Режим доступу: https://edtechbooks.org/hyflex_guide/ch6_hardware_software_applications_choices. – Дата звернення: 26.10.2025.

12. Technology Tools for Language Learning [Електронний ресурс] / Language Technology Lab, University of Iowa. – Режим доступу: https://languagetech.lab.uiowa.edu/technology-tools-language-learning. – Дата звернення: 26.10.2025.

13. Two Tools for Communication, Assessment, and More [Електронний ресурс] / edWeb. – Режим доступу: https://home.edweb.net/two-tools-for-communication-assessment-and-more/. – Дата звернення: 26.10.2025.

14. AI in Language Learning and Teaching [Електронний ресурс] / American Public University System. – Режим доступу: https://www.apu.apus.edu/area-of-study/arts-and-humanities/resources/ai-in-language-learning-and-teaching/. – Дата звернення: 26.10.2025.

15. CEFR: The Common European Framework of Reference for Languages [Електронний ресурс] / Cambridge English. – Режим доступу: https://www.cambridgeenglish.org/exams-and-tests/cefr/. – Дата звернення: 26.10.2025.

16. Lingoda: Online Language Courses [Електронний ресурс]. – Режим доступу: https://www.lingoda.com/en/. – Дата звернення: 26.10.2025.

17. Preply: Learn Languages Online [Електронний ресурс]. – Режим доступу: https://preply.com/. – Дата звернення: 26.10.2025.

18. italki: Language Learning Community [Електронний ресурс]. – Режим доступу: https://www.italki.com/. – Дата звернення: 26.10.2025.

19. Duolingo: Learn a Language for Free [Електронний ресурс]. – Режим доступу: https://www.duolingo.com/. – Дата звернення: 26.10.2025.

20. Babbel: Language Learning [Електронний ресурс]. – Режим доступу: https://www.babbel.com/. – Дата звернення: 26.10.2025.

21. Goethe-Zertifikat [Електронний ресурс]. – Режим доступу: https://www.goethe.de/en/spr/prf.html. – Дата звернення: 26.10.2025.

22. DELF/DALF [Електронний ресурс]. – Режим доступу: https://ifsuede.com/franska-spraket/sprakprov-i-franska/delf-dalf/?lang=sv. – Дата звернення: 26.10.2025.

23. Ross D. T. Structured Analysis and Design Technique (SADT) / D. T. Ross // SofTech, Inc. – 1973. – [Електронний ресурс]. – Режим доступу: https://segoldmine.ppi-int.com/taxonomy/term/844. – Дата звернення: 26.10.2025.

24. Black Box [Електронний ресурс] / Investopedia. – Режим доступу: https://www.investopedia.com/terms/b/blackbox.asp. – Дата звернення: 26.10.2025.

25. Richardson C. Microservice Architecture pattern [Електронний ресурс] / C. Richardson // Microservices.io. – Режим доступу: https://microservices.io/patterns/microservices.html. – Дата звернення: 26.10.2025.

26. Що таке ЄКТС і як вона працює в Україні? [Електронний ресурс] // Освіта.UA. – [Електронний ресурс]. – Режим доступу: https://osvita.ua/vnz/high_school/70499/. – Дата звернення: 26.10.2025.

27. PostgreSQL [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/. – Дата звернення: 26.10.2025.

28. Benefits of PostgreSQL [Електронний ресурс] / Prisma. – Режим доступу: https://www.prisma.io/dataguide/postgresql/benefits-of-postgresql. – Дата звернення: 26.10.2025.

29. Pros and Cons of Using PostgreSQL for Application Development [Електронний ресурс] / Aalpha. – Режим доступу: https://www.aalpha.net/blog/pros-and-cons-of-using-postgresql-for-application-development/. – Дата звернення: 26.10.2025.

30. PostgreSQL vs SQL [Електронний ресурс] / Google Cloud. – Режим доступу: https://cloud.google.com/learn/postgresql-vs-sql. – Дата звернення: 26.10.2025.

31. PostgreSQL vs MySQL: 360-Degree Comparison [Електронний ресурс] / EnterpriseDB. – Режим доступу: https://www.enterprisedb.com/blog/postgresql-vs-mysql-360-degree-comparison-syntax-performance-scalability-and-features. – Дата звернення: 26.10.2025.

32. Java [Електронний ресурс]. – Режим доступу: https://www.java.com/en/. – Дата звернення: 26.10.2025.

33. Oracle Java Documentation [Електронний ресурс] / Oracle. – Режим доступу: https://docs.oracle.com/en/java/. – Дата звернення: 26.10.2025.

34. Java [Електронний ресурс] / GeeksforGeeks. – Режим доступу: https://www.geeksforgeeks.org/java/. – Дата звернення: 26.10.2025.

35. Java Tutorials [Електронний ресурс] / Baeldung. – Режим доступу: https://www.baeldung.com/java-tutorials. – Дата звернення: 26.10.2025.

36. Java SE Release Notes [Електронний ресурс] / Oracle. – Режим доступу: https://www.oracle.com/java/technologies/javase/jdk-relnotes.html. – Дата звернення: 26.10.2025.

37. Jakarta EE [Електронний ресурс]. – Режим доступу: https://jakarta.ee/. – Дата звернення: 26.10.2025.

38. Servlets [Електронний ресурс] / Oracle. – Режим доступу: https://docs.oracle.com/javaee/7/tutorial/servlets.htm. – Дата звернення: 26.10.2025.

39. java.sql Module Summary [Електронний ресурс] / Oracle. – Режим доступу: https://docs.oracle.com/en/java/javase/17/docs/api/java.sql/module-summary.html. – Дата звернення: 26.10.2025.

40. HikariCP [Електронний ресурс] / Baeldung. – Режим доступу: https://www.baeldung.com/hikaricp. – Дата звернення: 26.10.2025.

41. Hikari Connection Pooling [Електронний ресурс] / Javarevisited // Medium. – Режим доступу: https://medium.com/javarevisited/hikari-connection-pooling-5600d765e5ae. – Дата звернення: 26.10.2025.

42. CompletableFuture [Електронний ресурс] / Oracle. – Режим доступу: https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/concurrent/CompletableFuture.html. – Дата звернення: 26.10.2025.

43. Java CompletableFuture [Електронний ресурс] / Javatpoint. – Режим доступу: https://www.javatpoint.com/java-completablefuture. – Дата звернення: 26.10.2025.

ДОДАТОК А

Декомпозиція моделі системи

Таблиця А.1 – Потоки даних активностей Студент/Користувач

Активність (Activity)	Входи (Inputs)	Виходи (Outputs)	Контроль (Controls)	Механізни (Mechanisms)
1	*2*	*3*	*4*	*5*
Зарахування на предмет	Вибір предмета	Підтвердження зарахування	Параметри зарахування	MVC-контролер для обробки запитів
	Перегляд інформація користувача	Оновлений список предметів	Перевірка ролі користувача	Підключення до БД
Скасування реєстрації на предмет	Ідентифікат. предм. для скасування	Підтвердження відписки	Правила відписки	Пул обробників. оновлен. БД
	Ідентифікат. користувача	Оновлений список предметів		Транзакції БД
Перегляд моїх предметів	Ідентифікат. користувача	Список зареєстрованих предметів	Керування конфіденц.	Модель для пошуку даних
	Додаткові фільтри			Запит до БД

Кінець Таблиці А.1

1	*2*	*3*	*4*	*5*
Перегляд усіх предметів	Фільтри пошуку	Список доступних предметів	Правила видимості	Рендер перегляду
				Кешовані результати БД

Таблиця А.2 – Потоки даних активностей Екзаменований/Користувач

Активність (Activity)	Входи (Inputs)	Виходи (Outputs)	Контроль (Controls)	Механізни (Mechanisms)
1	*2*	*3*	*4*	*5*
Реєстрація на іспит	ID іспиту	Підтвердження реєстрації	Параметри реєстрації	Асинхронний обробник сповіщення
	Дані учасника	Оновлений список учасників	Перевірки відповідності вимогам	Пул БД для вставки даних
	Вибрана дата/час			
Перенесення іспиту	Існуючий ID іспиту	Підтвердження перенесення	Обмеження перенесення	Контролер для перевірки
	Новий запит на дату/час	Оновлений розклад		Транзакція БД для оновлення
Скасування іспиту	ID іспиту для скасування	Підтвердження скасування	Політики скасування	Виконавець для фонового очищення
		Звільнений слот		Запит для видалення

Кінець Таблиці А.2

1	*2*	*3*	*4*	*5*
Перегляд моїх іспитів	ID учасника	Список зареєстрованих іспитів зі статусом	Елементи керування доступом	Запит до БД
	Додаткові фільтри		Параметри діапазону дат	Візуалізація відповіді додатку
Перегляд усіх іспитів	Додаткові фільтри	Список доступних іспитів з детальною інформацією	Правила публічної видимості	Представлення перегляду
			Параметри пошуку	Кешовані результати.

Таблиця Д.1.3 – Потоки даних активностей Адміністрування

Активність (Activity)	Входи (Inputs)	Виходи (Outputs)	Контроль (Controls)	Механізни (Mechanisms)
1	*2*	*3*	*4*	*5*
Додати нового користувача	Відомості про нового користувача	Запис нового користувача	Перевірка прав адмін.	Безпечна вставка даних
	Дані учасника	Сповіщення	Параметри унікальності	Обробник асинхронної операції
Редагувати профіль користувача	Ідентифікатор користувача	Оновлений профіль користувача	Правила редагування	Транзакція оновлення БД
	Оновлені дані	Журнал змін	Параметри перевірки	Форма представлення відображ.

Кінець Таблиці А.3

1	*2*	*3*	*4*	*5*
Додати новий предмет	Деталі предмету	Новий запис предмету	Перевірка унікальності	Панель адміністратора
		Оновлений каталог	Робочі процеси затвердження	Пул підключення до БД
Редагування деталей предмета	ID предмету	Оновлений предмет	Оцінка впливу	Асинхронний обробник сповіщення
	Перегл. деталей	Сповіщення зареєстров. користувач.		Оновлення БД
Створити новий іспит	Деталі іспиту	Новий розклад іспиту	Параметри планування	Інтеграція календаря
		Публічний список		
Редагування запланованого іспиту	ID іспиту	Оновлений розклад	Обмеження зміни розкладу	Асинхронний обробник сповіщення
	Оновлені деталі	Сповіщення зареєстр. користувачів	Механізми погодження зміни розкладу із учасниками процесу	Транзакція та запити до БД

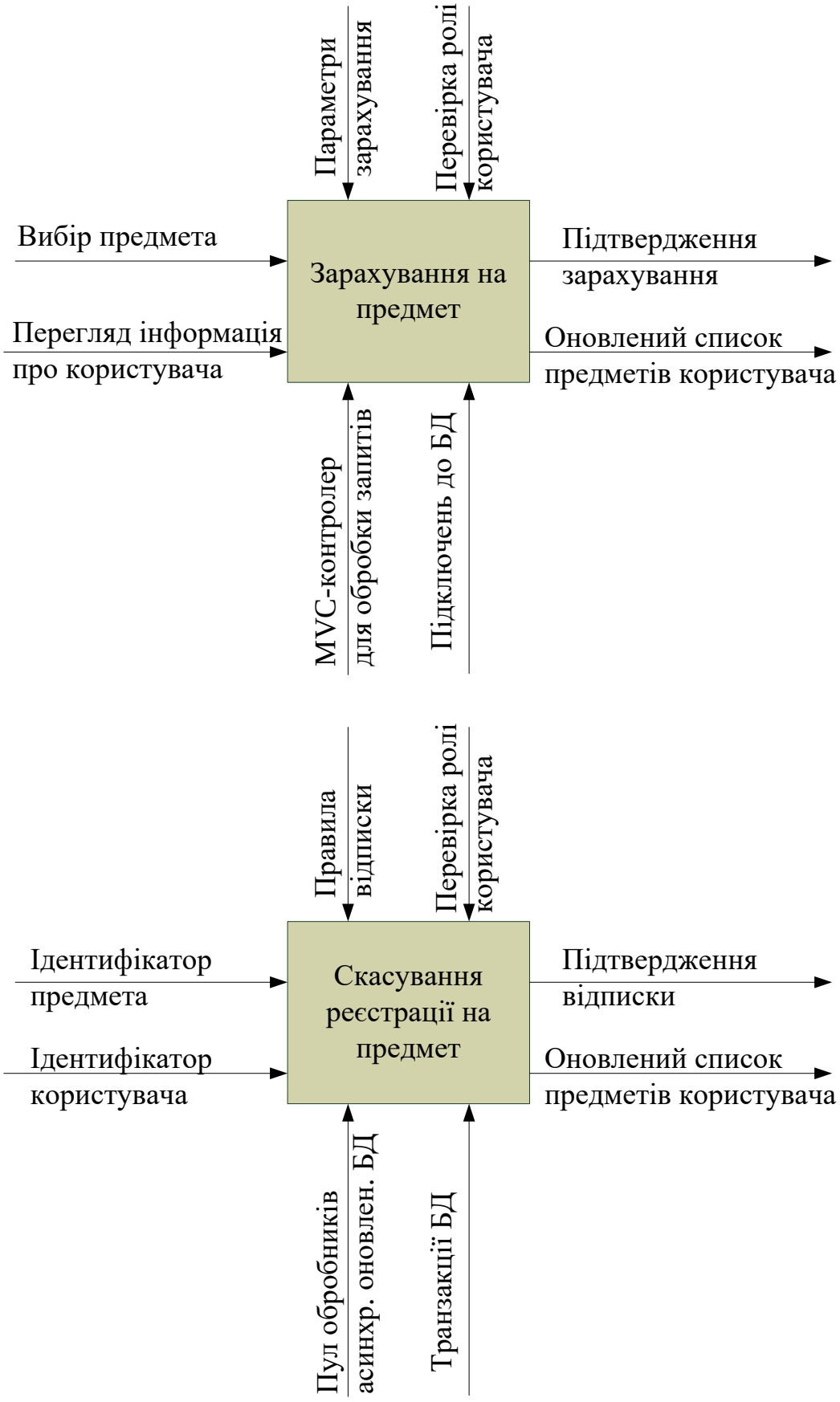


Рисунок А.1 – Активності «Зарахування на предмет» та «Скасування реєстрації»

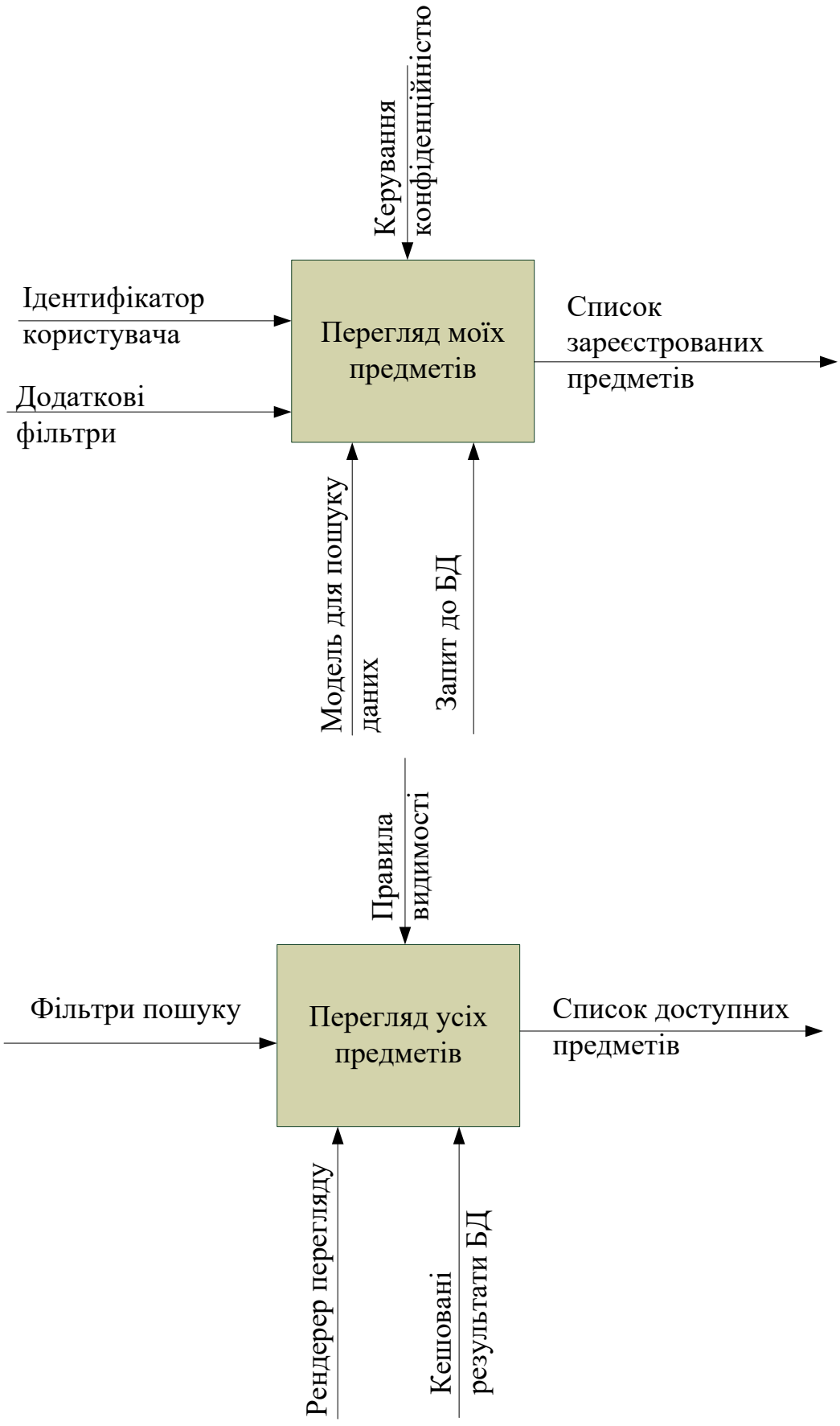


Рисунок А.2 – Активності «Перегляд мої / усіх предметів»

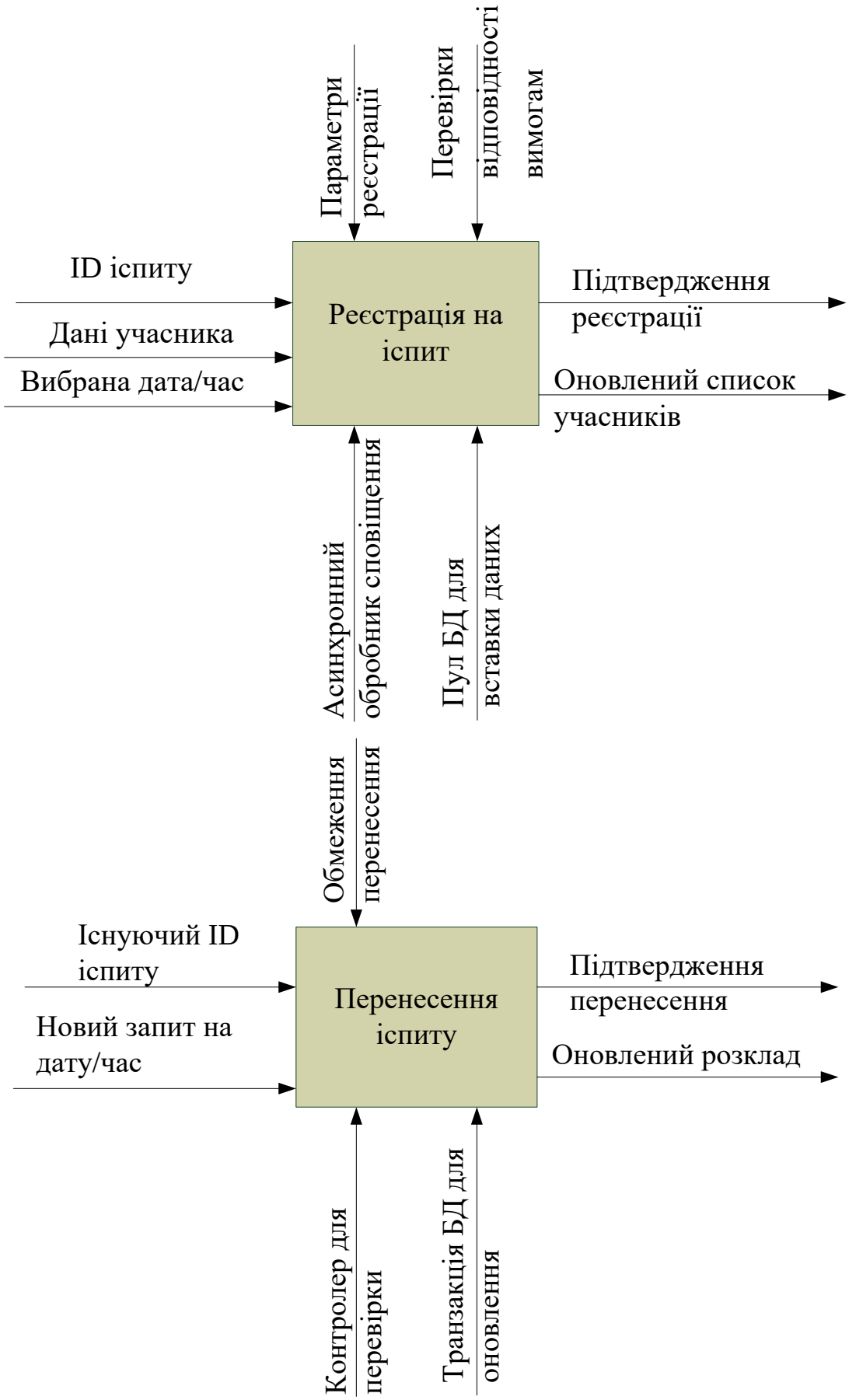


Рисунок А.3 – Активності «Реєстрація на іспит» та «Перенесення іспиту»

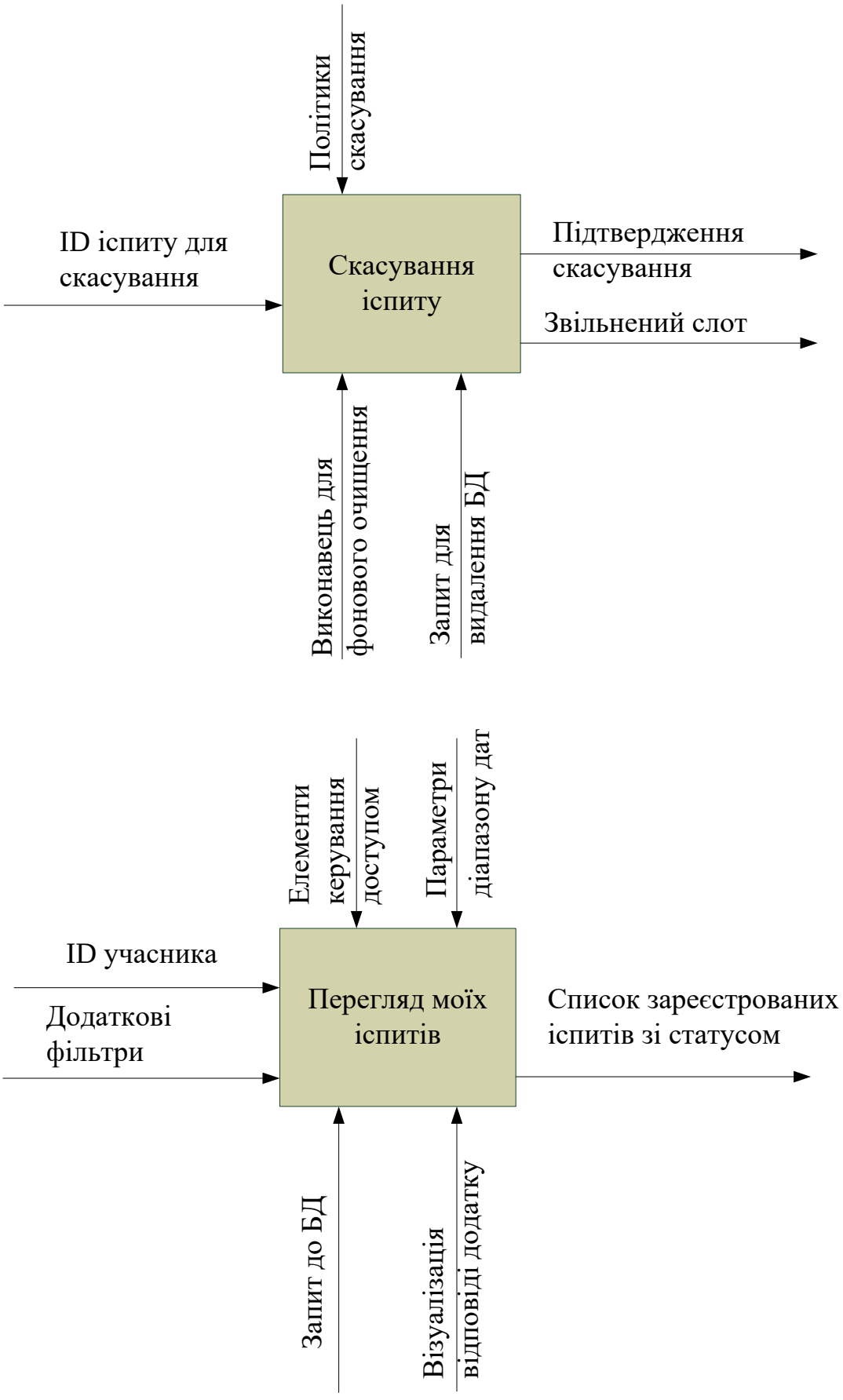


Рисунок А.4 – Активності «Скасування іспиту» та «Перегляд моїх іспитів»

