Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему *Метод побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом*

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

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

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконав: *студент 2 курсу, група КНм-23-1* — Підпис — *Назарій ПЛАХТІЙ*
Курс, група виконавця; Ім'я, прізвище

Керівник: *к.т.н., доцент кафедри КН* — Підпис — *Олександр ПАСІЧНИК*
Науковий ступінь, посада; Ім'я, прізвище

Нормоконтроль: *к.т.н., доцент кафедри КН* — Підпис — *Руслан БАГРІЙ*
Науковий ступінь, посада; Ім'я, прізвище

До захисту допускаю:
Зав. кафедри КН, д.т.н., професор — Підпис — *Олександр БАРМАК*
Ім'я, прізвище

16 12 24 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *магістр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук*
_____
(підпис)
д.т.н., професор Олександр БАРМАК
«02» 09 24 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

1. Тема кваліфікаційної роботи магістра: *«Метод побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом»*
2. Завдання видано студенту Назарію ПЛАХТІЮ
((ім'я, прізвище,)
3. Керівник роботи *доцент кафедри КН Олександр ПАСІЧНИК*
(ім'я, прізвище)
4. Затверджені наказом університету від «26» 08 24 р. № 60.
5. Зміст пояснювальної записки (перелік задач) та вихідні дані: Наведено характеристику предметної області з оглядом методів побудови структурно-логічної схеми освітніх компонентів, оглядом сучасних підходів та наявного програмного забезпечення. Визначено мету та задачі дослідження. Реалізовано метод побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом та виконана його програмна реалізація. Виконано тестування програмної реалізації та дослідження реалізованого методу.

Реферат

Кваліфікаційна робота магістра присвячена покращенню побудови структурно-логічної схеми освітніх компонентів, з використанням генетичних алгоритмів, що має на меті оптимізацію освітньої траєкторії для здобувачів вищої освіти як інформаційної системи.

Актуальність теми. В контексті змін сучасного світу, особливо у сфері технологій та знань, освіта стикається з величезними викликами у питанні своєї актуалізації та оптимізації. Питання покращення освітніх процесів стає все актуальнішим, оскільки і державні, і приватні освітні заклади прагнуть до прогресивних змін, які забезпечать ефективне та якісне навчання. Актуальність дослідження полягає в розробці та впровадженні інноваційних підходів до побудови структурно-логічних схем освітніх компонентів з використанням генетичних алгоритмів. Оптимізація у цьому контексті означає не лише покращення існуючих методів навчання, а й створення систем, здатних адаптуватися та відповідати індивідуальним потребам студентів.

Традиційно, оптимізація в освіті часто використовувалась у контексті економії ресурсів, що нерідко призводило до скорочення чисельності кадрів, зменшення бюджетного фінансування, або навіть до закриття навчальних закладів. Однак, сучасне розуміння та застосування оптимізації в освіті має на меті досягнення оптимальних результатів з використанням існуючих ресурсів, що веде до підвищення якості освіти та забезпечення можливості для студентів розвивати свої потенціали. В площині оптимізаційних процесів часто йдеться про мінімізацію витрат та максимізацію результативності, що можна розглядати у вигляді якості освіти, ефективності програм та задоволення потреб студентів. З цією метою важливо не лише наявність чітко сформульованих цілей, але й глибоке розуміння того, що є найкращим результатом в конкретних умовах та для конкретної групи здобувачів освіти.

В дослідженні, використовуючи генетичні алгоритми для розробки структурно-логічної схеми освітніх компонентів, розглядається можливість досягти насправді індивідуалізованого та гнучкого підходу до побудови навчальних програм,

що враховують значну кількість факторів та параметрів, спрямованих на підвищення ефективності та якості освітнього процесу.

Мета і задачі роботи. Мета кваліфікаційної роботи полягає у підвищення якості формування структурно-логічної схеми освітніх компонентів за генетичним алгоритмом.

Для досягнення цієї мети були поставлені такі завдання:

- провести аналіз методів побудови структурно-логічної схеми освітніх компонентів, можливостей, переваг та недоліків генетичного алгоритму;

- спроектувати метод побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом;

- визначити набір критеріїв для оцінки підвищення якості формування побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом;

- виконати програмну реалізацію методу побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом;

- провести експериментальне дослідження реалізованого методу за тестовими наборами даних шляхом оцінки якості побудови структурно-логічної схеми освітніх компонентів.

Об’єкт дослідження – процес формування структурно-логічної схеми освітніх компонентів за генетичним алгоритмом.

Предмет дослідження – моделі, алгоритми та засоби побудови структурно-логічної схеми освітніх компонентів.

Методи дослідження. Генетичний алгоритм, методи пошуку найкращих рішень, експериментальне тестування на реальних наборах даних.

Наукова новизна одержаних результатів. Удосконалено метод побудови структурно-логічної схеми освітніх компонентів, який використовує генетичний алгоритмів та дозволяє врахувати послідовність та повноту опанування освітніх компонент.

Апробація результатів кваліфікаційної роботи магістра та публікації. За темою кваліфікаційної роботи магістра автором виконано 1 наукову публікацію -

Плахтій Н., Пасічник О., Манзюк Е., Скрипник Т., Петровський С. Спосіб формування пулу вибіркових навчальних дисциплін на основі генетичного алгоритму // Вісник Хмельницького національного університету, № 3, 2024, (335), С. 182 – 185. DOI: 10.31891/2307-5732-2024-335-3-26.

Структура та обсяг роботи. Кваліфікаційна робота магістра містить завдання, реферат, зміст, перелік скорочень, вступ, 4 розділи, висновки, перелік посилань із 40 найменувань та 3 додатків. Загальний обсяг становить 110 сторінок, з них 72 сторінки основного тексту. У роботі наведено 49 рисунків та 13 таблиць.

Ключові слова: генетичний алгоритм, структурно-логічна схема, освітня траєкторія, компетентність, дисципліна, програмне забезпечення.

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
ГА	Генетичний алгоритм
ECTS	European Credit Transfer and Accumulation System
IOT	Індивідуальна освітня траєкторія
БД	База даних
CRUD	Create Read Update Delete
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
S.M.A.R.T.	Specific Measurable Achievable Relevant Time-bound
MVC	Model-View-Controller

Вступ

Актуальність теми. В контексті змін сучасного світу, особливо у сфері технологій та знань, освіта стикається з величезними викликами у питанні своєї актуалізації та оптимізації. Питання покращення освітніх процесів стає все актуальнішим, оскільки і державні, і приватні освітні заклади прагнуть до прогресивних змін, які забезпечать ефективне та якісне навчання. Актуальність дослідження полягає в розробці та впровадженні інноваційних підходів до побудови структурно-логічних схем освітніх компонентів з використанням генетичних алгоритмів. Оптимізація у цьому контексті означає не лише покращення існуючих методів навчання, а й створення систем, здатних адаптуватися та відповідати індивідуальним потребам студентів.

Традиційно, оптимізація в освіті часто використовувалась у контексті економії ресурсів, що нерідко призводило до скорочення чисельності кадрів, зменшення бюджетного фінансування, або навіть до закриття навчальних закладів. Однак, сучасне розуміння та застосування оптимізації в освіті має на меті досягнення оптимальних результатів з використанням існуючих ресурсів, що веде до підвищення якості освіти та забезпечення можливості для студентів розвивати свої потенціали. В площині оптимізаційних процесів часто йдеться про мінімізацію витрат та максимізацію результативності, що можна розглядати у вигляді якості освіти, ефективності програм та задоволення потреб студентів. З цією метою важливо не лише наявність чітко сформульованих цілей, але й глибоке розуміння того, що є найкращим результатом в конкретних умовах та для конкретної групи здобувачів освіти.

В дослідженні, використовуючи генетичні алгоритми для розробки структурно-логічної схеми освітніх компонентів, розглядається можливість досягти насправді індивідуалізованого та гнучкого підходу до побудови навчальних програм, що враховують значну кількість факторів та параметрів, спрямованих на підвищення ефективності та якості освітнього процесу.

Мета і задачі роботи. Мета кваліфікаційної роботи полягає у підвищення якості формування структурно-логічної схеми освітніх компонентів за генетичним алгоритмом.

Для досягнення цієї мети були поставлені такі завдання:

- провести аналіз методів побудови структурно-логічної схеми освітніх компонентів, можливостей, переваг та недоліків генетичного алгоритму;

- спроєктувати метод побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом;

- визначити набір критеріїв для оцінки підвищення якості формування побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом;

- виконати програмну реалізацію методу побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом;

- провести експериментальне дослідження реалізованого методу за тестовими наборами даних шляхом оцінки якості побудови структурно-логічної схеми освітніх компонентів.

Об'єкт дослідження – процес формування структурно-логічної схеми освітніх компонентів за генетичним алгоритмом.

Предмет дослідження – моделі, алгоритми та засоби побудови структурно-логічної схеми освітніх компонентів.

Методи дослідження. Генетичний алгоритм, методи пошуку найкращих рішень, експериментальне тестування на реальних наборах даних.

Наукова новизна одержаних результатів. Удосконалено метод побудови структурно-логічної схеми освітніх компонентів, який використовує генетичний алгоритмів та дозволяє врахувати послідовність та повноту опанування освітніх компонент.

Апробація результатів кваліфікаційної роботи магістра та публікації. За темою кваліфікаційної роботи магістра автором виконано 1 наукову публікацію - Плахтій Н., Пасічник О., Манзюк Е., Скрипник Т., Петровський С. Спосіб формування пулу вибіркових навчальних дисциплін на основі генетичного алгоритму

// Вісник Хмельницького національного університету, № 3, 2024, (335), С. 182 – 185. DOI: 10.31891/2307-5732-2024-335-3-26.

Структура та обсяг роботи. Кваліфікаційна робота магістра містить завдання, реферат, зміст, перелік скорочень, вступ, 4 розділи, висновки, перелік посилань із 40 найменувань та 3 додатків. Загальний обсяг становить 110 сторінок, з них 72 сторінки основного тексту. У роботі наведено 49 рисунків та 13 таблиць.

Розділ 1 Характеристика предметної області та постановка задачі

Освіта сьогодні виступає незмінним фундаментом у становленні розвиненого суспільства, відіграючи ключову роль у формуванні економічного, соціального, та культурного потенціалу кожної держави. У зв’язку з цим, існує потреба в постійному та швидкому оновленні освітніх програм і навчальних планів, що зумовлено суттєвою варіативністю вимог ринку праці та динамічним розвитком суспільства. Паралельно, величезні досягнення в сфері комп’ютерних наук відкривають нові можливості для підвищення ефективності навчання, інтеграції інноваційних технологій у освітній процес та розробки адаптивної освіти. Використання передових методів комп’ютерних наук у сфері освіти дозволяє не тільки покращити якість навчального матеріалу, але й отримати високоефективні рішення, які спрямовані на успішну реалізацію освітніх цілей.

1.1 Аналіз предметної області

Освітні системи по всьому світу відіграють ключову роль у формуванні кваліфікованих спеціалістів, які згодом стають рушійною силою економіки кожної країни. Україна не є винятком, і процес її освітніх реформ має на меті забезпечити якісне навчання та адекватну підготовку студентів до сучасних викликів. Освітні програми, що запроваджуються навчальними закладами, поділяються на кілька основних типів [1]. Початкову та загальну середню освіту регулює держава з метою забезпечення усіх необхідних знань та навичок згідно з віковими вимогами. Дані програми мають обов’язковий перелік предметів, які повинні вивчати учні [2]. Вища освіта поділяється на бакалаврат, магістратуру та аспірантуру, і кожен рівень передбачає певну комплексність та спеціалізацію навчання. Програми вищої освіти орієнтуються на глибоке засвоєння спеціальних знань і підготовку до професійної діяльності [3]. Професійна освіта зосереджена на підготовці кваліфікованих робітників і спеціалістів для конкретних галузей економіки. Навчальні програми профтехів акцентують увагу на практичних навичках [4]. Програми підвищення

кваліфікації та професійного розвитку є важливою частиною утримання актуальності кваліфікацій серед працюючих професіоналів, що дозволяє їм залишатися конкурентоспроможними на ринку праці [5].

Забезпечення якості освітніх програм є критичним аспектом їхнього впровадження. Це здійснюється через регулярну акредитацію, оцінку та відгуки від студентів та викладачів, що дозволяє виявляти потенційні слабкі місця та вдосконалювати навчальні процеси [6].