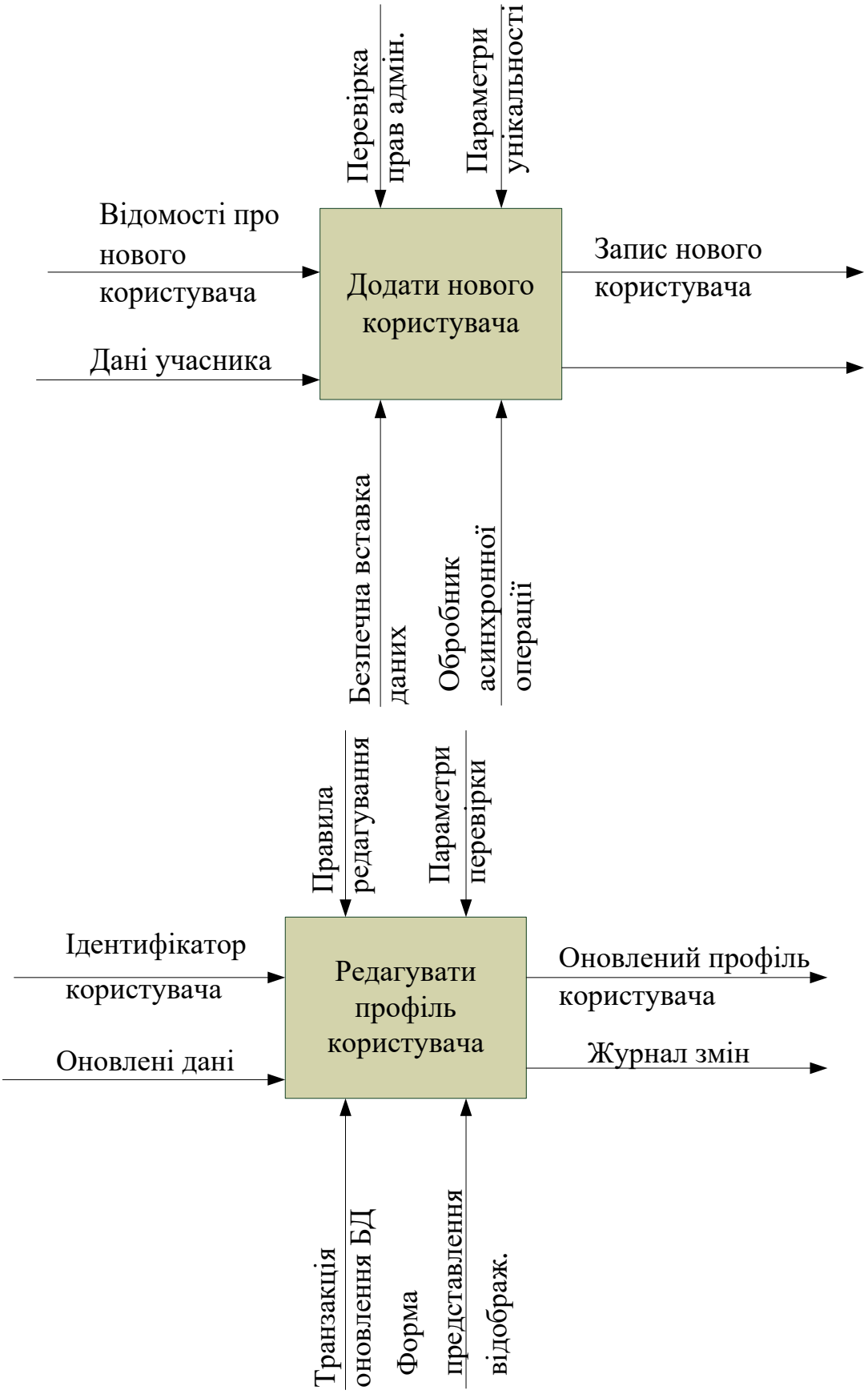


Рисунок А.5 – Активності «Додати нового користувача» та «Редагувати профіль користувача»

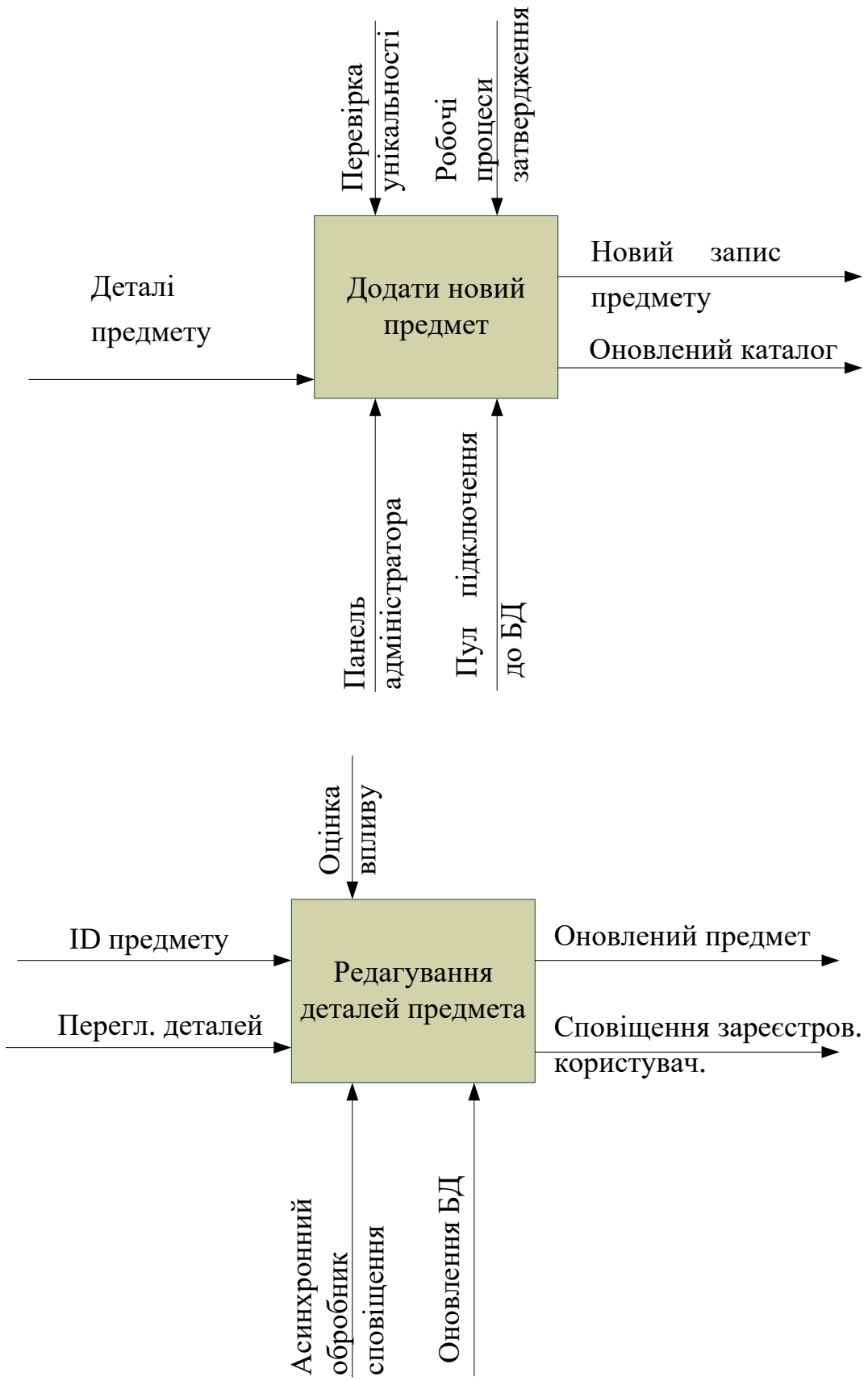


Рисунок А.6 – Активності «Додати нового предмет» та «Редагувати деталі предмету»

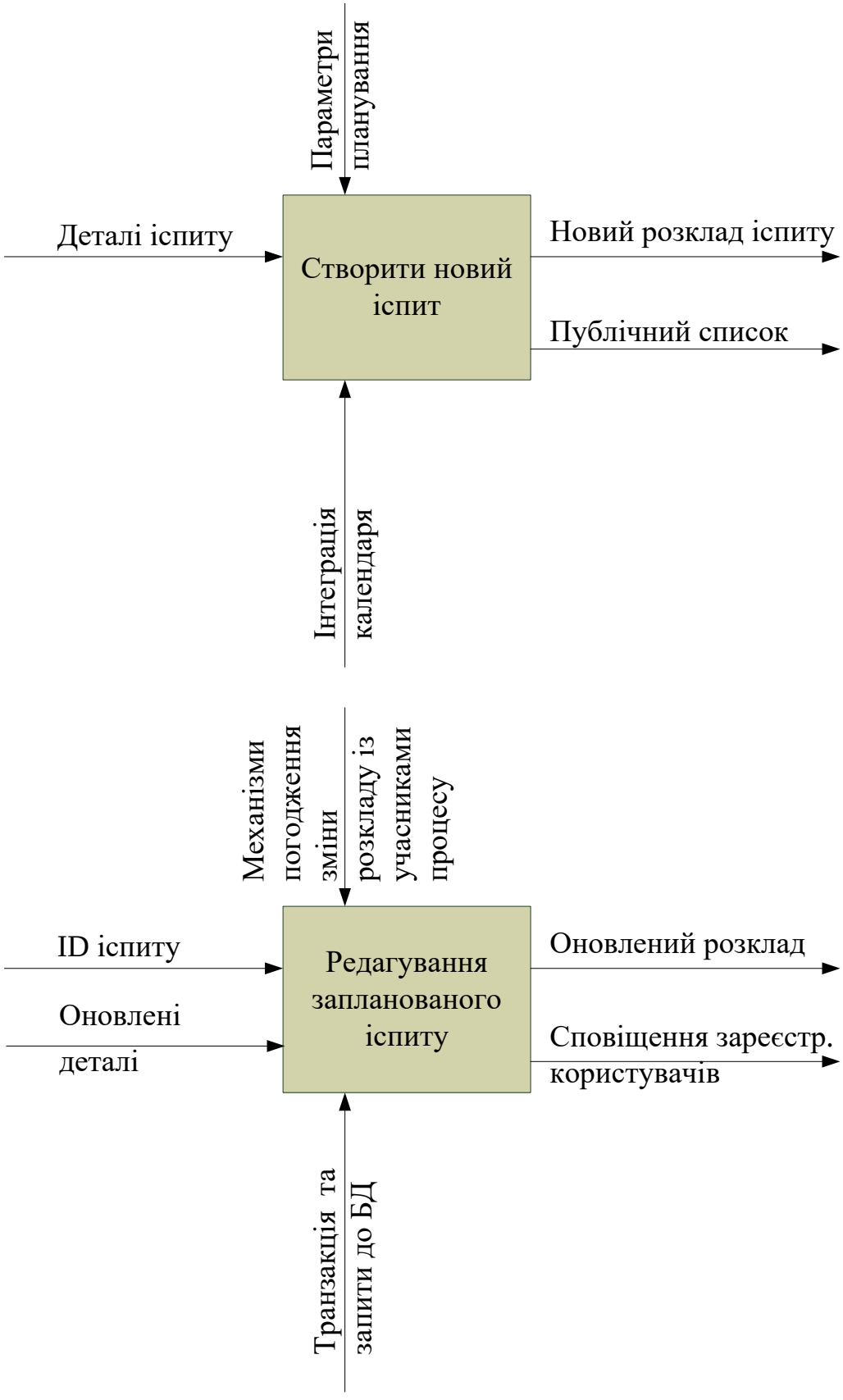
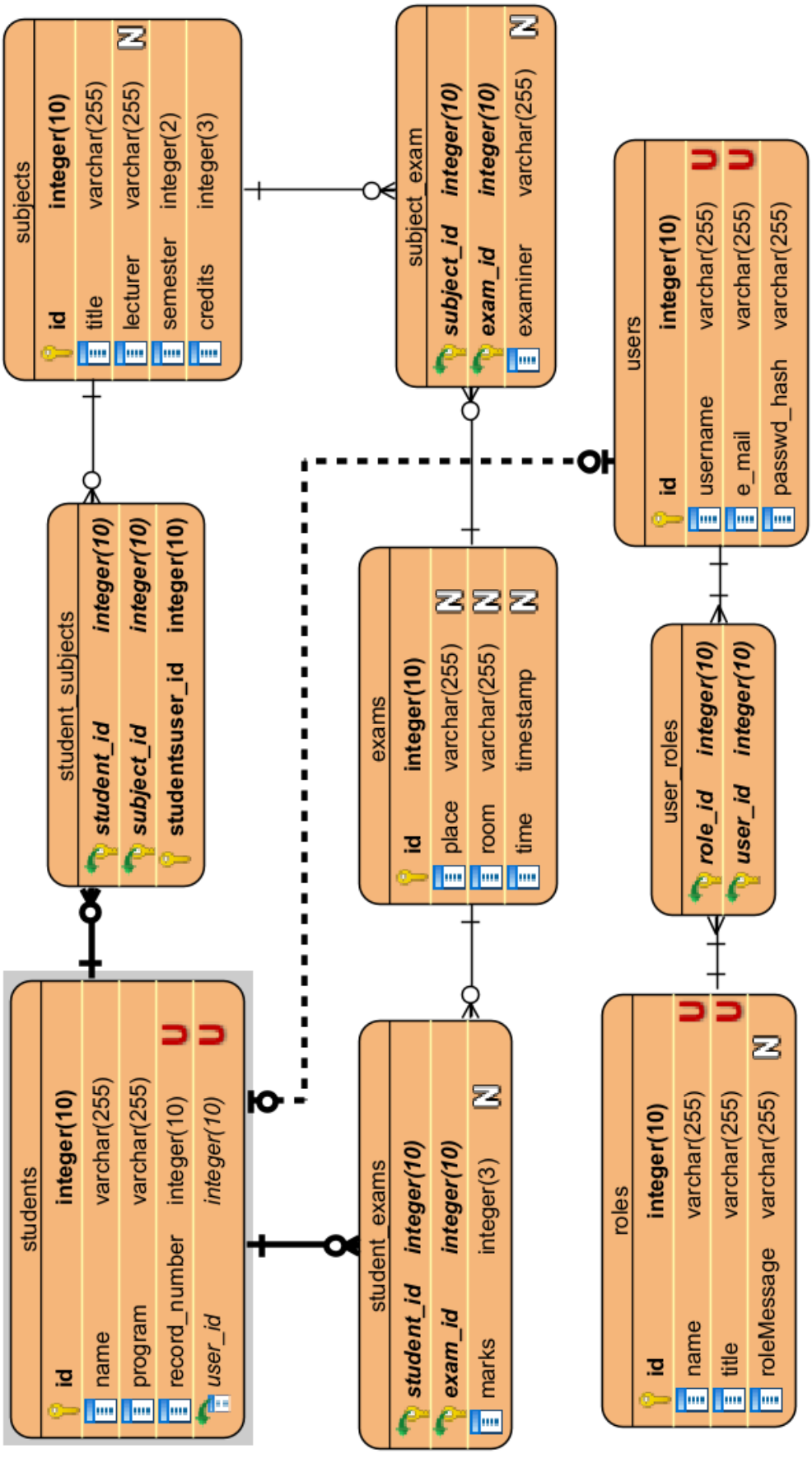


Рисунок В.7 – Активності «Створити новий іспит» та «Редагування запланованого іспиту»

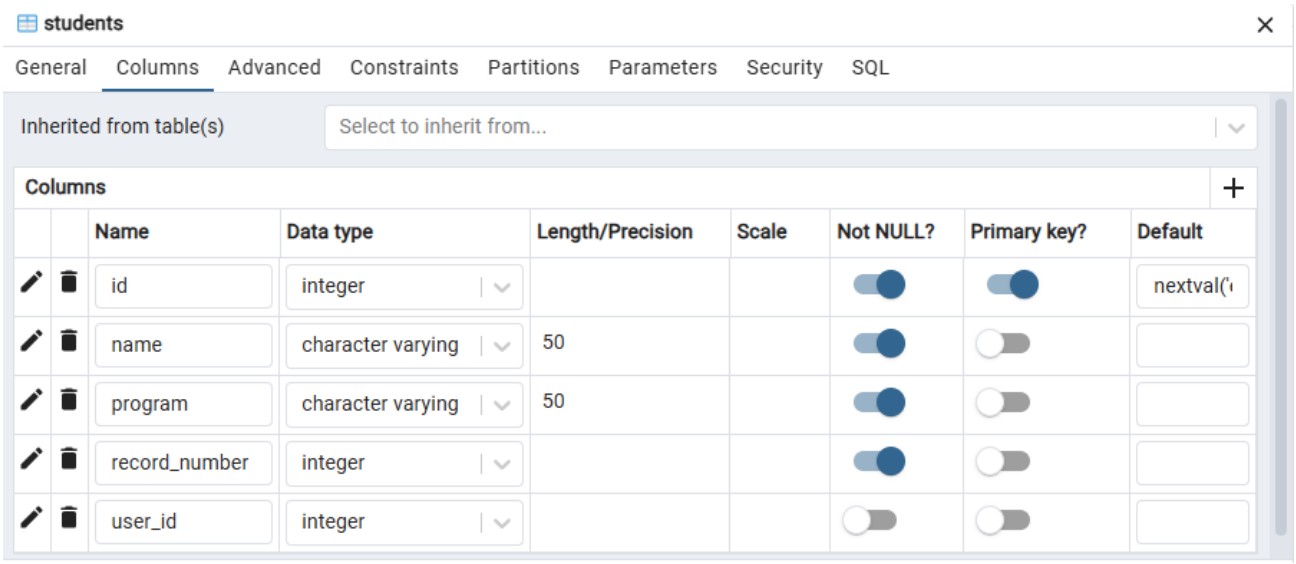
ДОДАТОК Б

Схема бази даних



ДОДАТОК В

Створення таблиць та заповнення їх даними в PostgreSQL



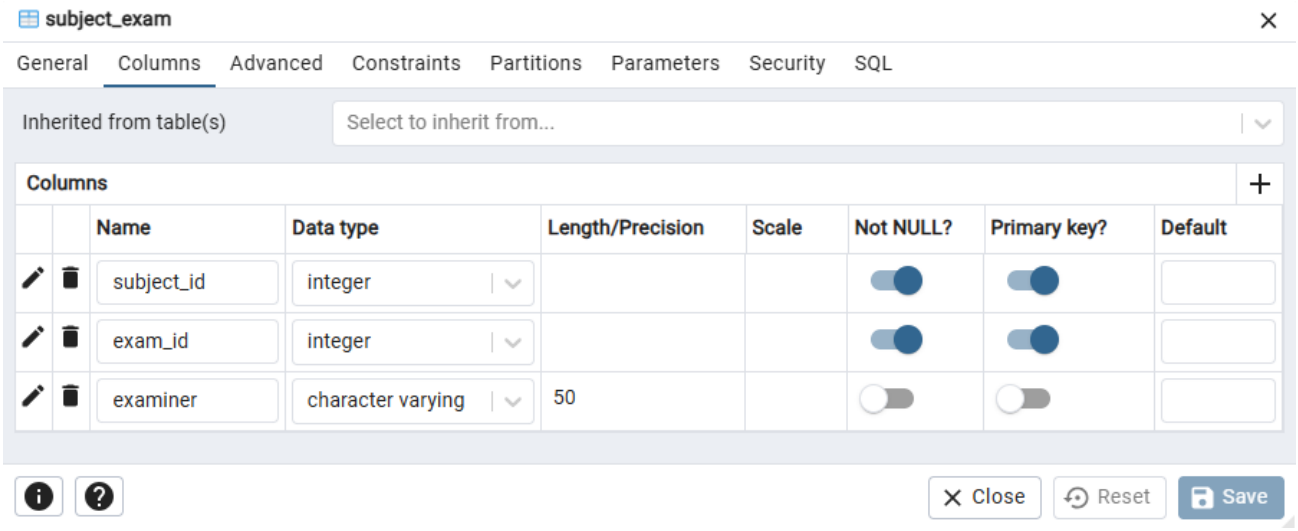
а)

	id [PK] integer	name character varying (50)	program character varying (50)	record_number integer	user_id integer
1	1	Joseph, Miller	Telecommunication and Radio Engineering	25025	2
2	2	Michael, Garcia	Computer Science	25026	[null]
3	3	Daniel, Hernandez	Computer Science	25027	3
4	4	Ava, Martinez	Applied Mathematics	25028	[null]
5	5	Mia, Doe	Telecommunication and Radio Engineering	25055	[null]
6	6	Sophia, Wilson	Telecommunication and Radio Engineering	25056	[null]
7	7	Joseph, Brown	Applied Mathematics	25057	[null]
8	8	Elizabeth, Moore	Applied Mathematics	25058	[null]
9	9	Michael, Wilson	Computer Science	25059	[null]
10	10	Mia, Davis	Computer Science	25060	[null]
11	11	Elijah, King	Applied Mathematics	24034	[null]
12	12	Amelia, Young	Applied Mathematics	24035	[null]
13	13	Benjamin, Allen	Applied Mathematics	24036	[null]
14	14	Charlotte, Hall	Applied Mathematics	24037	[null]
15	15	William, Walker	Computer Science	24038	[null]
16	16	Isabella, Lewis	Computer Science	24039	[null]
17	17	James, Clark	Computer Science	24012	[null]
18	18	Olivia, Harris	Telecommunication and Radio Engineering	24013	[null]

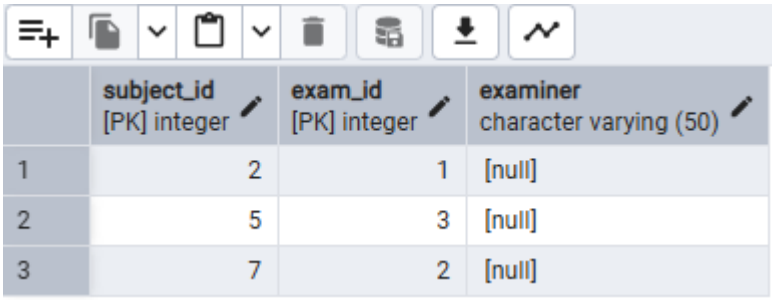
б)

Рисунок В.1 – Створення таблиці «students» (а)

та її заповнення тестовими даними (б)



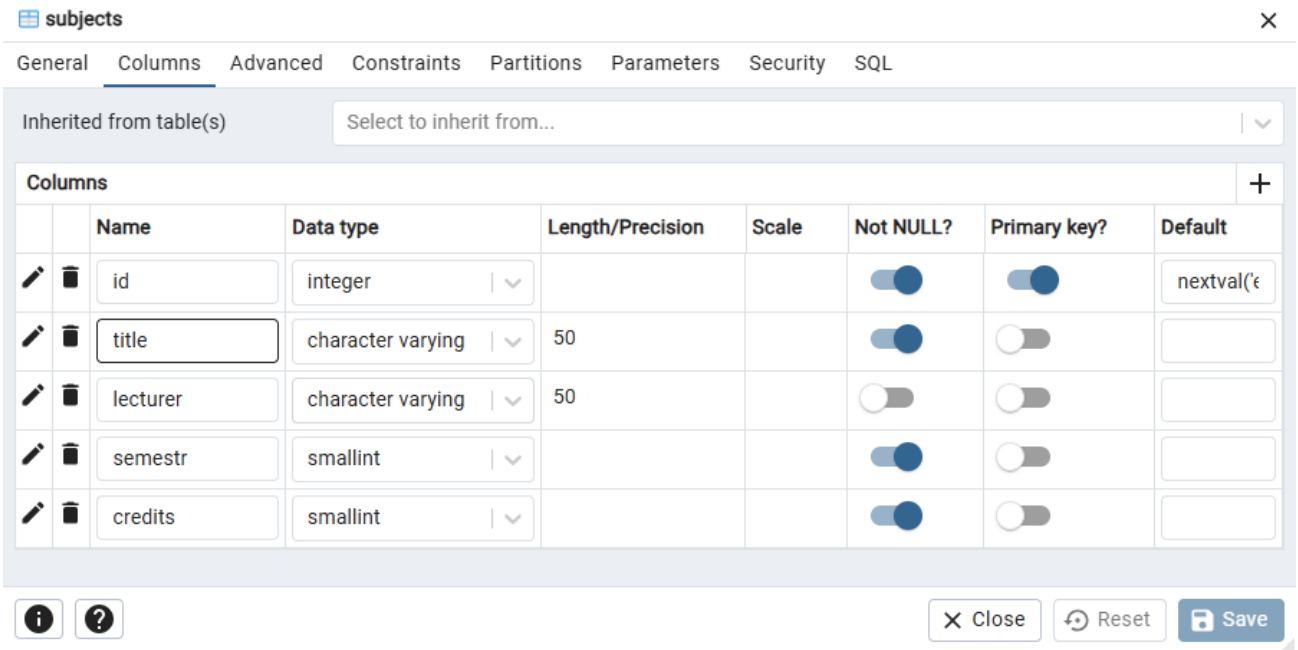
а)



	subject_id [PK] integer	exam_id [PK] integer	examiner character varying (50)
1	2	1	[null]
2	5	3	[null]
3	7	2	[null]

б)

Рисунок В.2 – Створення таблиці «subject_exam» (а)
та її заповнення тестовими даними (б)

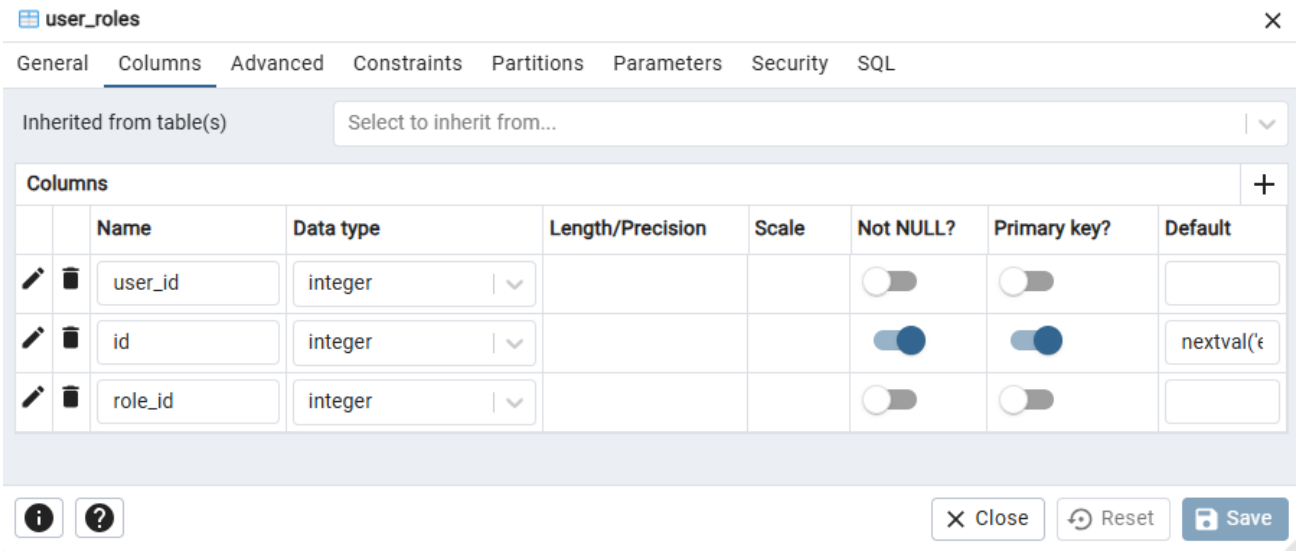


а)

	id [PK] integer	title character varying (50)	lecturer character varying (50)	semestr smallint	credits smallint
1	2	General English B2	Prof. M. L. Thompson, P...	1	40
2	3	English for Academy	Assoc. Prof. E. J. Kim, PhD	1	40
3	4	Basic English A1-A2	Assoc. Prof. E. J. Kim, PhD	1	40
4	5	German B2	Prof. M. L. Thompson, P...	1	40
5	6	Advanced English C1	Prof. R. S. Patel, PhD	1	40
6	7	Business English B2-C1	Prof. R. S. Patel, PhD	1	40

б)

Рисунок В.3 – Створення таблиці «subjects» (а)
та її заповнення тестовими даними (б)

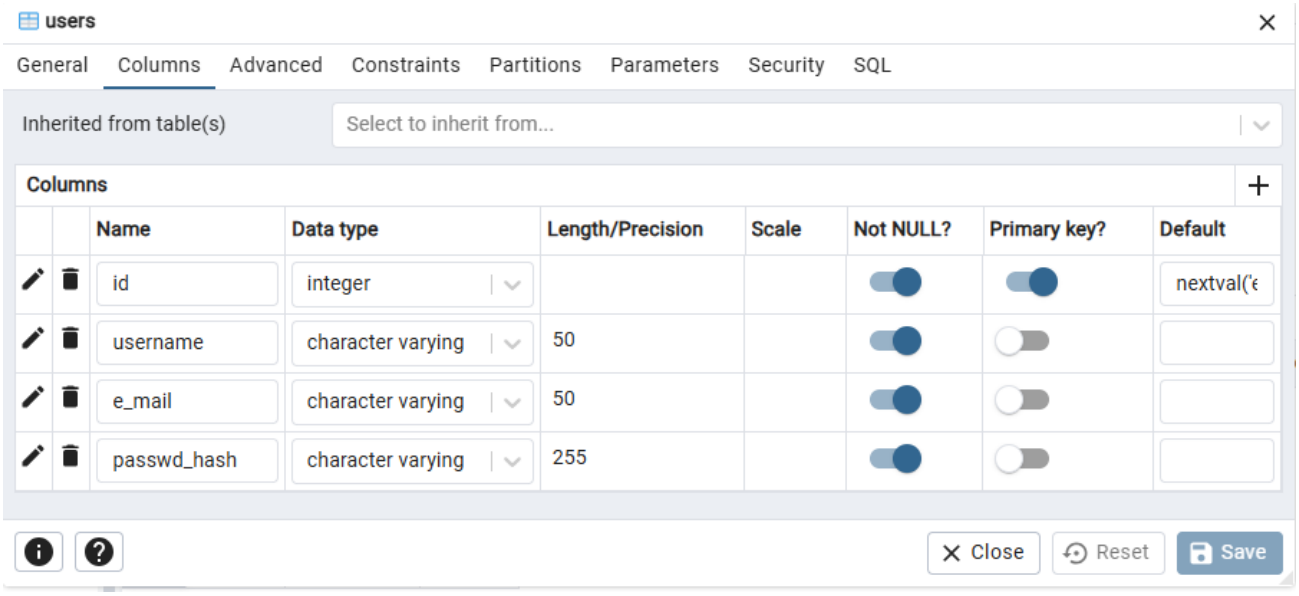


а)

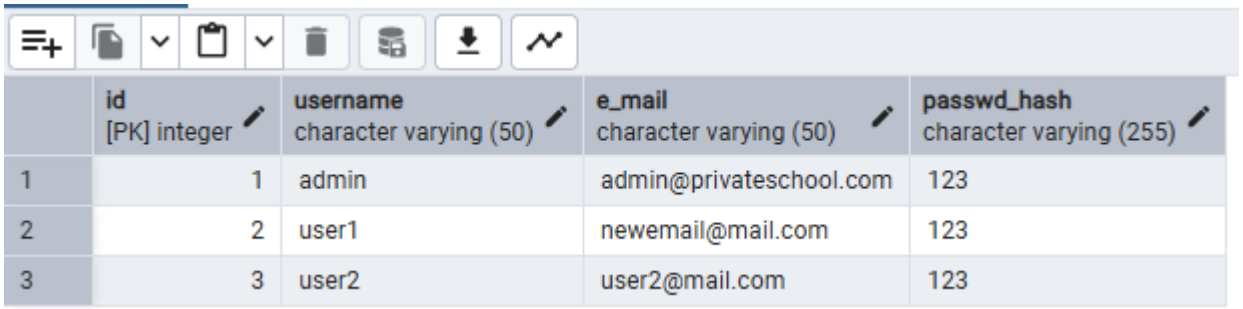
	user_id integer	id [PK] integer	role_id integer
1	2	1	2
2	3	2	3
3	1	3	1
4	3	4	2

б)

Рисунок В.4 – Створення таблиці «user_roles» (а)
та її заповнення тестовими даними (б)



а)



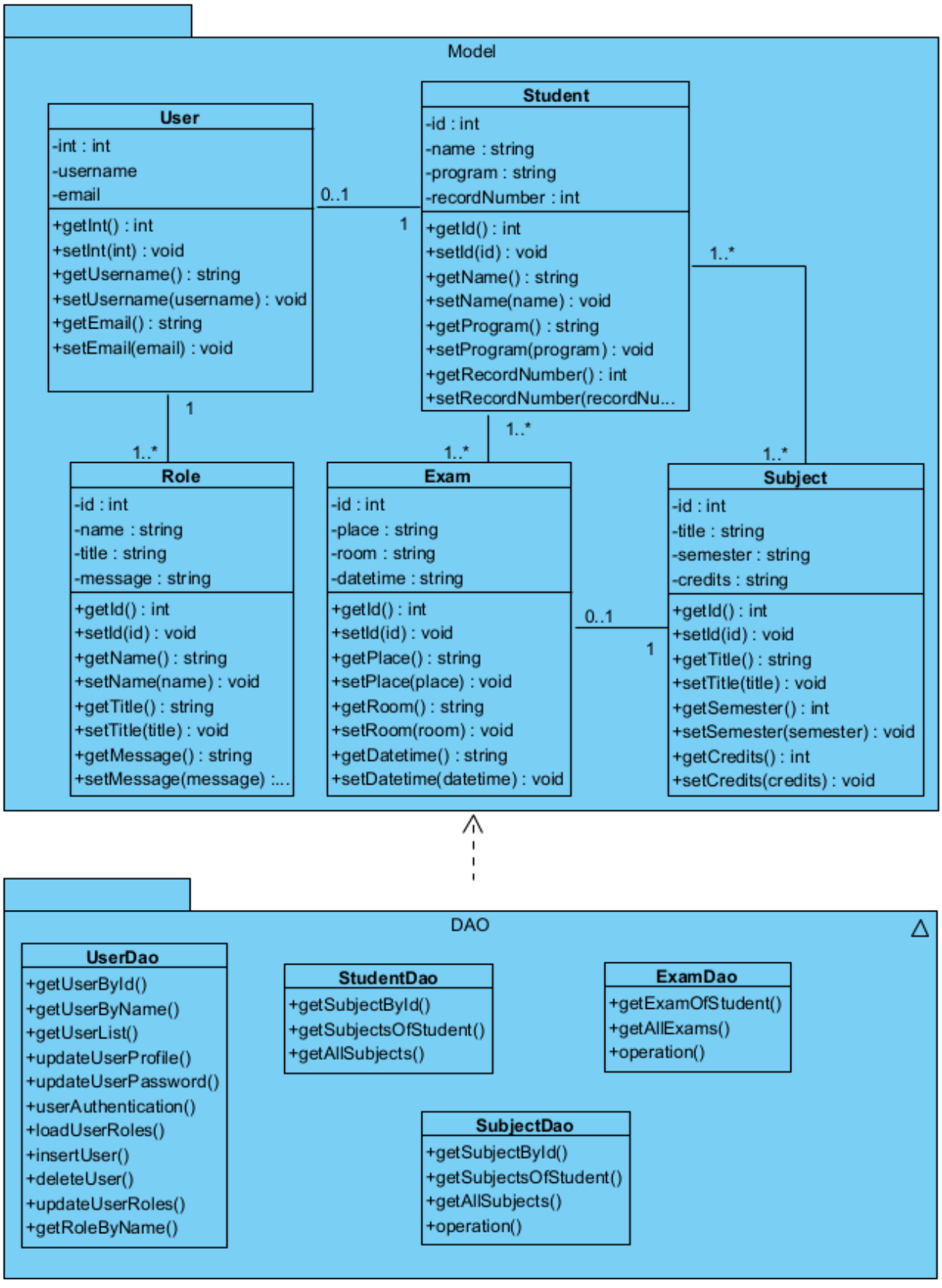
	id [PK] integer	username character varying (50)	e_mail character varying (50)	passwd_hash character varying (255)
1	1	admin	admin@privateschool.com	123
2	2	user1	newemail@mail.com	123
3	3	user2	user2@mail.com	123

б)

Рисунок В.5 – Створення таблиці «users» (а)
та її заповнення тестовими даними (б)

ДОДАТОК Г

Об’єктно-орієнтована архітектура програмного забезпечення



ДОДАТОК Д

Код програмних модулів

Лістинг Д.1 – Код класу модулі User

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

import org.max.education.privateshcool.models.Student;
import org.max.education.privateshcool.models.Subject;

public class SubjectDao {

     public static Subject getSubjectById(int id) {
          try(Connection conn = DbConnection.getConnection();
               PreparedStatement ps = conn.prepareStatement(
                         "select title, lecturer, semestr, credits "
+
                         "from exams.subjects where id = ?"
          )) {
               ps.setInt(1, id);
               ResultSet rs = ps.executeQuery();
               if (rs.next()) {
                    String title = rs.getString(1);
                    String lecturer = rs.getString(2);
                    int semester = rs.getInt(3);
                    int credits = rs.getInt(4);
                    return new Subject(id, title, lecturer,
semester, credits);
               }
          } catch (SQLException e) {
               // TODO Auto-generated catch block
               System.err.println("Get Subject By Id Error!");
               e.printStackTrace();
          }
          return null;
     }

     public static Subject[] getSubjectsOfStudent(Student student)
{
          try (Connection conn = DbConnection.getConnection();
                PreparedStatement ps = conn.prepareStatement(
                    "select s.id, s.title, s.lecturer, s.semestr,
s.credits " +
```

```
                    "from exams.subjects as s,
exams.student_subjects as ss, exams.subjects as st " +
                    "where st.id = ss.student_id and ss.subject_id =
s.id and st.id = ?"
          )) {
               ps.setInt(1, student.getId());
               ResultSet rs = ps.executeQuery();
               ArrayList<Subject> subjectList = new
ArrayList<Subject>();
               while (rs.next()) {
                    int id = rs.getInt(1);
                    String title = rs.getString(2);
                    String lecturer = rs.getString(3);
                    int semestr = rs.getInt(4);
                    int credits = rs.getInt(5);
                    subjectList.add(new Subject(id, title, lecturer,
semestr, credits));
               }
               return subjectList.toArray(new Subject[0]);

          } catch (SQLException e) {
               System.err.println("Get Subject List Of Student
Error!");
               e.printStackTrace();
          }
          return new Subject[0];
     }

     public static Subject[] getAllSubjects() {
          try (Connection conn = DbConnection.getConnection();
                Statement statement = conn.createStatement()) {
               ResultSet rs = statement.executeQuery(
                     "select id, title, lecturer, semestr, credits "
+
                          "from exams.subjects order by title"
               );
               ArrayList<Subject> subjects = new
ArrayList<Subject>();
               while (rs.next()) {
                    int id = rs.getInt(1);
                    String title = rs.getString(2);
                    String lecturer = rs.getString(3);
                    int semestr = rs.getInt(4);
                    int credits = rs.getInt(5);

                    subjects.add(new Subject(id, title, lecturer,
semestr, credits));
               }
               return subjects.toArray(new Subject[0]);

          } catch (SQLException e) {
               // TODO Auto-generated catch block
```

```
                e.printStackTrace();
            }
            return new Subject[0];
        }

        public static void main(String[] args) {
//          System.out.println(getSubjectById(2));
//          Student student = StudentDao.getStudentById(2);
//          Subject[] subjects = getSubjectsOfStudent(student);
            Subject[] subjects = getAllSubjects();
            for (Subject s : subjects) {
                System.out.println(s);
            }
        }
}
```

Лістинг Д.2 – Код класу модулі Role

```
public class Role {
    private int id;
    private String name;
    private String title;
    private String message;

    public Role(int id, String name, String title, String message)
{
        this.id = id;
        this.name = name;
        this.title = title;
        this.message = message;
    }
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getTitle() {
```

```
          return title;
     }

     public void setTitle(String title) {
          this.title = title;
     }

     public String getMessage() {
          return message;
     }

     public void setMessage(String message) {
          this.message = message;
     }
     @Override
     public String toString() {
          return "Role [id=" + id + ", name=" + name + ", title=" +
title + ", message=" + message + "]";
     }


}
```

Лістинг Д.3 – Код класу модулі Student

```
import java.util.ArrayList; import java.util.Objects;
public class Student { private int id; private String name; private
String program; private int recordNumber; private User
referencedUser; private ArrayList examList = new ArrayList();
public Student(int id, String name, String program, int
recordNumber) { this.id = id; this.name = name; this.program =
program; this.recordNumber = recordNumber; }
public Student(int id, String name, String program, int
recordNumber, User referencedUser) { this.id = id; this.name =
name; this.program = program; this.recordNumber = recordNumber;
this.referencedUser = referencedUser; }
public Student(String name, String program, int recordNumber) {
this.name = name; this.program = program; this.recordNumber =
recordNumber; }
public int getId() { return id; }
public void setId(int id) { this.id = id; }
public String getName() { return name; }
public void setName(String name) { this.name = name; }
public String getProgram() { return program; }
public void setProgram(String program) { this.program = program; }
public int getRecordNumber() { return recordNumber; }
public void setRecordNumber(int recordNumber) { this.recordNumber =
recordNumber; }
public User getReferencedUser() { return referencedUser; }
```

```
public void setReferencedUser(User referencedUser) {
this.referencedUser = referencedUser; }
public void addExam(Exam exam) { this.examList.add(exam); }
public Exam[] getExams() { return this.examList.toArray(new
Exam[0]); }
@Override public String toString() { return "Student [id=" + id +
", name=" + name + ", program=" + program + ", recordNumber=" +
recordNumber + ", referencedUser=" + referencedUser + ", examList="
+ examList + "]"; }
@Override public int hashCode() { return Objects.hash(id, name,
program, recordNumber); }
@Override public boolean equals(Object obj) { if (this == obj)
return true; if (obj == null) return false; if (getClass() !=
obj.getClass()) return false; Student other = (Student) obj; return
id == other.id && Objects.equals(name, other.name) &&
Objects.equals(program, other.program) && recordNumber ==
other.recordNumber; }
}
```

Лістинг Д.4 – Код класу модулі Exam

```
import java.sql.Date;
import java.time.OffsetDateTime;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Objects;

public class Exam {
     private int id;
     private String place;
     private String room;
     private Date datetime;
     private Subject subject;
     private ArrayList<Student> studentList = new
ArrayList<Student>();
     private Map<Student, Integer> marks = new HashMap<Student,
Integer>();

     public Exam(int id, String place, String room, Subject
subject, Date datetime) {
          this.id = id;
          this.place = place;
          this.room = room;
          this.datetime = datetime;
          this.subject = subject;
     }
```

```
	public Exam(int id, String place, String room, Date datetime)
{
		this.id = id;
		this.place = place;
		this.room = room;
		this.datetime = datetime;
	}

	public Subject getSubject() {
		return subject;
	}

	public void setSubject(Subject subject) {
		this.subject = subject;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getPlace() {
		return place;
	}

	public void setPlace(String place) {
		this.place = place;
	}

	public String getRoom() {
		return room;
	}

	public void setRoom(String room) {
		this.room = room;
	}

	public Date getDatetime() {
		return datetime;
	}

	public void setDatetime(Date datetime) {
		this.datetime = datetime;
	}

	public void addStudent(Student student) {
		this.studentList.add(student);
	}
```

```
    public Student[] getStudentList() {
        return this.studentList.toArray(new Student[0]);
    }

    public void setMark(Student student, int maks) {
        this.marks.put(student, maks);
    }

    public int getMark(Student student) {
        return marks.get(student);
    }

    @Override
    public String toString() {
        return "Exam [id=" + id + ", place=" + place + ", room=" +
room + ", datetime=" + datetime + ", subject="
                + subject + "]";
    }

    @Override
    public int hashCode() {
        return Objects.hash(datetime, id, place, room, subject);
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        Exam other = (Exam) obj;
        return Objects.equals(datetime, other.datetime) && id ==
other.id && Objects.equals(place, other.place)
                && Objects.equals(room, other.room) &&
Objects.equals(subject, other.subject);
    }


}
```

Лістинг Д.5 – Код класу модулі Subject

```
import java.util.ArrayList; import java.util.Objects;
public class Subject { private int id; private String title;
private String lecturer; private int semester; private int credits;
private ArrayList studentList = new ArrayList(); private ArrayList
examList = new ArrayList();
```

```
public Subject(int id, String title, String lecturer, int semester,
int credits) { this.id = id; this.title = title; this.lecturer =
lecturer; this.semester = semester; this.credits = credits; }
public int getId() { return id; }
public void setId(int id) { this.id = id; }
public String getTitle() { return title; }
public void setTitle(String title) { this.title = title; }
public String getLecturer() { return lecturer; }
public void setLecturer(String lecturer) { this.lecturer =
lecturer; }
public int getSemester() { return semester; }
public void setSemester(int semester) { this.semester = semester; }
public int getCredits() { return credits; }
public void setCredits(int credits) { this.credits = credits; }
public void addStudent(Student student) {
this.studentList.add(student); }
public Student[] getStudents() { return
this.studentList.toArray(new Student[0]); }
public void addExam(Exam exam) { this.examList.add(exam); }
public Exam[] getExams() { return this.examList.toArray(new
Exam[0]); }
@Override public int hashCode() { return Objects.hash(credits, id,
lecturer, semester, title); }
@Override public boolean equals(Object obj) { if (this == obj)
return true; if (obj == null) return false; if (getClass() !=
obj.getClass()) return false; Subject other = (Subject) obj; return
credits == other.credits && id == other.id &&
Objects.equals(lecturer, other.lecturer) && semester ==
other.semester && Objects.equals(title, other.title); }
@Override public String toString() { return "Subject [id=" + id +
", title=" + title + ", lecturer=" + lecturer + ", semester=" +
semester + ", credits=" + credits + ", studentList=" + studentList
+ ", examList=" + examList + "]"; }
}
```

Лістинг Д.6 - Код класу UserDao

```
import java.sql.Connection; import java.sql.PreparedStatement;
import java.sql.ResultSet; import java.sql.SQLException; import
java.util.ArrayList; import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService; import
java.util.concurrent.Executors;
import org.max.education.privateshcool.models.Role; import
org.max.education.privateshcool.models.User;
public class UserDao {
private static final ExecutorService executor =
Executors.newFixedThreadPool(10);
```