Альтернативними методами взаємодії навчальних дисциплін є введення курсів за вибором, які дають студентам можливість глибше зануритися в обрані ними предмети і краще підготуватися до майбутньої кар'єри [7]. Важливим аспектом є гнучкість освітньої програми, вона має відповідати змінам в індустрії та суспільстві, включаючи інтеграцію з новітніми технологіями і методиками навчання [8]. Визначення напрямків підготовки і спеціалізацій, які будуть включені до освітньої програми, є одним з ключових етапів планування. Освітні установи повинні аналізувати потреби ринку праці, сучасні тенденції в індустрії і наукових дослідженнях, а також розглядати фідбек від випускників і роботодавців для того, щоб їх програми були релевантними і актуальними [9]. На основі цього аналізу формуються дисципліни, які становлять основу навчального плану. Важливо пам'ятати, що кожна дисципліна повинна мати чітко визначені цілі, вимоги, методи оцінювання та результати навчання, які потрібно досягати [10]. Це дозволяє студентам ясно розуміти, що від них вимагається, і які знання та навички вони отримають після завершення курсу. Для адаптивності та гнучкості освітніх програм створюють опції вибіркових курсів. Це надає студентам можливість налаштувати своє навчання згідно індивідуальних інтересів та кар'єрних амбіцій [11]. Вибіркові курси можуть бути спрямовані на поглиблення знань у вузьких областях, що стосуються головної спеціалізації, або ж дозволяти отримати міждисциплінарні навички, які можуть бути корисними в майбутньому. Ефективність освітньої програми також значною мірою залежить від кваліфікації викладацького складу. Викладачі повинні бути не тільки глибоко освіченими в своїх предметних областях, але й володіти навичками передачі знань, інноваційними методиками навчання та

здатністю мотивувати студентів [12]. Крім того, багато чого залежить від обладнання та матеріально-технічної бази навчального закладу. Сучасне обладнання, лабораторії, бібліотеки та доступ до новітніх технологій і дослідницьких матеріалів сприяють більш ефективному навчанню і дозволяють студентам отримувати актуальні практичні навички [13].

Зв’язок з індустрією і роботодавцями може забезпечувати студентам місця для стажувань та практик, що є чудовим способом застосування теоретичних знань на практиці. Це також підвищує їхні шанси на успішне працевлаштування після завершення навчання [14]. Нарешті, постійний моніторинг і перегляд програми на предмет її актуальності є необхідним для забезпечення високих стандартів якості освіти. Зворотній зв’язок від студентів, випускників та роботодавців дозволяє вчасно вносити необхідні корективи до програми [15]. Подальше вдосконалення освітніх програм вимагає інтеграції міжнародного досвіду та глобальних освітніх практик. З підвищенням глобалізації, українські навчальні заклади можуть багато перейняти від міжнародних партнерів, впроваджуючи кращі освітні технології та методики [16]. Це може включати все, від спільних наукових проектів до обмінних студентських програм, які розширюють горизонти студентів і стимулюють інноваційне мислення. Також значна роль у підвищенні ефективності освітніх програм припадає на використання цифрових інструментів та ресурсів. Впровадження відкритих онлайн курсів і використання штучного інтелекту для персоналізації навчального процесу можуть суттєво підвищити доступність і якість освіти [17]. Цифровізація навчання дозволяє студентам вчитися в гнучкому режимі, адаптувати швидкість навчання під особисті потреби та водночас дає доступ до найновіших наукових досягнень. Залучення до розробки навчальних програм фахівців із різних галузей також є вагомим аспектом. Вони можуть надати важливу інформацію щодо практичних аспектів кар’єри, які навчальні заклади мають враховувати, щоб випускники були командними, інноваційними та готовими до майбутніх робочих викликів [18].

Оцінка успішності навчальних програм є ще однією критичною складовою. Цей процес включає аналіз результатів студентів, а також їхню зайнятість та кар’єрне розвиток після випуску. Такі дані допомагають освітнім установам визначити,

наскільки добре студенти готові до реальних викликів у їхній професійній діяльності і що потрібно покращувати в майбутньому [19]. Укріплення взаємодії з академічними і науковими товариствами також сприяє зростанню наукового потенціалу країни. Студенти і викладачі, які займаються дослідницькою роботою, можуть значно покращувати якість навчання і розширювати прикладні дослідження в промисловості і технологіях.

Таким чином, розробка та імплементація освітніх програм є комплексним процесом, що вимагає уваги до деталей, глибокого аналізу потреб ринку праці та забезпечення відповідних ресурсів для викладання і навчання. Це створює міцну основу для підготовки обізнаних і компетентних випускників.

1.2 Аналіз сучасних підходів до побудови схеми освітніх компонентів

Сучасна освіта виходить за рамки традиційного трансферу знань, натомість акцентуючись на розробці інтегрованих освітніх компонентів, які сприяють глибокому залученню студентів до навчального процесу. Розглянемо детально кілька ключових стратегій в сучасних освітніх підходах.

Міждисциплінарний підхід у сучасній освіті відіграє ключову роль у формуванні навчальних планів, які сприяють глибокому розумінню сучасних викликів [20]. Поєднання знань з різних наукових дисциплін дозволяє студентам отримувати більш комплексне уявлення про предмети, розглядаючи їх з різних перспектив. Наприклад, курс з екологічної політики може інтегрувати елементи з біології, хімії, соціології та політичних наук – такий підхід не тільки збагачує знання студентів, але й стимулює критичне мислення та творчий підхід до вирішення проблем. Це важливо у контексті глобалізації та швидкого розвитку технологій, де часто виникають задачі, які вимагають інтегративних знань [21].

Модульний підхід поділяє навчальний процес на окремі, самодостатні блоки, які мають конкретну тематику або направленість [22]. Кожен модуль спроектований так, щоб надати студентам гнучкість у виборі навчальних траєкторій – здобувачі освіти можуть вибирати модулі відповідно до своїх інтересів або професійних потреб.

Такий підхід також враховує міждисциплінарні зв’язки, що дозволяє студентам отримувати більш комплексне розуміння предмету. Модулі можуть бути застосовані в онлайн і офлайн форматах, включаючи практичні дослідження, лекції, воркшопи, що робить навчання більш динамічним і адаптивним до особистісних потреб кожного студента.

Компетентнісний підхід зосереджується на розвитку знань студентів, які вони можуть застосовувати в реальних ситуаціях професійної діяльності [23]. Такий підхід вимагає від освітніх установ визначення ключових компетенцій, які вважаються важливими для майбутньої кар’єри студентів, і відповідно розробку програм, які забезпечують засвоєння цих компетенцій. Ці компетенції необхідні не тільки для роботи за спеціальністю, але й для управління кар’єрною траєкторією та довгостроковим професійним зростанням. При цьому важливим є не тільки знання, але і уміння аналізувати інформацію, працювати в команді, вирішувати комплексні задачі та адаптуватися до змін в робочому середовищі.

Проектний підхід в освіті включає активну роботу студентів над реальними або симуляційними проектами, які відображають реальні професійні виклики та ситуації [24]. Цей підхід дозволяє студентам інтегрувати і застосовувати знання з різних дисциплін, розвивати критичне мислення, творчі і комунікативні навички. Проектна робота зазвичай включає планування, дослідження, виконання та оцінку проекту, що забезпечує студентам всебічний досвід роботи над комплексними завданнями. Це також сприяє розвитку навичок управління часом та ресурсами, які є важливими для будь-якої професійної діяльності. Проекти можуть бути індивідуальними або груповими і часто включають елементи дослідження, дизайну, реалізації та презентації. Проектний підхід є вкрай ефективним для підготовки студентів до динамічного та багатогранного ринку праці.

Технологічний розвиток відкриває нові можливості для освітнього процесу. Інтеграція інноваційних технологій, таких як штучний інтелект, машинне навчання, віртуальна та доповнена реальність, забезпечує створення реалістичних і інтерактивних навчальних середовищ [25]. Ці технології можуть використовуватися для створення симуляцій, які дозволяють студентам випробувати професійні ситуації

в безпечному і контрольованому середовищі, здійснювати експерименти та отримувати практичний досвід. Зростання використання відкритих освітніх ресурсів, таких як відкриті навчальні матеріали, онлайн-курси та платформи, демократизує доступ до якісної освіти. Це не тільки схприяє більш широкому доступу до навчання, але й стимулює розвиток освітніх інновацій, оскільки викладачі та навчальні заклади можуть використовувати, адаптувати та поєднувати ресурси для створення унікального навчального досвіду.

Систематичне оцінювання та зворотний зв'язок є важливими елементами ефективної освіти. Вони дозволяють студентам отримувати критичну інформацію про свій прогрес і виявляти сфери, які потребують додаткової уваги або вдосконалення. Сучасні підходи до оцінювання часто включають комбінації традиційних екзаменів, проектної роботи, портфоліо, а також неперервне формативне оцінювання, яке дозволяє вчителям та студентам вести постійний діалог про навчальний процес [26]. Залучення технологій також дозволяє використовувати адаптивні системи оцінювання, які автоматично налаштовують складність завдань в залежності від успішності студента, сприяючи більш персоналізованому підходу до навчання. Також, у сучасному динамічному світі важливою є здатність освітніх програм швидко адаптуватися до змін у технологіях, промисловості та суспільстві. Це включає не тільки періодичне оновлення курсів та навчальних планів, але й більш гнучку структуру навчання, що дозволяє студентам легко переходити між різними курсами або дисциплінами і брати активну роль у формуванні свого освітнього досвіду. Така гнучкість може бути підтримана через модульні і онлайн курси, що дозволяють студентам краще узгоджувати освіту з особистими та професійними обов'язками, а також вивчати нові тематики відповідно до змін у їхніх інтересах і цілях.

Навчальні заклади також прагнуть впроваджувати освітні практики, що сприяють інтеграції принципів екологічної свідомості, соціальної справедливості та економічної обізнаності в курси та проекти, що готують студентів до вирішення глобальних викликів сучасності [27]. Важливість підготовки студентів до глобалізованого світу через інтеграцію інформації про глобальні виклики, міжнародну співпрацю, та міжкультурне розуміння. Програми можуть включати

навчання іноземних мов, обмінні програми або міжнародні стажування, що сприяє глибшому розумінню культурних і економічних контекстів різних країн. Вивчення цих принципів сприяє формуванню випускників, які не тільки успішні у своєму професійному полі, але й свідомі громадяни, готові діяти на благо загальної стійкості та добробуту спільнот. Сучасна освітня схема визнає важливість не тільки інтелектуального, але й емоційного та соціального розвитку студентів. Підтримка здорового емоційного клімату в освітніх установах, розвиток навичок комунікації, лідерства, вміння працювати в команді та вирішувати конфлікти є невід'ємною частиною сучасних освітніх програм [28]. Інтеграція в освітній процес модулів, що розвивають емоційний інтелект і соціальні навички, такі як самосвідомість, управління емоціями, мотивація, емпатія та навички соціальної взаємодії. Програми, які розвивають такі якості, сприяють формуванню здоровішої робочої атмосфери, кращому розумінню та повазі у різноманітних соціокультурних контекстах.

У світі, де знання швидко застарівають, важливо виховувати у студентів ставлення до навчання як до тривалого, життєвого процесу [29]. Сучасні освітні системи прагнуть розробляти програми, які формують у студентів навички самонавчання та саморозвитку. Це може включати курси з основ дослідження, використання різноманітних джерел інформації, а також навчальні модулі, які розвивають здатність критично оцінювати інформацію та формувати власні знання на її основі.

Сучасні підходи до побудови освітніх компонентів наголошують на інтеграції, гнучкості, доступності та сталості знань, з метою підготувати студентів не лише до успішного вирішення професійних завдань, а й до відповідального й змістовного життя в динамічному і багатоманітному світі.

1.3 Аналіз існуючого програмного забезпечення

З огляду на розвиток інтернет-технологій, надання освітніх послуг в онлайн-форматі стає дедалі популярнішим. Дистанційне навчання дозволяє людям з усього світу здобувати нові знання та навички, не виходячи з дому. Це особливо актуально в

умовах, коли фізична присутність в аудиторіях може бути обмежена через різні зовнішні обставини. Разом з тим, освітні установи та бізнес-структури шукають способи оптимізації та інновацій для покращення навчальних процесів. В цьому контексті дедалі більшу роль відіграють інформаційні ресурси, які пропонують різноманітні освітні програми та курси. Розглянемо ключові аспекти цих рішень на прикладі декількох платформ.

Coursera є однією з провідних платформ онлайн-освіти [30], вона пропонує широкий спектр курсів, спеціалізацій, професійних сертифікатів та ступенів вищої освіти, партнерства з провідними університетами та індустріальними гігантами з усього світу (рисунок 1.1). Coursera надає можливість кожному отримати доступ до освіти незалежно від місцезнаходження.