```
public static CompletableFuture getUserByIdAsync(int id) { return
CompletableFuture.supplyAsync(() -> { return getUserById(id); },
executor ); }
public static CompletableFuture getUserByNameAsync(String username)
{ return CompletableFuture.supplyAsync(() -> { return
getUserByName(username); }, executor ); }
public static CompletableFuture getUserListAsync() { return
CompletableFuture.supplyAsync(() -> { return getUserList(); },
executor ); }
public static CompletableFuture getUserListAsync(int lim) { return
CompletableFuture.supplyAsync(() -> { return getUserList(lim); },
executor ); }
public static CompletableFuture getUserListAsync(String nameTerm,
String emailTerm) { return CompletableFuture.supplyAsync(() -> {
return getUserList(nameTerm, emailTerm); }, executor ); }
public static CompletableFuture getUserListAsync(String nameTerm,
String emailTerm, int lim) { return
CompletableFuture.supplyAsync(() -> { return getUserList(nameTerm,
emailTerm, lim); }, executor ); }
public static CompletableFuture updateUserProfileAsync(User user) {
return CompletableFuture.supplyAsync(() -> { return
updateUserProfile(user); }, executor ); }
public static CompletableFuture updateUserPasswordAsync(User user,
String newPasswdHash) { return CompletableFuture.supplyAsync(() ->
{ return updateUserProfile(user); }, executor ); }
public static CompletableFuture userAuthenticationAsync(String
login, String passwordHash) { return
CompletableFuture.supplyAsync(() -> { return
userAuthentication(login, passwordHash); }, executor ); }
public static CompletableFuture insertUserAsync(User user, String
passwdHash) { return CompletableFuture.supplyAsync(() -> { return
insertUser(user, passwdHash); }, executor ); }
public static CompletableFuture loadUserRolesAsync(User user) {
return CompletableFuture.supplyAsync(() -> { return
loadUserRoles(user); }, executor ); }
public static CompletableFuture deleteUserAsync(User user) { return
CompletableFuture.supplyAsync(() -> { return deleteUser(user); },
executor ); }
public static CompletableFuture updateUserRolesAsync(User user) {
return CompletableFuture.supplyAsync(() -> { return
updateUserRoles(user); }, executor ); }
public static CompletableFuture getRoleByNameAsync(String roleName)
{ return CompletableFuture.supplyAsync(() -> { return
getRoleByName(roleName); }, executor ); }
public static void shutdown() { executor.shutdown(); }
public static User getUserById(int id) { try (Connection connection
= DbConnection.getConnection(); PreparedStatement prepStatement =
connection.prepareStatement( "select username, e_mail from
exams.users where id = ?" )) { prepStatement.setInt(1, id);
ResultSet resultSet = prepStatement.executeQuery(); if
(resultSet.next()) { String username = resultSet.getString(1);
```

```
String email = resultSet.getString(2); User user = new User(id,
username, email); return user; }
} catch (SQLException e) { // TODO Auto-generated catch block
System.err.println("Get User By Id Error!"); e.printStackTrace();
};
return null; }
public static User getUserByName(String username) { try (Connection
connection = DbConnection.getConnection(); PreparedStatement
prepStatement = connection.prepareStatement( "select id, e_mail
from exams.users where username = ?" )) {
prepStatement.setString(1, username); ResultSet resultSet =
prepStatement.executeQuery(); if (resultSet.next()) { int id =
resultSet.getInt(1); String email = resultSet.getString(2); User
user = new User(id, username, email); return user; }
} catch (SQLException e) { // TODO Auto-generated catch block
System.err.println("Get User By Name Error!"); e.printStackTrace();
};
return null; }
public static User[] getUserList() { return getUserList("", "", -
1); }
public static User[] getUserList(int lim) { return getUserList("",
"", lim); }
public static User[] getUserList(String nameTerm, String emailTerm)
{ return getUserList(nameTerm, emailTerm, -1); }
public static User[] getUserList(String nameTerm, String emailTerm,
int lim) {
if (lim < 0) lim = Integer.MAX_VALUE; try (Connection connection =
DbConnection.getConnection(); PreparedStatement prepStatement =
connection.prepareStatement( "select id, username, e_mail from
exams.users " + "where username like ? and e_mail like ? " + "order
by username desc " + "limit ?" )) {
prepStatement.setString(1, "%" + nameTerm + "%");
prepStatement.setString(2, "%" + emailTerm + "%");
prepStatement.setInt(3, lim);
ArrayList users = new ArrayList(); ResultSet resultSet =
prepStatement.executeQuery(); while (resultSet.next()) { int id =
resultSet.getInt(1); String username = resultSet.getString(2);
String email = resultSet.getString(3); users.add(new User(id,
username, email)); } return users.toArray(new User[0]);
} catch (SQLException e) { // TODO Auto-generated catch block
System.err.println("Get User List Error"); e.printStackTrace(); }
return null; }
public static boolean updateUserProfile(User user) { try
(Connection connection = DbConnection.getConnection();
PreparedStatement prepStatement = connection.prepareStatement(
"update exams.users set username = ?, e_mail = ? where id = ?" )) {
prepStatement.setString(1, user.getUsername());
prepStatement.setString(2, user.getEmail());
prepStatement.setInt(3, user.getId());
prepStatement.executeUpdate(); return true; } catch (SQLException
e) { // TODO Auto-generated catch block System.err.println("User
Profile Updare error!"); e.printStackTrace(); } return false; }
```

```
public static boolean updateUserPassword(User user, String
newPasswdHash) { try (Connection connection =
DbConnection.getConnection(); PreparedStatement prepStatement =
connection.prepareStatement( "update exams.users set passwd_hash =
? where id = ?" )) { prepStatement.setString(1, newPasswdHash);
prepStatement.setInt(2, user.getId());
prepStatement.executeUpdate(); return true; } catch (SQLException
e) { // TODO Auto-generated catch block System.err.println("User
Password Updare error!"); e.printStackTrace(); } return false; }
public static boolean userAuthentication(String login, String
passwordHash) {
try (Connection connection = DbConnection.getConnection();
PreparedStatement prepStatement = connection.prepareStatement(
"select id, username from exams.users " + "where username = ? and
passwd_hash = ?" )) { prepStatement.setString(1, login);
prepStatement.setString(2, passwordHash); ResultSet resultSet =
prepStatement.executeQuery(); return resultSet.next();
} catch (SQLException e) { // TODO Auto-generated catch block
System.err.println("User Authentication Error!");
e.printStackTrace(); }
return false; }
public static User loadUserRoles(User user) { try (Connection
connection = DbConnection.getConnection(); PreparedStatement
prepStatement = connection.prepareStatement( "select role_id, name,
title, rolemessage from exams.roles, exams.user_roles " + "where
roles.id = user_roles.role_id and user_roles.user_id = ?" )) {
prepStatement.setInt(1, user.getId()); ResultSet resultSet =
prepStatement.executeQuery(); while (resultSet.next()) { int roleId
= resultSet.getInt(1); String roleName = resultSet.getString(2);
String roleTitle = resultSet.getString(3); String roleMsg =
resultSet.getString(4); user.addRole(new Role(roleId, roleName,
roleTitle, roleMsg)); } return user;
} catch (SQLException e) { // TODO Auto-generated catch block
System.err.println("Role Update Error!"); e.printStackTrace(); }
return null; } public static boolean insertUser(User user, String
passwdHash) { try (Connection connection =
DbConnection.getConnection(); PreparedStatement prepStatement =
connection.prepareStatement( "insert into exams.users (username,
e_mail, passwd_hash) " + "values (?, ?, ?)" )) {
prepStatement.setString(1, user.getUsername());
prepStatement.setString(2, user.getEmail());
prepStatement.setString(3, passwdHash);
if (prepStatement.executeUpdate() >= 1) { PreparedStatement ps =
connection.prepareStatement( "select id from exams.users where
username = ?" ); ps.setString(1, user.getUsername()); ResultSet
resultSet = ps.executeQuery(); if (resultSet.next()) { int userId =
resultSet.getInt(1); user.setId(userId); return true; } }
} catch (SQLException e) { // TODO Auto-generated catch block
System.err.println("Insert New User Error!"); e.printStackTrace();
} return false; }
public static boolean deleteUser(User user) { try (Connection
connection = DbConnection.getConnection(); PreparedStatement
```

```
prepStatement = connection.prepareStatement( "delete from
exams.users where id = ?" )) {
prepStatement.setInt(1, user.getId()); return
prepStatement.executeUpdate() == 1;
} catch (SQLException e) { // TODO Auto-generated catch block
e.printStackTrace(); } return false; }
public static boolean updateUserRoles(User user) { if
(user.getRoles().length == 0) return true; try (Connection
connection = DbConnection.getConnection()) {
connection.setAutoCommit(false);
PreparedStatement statementDelete = connection.prepareStatement(
"delete from exams.user_roles " + "where user_id = ?; " );
statementDelete.setInt(1, user.getId());
statementDelete.executeUpdate(); connection.commit();
try { for (Role role : user.getRoles()) { PreparedStatement
insertRoleStatement = connection.prepareStatement( "insert into
exams.user_roles (user_id, role_id) " + "values (?, ?); " );
insertRoleStatement.setInt(1, user.getId());
insertRoleStatement.setInt(2, role.getId());
insertRoleStatement.executeUpdate(); connection.commit(); } } catch
(SQLException exp) { // TODO: handle exception
System.err.println("Update User Roles Error (inner)!");
exp.printStackTrace(); connection.rollback(); }
connection.commit(); return true;
} catch (SQLException e) { // TODO Auto-generated catch block
System.err.println("Update User Roles Error!");
e.printStackTrace(); } return false; }
public static Role getRoleByName(String roleName) { try (Connection
connection = DbConnection.getConnection(); PreparedStatement
prepStatement = connection.prepareStatement( "select id, title,
rolemessage from exams.roles " + "where name = ?" )) {
prepStatement.setString(1, roleName); ResultSet resultSet =
prepStatement.executeQuery(); if (resultSet.next()) { int roleId =
resultSet.getInt(1); String roleTitle = resultSet.getString(2);
String roleMessage = resultSet.getString(3); return new
Role(roleId, roleName, roleTitle, roleMessage); } else return null;
} catch (SQLException e) { // TODO Auto-generated catch block
System.err.println("Role Load Error!"); e.printStackTrace(); }
return null; }
public static void main(String[] args) { // TODO Auto-generated
method stub //User user = getAuthorizedUser("user1", "123"); //
System.out.println(user); // user.setUsername("user1"); //
user.setEmail("mailmailmail@mail.com"); //
System.out.println(updateUserProfile(user, "123")); //
System.out.println(user); //
System.out.println(updateUserPappword(user, "123", "123"));
// User user = getAuthorizedUserAsync("user1", "123").get(); //
System.out.println(user);
// CompletableFuture futureUser = getAuthorizedUserAsync("user1",
"123"); // futureUser.thenAcceptAsync(user -> { //
System.out.println(user); // user.setEmail("newemail@mail.com"); //
// CompletableFuture futureUpdateFrofile =
```

```
updateUserProfileAsync(user, "123"); //
futureUpdateFrofile.thenAcceptAsync(success -> { //
System.out.println("Profile updated: " + success); //
System.out.println(user); // }).exceptionally(throwable -> { //
System.err.println("Error: " + throwable.getMessage()); // return
null; // }); // // CompletableFuture futureUpdatePassword =
updateUserPasswordAsync(user, "123", "123"); //
futureUpdatePassword.thenAcceptAsync(success -> { //
System.out.println("Password updated: " + success); //
}).exceptionally(throwable -> { // System.err.println("Error: " +
throwable.getMessage()); // return null; // }); // //
}).exceptionally(throwable -> { // System.err.println("Error: " +
throwable.getMessage()); // return null; // }); //
// User[] users = getUserList("", "", -1); // for (User user :
users) { // System.out.println(user); // }
// User user1 = getUserById(1); // System.out.println(user1);
// User user1 = getUserById(2); // System.out.println(user1); // //
int id = user1.getId(); // String oldmail = user1.getEmail(); //
String oldusername = user1.getUsername(); // //
user1.setEmail("user1@mail.com"); // user1.setUsername("user1"); //
updateUserProfile(user1); // // user1 = getUserById(id); //
System.out.println(user1);
// User newUser = new User("new_user_name",
"newuseremail@mail.com"); // System.out.println(incertUser(newUser,
"123"));
Role admin = getRoleByName("admin"); Role student =
getRoleByName("student"); Role testTaker =
getRoleByName("testtaker");
User newuser = new User("petya", "petya@mail.com");
System.out.println(insertUser(newuser, "123"));
newuser.addRole(admin); newuser.addRole(student);
newuser.addRole(testTaker);
CompletableFuture futureUpdateUserRoles =
updateUserRolesAsync(newuser);
futureUpdateUserRoles.thenAcceptAsync(b -> { System.out.println(b);
int id = newuser.getId(); CompletableFuture getUserByIdFuture =
getUserByIdAsync(id); getUserByIdFuture.thenAcceptAsync(user -> {
System.out.println(user); });
});
// User toDel = getUserByName("vasya"); //
System.out.println(toDel); // //
System.out.println(deleteUser(toDel));
}
}
```

Лістинг Д.7 – Код класу StudentDao

```
import java.sql.Connection; import java.sql.PreparedStatement;
import java.sql.ResultSet; import java.sql.SQLException; import
java.sql.Types; import java.util.ArrayList; import
java.util.Optional; //import java.util.concurrent.ExecutorService;
//import java.util.concurrent.Executors;

import org.max.education.privateshcool.models.Exam; import
org.max.education.privateshcool.models.Student; import
org.max.education.privateshcool.models.Subject;

public class StudentDao {

//private static final ExecutorService executor =
Executors.newFixedThreadPool(10);

public static Student getStudentById(int id) { try (Connection
connection = DbConnection.getConnection(); PreparedStatement ps =
connection.prepareStatement( "select name, program, record_number "
+ "from exams.students where id = ? order by id asc" )) {

ps.setInt(1, id); ResultSet rs = ps.executeQuery(); if (rs.next())
{ String name = rs.getString(1); String program = rs.getString(2);
int rec = rs.getInt(3); return new Student(id, name, program, rec);
}

} catch (SQLException e) { // TODO Auto-generated catch block
System.err.println("Get Student by Id Error"); e.printStackTrace();
} return null; }

public static Student getStudentByNumber(int number) { try
(Connection connection = DbConnection.getConnection();
PreparedStatement ps = connection.prepareStatement( "select id,
name, program " + "from exams.students where record_number = ?
order by id asc" )) {

ps.setInt(1, number); ResultSet rs = ps.executeQuery(); if
(rs.next()) { int studId = rs.getInt(1); String name =
rs.getString(2); String program = rs.getString(3); return new
Student(studId, name, program, number); }

} catch (SQLException e) { // TODO Auto-generated catch block
System.err.println("Get Student by Number Error");
e.printStackTrace(); } return null; }

public static Student[] getStudentsList( Optional name, Optional
program) { String nameTerm = name.orElse(""); String programTerm =
program.orElse(""); try (Connection conn =
```

```
DbConnection.getConnection(); PreparedStatement ps =
conn.prepareStatement( "select id, name, program, record_number " +
"from exams.students where " + "name like ? and program like ? " +
"order by name" )) { ps.setString(1, "%" + nameTerm + "%");
ps.setString(2, "%" + programTerm + "%"); ResultSet rs =
ps.executeQuery(); ArrayList list = new ArrayList(); while
(rs.next()) { int studId = rs.getInt(1); String studName =
rs.getString(2); String studProgram = rs.getString(3); int
studRecord = rs.getInt(4); list.add(new Student(studId, studName,
studProgram, studRecord)); } return list.toArray(new Student[0]);

} catch (SQLException e) { // TODO Auto-generated catch block
System.err.println("Get Student List Error!"); e.printStackTrace();
}

return new Student[0];

}

public static Student[] getStudentsOfSubject(Subject subject) { try
(Connection conn = DbConnection.getConnection(); PreparedStatement
ps = conn.prepareStatement( "select st.id, st.name, st.program,
st.record_number " + "from exams.students as st,
exams.student_subjects as ss, exams.subjects as s " + "where st.id
= ss.student_id and ss.subject_id = s.id and s.id = ? " + "order by
st.name" )) { ps.setInt(1, subject.getId()); ResultSet rs =
ps.executeQuery(); ArrayList studList = new ArrayList(); while
(rs.next()) { int id = rs.getInt(1); String name = rs.getString(2);
String program = rs.getString(3); int recordNum = rs.getInt(4);
studList.add(new Student(id, name, program, recordNum)); } return
studList.toArray(new Student[0]); } catch (SQLException e) { //
TODO Auto-generated catch block System.err.println("Get Students Of
Subject Error!"); e.printStackTrace(); } return new Student[0];

}

public static Student[] getStudentsOfExam(Exam exam) { try
(Connection conn = DbConnection.getConnection(); PreparedStatement
ps = conn.prepareStatement( "select st.id, st.name, st.program,
st.record_number " + "from exams.students as st,
exams.student_exams as ste, exams.exams as e " + "where st.id =
ste.student_id and ste.exam_id = e.id " + "and e.id = ? order by
st.name" )) { ps.setInt(1, exam.getId()); ResultSet rs =
ps.executeQuery(); ArrayList studList = new ArrayList(); while
(rs.next()) { int id = rs.getInt(1); String name = rs.getString(2);
String program = rs.getString(3); int recordNum = rs.getInt(4);
studList.add(new Student(id, name, program, recordNum)); } return
studList.toArray(new Student[0]); } catch (SQLException e) { //
TODO Auto-generated catch block System.err.println("Get Students Of
Exams Error!"); e.printStackTrace(); } return new Student[0];

}
```

```
public static boolean updateStudent(Student student) { try
(Connection conn = DbConnection.getConnection(); PreparedStatement
ps = conn.prepareStatement( "update exams.students " + "set name =
?, program = ?, record_number = ?, user_id = ? " + "where id = ?"
)) {

ps.setString(1, student.getName()); ps.setString(2,
student.getProgram()); ps.setInt(3, student.getRecordNumber()); if
(student.getReferencedUser() != null) { ps.setInt(4,
student.getReferencedUser().getId()); } else { ps.setNull(4,
Types.INTEGER); } ps.setInt(5, student.getId());

return ps.executeUpdate() == 1;

} catch (SQLException ex) { System.err.println("Student Profile
Update Error!"); ex.printStackTrace(); } return false; }

public static boolean insertStudent(Student student) { try
(Connection conn = DbConnection.getConnection(); PreparedStatement
ps = conn.prepareStatement( "insert into exams.students (name,
program, record_number) " + "values (?, ?, ?)" )) {

ps.setString(1, student.getName()); ps.setString(2,
student.getProgram()); ps.setInt(3, student.getRecordNumber());

return ps.executeUpdate() == 1;

} catch (SQLException ex) { System.err.println("Student Insert
Error!"); ex.printStackTrace(); } return false; }

public static boolean deleteStudent(Student student) { try
(Connection conn = DbConnection.getConnection(); PreparedStatement
ps = conn.prepareStatement( "delete from exams.students " + "where
id = ?" )) {

ps.setInt(1, student.getId());

return ps.executeUpdate() == 1;

} catch (SQLException ex) { System.err.println("Student Delete
Error!"); ex.printStackTrace(); } return false; }

public static void main(String[] args) { // int id = 1; // Student
stud = getStudentById(id); // System.out.println(stud);

// int number = 24010; // Student stud =
getStudentByNumber(number); // System.out.println(stud);

// String nameterm = "Jo"; // String protramTerm = "Te"; // //
Student[] students = getStudentsList(Optional.empty(),
Optional.of(protramTerm)); // for (Student student : students) { //
System.out.println(student); // }
```

```
//Subject subject = new Subject(3, null, null, 0, 0); // Exam exam
= new Exam(3, null, null, null, null); // // //Student[] students =
getStudentsOfSubject(subject); // Student[] students =
getStudentsOfExam(exam); // for (Student student : students) { //
System.out.println(student); // }

// Student student = new Student(4, "Jane", "Doe", 11111); //
System.out.println(updateStudent(student)); // student =
getStudentById(student.getId()); // System.out.println(student);

// Student student = new Student(4, "", "", 0); //
System.out.println(deleteStudent(student)); // Student student2 =
new Student(21, "", "", 0); //
System.out.println(deleteStudent(student2));

} }
```

Лістинг Д.8 – Код класу ExamDao

```
package org.max.education.privateshcool.dao;

import java.sql.Connection;
import java.sql.Date;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

import org.max.education.privateshcool.models.Exam;
import org.max.education.privateshcool.models.Student;

public class ExamDao {
     @SuppressWarnings("deprecation")
     public static Exam[] getExamOfStudent(Student student) {
          try (Connection connection = DbConnection.getConnection();
                PreparedStatement ps = connection.prepareStatement(

                    "select e.id, e.place, e.room, e.time " +
                    "from exams.exams as e, exams.student_exams as
se, exams.students as s " +
                    "where e.id = se.exam_id and se.student_id =
s.id and s.id = ?"
             )) {
                ps.setInt(1, student.getId());
                ResultSet rs = ps.executeQuery();
                ArrayList<Exam> exams = new ArrayList<Exam>();
                while (rs.next()) {
                     int id = rs.getInt(1);
                     String place = rs.getString(2);
```

```
                String room = rs.getString(3);
                Date date = rs.getDate(id);
                exams.add(new Exam(id, place, room, date));
            }
            return exams.toArray(new Exam[0]);

        } catch (SQLException e) {
            System.err.println("Get Exam Students Error!");
            e.printStackTrace();
        }
        return new Exam[0];
    }

    public static Exam[] getAllExams() {
        return new Exam[0];
    }

    public static void main(String[] args) {
        Student student = new Student(3,"","",0);
        Exam[] exams = getExamOfStudent(student);
        for (Exam e : exams) {
            System.out.println(e);
        }
    }
}
```

Лістинг Д.9 – Код класу SubjectDao

```
package org.max.education.privateshcool.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

import org.max.education.privateshcool.models.Student;
import org.max.education.privateshcool.models.Subject;

public class SubjectDao {

    public static Subject getSubjectById(int id) {
        try(Connection conn = DbConnection.getConnection();
            PreparedStatement ps = conn.prepareStatement(
                    "select title, lecturer, semestr, credits "
+
                    "from exams.subjects where id = ?"
        )) {
            ps.setInt(1, id);
```

```
                ResultSet rs = ps.executeQuery();
                if (rs.next()) {
                     String title = rs.getString(1);
                     String lecturer = rs.getString(2);
                     int semester = rs.getInt(3);
                     int credits = rs.getInt(4);
                     return new Subject(id, title, lecturer,
semester, credits);
                }
           } catch (SQLException e) {
                // TODO Auto-generated catch block
                System.err.println("Get Subject By Id Error!");
                e.printStackTrace();
           }
           return null;
     }

     public static Subject[] getSubjectsOfStudent(Student student)
{
           try (Connection conn = DbConnection.getConnection();
                 PreparedStatement ps = conn.prepareStatement(
                     "select s.id, s.title, s.lecturer, s.semestr,
s.credits " +
                     "from exams.subjects as s,
exams.student_subjects as ss, exams.subjects as st " +
                     "where st.id = ss.student_id and ss.subject_id =
s.id and st.id = ?"
           )) {
                ps.setInt(1, student.getId());
                ResultSet rs = ps.executeQuery();
                ArrayList<Subject> subjectList = new
ArrayList<Subject>();
                while (rs.next()) {
                     int id = rs.getInt(1);
                     String title = rs.getString(2);
                     String lecturer = rs.getString(3);
                     int semestr = rs.getInt(4);
                     int credits = rs.getInt(5);
                     subjectList.add(new Subject(id, title, lecturer,
semestr, credits));
                }
                return subjectList.toArray(new Subject[0]);

           } catch (SQLException e) {
                System.err.println("Get Subject List Of Student
Error!");
                e.printStackTrace();
           }
           return new Subject[0];
     }

     public static Subject[] getAllSubjects() {
```

```
          try (Connection conn = DbConnection.getConnection();
                Statement statement = conn.createStatement()) {
               ResultSet rs = statement.executeQuery(
                     "select id, title, lecturer, semestr, credits "
+
                          "from exams.subjects order by title"
               );
               ArrayList<Subject> subjects = new
ArrayList<Subject>();
               while (rs.next()) {
                    int id = rs.getInt(1);
                    String title = rs.getString(2);
                    String lecturer = rs.getString(3);
                    int semestr = rs.getInt(4);
                    int credits = rs.getInt(5);

                    subjects.add(new Subject(id, title, lecturer,
semestr, credits));
               }
               return subjects.toArray(new Subject[0]);