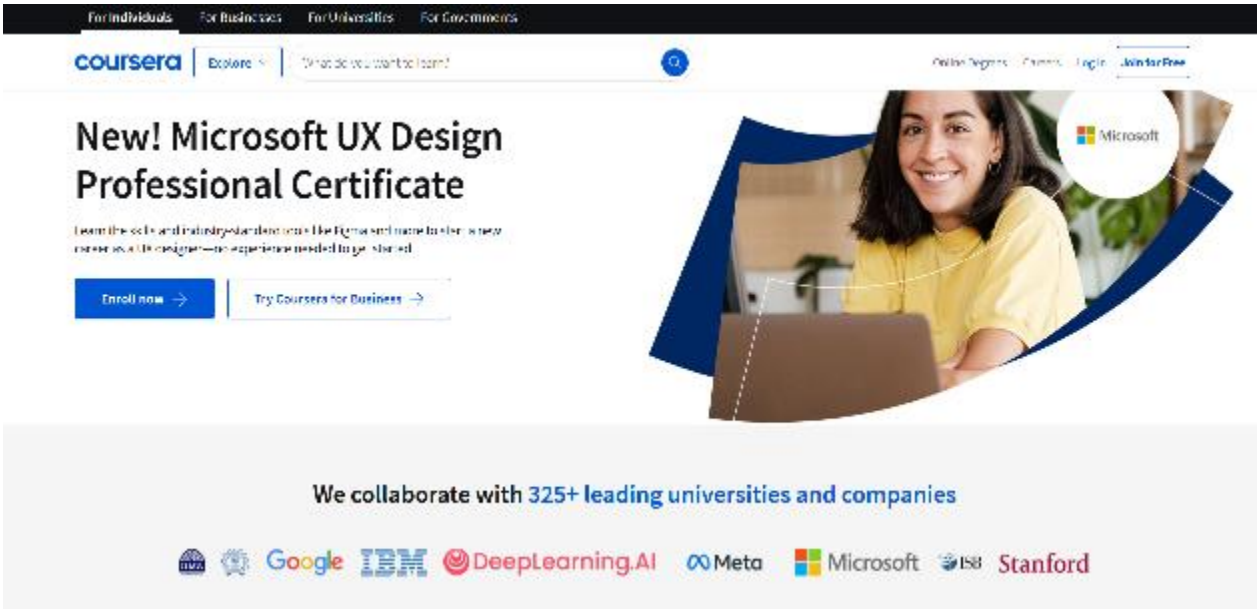


Рисунок 1.1 – Головна сторінка платформи Coursera

На Coursera можна знайти курси майже з усіх сфер знань, включаючи науку про дані, комп’ютерні науки, бізнес, особистісний розвиток, мистецтво та гуманітарні науки (рисунок 1.2). Більшість курсів можна проходити безкоштовно в аудиторному режимі, але для отримання сертифіката потрібно заплатити. Спеціалізації та професійні сертифікати необхідні для розвитку навичок у конкретній професійній сфері, а програми ступенів вищої освіти пропонують можливість здобуття бакалавра

або магістра онлайн (рисунок 1.3). Coursera забезпечує високий рівень інтерактивності: студенти можуть дивитися відео лекції, виконувати практичні завдання, брати участь у дискусіях на форумах, проходити оцінювання та тести для перевірки знань.

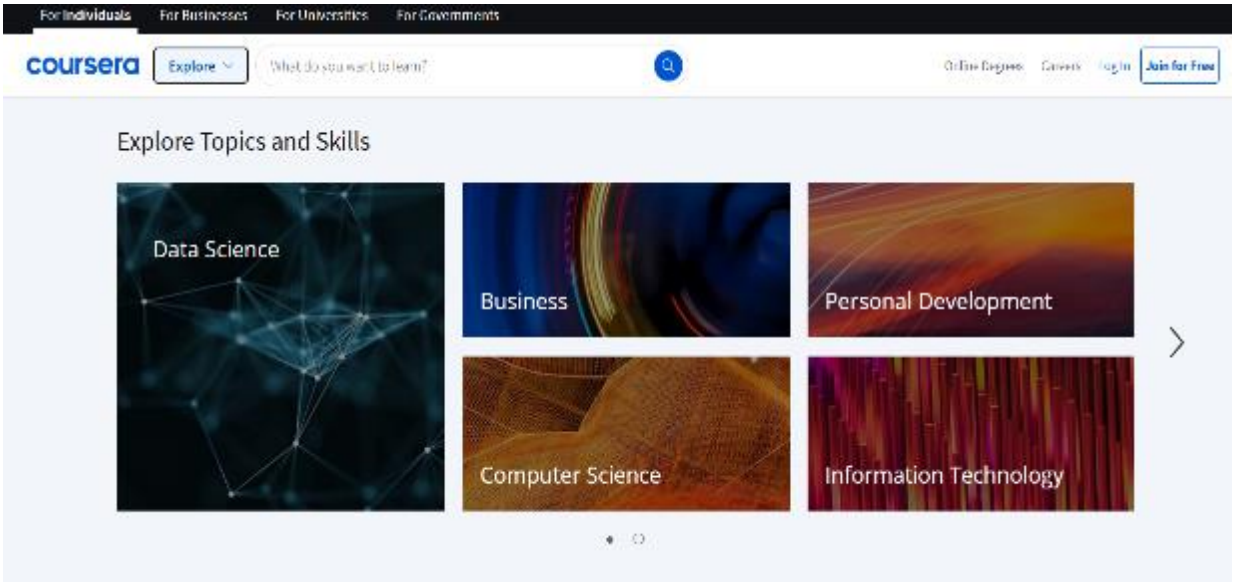


Рисунок 1.2 – Сторінка з переліком спеціальностей

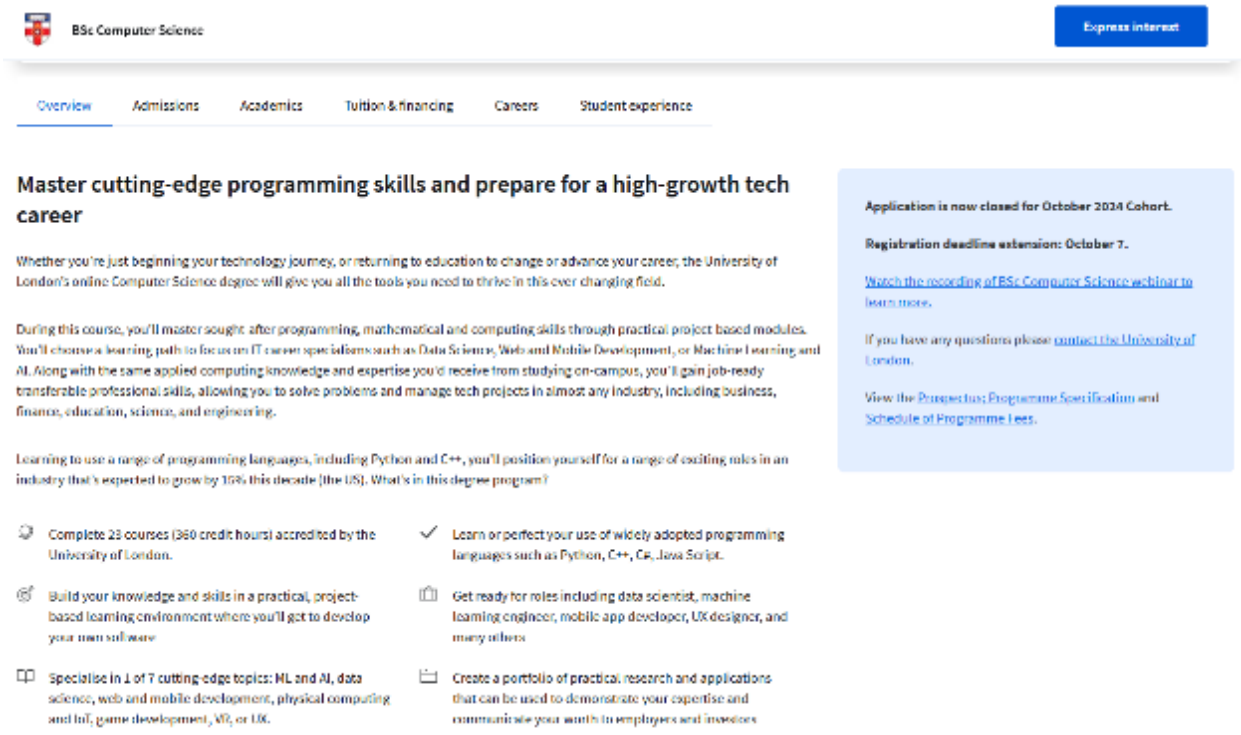


Рисунок 1.3 – Сторінка з описом освітньої програми

Проаналізувавши даний ресурс можна виділити наступні переваги та недоліки. До переваг можна віднести: широкий спектр спеціальностей від технічних наук до мистецтва; партнерство з відомими університетами та компаніями; гнучкість у навчанні, багато курсів пропонують безкоштовний доступ до матеріалів. До недоліків можна віднести: вартість отримання сертифіката; відсутність персоналізації хоча платформа і пропонує рекомендації, комплексного індивідуального підходу до кожного студента, як у випадку з традиційним освітнім закладом, немає; через масовий характер курсів, індивідуальна взаємодія з викладачами обмежена, що може впливати на якість навчального процесу для певних курсів або студентів.

Наступний освітній ресурс – Khan Academy є некомерційною освітньою організацією [31], що пропонує повний спектр курсів, в основному зосереджених на математиці, науці, програмуванні, історії, мистецтві, економіці та інших предметних областях для студентів починаючи з дитячого садка і закінчуючи вищою школою. Khan Academy також включає матеріали для підготовки до SAT, LSAT, MCAT і інших стандартизованих тестів (рисунок 1.4).

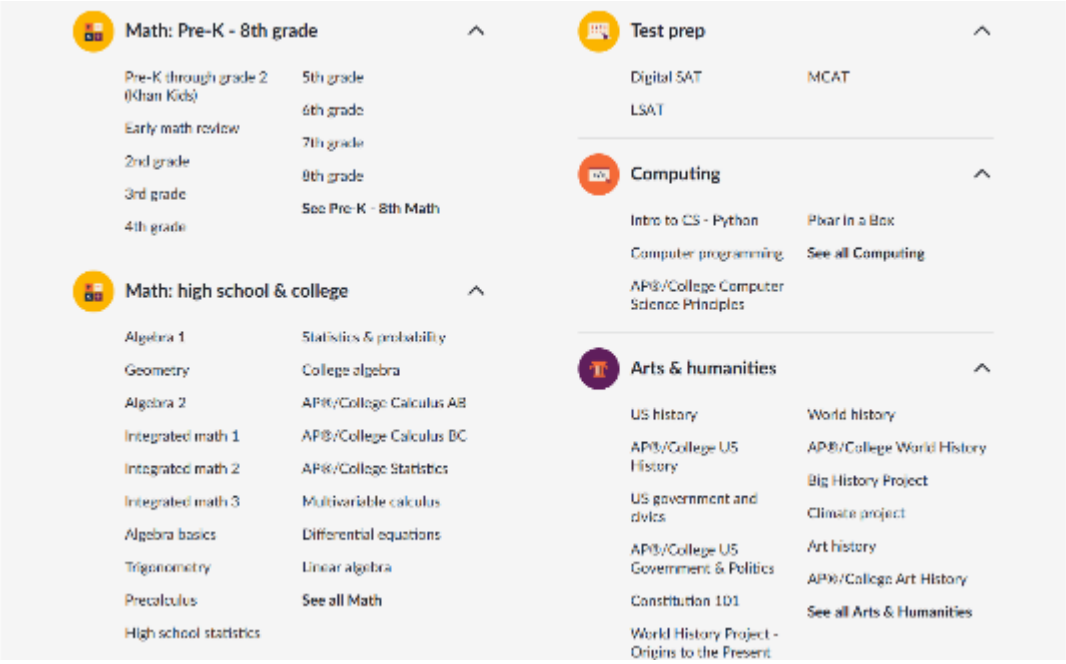


Рисунок 1.4 – Сторінка з переліком доступних курсів

Khan Academy використовує відео-уроки, інтерактивні завдання та вправи для забезпечення глибокого розуміння навчального матеріалу. Студенти можуть відстежувати свій прогрес, встановлювати цілі навчання і отримувати рекомендації для подальших кроків завдяки системі персоналізації. Одна з ключових особливостей Khan Academy – це орієнтація на самостійне навчання та індивідуальні темпи засвоєння матеріалу, що дозволяє студентам концентруватися на найбільш потрібних компетентностях.

Перейдемо до опису переваг та недоліків, до переваг в першу чергу можна віднести: безкоштовний доступ до всіх матеріалів; Khan Academy пропонує різноманітні курси в багатьох сферах; студенти можуть навчатися в будь-який час і працювати у зручному темпі; система відстеження прогресу і персоналізовані навчальні траєкторії допомагають студентам зосередитися на прогалинах у знаннях. Щодо недоліків то вивчені курси не передбачають отримання ступенів або формальних кваліфікацій, також незважаючи на наявність інтерактивних вправ і завдань, брак можливості для обговорення з викладачами або однолітками може зменшити ефективність навчання для студентів.

Підсумовуючи аналіз освітніх платформ, стає очевидним наявний потенціал для продовження дослідження та застосування новітніх методів у галузі освітніх технологій. Одним із таких перспективних напрямків є використання генетичних алгоритмів для імплементації у системи навчання з метою поліпшення ефективності та адаптивності. Генетичні алгоритми, натхненні процесами природного відбору, можуть слугувати механізмом для оптимізації освітніх траєкторій, роблячи процес навчання більш індивідуалізованим та гнучким.

1.4 Про генетичний алгоритм

Генетичні алгоритми (ГА) є ефективним інструментом у галузі штучного інтелекту, зокрема в області еволюційного обчислення [32]. ГА імітують процеси природного відбору, які спостерігаються в біології, адаптуючи їх для вирішення задач оптимізації та пошуку рішень у різних доменах. Заснований на механізмах

біологічної еволюції, таких як спадковість, мутація, рекомбінація (кросинговер) та селекція, генетичний алгоритм стимулює пошук найкращих рішень, адаптуючи популяцію кандидатів з покоління в покоління.

ГА починається з генерації випадкової початкової популяції кандидатів, які представляють можливі рішення задачі [33]. Кожен кандидат, відомий як хромосома, містить сет інформації, яка кодується шляхом генів. В основному, кожен ген представляє певний аспект рішення. Наступним кроком є оцінка кожної хромосоми для визначення її придатності. Придатність вимірюється з допомогою функції пристосованості, специфічної для кожної задачі, яка оцінює як добре хромосома вирішує задачу. На основі цих оцінок, хромосоми обираються для участі у наступних кроках.

Кросинговер – це процес, в якому дві хромосоми обмінюються сегментами своєї ДНК, що призводить до створення нових хромосом, що мають ознаки обох батьківських хромосом [34]. Мутація є випадковою модифікацією генів у хромосомі [35]. Мутація необхідна для запобігання зупинки еволюції та допомагає алгоритму вийти з локальних мінімумів. Ці процеси створюють нове покоління, яке потім знову оцінюється, і процес повторюється протягом багатьох поколінь. Кінцевою метою є знаходження оптимального рішення задачі.

Генетичні алгоритми (ГА) мають цілий ряд переваг, які роблять їх вкрай ефективними для рішення складних задач оптимізації. Серед основних переваг можна виділити такі як:

1. Глобальний пошук – генетичні алгоритми здійснюють глобальний пошук по можливих рішеннях, що знижує ймовірність застрягання в локальних мінімумах. Це особливо важливо у складних просторах пошуку, де існують численні локальні оптимуми [36].

2. Адаптивність – ГА можуть досить легко адаптуватися до нових умов без необхідності кардинальної переробки самого алгоритму.

3. Паралелізм – завдяки природі популяційного пошуку, ГА піддаються паралелізму, де кілька процесів можуть обчислювати рішення незалежно, що значно знижує загальний час пошуку та збільшує ефективність.

4. Стійкість – ГА відрізняються високою стійкістю до шумів і змін у вхідних даних.

5. Масштабованість – ГА мають хороші властивості масштабування, що дозволяє їм ефективно розв’язувати великі та складні задачі.

6. Гнучкість – ГА є методологічно гнучкими, здатними розв’язувати задачі з різних галузей, від інженерії, до фінансів та біології, без необхідності значних модифікацій у самій структурі алгоритму.

7. Робота з обмеженнями – ГА можуть ефективно працювати з різними обмеженнями, знаходячи рішення, яке задовольняє всі, або більшість з них.

8. Здатність до самонавчання – з кожним новим поколінням хромосом генетичні алгоритми «навчаються» від найкращих рішень, застосовуючи досягнуті знання для створення ще більш ефективних рішень.

9. Простота впровадження – незважаючи на свою потужність і складність, базові генетичні алгоритми досить прості для розуміння та реалізації. Це робить їх доступними навіть для тих, хто не має глибоких знань у галузі комп’ютерних наук.

10. Різноманітність рішень – один із аспектів ГА полягає в здатності генерувати дуже різноманітні рішення для задачі. Завдяки механізмам кросинговеру та мутацій, рішення може еволюціонувати несподіваними шляхами, що відкриває нові перспективи вирішення проблем.

11. Вирішення NP-складних задач – ГА демонструють вражаючі результати у вирішенні NP-складних задач, де інші методи зазвичай не ефективні [37]. Це робить їх ідеальним інструментом для різних прикладів, які включають графи, маршрутизацію та оптимізацію розкладів.

12. Ефективність у мульти-об’єктній оптимізації – ГА особливо ефективні у задачах, де потрібно знайти баланс між декількома конфліктними цілями, такими як якість, вартість, час і ресурси [38]. Алгоритм може адаптуватись для досягнення прийнятного компромісу між цими цілями.

Хоча ГА мають багато переваг і широко застосовуються для рішення різноманітних оптимізаційних задач, існують і деякі недоліки, які можуть

обмежувати їх ефективність у певних ситуаціях. Недоліки генетичних алгоритмів включають:

1. Висока обчислювальна вартість – ГА потребують значних ресурсів для обчислень, особливо при роботі з великими популяціями або складними генетичними структурами. Це може стати проблемою при обмежених обчислювальних ресурсах або потребі в швидкій обробці.

2. Складність налаштування параметрів – правильне налаштування параметрів ГА, таких як розмір популяції, ймовірність мутації та кросинговеру, відбір і інші, є критичним для успіху алгоритму. Неправильне налаштування може призвести до поганих результатів, таких як передчасна конвергенція.

3. Ризик передчасної конвергенції – ГА можуть бути схильні до передчасної конвергенції, коли вся популяція вказує на одне рішення занадто рано, втрачаючи генетичне розмаїття і спроможність ефективно досліджувати простір рішень [39].

4. Залежність від випадковості – оскільки ГА залежать від таких випадкових процесів, як мутація та відбір, результати можуть неодноразово варіюватись, навіть при одній і тій же вихідній конфігурації, що ускладнює тестування алгоритму.

5. Необхідність у великій кількості оцінок – для більшості задач оптимізації ГА вимагають великої кількості оцінок функції придатності, що може бути витратним як в плані часу, так і в ресурсах, особливо для складних у обчисленні функцій.

6. Підтримка різноманітності – без спеціальних засобів підтримки різноманітності генів, ГА можуть швидко втратити генетичне розмаїття, що необхідне для ефективного пошуку, що особливо актуально при відборі та репродукції.

Генетичні алгоритми знайшли своє застосування в різноманітних сферах діяльності завдяки спроможності ефективно вирішувати складні оптимізаційні проблеми [40]. Серед найбільш значущих галузей, де вони демонструють високу ефективність можна виділити: інженерію, промисловий дизайн, фінанси, транспорт, логістика, енергетика, машинне навчання. Генетичні алгоритми застосовуються для проектування машин, автомобілів та інших інженерних структур з оптимальними характеристиками. Це може включати оптимізацію форми для мінімізації

аеродинамічного опору або оптимізацію ваги металоконструкцій, забезпечуючи при цьому їх міцність та довговічність. У біоінформатиці ГА часто використовують для аналізу та реконструкції філогенетичних дерев, а також для моделювання еволюційних процесів. Вони дозволяють моделювати, як гени та білки взаємодіють між собою, що є ключем для розуміння багатьох біологічних процесів і розвитку захворювань. У фінансовій індустрії ГА використовують для оптимізації портфелів, автоматизації торгових стратегій, а також для прогнозування ринкових тенденцій. ГА допомагають визначати найбільш прибуткові стратегії управління активами, мінімізуючи ризики та максимізуючи доходи. ГА допомагають оптимізувати маршрути доставки, створення розкладів і управління складськими запасами, що приводить до зниження витрат на транспортування та покращення ефективності логістичних операцій. У галузі енергетики генетичні алгоритми використовують для оптимізації розподілу ресурсів, управління навантаженнями та планування виробництва енергії, особливо у відновлюваних джерелах енергії, таких як вітрові та сонячні електростанції. Завдяки спроможності до оптимізації та автоматизації, ГА використовують для розробки та покращення алгоритмів штучного інтелекту. Алгоритм застосовується для налаштування нейронних мереж та інших моделей машинного навчання, що підвищує їх ефективність і адаптивність.

Отже, беручи все вище описане до уваги, ГА є надзвичайно потужним інструментом для розв’язання широкого спектру проблем, зокрема тих, що включають складні обчислення та великі простори пошуку рішень. Це також зумовлює їх популярність в академічних дослідженнях та реальних застосуваннях на практиці.

1.5 Постановка задачі

Мета кваліфікаційної роботи магістра полягає у підвищенні якості формування структурно-логічної схеми освітніх компонентів за генетичним алгоритмом. Цей підхід має на меті створення адаптивної, індивідуалізованої освітньої траєкторії, яка

відповідатиме потребам здобувачів вищої освіти та враховуватиме академічні правила. Для виконання поставленого завдання необхідно:

1. Проведення детального аналізу методів побудови структурно-логічної схеми освітніх компонентів, можливостей, переваг та недоліків генетичного алгоритму.

2. Спроектувати метод побудови структурно-логічної схеми освітніх компонентів з використанням генетичних алгоритмів, що включає розробку алгоритмічної структури, яка зможе автоматично генерувати оптимальні шляхи навчання на основі вхідних параметрів.

3. Визначити набір критеріїв для оцінки підвищення якості формування побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом.

3. Виконати програмну реалізацію методу побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом.

4. Провести експериментальне дослідження реалізованого методу за тестовими наборами даних шляхом оцінки якості побудови структурно-логічної схеми освітніх компонентів.

РОЗДІЛ 2 Метод побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом

2.1 Загальні положення щодо побудови структурно-логічної схеми освітніх компонентів

Фундаментом для створення та оптимізації адаптивних освітніх систем служить структурно-логічна схема освітніх елементів, що представляє собою інтеграцію різноманітних навчальних ресурсів у цілісну структуру. Ця структура має на меті об’єднати теоретичні матеріали, практичні завдання, тести для перевірки знань та інші важливі компоненти, адаптуючи їх під індивідуальні особливості кожного учня (рисунок 2.1).

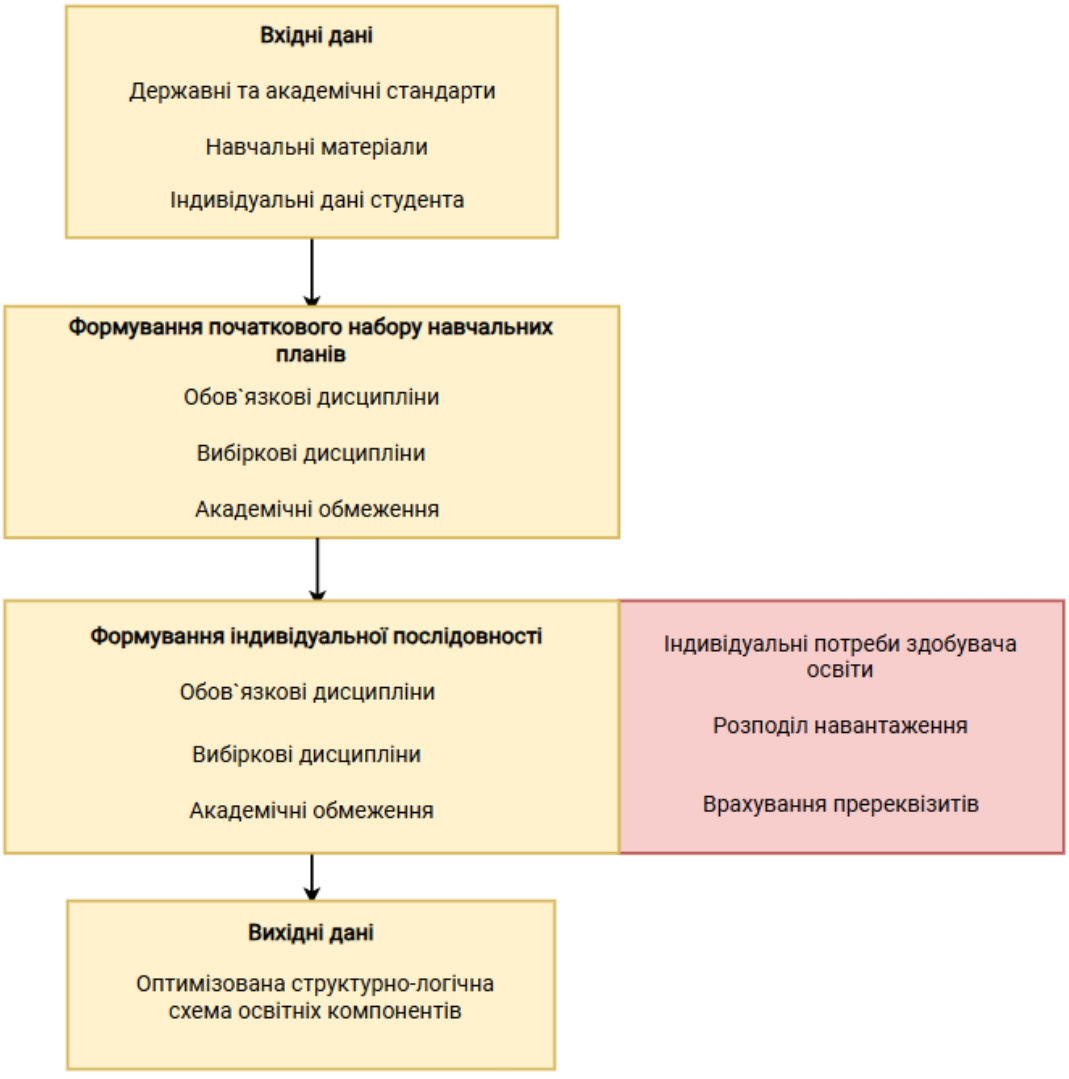


Рисунок 2.1 – Послідовність формування структурно-логічної схеми

До вхідних даних, що слугують основою для формування початкового набору можливих рішень та подальшої оптимізації, відносяться визначені освітні цілі, індивідуальні потреби та дані учнів, а також базовий навчальний контент, що містить всі необхідні для навчання елементи. У контексті створення адаптивної освітньої системи, важливість і різноманітність вхідних даних відіграє дуже важливу роль. Зібрані дані про кожного учня, що включають інформацію про попередній досвід навчання, інтереси, переваги у форматах навчання, досягнення, а також індивідуальні сильні та слабкі сторони, такі дані дозволяють здійснювати персоналізацію навчального процесу. Правильно сформульовані освітні цілі та компетенції, які учень має набути в результаті навчання, повинні бути визначені на основі державних освітніх стандартів, специфіки предмету, або конкретних вимог до професійної підготовки. Також, базова бібліотека навчального контенту, що включає теоретичні матеріали, практичні завдання, відео-уроки, тести для самоперевірки тощо, відіграє важливу роль в створенні структурно-логічної схеми.