          } catch (SQLException e) {
               // TODO Auto-generated catch block
               e.printStackTrace();
          }
          return new Subject[0];
     }

     public static void main(String[] args) {
//        System.out.println(getSubjectById(2));
//        Student student = StudentDao.getStudentById(2);
//        Subject[] subjects = getSubjectsOfStudent(student);
          Subject[] subjects = getAllSubjects();
          for (Subject s : subjects) {
               System.out.println(s);
          }
     }
}
```

Лістинг Д.10 – Код файлу сторінки index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
```

```
    <meta charset="utf-8">
    <link rel="stylesheet" href="<c:url value='/css/styles.css'/>">
    <title>Private School Website</title>
</head>
<body>
<jsp:include page="/WEB-INF/views/header.jsp"/>
    <main class="main">
     <jsp:include page="/WEB-INF/views/sidebars.jsp"/>
     <jsp:include page="/WEB-INF/views/content.jsp"/>
    </main>
    <jsp:include page="/WEB-INF/views/footer.jsp"/>
</body>
</html>
```

Лістинг Д.11 – Код файлу сторінки header.jsp

```
<header id="header">
     <aside id="userpanel" class="aside">
          <div class="userinfo">
               <span>Welcome, Guest</span> <br> <a
href="/login">Login</a>
          </div>
     </aside>
     <h1>My Page</h1>
     <nav id="nav">
          <ul>
               <li><a href="#">Menu Item 1</a></li>
               <li><a href="#">Menu Item 2</a></li>
               <li><a href="#">Menu Item 3</a></li>
          </ul>
     </nav>
</header>
```

Лістинг Д.12 – Код файлу сторінки sidebars.jsp

```
<aside id="sidebar1" class="aside">
     <h2>Lorem, ipsum dolor sit amet consectetur</h2>
     <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
Ipsam
          illum porro consequuntur, sequi nesciunt modi facilis qui
dicta vel
          temporibus.</p>
</aside>

<aside id="sidebar2" class="aside">
     <h2>Lorem ipsum dolor sit amet consectetur adipisicing elit.
          Accusamus, odio?</h2>
```

```
     <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
Iure
          illo alias perferendis adipisci consequuntur aliquid
reiciendis nulla
          dignissimos voluptate nesciunt! Accusamus consequatur ea,
voluptas ut
          quod natus officiis vero et?</p>
     <h3>Options</h3>
     <ul>
          <li>Item 1</li>
          <li>Item 2</li>
     </ul>
</aside>
```

Лістинг Д.13 – Код файлу сторінки content.jsp

```
<article id="article">
     <h2>What is Lorem Ipsum?</h2>
     <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.
          Consequuntur, quibusdam eligendi velit iste quam sit
cumque ipsam
          cupiditate temporibus quisquam nesciunt mollitia odit
aspernatur culpa
          nihil blanditiis accusantium facere dolore.</p>
     <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
Ducimus
          maxime tempora, quaerat quasi quod dicta corrupti odit
doloribus
          consequatur a, quisquam sequi velit aut. Possimus cumque
repudiandae
          nisi blanditiis ab.</p>
</article>
```

Лістинг Д.14 – Код файлу сторінки footer.jsp

```
<footer id="footer">
     <p>Contacts: admin@dummymail.com</p>
     <p>Copyright © Private Language School, 2025</p>
</footer>
```

Інфокомунікаційна система екзаменаційно-тренінгового центру іноземних мов

ДИПЛОМНА РОБОТА

Спеціальність 172 – Електронні комунікації та радіотехніка

Виконав: студент 2 курсу, група $ЕКР_{м}$-24-1 М.М. Коржан
Керівник: д-р техн. наук, професор С. К. Підченко

Факультет *інформаційних технологій*
Кафедра *телекомунікацій, медійних та інтелектуальних технологій*
Освітня-професійна програма *Електронні інформаційно-комунікаційні системи та мережі*

Вступ та актуальність проблеми

1. Технології та засоби електронних комунікацій сприяють автоматизації процесів функціонування систем різного призначення. Зокрема, в освітній діяльності доцільно керувати навчальним процесом через централізоване програмне забезпечення інфокомунікаційної системи. В процесі вивчення іноземних мов та проведення іспитів виникають проблеми, що пов'язані із керуванням взаємодії учасників освітнього процесу, що особливо актуально для дистанційної форма навчання. Крім того, сучасні інформаційні технології та мережа Інтернет надає цілодобовий доступ до інформаційного ресурсу з розкладом, графіками іспитів, домашнім завданим тощо.

2. Незважаючи на доволі об'ємну матеріально-технічну базу та методичні напрацювання щодо розробки програмного забезпечення для інфокомунікаційних освітніх систем, актуальною залишається проблема забезпечення надійного та безвідмовного функціонування останніх. Особливо гостро ця проблема ставиться у випадку суто дистанційного (онлайн) навчання, для великої кількості учасників, що одночасно здійснюють запити до системи. В таких випадках відмова одного або декількох ключових програмних компонентів призводять до відмови усієї системи.

Мета та завдання роботи

Мета роботи: підвищення надійності програмних компонентів інфокомунікаційної системи екзаменаційно-тренінгового центру іноземних мов.

Для досягнення поставленої мети сформульовані та вирішені такі ***задачі дослідження***:

1. Провести аналіз предметної області, актуальних публікацій згідно теми, розглянути переваги та недоліки існуючих програмних систем.
2. Розробка формальної моделі інфокомунікаційної системи екзаменаційно-тренінгового центру іноземних мов з урахуванням головних процесів та учасників. Провести декомпозиція системи у відповідності до структурного підходу з визначенням вхідних та вихідних потоків даних, параметрів, а також керуючих сигналів.
3. Розробка моделі надійності системи на основі резервування найбільш вразливих програмних компонентів.
4. Розробка моделі варіантів використання ІКС на основі функціональних вимог; алгоритмів та моделювання інфокомунікаційних процесів взаємодії акторів та системи; моделі даних та практична реалізація бази даних.
5. Програмна реалізація системи згідно триступеневої архітектури з дотриманням нефункціональних вимог. Реалізація моделі даних та компонентів взаємодії з базою даних, компонентів автоматизації інфокомунікаційних процесів системи та елементів інтерфейсу.

Об’єкт та предмет дослідження

Об’єкт дослідження:
інформаційно-комунікаційні процеси електронної комунікації між учасниками екзаменаційно-тренінгового центру іноземних мов.

Предмет дослідження:
програмне забезпечення та метод підвищення надійності компонентів інфокомунікаційної системи екзаменаційно-тренінгового центру іноземних мов.

Наукова новизна та практична значення

1. Отримав подальшого розвитку метод ковзного резервування програмних компонентів доступу до бази даних серверної частини програмного забезпечення, що дозволяє збільшити середній час напрацювання на відмову до 154 годин (більше 6 днів), що понад у 12 разів більше за відповідний показник для лише одного елементу із базовими показниками надійності, з середнім часом відновлення до 7 хвилин.

2. Розроблено прототип програмного забезпечення інфокомунікаційної системи екзаменаційно-тренінгового центу іноземних мов на базі набору технологій Java EE, що дозволяє виконувати базові функції згідно технічного завдання. Додаток може бути використаний для тестування процесів функціонування відповідних систем з метою оптимізації бізнес-процесів освітньої діяльності.

Апробація результатів: за матеріалами роботи підготовлені тези доповіді для XVII всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук», АПКН–2025.

1 Огляд існуючих засобів забезпечення функціонування інфокомунікаційних систем освіти
1.1 Огляд програмно-апаратних засобів

Таблиця 1.1 – Апаратні засоби ІКС

Категорія	Опис та приклади	Застосування в освіті
Обчислювальні пристрої	Комп'ютери, ноутбуки, планшети, мейнфрейми (для централізованих систем).	Доступ до онлайн-курсів мов; проведення тестів з оцінкою засобами ШІ.
Комунікаційне обладнання	Камери, мікрофони, гарнітури, мережеві пристрої (маршрутизатори LAN/WAN).	Відеоконференції для розмовної практики мов; запис відповідей для атестації.
Дисплейні та інтерактивні пристрої	Розумні дошки, проектори, гарнітури доповненої реальності, мультимедійні комп'ютери.	Використання прогресивних технологій віртуальної та доповненої реальності; інтерактивні тести на екранах.

Програмні засоби ІКС

Таблиця 1.2 – Програмні засоби ІКС

Категорія	Опис та приклади	Застосування в освіті
Системи управління навчанням (СКН)	Платформи для організації навчання: Google Classroom, Moodle, Blackboard.	Курси іноземних мов з модулями; онлайн-атестація з контролем прогресу.
Інструменти для комунікації	Відеоконференції: Zoom, Microsoft Teams, Google Meet; синхронні (інтерактивні чати) та асинхронні (e-mail, форуми).	Практика розмовної мови в реальному часі; групові проекти для атестації.
Додатки для вивчення іноземних мов	CALL-програми: Duolingo, Rosetta Stone, Ellis, Dynamic English; мультимедіа з розпізнаванням мови (SpeechViewer).	Самостійне вивчення граматики/словника; інтегративні симуляції для практики.
Інструменти атестації та ШІ	Платформи оцінювання: Quizlet, Edulastic; ШІ-інструменти для зворотного зв'язку та перевірка граматики.	Формативні тести з даними; персоналізована оцінка прогресу студентів.

Огляд існуючих ІКС

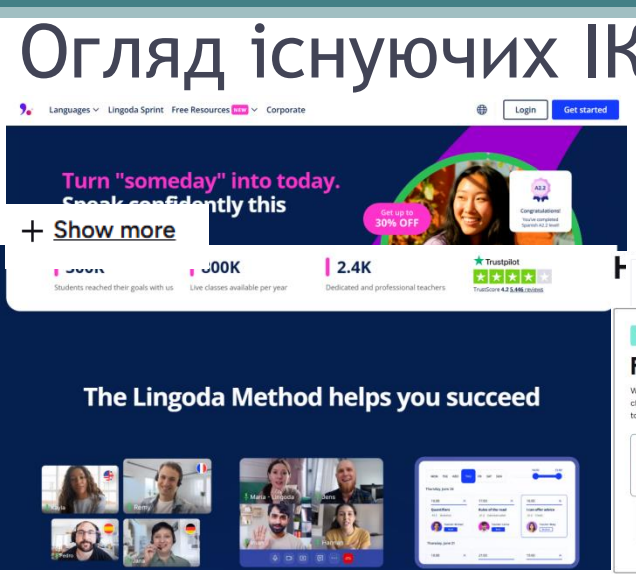


Рисунок 1.1 – Інтерфейс сайту платформи Lingoda

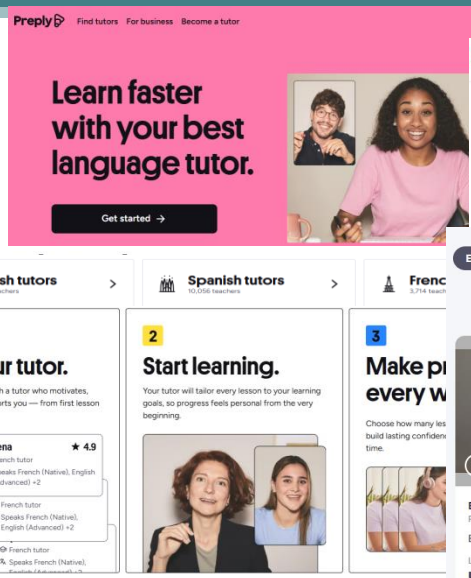


Рисунок 1.2 – Інтерфейс сайту платформи Preply

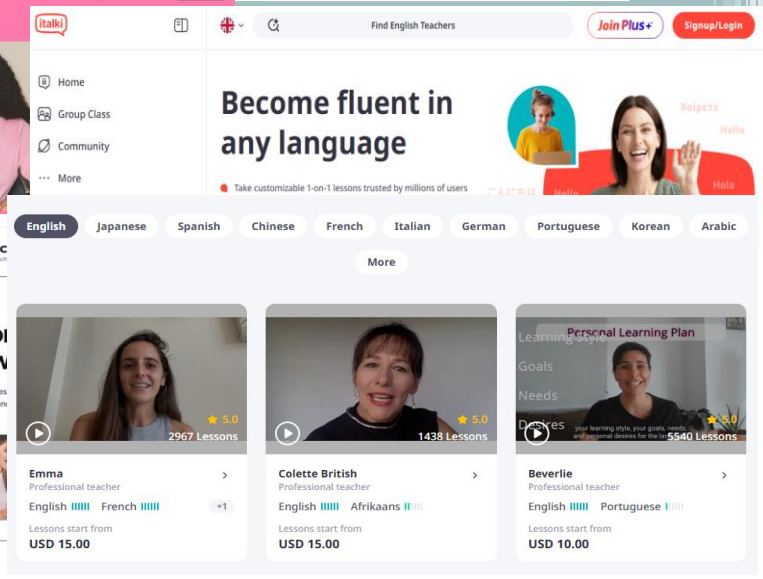


Рисунок 1.3 – Інтерфейс сайту платформи Italki

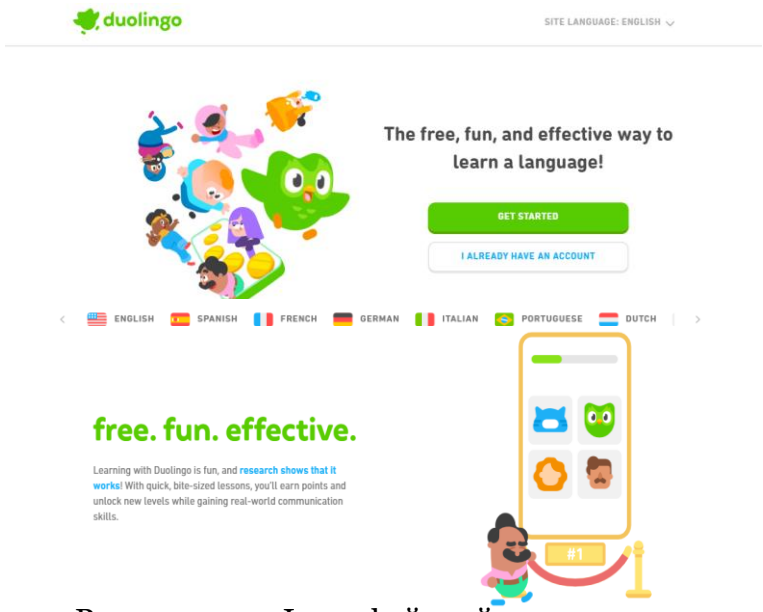


Рисунок 1.4 – Інтерфейс сайту платформи Duolingo

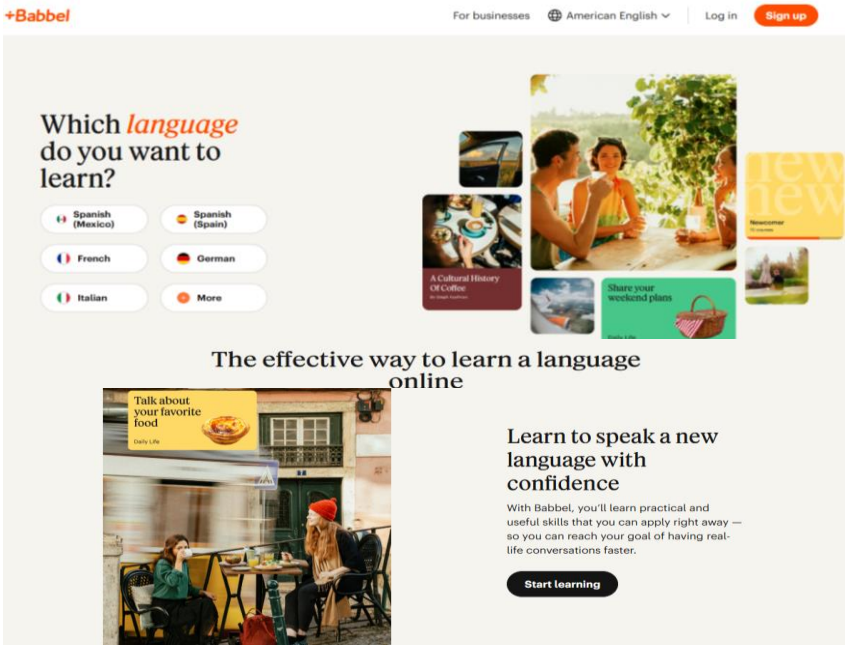


Рисунок 1.5 – Інтерфейс сайту платформи Babbel

Результати порівняльного аналізу вищеописаних онлайн-платформ для вивчення іноземних мов та атестації знань зведено до Таблиці 1.3.

Назва	Кількість мов	Програмні засоби	Функції	Атестація знань	Орієнтовна вартість
1	2	3	4	6	7
Lingoda	5 (англійська, німецька, франц., іспанська, бізнес-англійська)	Віртуальні класи Zoom, інтеракт. PDF, словники, домашні завдання.	Груп. та інд. уроки з носіями, рівні A1-C2, розм. практика, граматика, письмо, аудіювання.	Тест на рівень (CEFR), сертифікати після завершення рівня (50-60 занять), підготовка до іспитів.	Групові: від $11/урок, індивідуальні: від $20/урок, пакети 4-40 занять/міс, 7-денний пробний період.
Preply	Понад 90 (англійська, франц., іспанська, німецька тощо)	Віртуальний клас, дошки, флеш-карти, ШІ-аналіз для нотаток та словників, домашні завдання.	Індивід. уроки з репетиторами-носіями, гнучкий розклад, індивідуал. плани для розмовної практики та граматики.	Безкоштовн. тест рівня мови, підготовка до DELF, TELC, DALF з прикладами завдань іспитів.	Від $15-$50/урок, місячні підписки, пробний урок.
Italki	Понад 150 (англійська, японська, іспанська, китайська тощо)	Віртуальний клас, чат, спільнота для практики, записи уроків.	Індивід. та груп. роки, гнучкий розклад, фокус на розмовну практику, спільнота для обміну знань.	Безкоштовн. тест рівня мови, відстеження прогресу.	Вартість залежить від викладача (від $10/урок), без фіксованої підписки.
1	2	3	4	6	7
Duolingo	43 (англійська, франц., іспанська, німецька, японська тощо	Інтерактивні вправи, аудіо від носіїв, AI-рольові ігри, історії, пояснення помилок,	Ігрові уроки, курси за CEFR, фокус на словник, граматику, аудіювання, розмовні фрази.	Duolingo English Test ($70, слухання/читання/говоріння/письмо, Duolingo Score.	Безкоштовна базова версія, преміум-підписка від $7/міс.
Babbel	14 (іспанська, франц., німецька, італійська тощо)	Інтерактивні уроки, розпізнавання. мови, подкасти, Babbel Live (віртуальний клас), тематичні курси.	Короткі уроки, аудіо від носіїв, Babbel Live для груп. та інд. занять.	Тест на рівень (CEFR A1-C1), Babbel English Test ($40, слухання/читання, A1-B1, сертифікат), підготовка до іспитів.	Від $22/міс, безкоштовний пробний урок для Babbel Live.

Таблиця 1.3 – Порівняльна таблиця п’яти відомих онлайн платформ для вивчення іноземних мов та атестації знань

1.2 Аналіз інфокомунікаційних процесів та головних вимог

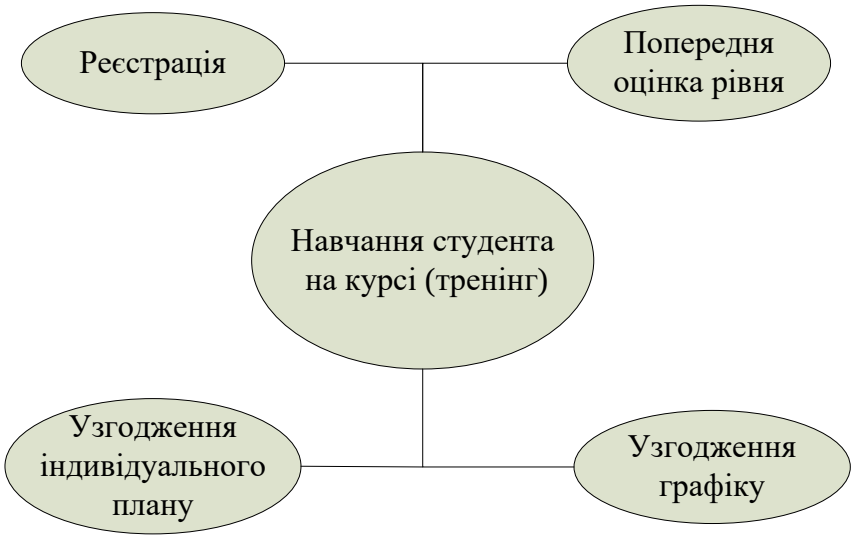


Рисунок 1.6 – Процес «Навчання студента на курсі»

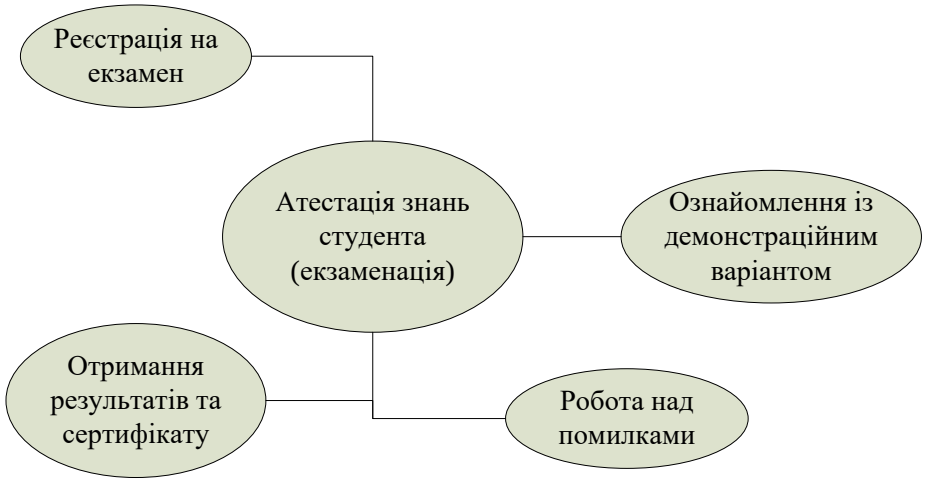


Рисунок 1.7 – Процес «Атестація знань»

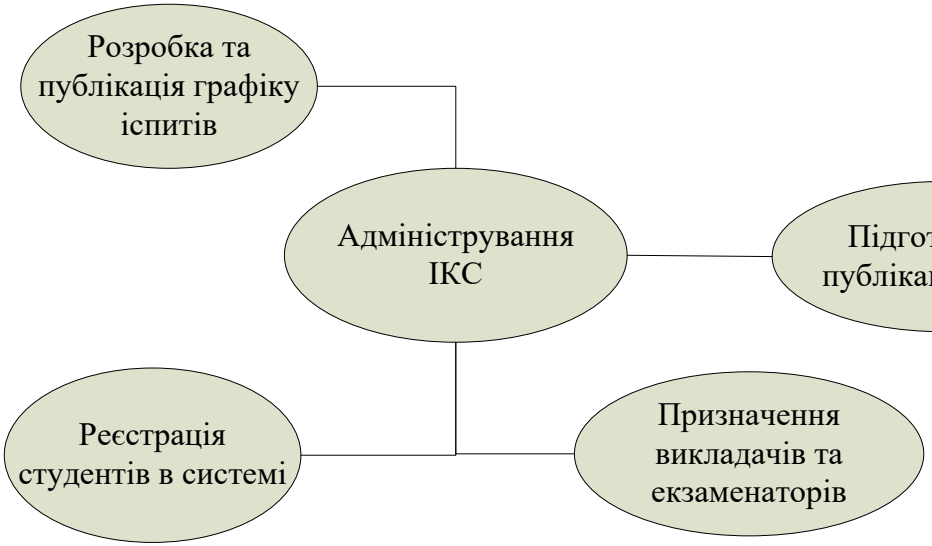


Рисунок 1.8 – Процес «Адміністрування ІКС»

Аналів вимог та постановка задачі

Таблиця 1.4 – Головні вимоги до системи

Користувач (User)	Вимоги функціональні (System requirements)	Вимоги нефункціональні (General requirements)
Студент (Student)	1. Переглянути список доступних курсів	15. Обробка запитів з часом відгуку не більше 2 секунд з навантаженням до 1000 одночасних користувачів.
	2. Записатись на курс	
	3. Переглянути список своїх курсів	
	4. Припинити навчання на курсі	16. Використання HTTPS-протокол для всіх сесій, шифрування персональних даних за стандартом AES-256.
Екзаменований (Exam Taker)	5. Переглянути список доступних екзаменів	
	6. Зареєструватись на екзамен	17. Система повинна бути сумісною з основними браузерами (Chrome, Firefox, Safari, Edge) версій не старше 2 років, та операційними системами (Windows, macOS, iOS, Android), забезпечуючи коректне відтворення мультимедіа для мовних вправ (аудіо/відео).
	7. Переглянути список своїх екзаменів	
	8. Скасувати або перенести екзамен	
Адміністратор (Admin)	9. Додати нового користувача (студента або екзаменованого)	18. Система повинна мати високий рівень доступності з резервним копіюванням даних та можливістю відновлення сесії тесту у разі збою мережі.
	10. Редагувати профіль користувача	
	11. Додати новий курс	
	12. Редагувати курс	
	13. Запланувати екзамен	
	14. Змінити деталі екзамену (дату, час, екзаменатор тощо)	

Вимоги (функціональні та нефункціональні) до ІКС екзаменаційно-тренінгового центу іноземних мов наведені в Таблиці 1.4.

Відповідно до проведеного аналізу предметної області, відомих ІКС згідно обраної тематики, а також визначених функціональних та нефункціональних вимог, сформулюємо основні задачі, які необхідно вирішити в рамках дипломної роботи магістра:

1. Розробка формальної моделі ІКС екзаменаційно-тренінгового центру іноземних мов з урахуванням головних процесів та учасників.

2. Декомпозиція системи у відповідності до структурного підходу з визначенням вхідних та вихідних потоків даних, параметрів, а також керуючих сигналів.

3. Розробка моделі надійності системи на основі резервування найбільш вразливих програмних компонентів.

4. Розробка моделі варіантів використання ІКС на основі функціональних вимог Таблиці 1.4.

5. Розробка алгоритмів та моделювання інфокомунікаційних процесів взаємодії акторів та системи.

6. Розробка моделі даних та практична реалізація бази даних (БД).

7. Обгрунтування вибору набору технологій до практичної реалізації ІКС.

8. Програмна реалізація системи згідно триступеневої архітектури з дотриманням нефункціональних вимог Таблиці 1.4. Реалізація моделі даних та компонентів взаємодії з БД, компонентів автоматизації інфокомунікаційних процесів системи та елементів інтерфейсу.

2 Моделі та методи інфокомунікаційної системи екзаменаційно-тренінгового центру іноземних мов

2.1 формальна модель ІКС екзаменаційно-тренінгового центру іноземних мов

Проведемо формалізацію системи згідно методології структурного аналізу та дизайну (Structured analysis and design technique, SADT) [23] та загальних принципів моделювання систем, зокрема, з використанням методу «чорного ящика» [24].

Представимо ІКС I_S у вигляді трійки:

$$I_S = \{X, Y, Z\}, \quad (2.1)$$

де X – множина вхідних потоків даних (Inputs);
Y – множина вихідних потоків (Outputs);
Z – параметри системи, що в свою чергу згідно методології SADT поділяють на сигнали контролю (Controls – C) та механізми виконання (Mechanisms – M):

$$Z = \{C, M\}. \quad (2.2)$$

Схематично система може бути представлена у спрощеному вигляді, так як це показано на Рисунку 2.1.

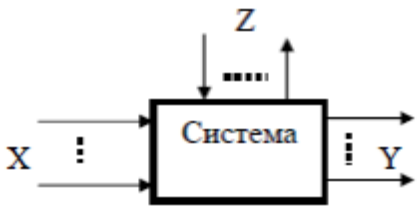


Рисунок 2.1 – Модель системи за методом «чорного ящика»

Таблиця 2.1 – Входи (Inputs) контекстної моделі A-0

Назва	Приклад даних	Опис
1	2	3
Облікові дані автентифікації користувача	Ім'я користувача, пароль	Дані для входу та початку сеансу.
Запити на реєстрацію	Ідентифікатори предмета та студента	Надходять від студентів, які реєструються на предмети.
Запити на реєстрацію на екзамен	Ідентифікатори екзамену та екзаменованого, бажана дата.	Надходить від екзаменованого.
Запити на відрахування або скасування	Ідентифікатор предмета або ідентифікатор іспиту для видалення.	З обгрунтуванням причини.

Таблиця 2.2 – Виходи (Outputs) контекстної моделі A-0

Назва	Приклад даних	Опис
1	2	3
Підтвердження реєстрації	Повідомлення про успішне завершення, оновлений список предметів для користувача	Передача користувачу облікових даних через електронну пошту.
Підтвердження реєстрації/перенесення/скасування іспиту	Оновлений статус іспиту, новий розклад	Із внесенням змін до переліку задач в календарі.
Результати запитів	Список предметів, іспитів, зареєстрованих предметів або зареєстрованих іспитів	Виведення у вигляді таблиць.

Таблиця 2.3 – Виходи (Outputs) контекстної моделі A-0

Назва	Приклад даних	Опис
1	2	3
Політики автентифікації	Доступ на основі ролей: користувач/студент може лише реєструватися, Екзаменований – для іспитів, адміністратор – для управління.	На розсуд адміністратора інфраструктури.
Правила перевірки	Передумови для предметів, критерії відповідності іспитам, перевірки дати і часу.	Погоджується із навчально-методичним відділом.

Рисунок 2.2 – Контекстна діаграма моделі системи

2.2 Декомпозиція моделі системи та розробка моделі надійності програмних компонентів

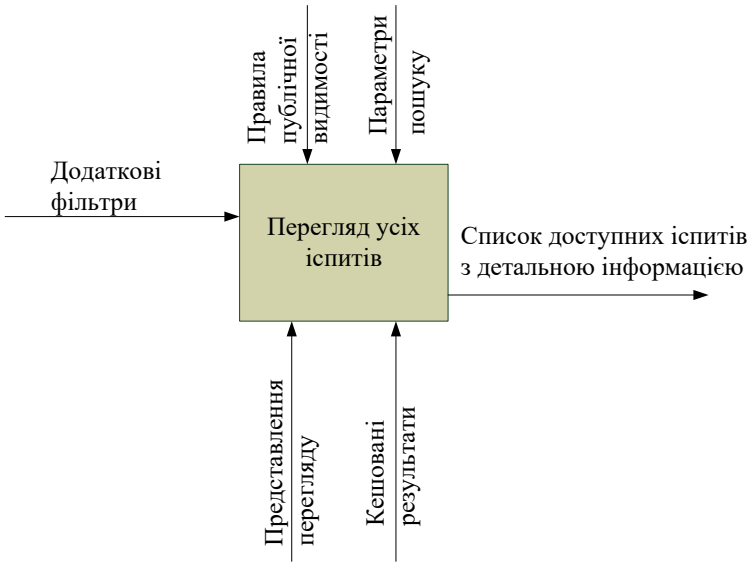


Рисунок 2.3 – Активність «Перегляд усіх іспитів»

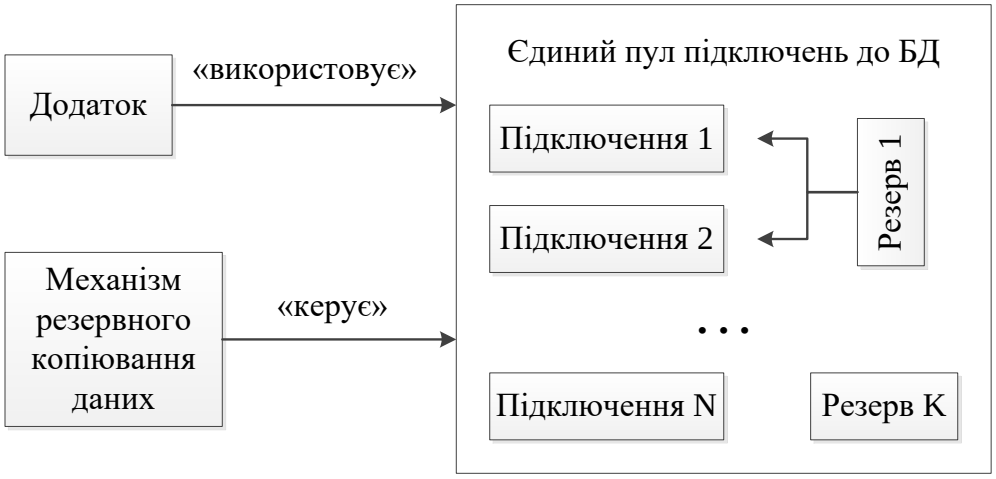


Рисунок 2.4 – Структура програмного модуля із резервуванням підключень до БД

Таблиця 2.5 – Параметри надійності елементів системи

Параметр № Елемента	1	2	3
Назва	Підключ. 1	Підключ. 1	Резерв 1
Середній час напрацювання на відмові (Mean Time Between Failures, MTBF), години	12	8	8
Середній час відновлення (Mean Time To Recovery, MTTR), хвилини	10	10	5
Середня інтенсивність відмови λ_i, год$^{-1}$	0,083	0,125	0,125
Середня інтенсивність відновлення μ_i, год$^{-1}$	6	6	12

На Рисунку 2.5 стани, що відповідають справності усіх елементів розміщені скраю зліва рисунку, наприклад «01» означає, що перший елемент знаходиться в резерві, а працюють другий і третій елементи і т.д. Стани, які позначені прямокутниками, відповідають відмовам.

$$\boldsymbol{AP} = \boldsymbol{B}, \qquad (2.3)$$

де $\boldsymbol{A}$ – матриця коефіцієнтів системи розміром (m×n), що визначається параметрами λ_i та μ_i (див. Таблицю 2.5);

$\boldsymbol{P}$ – вектор довжиною n ймовірностей перебування системи в кожному з можливий станів;

$\boldsymbol{B}$ – вектор довжиною m вільних членів правої частини лінійного рівняння.

Оскільки ймовірності $p_k \in \boldsymbol{P}$, $\overline{k = 1 \dots n}$ становлять повну групу, їх сума повинна дорівнювати одиниці:

$$\sum_{k=1}^{n} p_k = 1. \qquad (2.4)$$

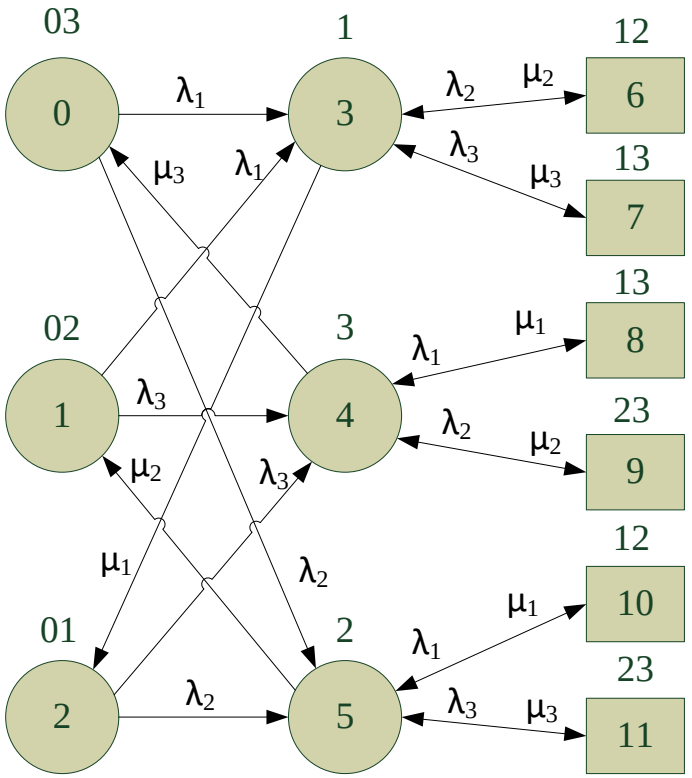


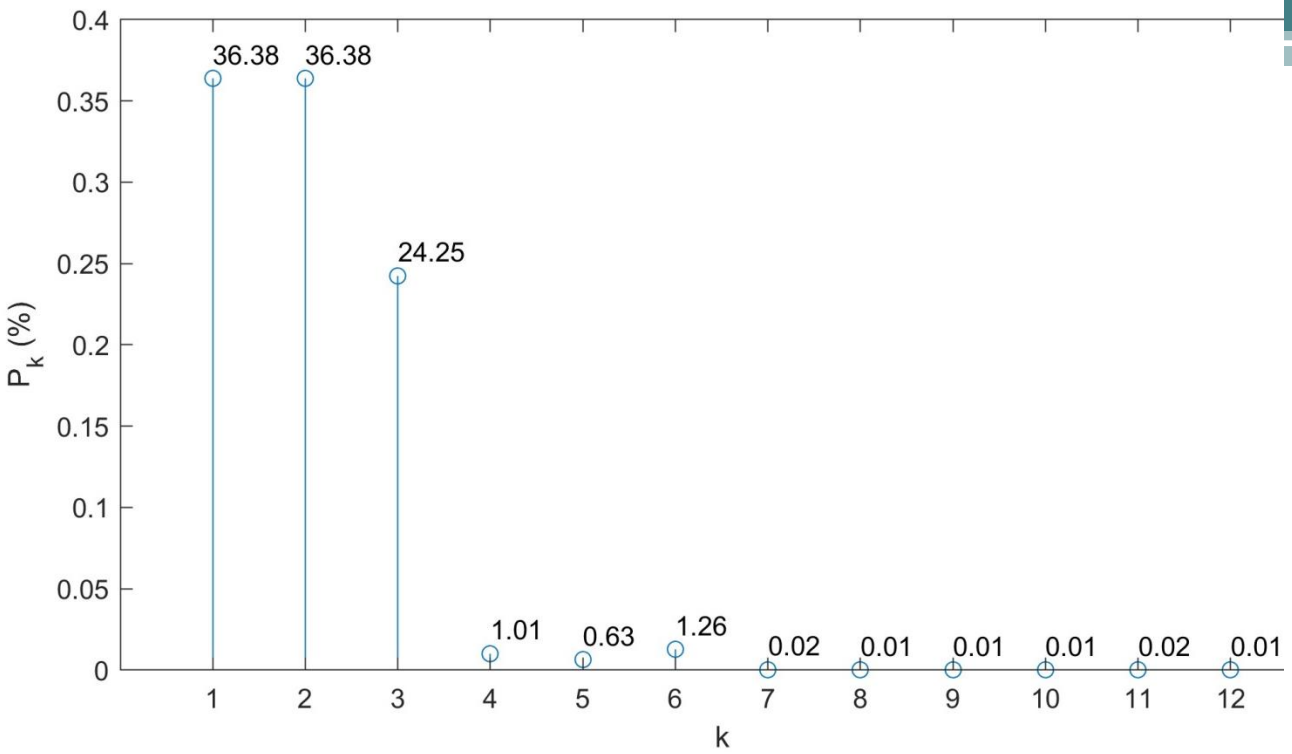
Рисунок 2.5 – Граф станів системи

Коефіцієнти матриці **A** параметрів системи (2.3) наведені в таблиці 2.6.

Таблиця 2.6 – Коефіцієнти матриці **A** рівняння (2.3)

m n	1	2	3	4	5	6	7	8
1	$-(\lambda_1+\lambda_2)$	0	0	0	μ_3	0	0	0
2	0	$-(\lambda_1+\lambda_3)$	0	0	0	μ_2	0	0
3	0	0	$-(\lambda_2+\lambda_3)$	μ_1	0	0	0	0
4	λ_1	λ_1	0	$-(\mu_1+\lambda_2+\lambda_3)$	0	0	μ_2	0
5	0	λ_3	λ_3	0	$-(\mu_3+\lambda_1+\lambda_2)$	0	0	μ_1
6	λ_2	0	λ_2	0	0	$-(\mu_3+\lambda_1+\lambda_3)$	0	0
7	0	0	0	λ_2	0	0	$-\mu_2$	0
8	0	0	0	λ_3	0	0	0	$-\mu_3$
9	0	0	0	0	0	λ_1	0	0
10	0	0	0	0	λ_2	0	0	0
11	0	0	0	0	0	λ_1	0	0
12	0	0	0	0	0	0	λ_3	0

m n	9	10	11	12
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0
9	$-\mu_1$	0	0	0
10	0	$-\mu_2$	0	0
11	0	0	$-\mu_1$	0
12	0	0	0	$-\mu_3$



Розв'язавши рівняння (2.3) з урахуванням (2.4), отримуємо вектор стаціонарних ймовірностей **P**, значення якого наведені на Рис. 2.6.

Відповідно до моделі на Рис. 2.5, робочими є стани 0-5, отже, коефіцієнт готовності становить суму відповідний ймовірностей:

$K_s = \sum_{k=0}^{5} P_k = 99{,}92\ \%$. (2.5)

Характеристика потоку відмов:

$$\omega = (\lambda_2 + \lambda_3)p_4 + (\lambda_1 + \lambda_2)p_5 + (\lambda_1 + \lambda_3)p_6 = 0{,}0893\ (\text{год.}^{-1}) \qquad (2.6)$$

Час напрацювання на відмову:

$$T_f = \frac{K_r}{\omega} = 154\ \text{год. 20 хв.} \qquad (2.7)$$

Середній час відновлення:

$$T_r = \frac{1-K_r}{\omega} = 7\ \text{хв. 48 сек.} \qquad (2.8)$$

Отже, запропонований метод дозволяє збільшити середній час напрацювання на відмову до 154 годин (більше 6 днів), що понад у 12 разів більше за відповідний показник для лише одного елементу із базовими показниками надійності, з середнім часом відновлення до 7 хвилин.

3 Розробка варіантів використання та бази даних
3.1 Розробка моделі варіантів використання системи

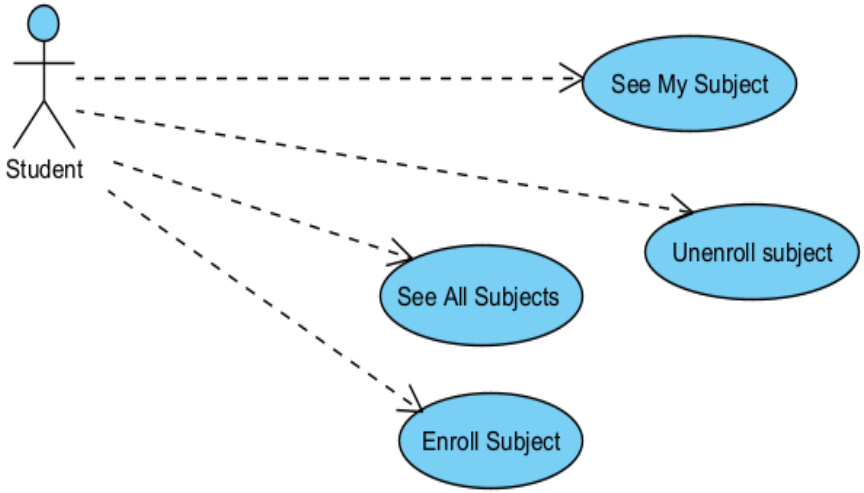


Рисунок 3.1 – Діаграма варіантів використання актора Student

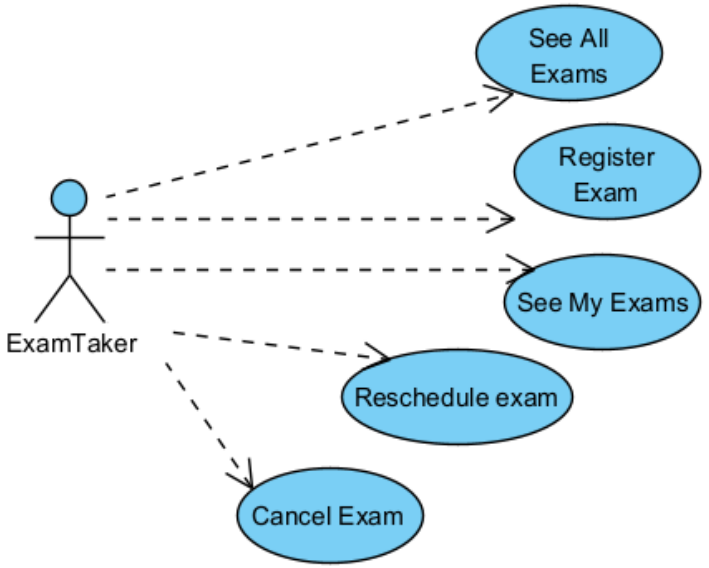
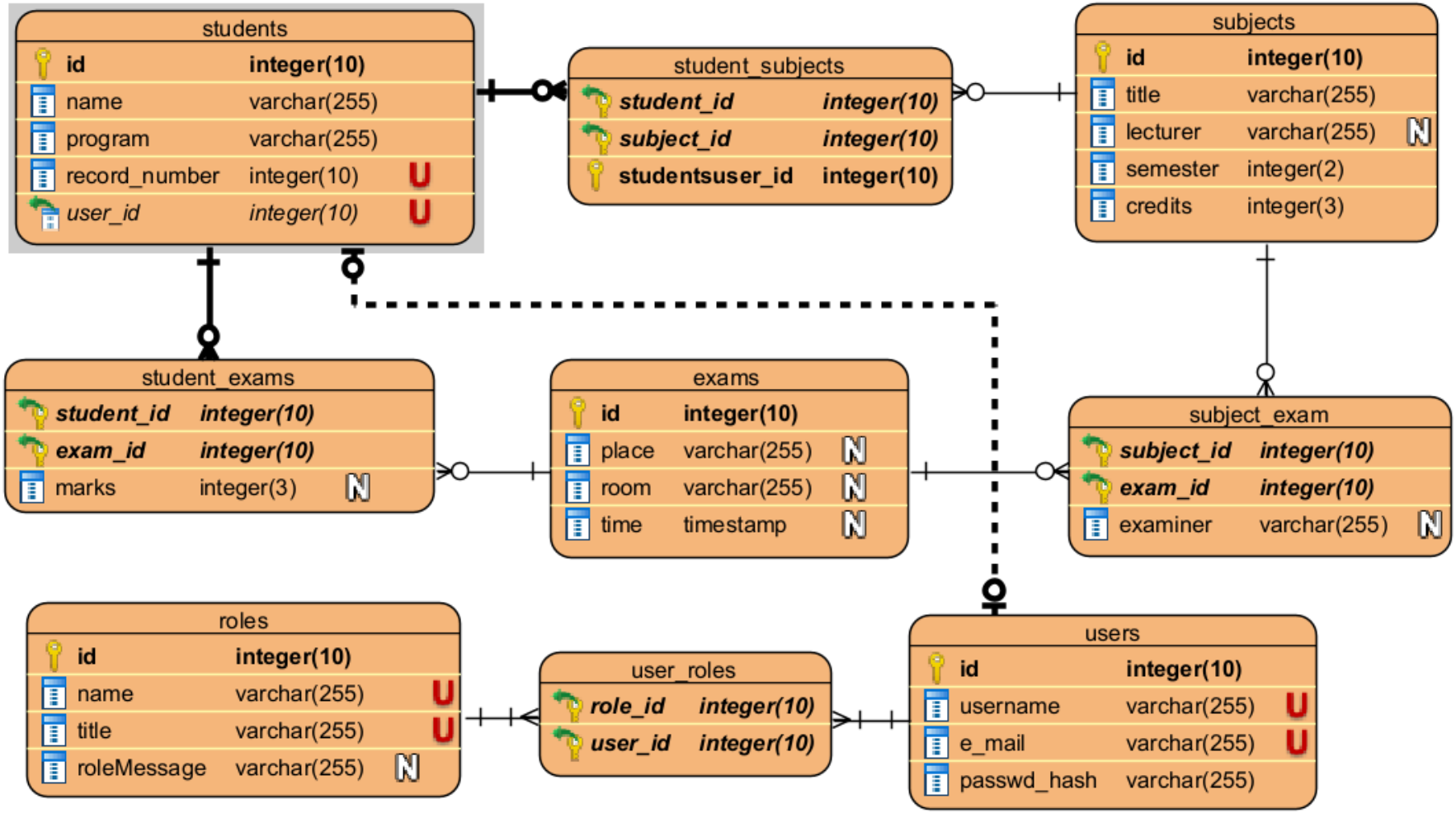