Окрім вхідних даних, формування початкового набору навчальних планів є важливим кроком у розробці адаптивних освітніх систем. Цей процес передбачає створення сукупності варіантів освітніх програм, кожна з яких балансує між виконанням академічних стандартів та наданням можливості для індивідуалізації. До базових компонентів відносяться обов’язкові дисципліни, які є фундаментальною частиною будь-якого навчального плану, вони визначаються академічними правилами та стандартами відповідного закладу освіти. Для забезпечення гнучкості та адаптації до інтересів студентів, до навчального плану додають вибіркові дисципліни. Під час формування наборів необхідно стежити за дотриманням академічних обмежень, таких як максимальна та мінімальна кількість кредитів ECTS за семестр, баланс між теоретичними та практичними заняттями, а також забезпечення послідовності вивчення предметів. Після ініціалізації початкових навчальних планів важливо здійснити їх аналіз на предмет відповідності навчальних цілей та забезпечення раціонального розподілу навчального навантаження. Здійснення корекцій дозволяє оптимізувати кожен план з метою досягнення найкращих можливих освітніх результатів.

Після того, як базовий набір навчальних планів був сформований, наступним кроком є формування фінальної послідовності, яка відображає оптимізовану структуру навчального процесу. Цей процес включає ряд кроків для забезпечення того, щоб кожен навчальний план виконував встановлені освітні інтенції та відповідав академічним вимогам, важливо перевірити, щоб усі навчальні плани відповідали основним критеріям, серед яких вимоги щодо кількості кредитів ECTS, рівновага між лекційними та практичними заняттями, покриття визначених навчальних цілей. Освітня програма має бути не лише відповідною до стандартів, але й достатньо гнучкою, щоб адаптуватися до індивідуальних потреб та інтересів студентів. Це означає, що вибіркові предмети і курси мають бути інтегровані таким чином, щоб учні могли формувати власний індивідуальний навчальний шлях. Також слід враховувати розподіл навчального навантаження протягом семестрів, щоб впевнитись, що студенти матимуть збалансоване навчання без перевантажень. Останнім кроком є оптимізація кожного навчального плану з урахуванням зібраних деталей та зворотного зв’язку, валідація відповідними академічними органами. Це забезпечує, що кожен навчальний план є валідним, реалістичним та відповідає очікуванням як студентів, так і навчальної установи.

В результаті, вихідними даними стає оптимізована структурно-логічна схема, що враховує як загальні освітні цілі, так і особливості здобувача освіти. Ця схема може бути легко інтегрована в освітні системи для забезпечення більш ефективного навчання.

2.2 Особливості використання генетичного алгоритму в задачі побудови структурно-логічної схеми освітніх компонентів

Цей підрозділ присвячений детальному розгляду особливостей адаптації генетичного алгоритму для задачі побудови структурно-логічної схеми освітніх компонентів. Така схема представляє собою комплексну модель, яка спрямована на систематизацію навчальних дисциплін, курсів та інших освітніх елементів у збалансовану і логічно організовану структуру, що відповідає цілям і завданням

специфічної освітньої програми. При створенні таких схем виникає багато викликів, пов’язаних із необхідністю врахування великої кількості факторів, таких як пререквізити між курсами, індивідуальні освітні потреби студентів. Генетичні алгоритми, зі здатністю ефективно оптимізувати рішення в умовах великої кількості обмежень та критеріїв, пропонують унікальні можливості для розробки та вдосконалення структурно-логічних схем освітніх програм.

Поняття «популяція» в контексті генетичного алгоритму передбачає множину потенційних рішень задачі, які розглядаються і еволюціонують з часом. У випадку побудови структурно-логічної схеми освітніх компонентів, популяцією може слугувати набір з декількох навчальних планів, кожен з яких намагається досягти оптимальної структури та збалансованого навантаження серед дисциплін (рисунок 2.2).

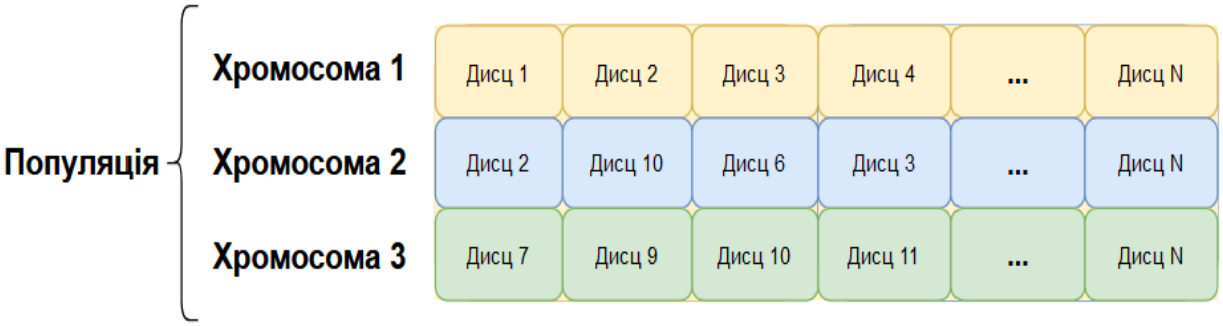


Рисунок 2.2 – Приклад структури популяції

Хромосома в генетичному алгоритмі є представленням індивідуального рішення чи індивідуума. Хромосома містить «генетичний код», який кодує інформацію про характеристики цього рішення. У нашому прикладі, кожна хромосома представлятиме окремий навчальний план, де генетичні елементи (гени) кодують вибір та розташування навчальних дисциплін (рисунок 2.3).

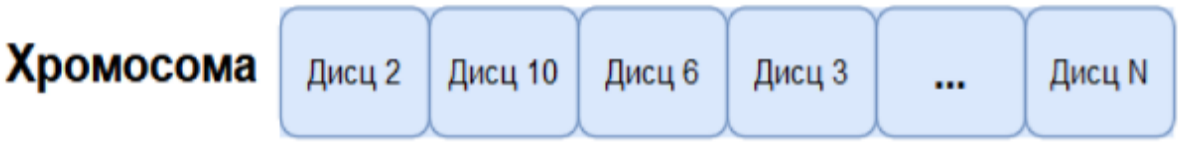


Рисунок 2.3 – Приклад структури хромосоми

Геном у хромосомі є базовою одиницею інформації, що визначає певну характеристику індивідуума. У контексті нашої задачі, кожен ген відповідає за конкретну навчальну дисципліну в навчальному плані, зазначаючи її характеристики, такі як назва, семестр, кредити та будь-які специфічні властивості, такі як пререквізити, компетенції які покриває дисципліна (рисунок 2.4).

Рисунок 2.4 – Приклад структури гена

На самому початку роботи генетичного алгоритму ключовим завданням є створення стартової популяції, яка використовуватиметься для подальших етапів оптимізації. Стартова популяція формується з набору навчальних планів, кожен з яких складається із конкретних дисциплін (рисунок 2.5), пов’язаних із обраною спеціальністю користувача:

1. Створення кожного навчального плану у стартовій популяції розпочинається з інтеграції усіх обов’язкових дисциплін, які є необхідними для здобуття відповідної кваліфікації за спеціальністю. Це забезпечує, що кожен план відповідає базовим академічним вимогам.

2. Після інтеграції обов’язкових дисциплін до структури плану додаються додаткові, вибіркові дисципліни. Вибір цих дисциплін відбувається випадковим чином з урахуванням загального обсягу ECTS кредитів, що необхідно набрати для завершення навчання. Цей процес триває до тих пір, поки загальний ліміт кредитів не буде вичерпаний.
3. Для кожної дисципліни, яка включена до навчального плану, здійснюється перевірка на наявність та включення відповідних пререквізитів. Це означає, що перед додаванням певної дисципліни до плану враховуються всі необхідні передумови її вивчення, що забезпечує логічну послідовність і підготовку студентів до її освоєння.

Рисунок 2.5 – Схема будови стартової популяції

За реалізацію розподілу дисциплін за семестрами та роками з урахуванням обов’язкових пререквізитів відповідає окремий алгоритм (рисунок 2.6). Мета цього алгоритму полягає у забезпеченні збалансованої та спланованої структури навчальних планів. Алгоритм складається з таких етапів:

1. Створюється граф, де кожен вузол відповідає окремому курсу. Граф використовується для зберігання інформації про курси та їх пререквізити.

2. Для кожного курсу визначаються його пререквізити. До прикладу, якщо курс «**B**» є пререквізитом для курсу «**A**», то між ними створюється направлений зв’язок в графі з «**B**» до «**A**». Це означає, що перед вивченням «**A**» необхідно завершити «**B**».
3. Кожному курсу, що не має пререквізитів (тобто початковим вузлам графа, індеград яких дорівнює нулю), призначається початковий семестр і рік. Вони використовуються як основа для подальшого розподілу по семестрах і роках.
4. Алгоритм починає обхід з курсів, що не мають пререквізитів, і послідовно переходить до курсів, які вони відкривають, зменшуючи рівень індеграда наступних курсів. Як тільки індеград курсу досягає нуля, цей курс може бути вивчений.
5. Курси розподіляються по семестрам. Кожен семестр може включати кілька курсів на основі їх залежностей, послідовності вивчення та обмежень кредитів. Після завершення двох семестрів, рік збільшується на одиницю.

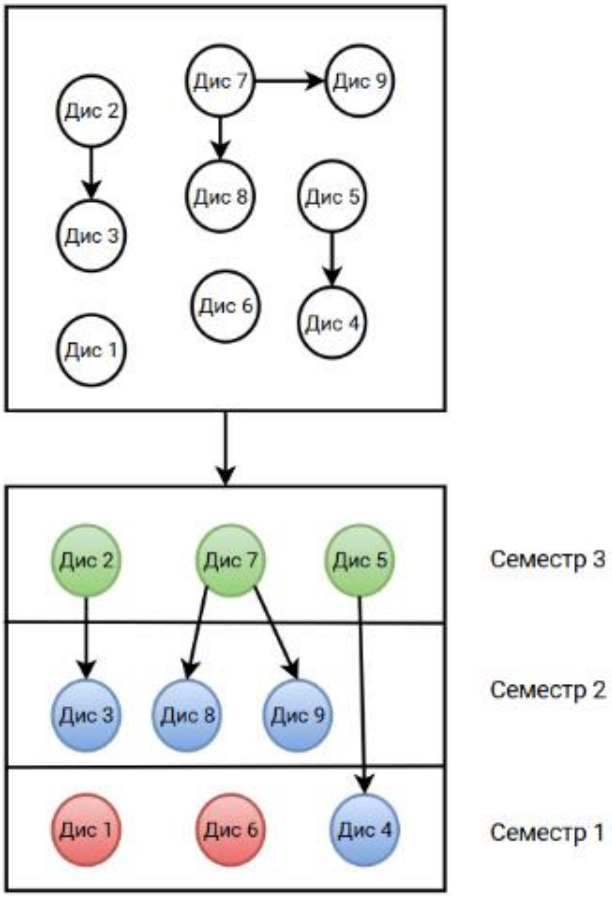


Рисунок 2.6 – Схема роботи алгоритму розподілу дисциплін

Після формування стартового набору навчальних планів, наступним критично важливим етапом у роботі алгоритму є розрахунок функції пристосованості для кожного навчального плану. Цей крок забезпечує вимірювання ефективності кожного потенційного рішення згідно з заданими критеріями, включаючи відповідність обраним компетентностям та оптимізацію загального навантаження за кредитами ECTS. Правильно розрахована функція пристосованості відіграє ключову роль у визначенні успішності різних навчальних планів, а також в подальшому відборі та схрещуванні особин для генерації нових, потенційно кращих рішень. Розрахунок функції пристосованості здійснюється в кілька етапів (рисунок 2.7).