Рисунок 3.2 – Діаграма варіантів використання актора ExamTaker

3.2 Розробка схеми даних



Створення таблиць та заповнення їх даними в PostgreSQL

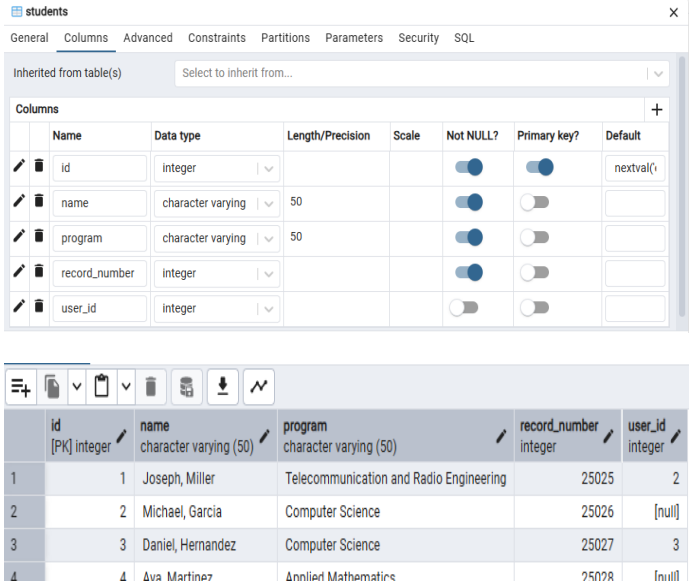


Рисунок В.3 – Створення таблиці «subjects» та її заповнення тестовими даними

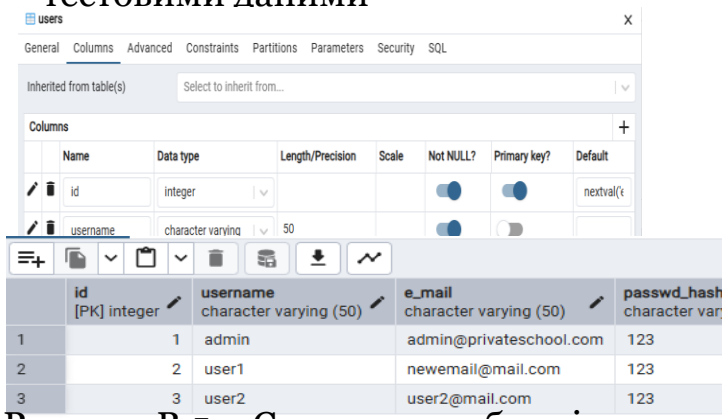


Рисунок В.5 – Створення таблиці «users» та її заповнення тестовими даними

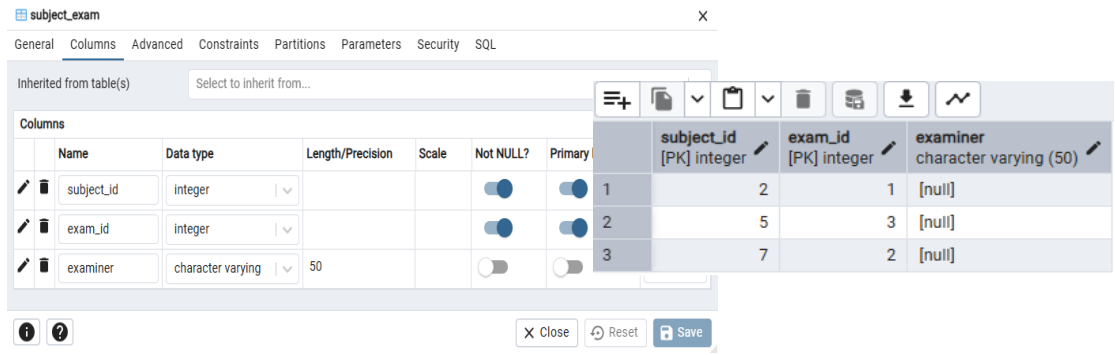


Рисунок В.2 – Створення таблиці «subject_exam» та її заповнення тестовими даними

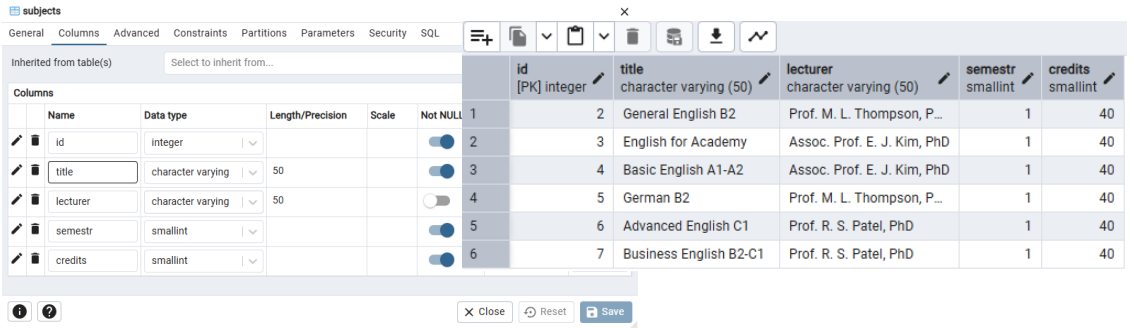


Рисунок В.3 – Створення таблиці «subjects» та її заповнення тестовими даними

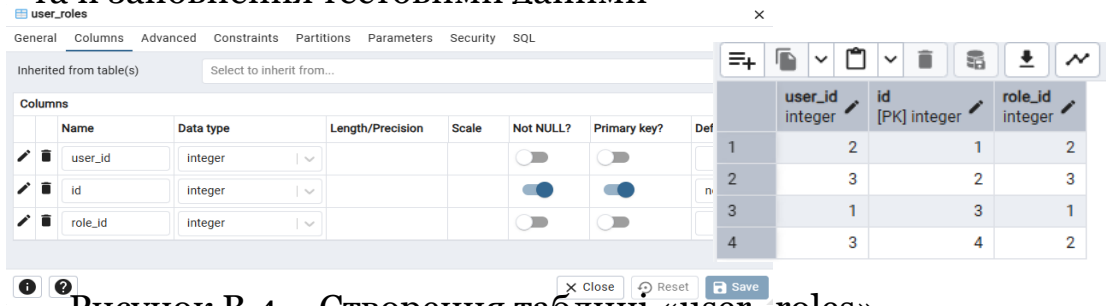


Рисунок В.4 – Створення таблиці «user_roles» та її заповнення тестовими даними

4 Програмна реалізація

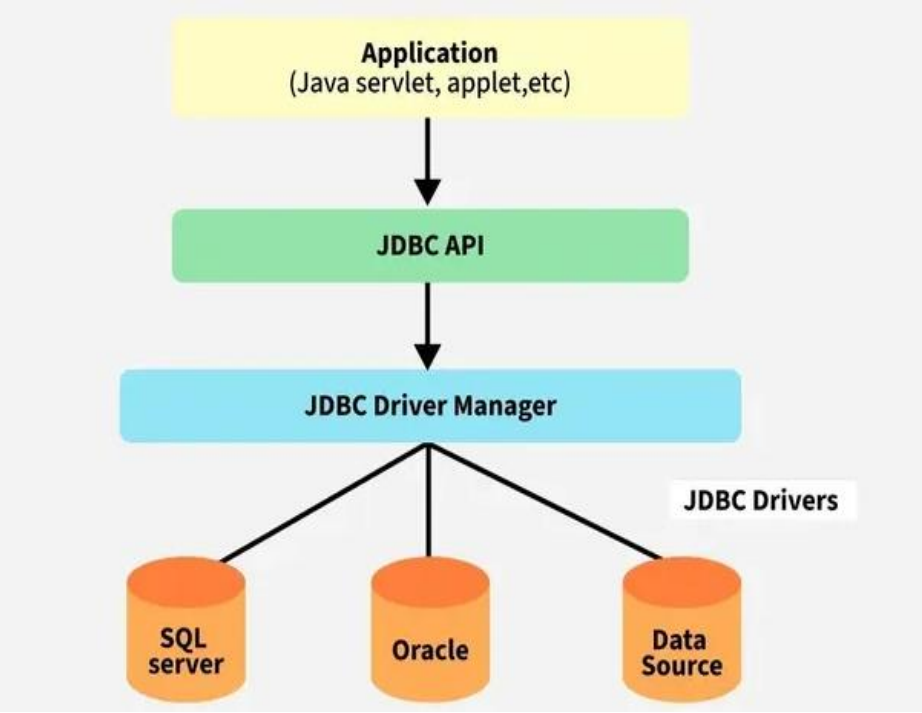


Рисунок 4.1 – Архітектура компонентів взаємодії з БД

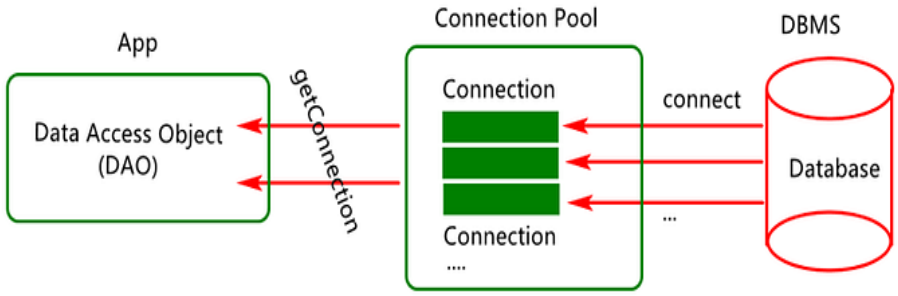
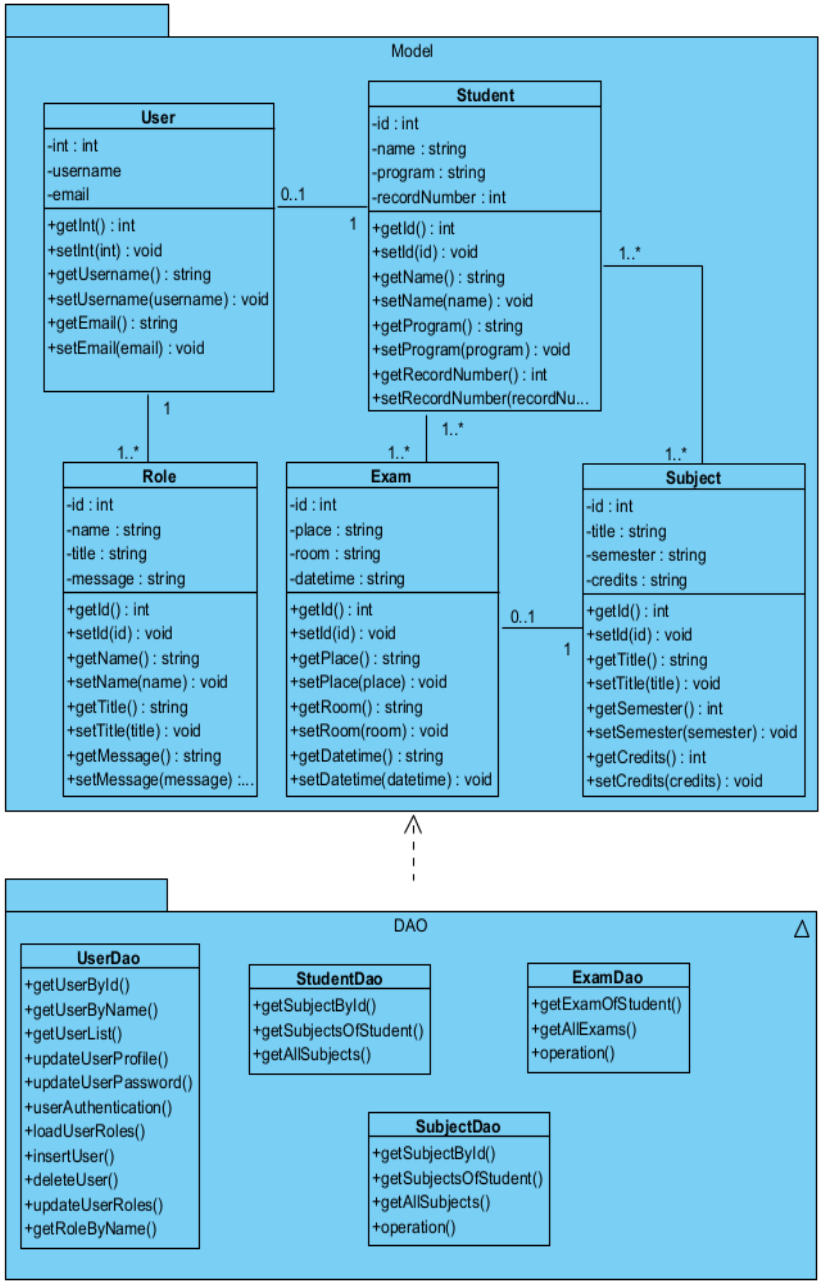


Рисунок 4.2 – Взаємодія з СУБД через пул підключень



ДОДАТОК Г – Архітектура системи

4.2.3 Реалізація елементів інтерфейсу користувача

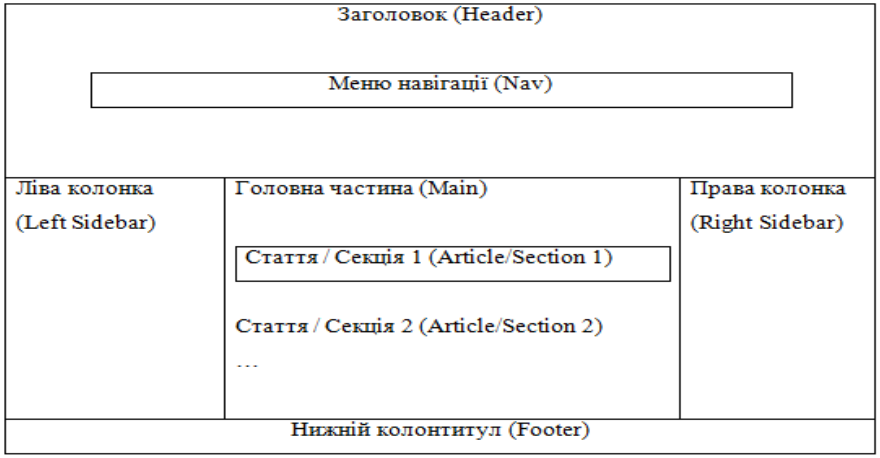


Рисунок 4.3 – Структура шаблону сторінок сайту

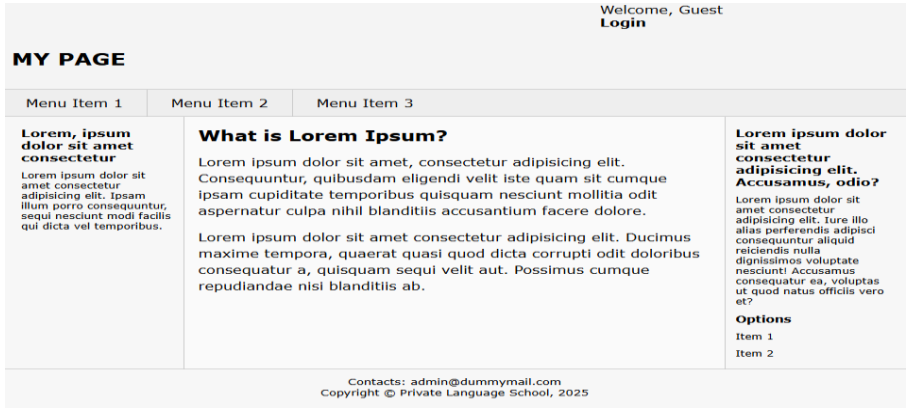


Рисунок 4.4 – Шаблон сторінки сайту системи

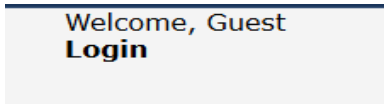


Рисунок 4.5 – Вітання гостя системи із запрошенням авторизації

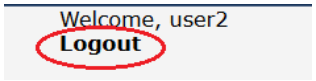


Рисунок 4.7 – Вітання авторизованого користувача та кнопка виходу із системи

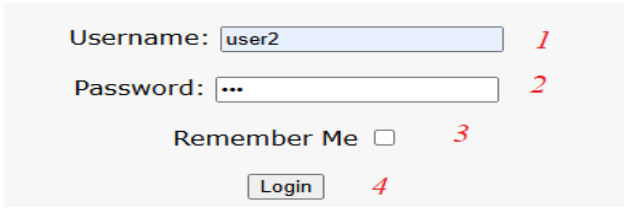


Рисунок 4.6 – Форма авторизації користувача

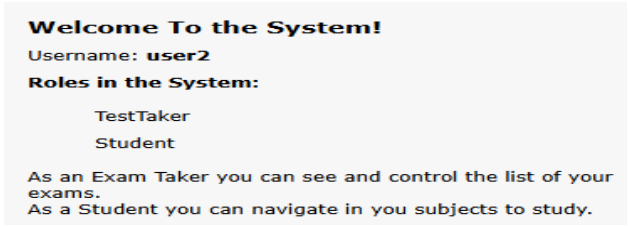


Рисунок 4.8 – Ліва бічна панель із інформацією про користувача та ролі

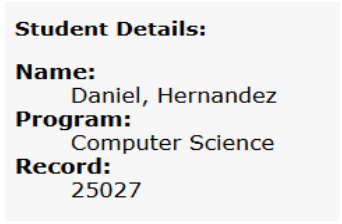


Рисунок 4.9 – Права бічна панель

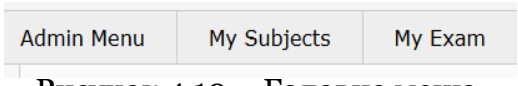


Рисунок 4.10 – Головне меню

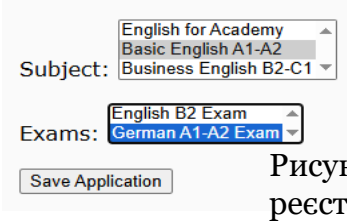
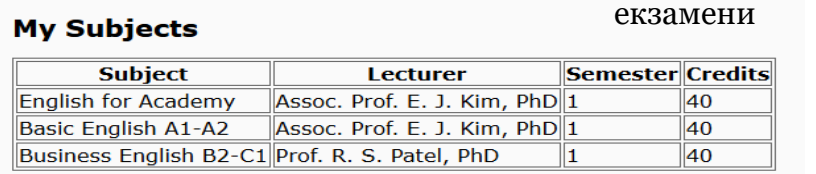
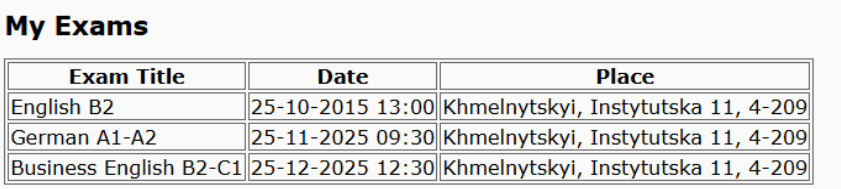


Рисунок 4.11 – Форма реєстрації на курси та екзамени



My Subjects

Subject	Lecturer	Semester	Credits
English for Academy	Assoc. Prof. E. J. Kim, PhD	1	40
Basic English A1-A2	Assoc. Prof. E. J. Kim, PhD	1	40
Business English B2-C1	Prof. R. S. Patel, PhD	1	40



My Exams

Exam Title	Date	Place
English B2	25-10-2015 13:00	Khmelnytskyi, Instytutska 11, 4-209
German A1-A2	25-11-2025 09:30	Khmelnytskyi, Instytutska 11, 4-209
Business English B2-C1	25-12-2025 12:30	Khmelnytskyi, Instytutska 11, 4-209

Рисунок 4.12 – Список навчальних дисциплін та екзаменів

4.3 Розгортання програмних компонентів на обчислювальних вузлах системи

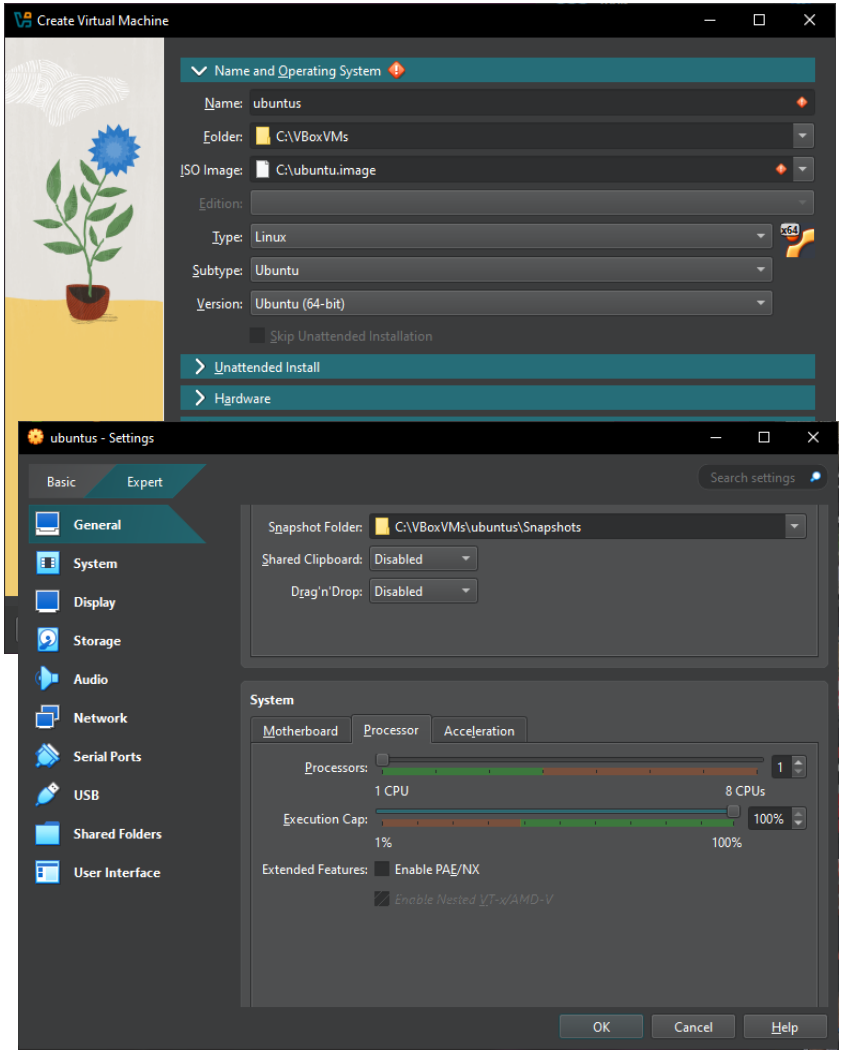


Рисунок 4.13 – Створення та налаштування віртуальної машини у VirtualBox

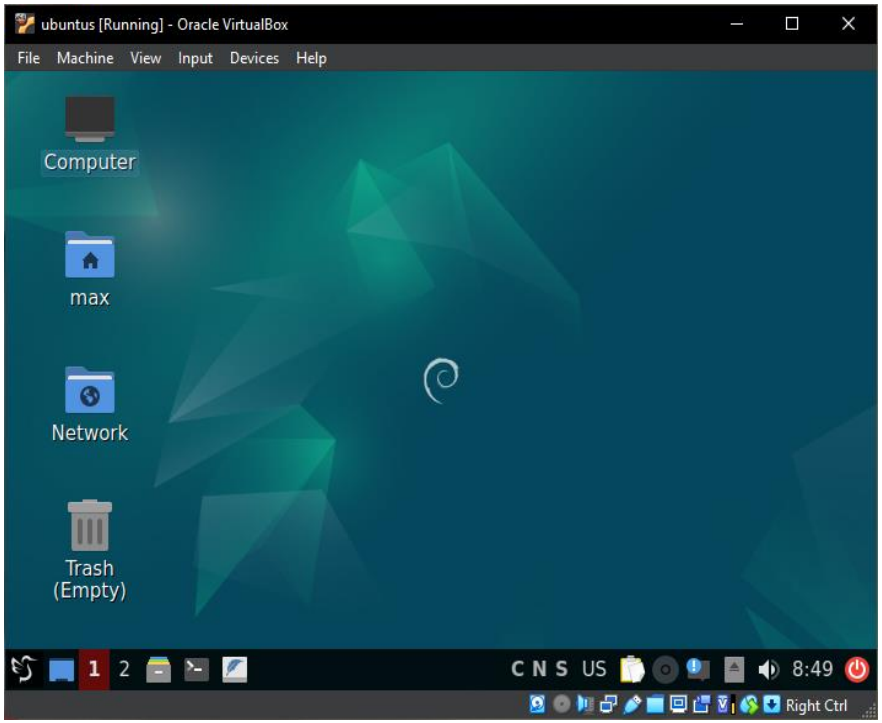


Рисунок 4.14 – Інтерфейс Ubuntu з робочим столом LXQT

ВИСНОВКИ

1. Кваліфікаційна робота магістра за спеціальність 172 – Електронні комунікації та радіотехніка виконана у повному обсязі у відповідності до поставлених завдань. Передумовами для роботи є ретельний аналіз предметної області, а саме освітній процесів, що виникають під час комунікації учасників та користувачів інфокомунікаційними системами освіти на прикладі екзаменаційно-тренінгового центу іноземних мов. В результаті огляду існуючих програмних систем даної тематики та аналізу літературних джерел було сформульовано технічне завдання та науково-технічні проблеми, що були вирішені в ході виконання.

2. В якості предмету дослідження в роботі було зафіксовано програмне забезпечення інфокомунікаційної системи екзаменаційно-тренінгового центру іноземних мов. В процесі реалізації останнього була вирішена актуальна науково-практична проблема – підвищено надійність критично важливий програмних компонентів системи, а саме модулів доступу до бази даних шляхом введення прийнятного рівня надлишковості та задіяння механізму резервного копіювання даних.

3. Розроблена модель інфокомунікаційних процесів з детальною декомпозицією структурних елементів, що згодом слугувала вихідним джерелом вимог для програмування компонентів системи.

4. Програмна реалізація виконана у вигляді веб-додатку засобами мови програмування високого рівня Java та набору специфікацій корпоративного програмного забезпечення Java EE. Реалізовано схему реляційної бази даних, алгоритми бізнес-логіки додатку та елементи інтерфейсу користувачів системи. В результаті було отримано повнофункціональну версію програмного забезпечення, яка може бути проваджена в освітній процес екзаменаційно-тренінгового центу іноземних мов для безпосереднього використання, а також для глибокого тестування, усунення можливих помилок та розширення функціоналу системи в рамках життєвого циклу програмного забезпечення.

ДЯКУЮ ЗА УВАГУ!

Міністерство освіти і науки України
Хмельницький національний університет

ЗБІРНИК НАУКОВИХ ПРАЦЬ

за матеріалами XVII Всеукраїнської науково-практичної конференції
«Актуальні проблеми комп'ютерних наук АПКН-2025»

14-15 листопада 2025

Хмельницький 2025

УДК 004:37:001:62

Збірник наукових праць за матеріалами XVII Всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2025». Хмельницький. 2025. 500с.

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, друкуються в авторській редакції та наведені в алфавітному порядку прізвищ авторів. При макетуванні можливі незначні зміни компоновки контенту авторських оригіналів. Відповідальність за якість та зміст публікацій несе автор.

Участь у конференції та складові всіх її етапів (розгляд праць, перевірка на плагіат, макетування, публікація збірника наукових праць та видача сертифікатів) є безкоштовними для всіх учасників. 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 - 2025

XVII Всеукраїнська науково-практична конференція

Метою конференції є висвітлення актуальних проблем комп'ютерних наук, інформатики та інформаційних технологій.

Робочі мови конференції:

українська, англійська

СЕКЦІЇ КОНФЕРЕНЦІЇ:

1. Комп'ютерні науки, штучний інтелект 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.

СПИСОК ОРГАНІЗАЦІЙ, ПРЕДСТАВНИКИ ЯКИХ БРАЛИ УЧАСТЬ У РОБОТІ КОНФЕРЕНЦІЇ:

Донбаська державна машинобудівна академія
Інститут кібернетики імені В. М. Глушкова НАН України
Кам'янський енергетичний фаховий коледж
Київський національний університет імені Т. Г. Шевченка
Національного аерокосмічного університету імені М. Є. Жуковського «Харківський авіаційний інститут»
Національний технічний університет «Харківський політехнічний інститут»
Сумський державний університет
Харківський національний університет радіоелектроніки
Хмельницький національний університет
Хмельницький фаховий економіко-технологічний коледж УЕП

ОРГКОМІТЕТ КОНФЕРЕНЦІЇ:

СИНЮК О. М. – голова оргкомітету, проректор Хмельницького національного університету з наукової роботи, доктор технічних наук, професор.

ГОВОРУЩЕНКО Т. О. – заступник голови оргкомітету, декан факультету інформаційних технологій Хмельницького національного університету, доктор технічних наук, професор.

БАРМАК О. В. – заступник голови оргкомітету, завідувач кафедри комп'ютерних наук Хмельницького національного університету, доктор технічних наук, професор.

САВЕНКО О. С. – професор кафедри комп'ютерної інженерії та інформаційних систем Хмельницького національного університету, доктор технічних наук, професор.

ВИСОЦЬКА О. В. – завідувач кафедри радіоелектронних та біомедичних комп'ютеризованих засобів і технологій Національного аерокосмічного університету ім. М. Є. Жуковського «Харківський авіаційний інститут», доктор технічних наук, 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, доцент кафедри комп'ютерних наук Хмельницького національного університету, кандидат технічних наук, доцент.

МОЛЧАНОВА М. О. – секретар конференції, старший викладач кафедри комп'ютерних наук Хмельницького національного університету, доктор філософії з комп'ютерних наук.

КОНТАКТНА ІНФОРМАЦІЯ:

e-mail для листування: *apkt.khnu@gmail.com*

УДК 004.4

Коржан М.М., Підченко С.К.

Хмельницький національний університет

ІНФОКОМУНІКАЦІЙНА СИСТЕМА ЕКЗАМЕНАЦІЙНО-ТРЕНІНГОВОГО ЦЕНТРУ ІНОЗЕМНИХ МОВ

Розглянуто методи захисту інформаційних систем від загрозливих програм на основі механізмів контролю доступу та розмежувальних політик. Запропоновано підхід до побудови багаторівневої системи захисту, що поєднує дискреційний, мандатний та рольовий контроль доступу з динамічними політиками безпеки. Результати дослідження демонструють, що правильно налаштовані механізми контролю доступу можуть суттєво знизити ймовірність успішного зараження системи та обмежити масштаби можливих наслідків.

Methods of protecting information systems from malicious programs based on access control mechanisms and separation policies are considered. An approach to building a multi-level protection system is proposed, combining discretionary, mandatory, and role-based access control with dynamic security policies. The results of the study demonstrate that properly configured access control mechanisms can significantly reduce the likelihood of successful system infection and limit the scale of possible consequences.

Забезпечення функціонування інфокомунікаційних систем (ІКС), в тому числі освітніх, покладено на всебічне використання програмних, апаратних та мережевих засобів [1-2]. Незалежно від масштабу та форми власності закладу освіти (ЗО), такі технології покращують якість надання послуг здобувачам освіти (студентам) за рахунок автоматизації процесів.

Відповідно до досліджень [3-4], використання ІКС для вивчення іноземних мов сприяє підвищенню мотивації, ефективності взаємодії учасників освітнього процесу, а також дозволяє моделювати реальні комунікаційні ситуації в штучному середовищі.

Такі інструменти особливо корисні для вивчення іноземних мов, де акцент робиться на розмовній практиці, аудіюванні та культурному обміні, а також для атестації знань студентів через, автоматизовану оцінку та персоналізований зворотний зв'язок.

Практика свідчить, що застосування ІКС в освіті сприяє розвитку від монотонного заучування до появи інтегральних методів, що суттєво підвищують якість освітніх послуг.

Природною є класифікація засобів підтримки ІКС на апаратні і програмні [3].

Представимо ІКС I_S у вигляді трійки:

$$Is = \{X,Y,Z\}, \quad (1)$$

де X – множина вхідних потоків даних (Inputs);

Y – множина вихідних потоків (Outputs);

Z – параметри системи, що в свою чергу згідно методології SADT поділяють на сигнали контролю (Controls – C) та механізми виконання (Mechanisms – M):

$$Z = \{C,M\}. \quad (2)$$

Схематично система може бути представлена у спрощеному вигляді, на основі базової моделі «чорного ящика» з подальшою розробкою контекстної діаграми (A-0), яка описує всю систему у вигляді єдиного блоку активності Activity A-0: «Керування реєстрацією на курси іноземних мов та екзамени».

Ця активність охоплює всі взаємодії з користувачами, обробку даних та адміністративні функції через веб-додаток на основі триступеневої архітектури (Model-View-Controller, MVC), БД (доступ до якої здійснюється через драйвер з пулом підключень) та асинхронні обробники (для паралельного виконання).

З огляду на важливість забезпечення безвідмовного доступу ПЗ до БД, розглянемо моделі резервування програмних компонентів для підключення програного додатку до БД.

Відома модель мікросервісів [5], згідно якої архітектура ПЗ розділяється на множину окремих компонентів (сервісів), що функціонують паралельно в єдиній екосистемі. Використання таких сервісів сприяє підвищенню надійності системи та спрощення структури окремих сервісів, які виконують кожен свою просту функцію (спрощено).

Представимо механізм доступу до БД у вигляді програмного модуля з багатосерсвісною архітектурою, що функціонує з метою, наприклад, використання одним додатком декількох БД, розташованих фізично на різних серверних платформах.

Об'єктно-орієнтована модель ПЗ дозволяє абстрагуватись від конкретної системи управління базою даних (СУБД), її драйвера та параметрів підключення, інкапсулюючи ці дані у вигляді об'єкта підключення.

На Рисунку 1 показана структура вищеназваних програмних компонентів, де головним елементом є пул підключень до БД, який складається із N об'єктів підключень та K резервних об'єктів. Метод резервування в загальному випадку може бути довільним, на розсуд архітектора або розробника ПЗ, однак, в рамках даної роботи обмежимось доволі простим випадком відновлювальної резервованої системи з резервом кратності 1/2. Така система складається із трьох елементів різної надійності.

Як видно із Рисунку 1, концепція використання пулу підключень до БД полягає у постійному використання підключень для нормальної роботи

програмного додатку, а також у забезпеченні резервного копіювання даних між основними та резервними сховищами для можливості найшвидшого відновлення у випадку відмови.

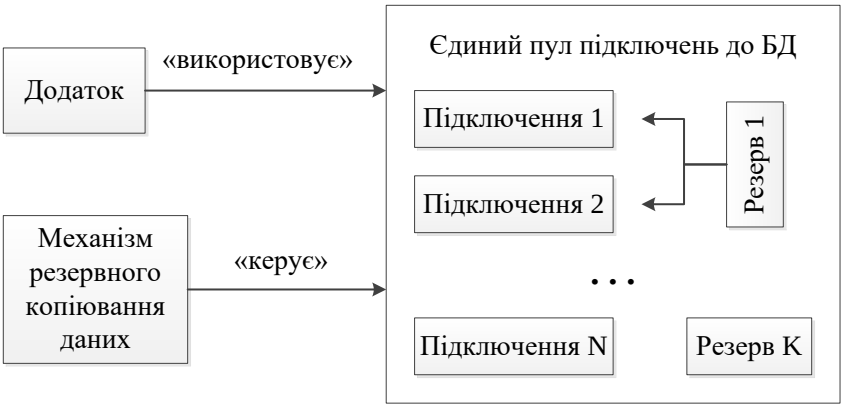


Рисунок 1 – Структура програмного модуля

Виконаємо моделювання такої системи в умовах високого навантаження невеликим часом напрацювання відмови програмних компонентів. В Таблиці 1 подані відповідні параметри надійності.

Таблиця 1 – Параметри надійності елементів системи

Параметр / **№ Елемента**	**1**	**2**	**3**
Назва	Підключ. 1	Підключ. 1	Резерв 1
Середній час напрацювання на відмові (Mean Time Between Failures, MTBF), години	12	8	8
Середній час відновлення (Mean Time To Recovery, MTTR), хвилини	10	10	5
Середня інтенсивність відмови, λ_i, год$^{-1}$	0,083	0,125	0,125
Середня інтенсивність відновлення μ_i, год$^{-1}$	6	6	12

Над вершинами графу 6-11 позначені номери елементів, які відмовили. Так, до прикладу, «1» означає випадок відмови першого елементу за роботи решти двох. Після відновлення неробочого елементу система переходить у стан справності усіх елементів, однак штатно працюватимуть перший та третій елементи.

На Рисунку 2 стани, що відповідають справності усіх елементів розміщені скраю зліва рисунку, наприклад «01» означає, що перший елемент знаходиться в резерві, а працюють другий і третій елементи і т.д. Стани, які позначені прямокутниками, відповідають відмовам.

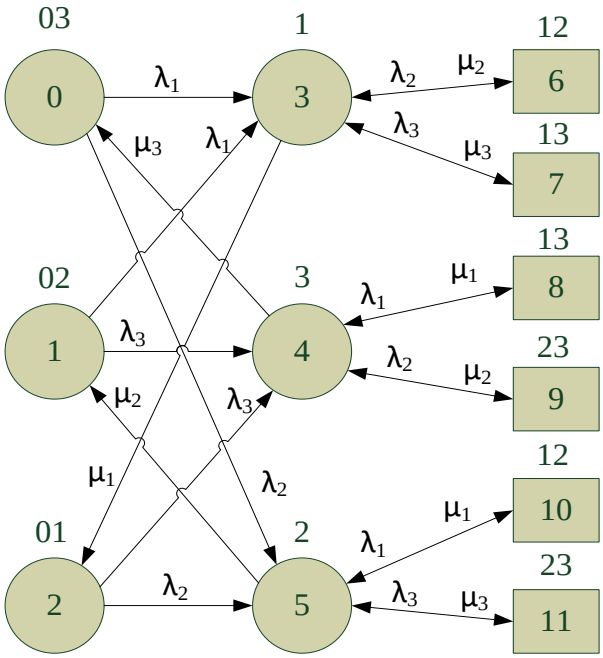


Рисунок 2 – Граф станів системи

Провівши серію імітаційних моделювань, отримуємо вектор стаціонарних ймовірностей P, значення якого наведені на Рисунку 3.

Відповідно до моделі на Рисунку 2, робочими є стани 0-5, отже, коефіцієнт готовності становить суму відповідний ймовірностей: $K_s = \sum_{k=0}^{5} P_k = 99{,}92\ \%$. З середнім часом напрацювання на відмові 154 год. 20 хв та часом відновлення 7 хв.50 сек.

Функціональний граф станів системи показано на Рисунку 3. Стани системи показані вершинами графу 0-11.

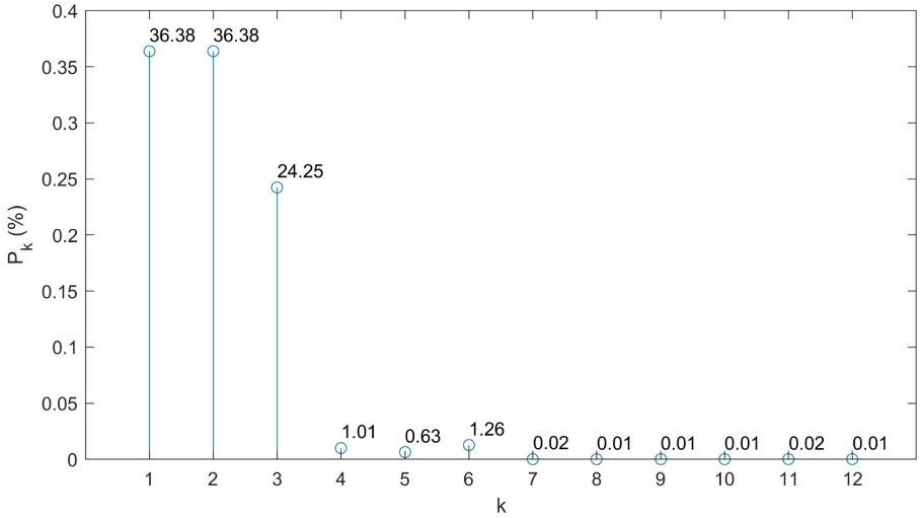


Рисунок 3 – Розраховані стаціонарні ймовірності перебування системи в k-му стані

Отже, запропонований метод дозволяє збільшити середній час напрацювання на відмову до 154 годин (більше 6 днів), що понад у 12 разів більше за відповідний показник для лише одного елементу із базовими показниками надійності, з середнім часом відновлення не більше 8 хвилин.

Перелік посилань

1. Довбиш А. С. Вступ до інформаційного аналізу і синтезу інфокомунікаційних систем : [монографія] / А. С. Довбиш. – Київ, 2016.
2. Осадча Ю. В. Аналіз програмного та інфокомунікаційного забезпечення освітнього процесу : [монографія] / Ю. В. Осадча. – Київ, 2022.
3. Classification of Technology-based Language Learning Resources according to Pedagogical and Functional Criteria / R. Seiz Ortiz, M. L. Carrió Pastor // Revista Electrónica de Lingüística Aplicada. – 2025. – Vol. 23, Iss. 1. – [Електронний ресурс]. – Режим доступу: https://matrix.aesla.org.es/RAEL/article/view/676. – DOI: https://doi.org/10.58859/rael.v23i1.676.
4. Information and communication technologies in foreign language learning [Електронний ресурс]. – Режим доступу: https://eprints.soton.ac.uk/192773/. – Дата звернення: 08.10.2025.
5. Microservice Architecture pattern / C. Richardson // Microservices.io. – [Електронний ресурс]. – Режим доступу: https://microservices.io/patterns/microservices.html. – Дата звернення: 08.10.2025.

АКТУАЛЬНІ ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК 2025

ЗБІРНИК НАУКОВИХ ПРАЦЬ

Комп'ютерна верстка: **Мазурець О. В.**

Підписано до друку 15.11.2025.
Версія друку «APKN2025_CorpusPaper v5mod93 Final».

E-mail: *apkt.khnu@gmail.com*
ХНУ. м. Хмельницький, вул. Інститутська,11.

Завідувачу кафедри
телекомунікацій, медійних та
інтелектуальних технологій (ТМІТ)
Сергію ПІДЧЕНКУ
студента 2 курсу, гр. ЕКРм-24-1
Миколи КОРЖАНА

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщена та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (StrikePlagiarism та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

26.11
дата

підпис

Коржан М.М.

Anti-Plagiarism (UA) v-16.675

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: UA, US, RU. **Помилок в документах: 22%**

ID: 251660 Назва: Інфокомунікаційна система екзаменаційно-тренінгового центру іноземних мов Додано в БД: 2025-12-04 Автора: Коржан Микола Миколайович Керівники: Підченко Сергій Костянтинович Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	129198	1016	1985 (2%)	28 (3%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Микола КОРЖАН ЕКРм-24-1

Співавтор:

Назва: Інфокомунікаційна система екзаменаційно- тренінгового центру іноземних мов

Науковий керівник: Сергій ПІДЧЕНКО, д.т.н., проф.

Підрозділ: Кафедра телекомунікацій, медійних та інтелектуальних технологій

Коефіцієнт подібності 1:5.5%

Коефіцієнт подібності 2:0.9%

Мікропробіли: 0

Заміна букв: 5

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-12-05 03:13:58.0

Після аналізу Звіту подібності констатую наступне:

☑ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

Дата 05.12.2025 експерт Живодар Олег Сергійович

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

ТЕЛЕКОМУНІКАЦІЙ, МЕДІЙНИХ ТА ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва: *Інфокомунікаційна система екзаменаційно-тренінгового центру іноземних мов*

Автор: **Коржан Микола Миколайович**

Освітня програма: **Телекомунікації, медійні технології та інтелектуальні мережі**

Рівень вищої освіти другий (магістерський) рівень

Спеціальність: **172 Телекомунікації та радіотехніка**

Науковий керівник: **д.т.н., проф. Підченко Сергій Костянтинович**

На основі аналізу кваліфікаційної роботи на дотримування вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	+
1.2	Виявлені запозичення не є академічним 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.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	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.	
2	Інші види порешень академічної доброчесності	

Підтвердження:

Виявленні запозичення не є плагіатом, так як розміщені у розділах, які не описують безпосередньо авторське дослідження (є власні терміни, визначення тощо), коефіцієнти подібності складають 5,5% та 0,9%, співпадіння мають посилання на приведений список літературних джерел.

«5» 12 2025 р.

Завідувач кафедри _____ Підпис Сергій ПІДЧЕНКО

Гарант освітньої програми _____ Підпис Сергій ПІДЧЕНКО

Керівник кваліфікаційної роботи _____ Підпис Сергій ПІДЧЕНКО

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ

Направляється студент Коржан Микола Миколайович на захист дипломного проєкту (роботи)
(прізвище, ім'я, по батькові)

за спеціальністю 172 - Електронні комунікації та радіотехніка

На тему: Інфокомунікаційна система екзаменаційно-тренінгового центру іноземних мов

Дипломний проєкт (робота), рецензія і довідка про перевірку на плагіат додаються.

Декан факультету _____ Світлана ГОВОРУЩЕНКО
(підпис) (ім'я, прізвище)

ДОВІДКА УСПІШНОСТІ

Коржан М.М. з 2024 по 2025 роки повністю виконав навчальний план спеціальності з таким розподілом оцінок за:
національною шкалою: відмінно 0,00 %, добре 12,50 %, задовільно 87,50 %.
шкалою ЄКТС: A 0,00 %, B 0,00 %, C 7,69 %, D 7,69 %, E 84,62 %.

Методист факультету _____ Тамара ЯСЮК
(підпис) (ім'я, прізвище)

ВИСНОВОК КЕРІВНИКА ДИПЛОМНОГО ПРОЄКТУ (РОБОТИ) ТА ОБГРУНТУВАННЯ ОЦІНКИ

Студент Коржан Микола Миколайович виконав дипломну магістерську роботу у повній відповідності до завдання. Робота пройшла апробацію на XVII всеукраїнській науково-практичній конференції «Актуальні проблеми комп'ютерних наук» АПКН-2025. Заслуговує на оцінку «добре».

Оцінка дипломного проєкту (роботи) добре

Керівник дипломного проєкту _____ Підченко С.К.
(підпис) (ім'я, прізвище)

"8" 12 2025 р.

ВИСНОВОК КАФЕДРИ ПРО ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ)

Дипломний проєкт (роботу) розглянуто. Студент Коржан М.М. допускається до захисту цього проєкту (роботи) в екзаменаційній комісії.

Завідувач кафедри ТМІТ
(назва)

_____ Підченко С.К.
(підпис, ім'я, прізвище)

"8" 12 2025 р.

РЕЦЕНЗІЯ

на магістерську дипломну роботу
студента групи ЕКРм-24-І Миколи КОРЖАНА
«Інфокомунікаційна система екзаменаційно-тренінгового центру іноземних мов»

Технології та засоби електронних комунікацій сприяють автоматизації процесів функціонування систем різного призначення, зокрема в освітній діяльності. В процесі вивчення іноземних мов та проведення іспитів виникають проблеми, що пов'язані із керуванням взаємодії учасників освітнього процесу, тому обрана тема роботи є актуальною, особливо для дистанційної форма навчання.

Метою роботи є підвищення надійності програмних компонентів інфокомунікаційної системи екзаменаційно-тренінгового центру іноземних мов.

Для досягнення поставленої мети в роботі вирішуються наступні основні завдання:

1. Розробка формальної моделі інфокомунікаційної системи екзаменаційно-тренінгового центру іноземних мов з урахуванням головних процесів та учасників.
2. Програмна реалізація системи згідно триступеневої архітектури з дотриманням нефункціональних вимог. Реалізація моделі даних та компонентів взаємодії з БД, компонентів автоматизації інфокомунікаційних процесів системи та елементів інтерфейсу.

Наукова-практична новизна роботи полягає у розробці прототипу програмного забезпечення інфокомунікаційної системи екзаменаційно-тренінгового центу іноземних мов на базі набору технологій Java EE, що дозволяє виконувати базові функції згідно технічного завдання. Додаток може бути використаний для тестування процесів функціонування відповідних систем з метою оптимізації бізнес-процесів освітньої діяльності..

За результатами роботи опубліковано тези доповіді у Збірнику наукових праць за матеріалами XVII Всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2025»: Хмельницький, 2025.

В цілому магістерська дипломна робота Миколи КОРЖАНА є актуальною у сфері сучасних технологій електронних комунікацій та радіотехніки, виконана на достатньому науково-технічному рівні та заслуговує оцінки «добре».

Рецензент:
зав. кафедри кібербезпеки,
к.т.н., доцент Юрій КЛЬОЦ

8.12.25р.