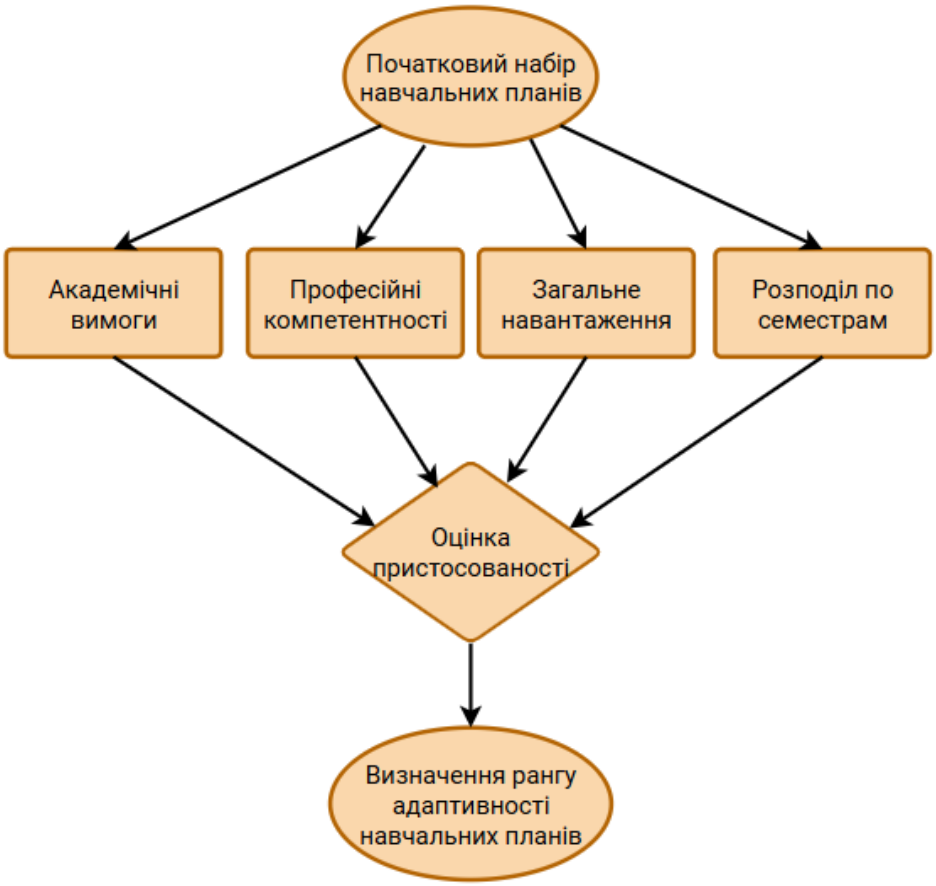


Рисунок 2.7 – Схема розрахунку функції пристосованості

Індивідуальний ранг кожного навчального плану встановлюється на основі перевірки, наскільки добре він відповідає не лише базовим академічним вимогам, а й прагненню студента до отримання певних професійних компетентностей, також

оцінюється загальний обсяг кредитів ECTS, а також розподіл між роками навчання та семестрами, щоб ідентифікувати потенційні перевантаження або дисбаланс. Комбінуючи ці фактори, визначається кінцевий показник пристосованості для кожного навчального плану. Вища пристосованість вказує на більшу відповідність навчального плану до встановлених критеріїв та, відповідно, вищий потенціал для включення у наступні генерації. Ефективність цього кроку безпосередньо впливає на здатність алгоритму ітеративно вдосконалювати навчальні плани, просуваючись до знаходження найбільш оптимізованих і ефективних навчальних структур.

Після ретельного аналізу та оцінення кожного навчального плану, всередині набору, наступним етапом роботи генетичного алгоритму є селекція. Селекція – це процес вибору особин, які будуть використовуватися для генерації наступного покоління через схрещування. Основна мета цього кроку – обрати найбільш пристосованих особин, що відповідають встановленим критеріям, для того, щоб вони передавали генетичні характеристики майбутнім поколінням, сприяючи таким чином покращенню якості навчальних планів в цілому.

Турнірна селекція – це метод селекції, що передбачає відбір групи особин, випадковим чином, і та особина, що має найвищу пристосованість у групі, обирається для схрещування (рисунок 2.8). Метод дає можливість контролювати тиск селекції через розмір турніру.

За допомогою цього методу визначаються пари навчальних планів, які будуть батьками наступного покоління. Під час селекції важливо підтримувати різноманіття дисциплін всередині наборів, щоб уникнути ранньої конвергенції до непотрібних локальних оптимумів. Різноманіття є запорукою вдалих інноваційних рішень, які можуть бути знайдені тільки через пошук у широких областях простору рішень. У зв’язку з цим, процес селекції є критичним етапом генетичного алгоритму, коли потрібно балансувати між забезпеченням успіху найбільш пристосованих особин та збереженням достатнього рівня різноманіття для здорової еволюції наборів. Саме цей крок ставить перед алгоритмом завдання не просто відібрати «найкращих» в поточному поколінні, а й забезпечити потенціал для подальшого розвитку та вдосконалення.

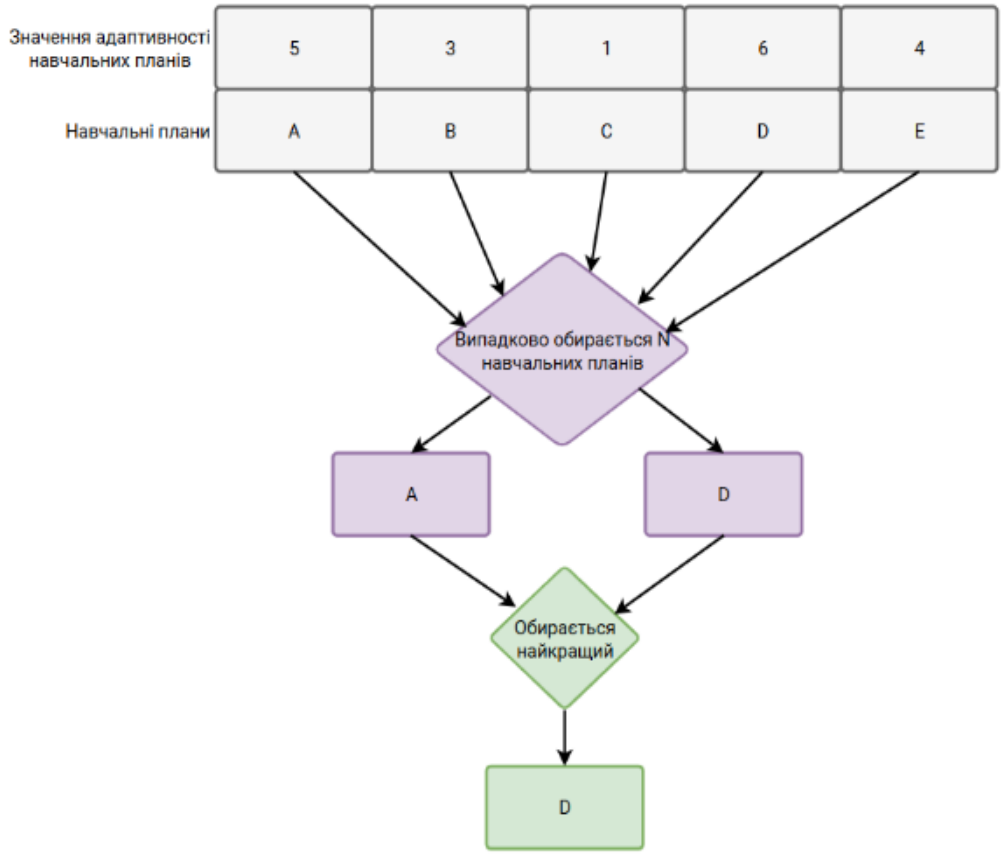


Рисунок 2.8 – Метод турнірного відбору

Після того, як було здійснено відбір, наступний критичний етап у роботі генетичного алгоритму – це схрещування. Схрещування відіграє ключову роль у генерації нових наборів, об’єднуючи генетичний матеріал батьківських хромосом для створення нових навчальних траєкторій із новими характеристиками. У контексті побудови структурно-логічних схем освітніх компонентів, схрещування має за мету комбінувати елементи різних навчальних планів для знаходження оптимальної структури курсів та їх розкладу.

Уніформне схрещування є особливим методом, що дозволяє кожному елементу «дитячого» навчального плану бути наслідуваним з випадкового «батька» з рівною ймовірністю для кожної дисципліни (рисунок 2.9). Це означає, що кожен елемент нової траєкторії обирається незалежно, від одного з двох батьків, за допомогою випадкового вибору. Процес уніформного схрещування складається з таких етапів:

1. Для кожного акту схрещування спочатку визначаються два навчальні плани за методом селекції, описаним на попередньому етапі.
2. Кожна дисципліна в новому навчальному плані випадковим чином обирається від одного з двох батьківських планів. Оскільки вибір відбувається з рівною ймовірністю, це забезпечує різноманітність новоствореного рішення.

Уніформне схрещування допомагає підтримувати високий рівень різноманіття, в навчальних траєкторіях, що особливо важливий для уникнення локальних мінімумів пристосованості. Кожна дисципліна має однаковий шанс бути обраною для передачі новостворениим навчальним планам, що сприяє ретельному дослідженню простору рішень. Уніформне схрещування ефективно застосовується у задачах побудови структурно-логічних схем освітніх компонентів для експериментів з різноманітними поєднаннями навчальних дисциплін і виявлення найбільш оптимальних структур навчальних планів. Цей механізм найкраще працює в умовах, коли необхідно вивчати великий простір можливих рішень з метою ідентифікації найефективніших навчальних стратегій.

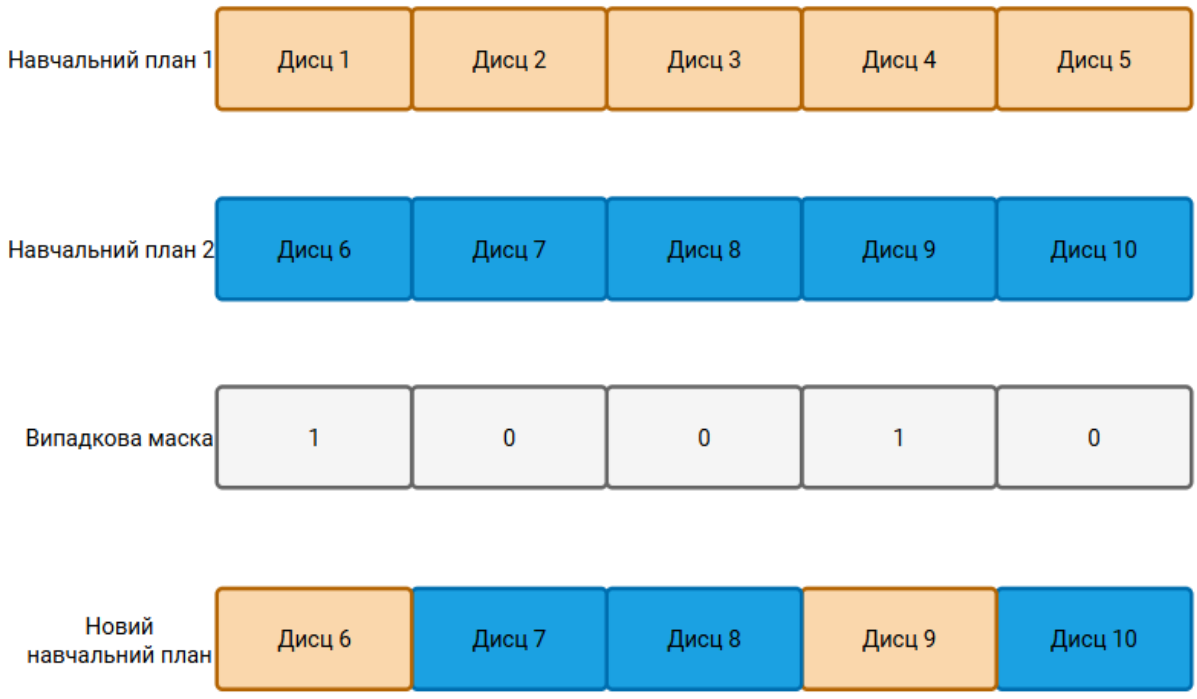


Рисунок 2.9 – Схема уніформного схрещування

Мутація у генетичному алгоритмі слугує як засіб внесення новизни та диверсифікації до навчальних траєкторій, що може допомогти долати потенційні локальні мінімуми та збільшити шанси на знаходження глобального оптимального рішення (рисунок 2.10). В контексті побудови навчальних планів, мутація дозволяє досліджувати нові комбінації дисциплін та їх розклад з метою виявлення ефективніших освітніх схем. Однак, для забезпечення продуктивності такого підходу, метод мутації потребує обережного та виваженого застосування. Мутація виконується наступним чином:

1. Серед усіх освітніх наборів випадковим чином обирається однин або декілька навчальних планів, які підлягатимуть мутації. Ймовірність мутації зазвичай встановлюється низькою, щоб забезпечити, що більшість траєкторій залишаються незмінними, зберігаючи цінний генетичний матеріал.
2. У обраного навчального плану замінюється одна дисципліна на іншу випадково обрану дисципліну, яка до цього моменту не була його частиною. Важливо, щоб нова дисципліна відповідала загальним вимогам програми.

Хоча випадковість мутації може здатися ризикованою, вона необхідна компонента для повноцінного функціонування генетичного алгоритму, надаючи необхідний рівень експерименту у пошуках ідеального навчального плану. Таким чином, уміло збалансоване застосування мутації сприяє підвищенню адаптивності та ефективності освітніх програм.

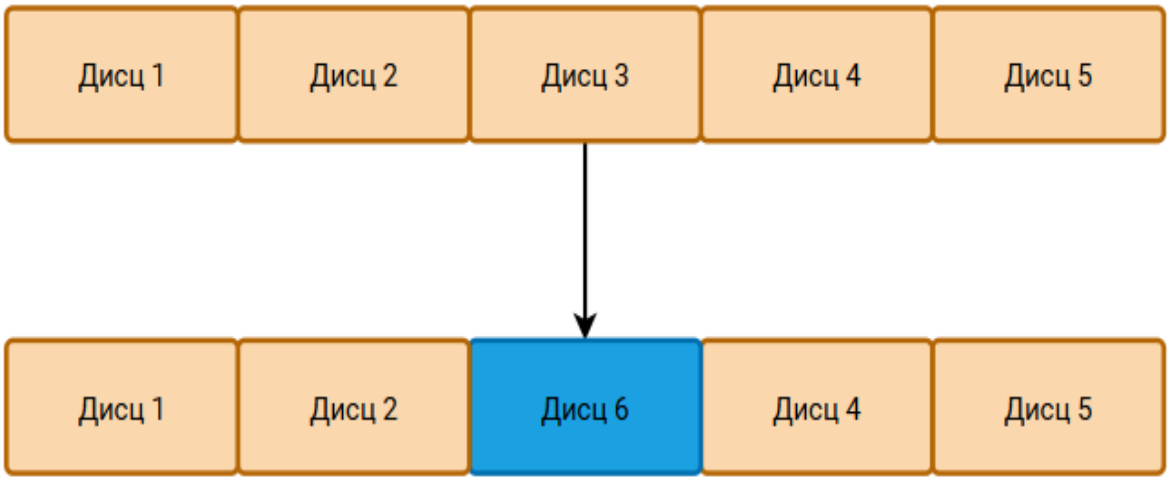


Рисунок 2.10 – Схема роботи методу мутації

Після завершення процесів селекції, схрещування та мутації навчальних планів, генетичний алгоритм переходить до формування нового пулу освітніх схем, що випливає з цих операцій (рисунок 2.11). Цей крок є досить важливим в еволюційному циклі, де визначається які навчальні плани будуть збережені. Процес формування нового пулу базується на рангуванні навчальних траєкторій, що відображає їхню загальну пристосованість та успішність. Рангування особин здійснюється на основі функції пристосованості, навчальні плани з вищою пристосованістю отримують кращі ранги, тим самим збільшуючи шанс бути обраними для нового пулу навчальних планів. При цьому здійснюються наступні дії:

1. Деяка кількість найкращих навчальних траєкторій, попереднього покоління, одразу проходить в новий набір, забезпечуючи збереження найсильніших характеристик.
2. Решта навчальних планів буде призначена на основі результатів селекційного процесу для підвищення різноманітності та потенціалу до розвитку.
3. Отриманий новий набір навчальних планів аналізується на предмет збереження генетичного різноманіття та загальної пристосованості. За потреби коригується структура освітніх схем для вирішення ідентифікованих проблем.

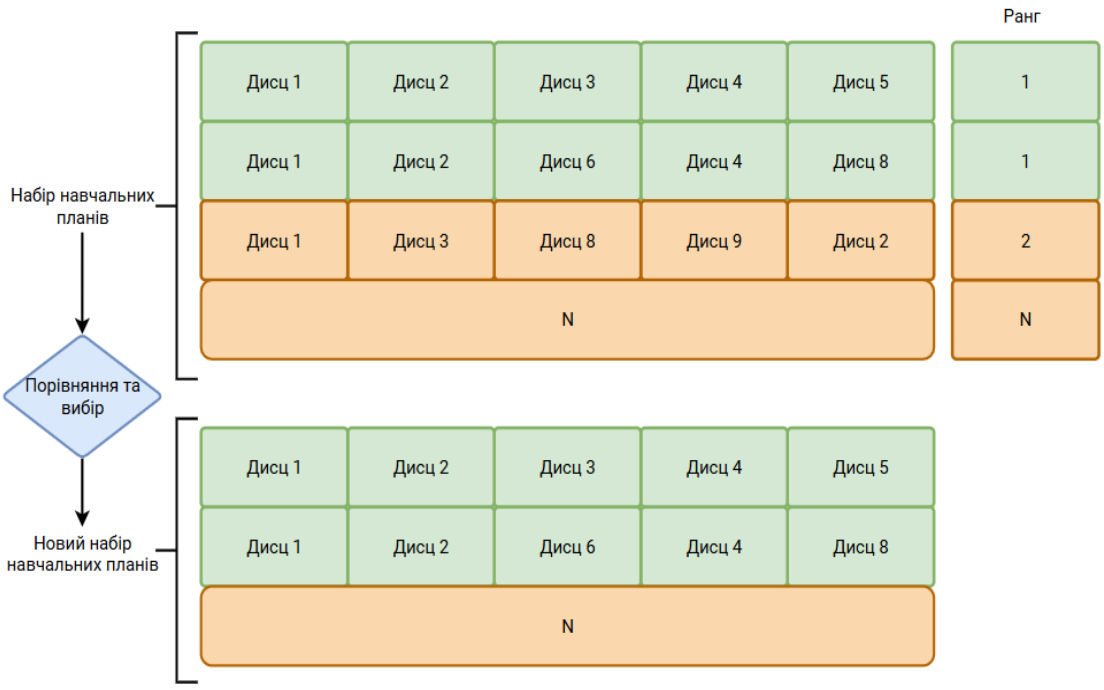


Рисунок 2.11 – Процес формування нового пулу навчальних планів

Останнім критичним етапом в роботі генетичного алгоритму є вибір оптимального рішення з новоствореного набору освітніх схем. Після завершення циклу еволюції алгоритм повинен визначити найкраще рішення. Цей вибір втілює кульмінаційний крок алгоритмічного процесу, де оцінюються ранг та пристосованість кожного навчального плану з фінального набору (рисунок 2.12).

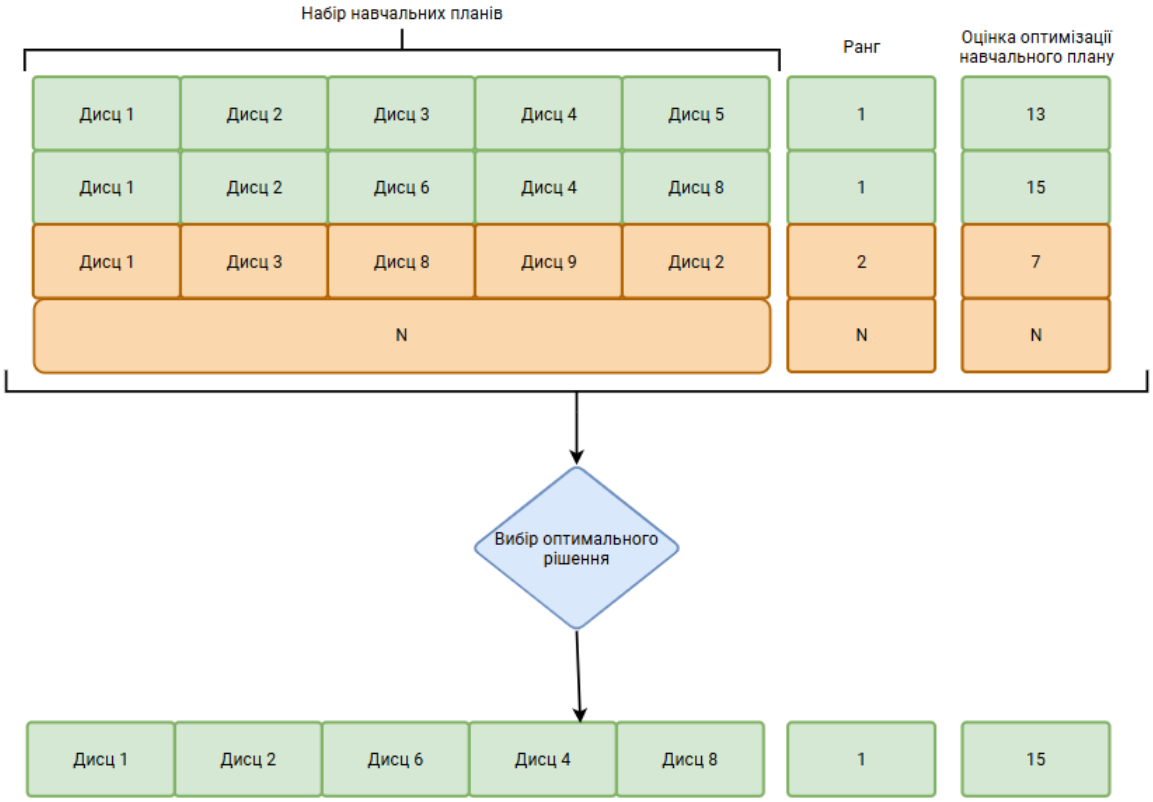


Рисунок 2.12 – Демонстрація процесу вибору оптимального рішення

Вибір оптимального рішення базується на порівняльному аналізі рангу та показників ефективності, що визначаються функцією пристосованості кожного окремого навчального плану. Траєкторії з вищим рангом мають перевагу, оскільки демонструють кращу адаптацію до встановлених вимог:

1. Переглядаються ранги всіх освітніх схем у фінальному наборі навчальних схем. Особини з вищим рангом отримують перевагу, як потенційні кандидати на оптимальне рішення.

2. Детальний аналіз показників оцінки пристосованості здійснюється для ідентифікації освітніх траєкторій, які найкраще задовольняють задані критерії ефективності та якості.
3. Кінцеве рішення обирається на основі балансу між високим рангом та високими показниками пристосованості.

Фактори, що враховуються при виборі оптимального рішення (рисунок 2.13):

1. Наскільки добре рішення відповідає загальним освітнім цілям та вимогам програми.
2. Забезпечення різноманітності дисциплін та балансу навчального навантаження.
3. Ефективне розподілення курсів, оптимізація використання часових та людських ресурсів.

Рисунок 2.13 – Схема критеріїв вибору оптимального рішення

Визначення оптимального рішення є ключовим для досягнення мети генетичного алгоритму – знайти найефективніший варіант розв’язку задачі, в даному випадку, розробки структурно-логічної схеми освітніх компонентів. Таким чином, алгоритм виявляє рішення, котрі не лише відповідають зазначеним критеріям, а й мають потенціал для подальших інновацій та розвитку.

2.3 Імплементація генетичного алгоритму у процес побудови структурно-логічної схеми освітніх компонентів

Імплементація ГА в процес побудови структурно-логічної схеми освітніх компонентів передбачає використання систематичного підходу для покращення якості та ефективності освітньої програми. Розглядаючи весь процес у цілісності, від аналізу вхідних даних до отримання вихідних результатів, стає зрозуміло, що алгоритм може бути використаний, як на етапі формування початкового набору навчальних планів, так і на етапі формування індивідуальної послідовності.

Процес формування початкового набору навчальних планів є важливим етапом в побудові ефективної і адаптивної освітньої структури. Першим кроком є детальний аналіз доступних вхідних даних, які включають інформацію про студентів, освітні середовища і академічні вимоги. Це дозволяє ідентифікувати ключові параметри для формування навчальних планів. На основі аналізу формуються основні компоненти, які мають бути включені в кожний навчальний план. Ці компоненти поділяються на обов’язкові та вибіркові дисципліни з урахуванням академічних правил та обмежень, після цього стає доступною можливість генерувати початкові варіанти навчальних планів, які різняться композицією та послідовністю дисциплін. На цьому етапі особлива увага приділяється забезпеченню відповідності планів до встановлених критеріїв та обмежень.

Після того як були сформовані початкові набори освітніх планів стає можливим формування індивідуальних послідовностей. На цьому етапі алгоритм оптимізує кожен навчальний план, відштовхуючись від індивідуальних потреб студента, розподілу навчального навантаження та необхідності врахування пререквізитів для курсів. Це забезпечує створення та підбір найбільш відповідного навчального плану для кожного учня. Розглянемо детальніше кожен з цих аспектів:

1. Визначення специфічних освітніх цілей та уподобань кожного студента. Це може включати інтерес до певних дисциплін, професійних амбіцій, тощо. Такий аналіз дозволяє ідентифікувати ключові компоненти для формування індивідуалізованих навчальних планів.

2. Важливим аспектом під час розробки індивідуальної послідовності є забезпечення збалансованого розподілу навчального навантаження. Це означає створення такого розкладу, щоб студент міг ефективно впоратися з навчальним матеріалом, не відчуваючи перевантаження.
3. Врахування пререквізитів при виборі курсів. Пререквізити гарантують, що студенти мають необхідні знання та навички для успішного проходження курсу. Під час формування індивідуальної послідовності слід забезпечити, щоб всі необхідні пререквізити були включені до індивідуального плану, до початку більш складних курсів.

Імплементація алгоритму у вказані етапи процесу побудови структурно-логічної схеми освітніх компонентів створює потенціал для значних покращень, забезпечуючи освітнім закладам засіб для реалізації більш персоналізованих і гнучких навчальних програм.

2.4 Критерії оцінки підвищення якості формування побудови структурно-логічної схеми освітніх компонентів

Імплементація інноваційних алгоритмів в освітні системи має потенціал значно покращити їхню ефективність та адаптивність. При введенні алгоритму для автоматизації складання оптимізованих навчальних планів, важливо мати чіткі критерії, за якими ми можемо оцінити внесок і покращення, які це впровадження принесло.

Економія часу є одним із найбільших покращень, яке імплементація нового алгоритму може принести в освітню систему, особливо у контексті формування та оптимізації навчальних планів. Автоматизація процесів, що раніше вимагали значних часових витрат від викладачів та адміністрації, може звільнити їх ресурси для зосередження на більш якісних аспектах навчання, таких як розвиток нових курсів, індивідуальна робота зі студентами тощо. В традиційному підході складання навчального плану з врахуванням різних навчальних блоків, потребує багато часу для ручного аналізу та ухвалення рішень. Алгоритм, за допомогою методів оптимізації,

може автоматично створювати збалансовані навчальні плани, з урахуванням академічних вимог та індивідуальних потреб студентів, тим самим значно зменшуючи час, що витрачається на даний процес. Зазвичай пошук оптимального рішення може тривати декілька робочих днів, особливо у великих освітніх установах. Алгоритмізоване рішення дозволяє проаналізувати сотні можливих варіантів за лічені хвилини, обираючи найбільш ефективний сценарій, що задовольнить усіх учасників освітнього процесу. Автоматична перевірка планів на відповідність стандартам і правилам дозволяє швидко ідентифікувати помилки чи колізії, що можуть виникнути під час ручного складання планів, тим самим сприяючи їх оперативному виправленню без значних часових витрат.

Пререквізити є важливим чинником для забезпечення послідовності освітнього процесу, дозволяючи студентам послідовно набувати знань та компетенцій, підвищуючи ефективність навчання. Впровадження алгоритму, який чітко враховує ці пререквізити під час формування навчальних планів, забезпечує плавний процес навчання і дозволяє уникнути перевантаження студентів інформацією, яку вони ще не готові засвоїти. Інтеграція пререквізитів у процес формування індивідуальних освітніх траєкторій дозволяє автоматизувати створення оптимальних навчальних планів таким чином, що кожен наступний курс підтримує та розширює знання, отримані в попередніх. Це допомагає створити більш ефективний та змістовний навчальний досвід для студентів. Впровадження алгоритму, що автоматично враховує пререквізити, забезпечує цілісність навчального процесу, оптимізує організацію навчання, і значно полегшує роботу викладачів та адміністрації з планування навчальних програм. Відповідно, це стає одним із ключових критеріїв оцінки покращення якості освітнього процесу.

Імплементація алгоритму має зробити освітні системи більш гнучкими, здатними враховувати індивідуальні потреби і переваги студентів. Оцінка адаптивності може базуватися на здатності системи пропонувати персоналізовані навчальні плани та швидко реагувати на запити зміни в курсах або навчальних потребах.

Комплексний ефект від цих покращень не тільки звільняє дорогоцінний час, але й підвищує загальну ефективність навчального процесу, створюючи можливості для розвитку та вдосконалення освітніх програм та методик навчання.

2.5 Висновки другого розділу

У другому розділі було розроблено метод побудови структурно-логічної схеми освітніх компонентів, що базується на застосуванні генетичного алгоритму.

Було адаптовано генетичний алгоритм для вирішення задачі побудови найкращої структурно-логічної схеми освітніх компонентів. Генетичний алгоритм було інтегровано до процесу побудови структурно-логічної схеми освітніх компонентів.

Визначено ключові критерії, за якими оцінюються покращення в освітній системі, внесені застосуванням алгоритму. Такими критеріями визначено економія часу, підвищення адаптивності, заберпечення повноти охоплення та послідовності опанування освітніх компонент.

РОЗДІЛ 3 Програмна реалізація методу

3.1 Опис програмної реалізації

Враховуючи, що більшість користувачів смартфонів, персональних комп'ютерів та інших мобільних пристроїв віддають перевагу використанню програмного забезпечення через веб-інтерфейси, було вирішено програмно реалізувати метод у вигляді веб-застосунку. Це пояснюється тим, що веб-додатки володіють високою швидкістю відгуку, зручністю у користуванні та легкістю розширення функціоналу. Розроблений веб-сайт передбачає наявність двох основних ролей: адміністратора та користувача в ролі студента.

Адміністрація веб-сайту відіграє ключову роль у керуванні процесом створення та адміністрування освітніх матеріалів. Адміністратор відповідає за попереднє налаштування сутностей, що будуть використовуватись для побудови структурно-логічних схем освітніх компонентів. Адміністратору надаються інструменти для створення та управлінням сутностями застосунку, такими як: факультети, кафедри, вчителі, дисципліни та компетентності (рисунок 3.1).

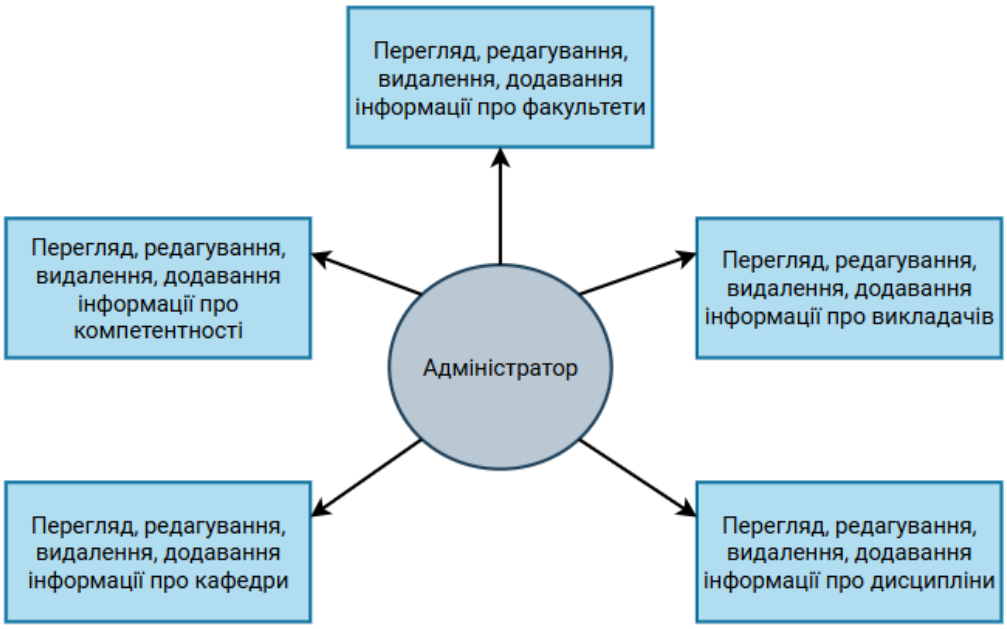


Рисунок 3.1 – Функції адміністратора

Роль студента дозволяє персоналізувати власний навчальний процес за допомогою вибору зі списку доступних компетентностей, які користувач бажає опанувати, результатом буде формування структурно-логічної схеми освітніх компонентів, що відповідатиме особистим академічним інтересам (рисунок 3.2).

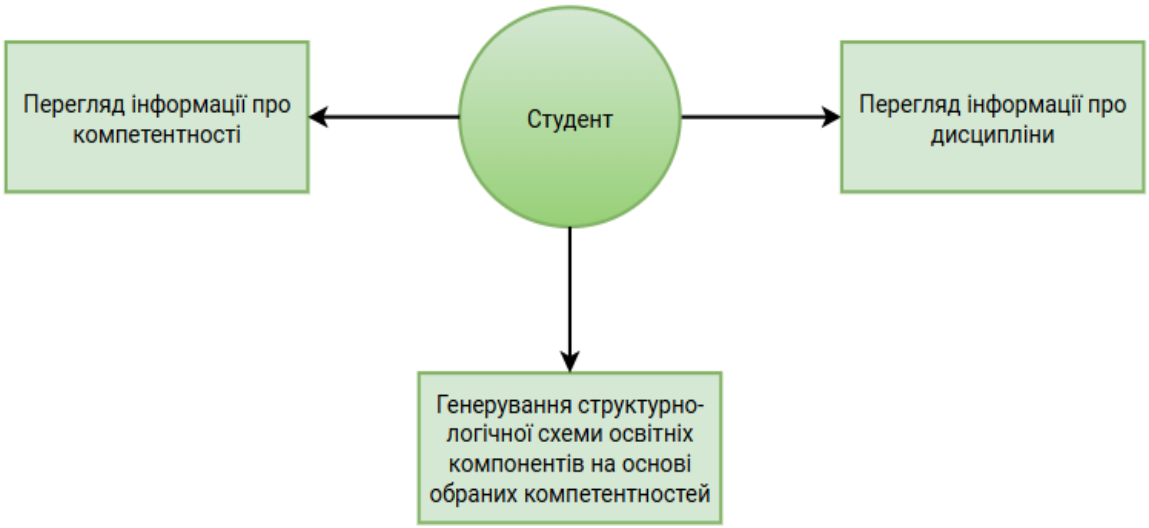


Рисунок 3.2 – Функції студента

Для забезпечення функціональності та взаємодії користувачів із веб-застосунком необхідно створити структуровану систему сутностей бази даних. Ці сутності є ключовими компонентами управління даними, дозволяючи ефективно зберігати, оновлювати, видаляти та відновлювати інформацію, необхідну для освітнього процесу. Нижче представлено опис основних сутностей, з якими буде працювати застосунок:

1. Користувач.
2. Роль.
3. Факультет.
4. Спеціальність.
5. Компетентність.
6. Викладач.
7. Дисципліна.

Сутність користувача зберігає інформацію про кожного, хто взаємодіє з системою. Це можуть бути студенти або адміністратори. Для кожного користувача зберігаються особисті дані, логін та пароль для входу в систему, спеціальність, а також посилання на роль, яку він виконує в контексті застосунку (рисунок 3.3).

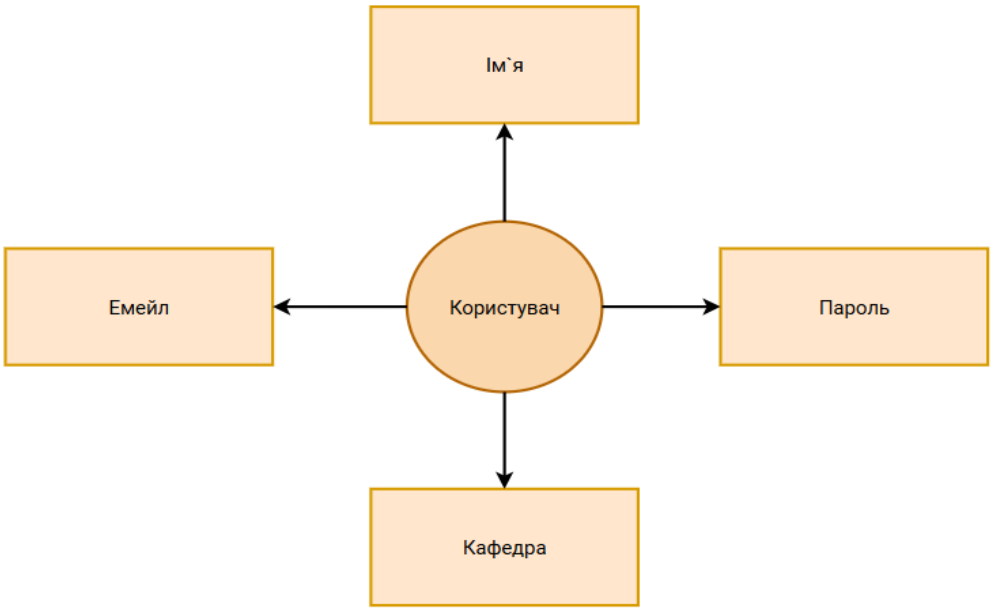


Рисунок 3.3 – Атрибути користувача

Сутність «Роль» визначає рівень доступу та можливості, які має користувач у системі. Типові ролі можуть включати адміністратора, який має доступ до управління всіма компонентами системи, та студента, який може переглядати матеріали та формувати власний навчальний план (рисунок 3.4).

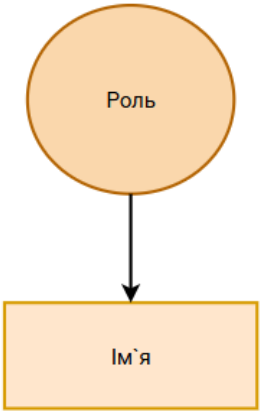


Рисунок 3.4 – Атрибути сутності ролі

Сутність «Факультет» представляє структурний підрозділ вищого навчального закладу, що об’єднує кілька спеціальностей. Для кожного факультету фіксуються такі характеристики, як назва факультету та набір спеціальностей (рисунок 3.5).

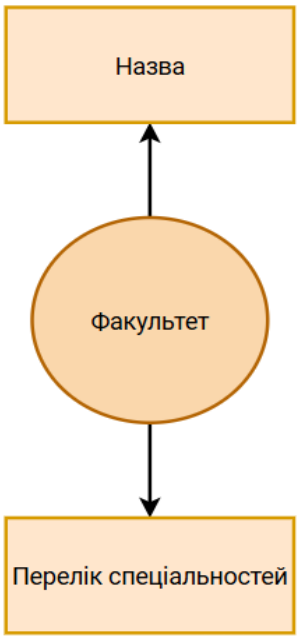


Рисунок 3.5 – Атрибути факультету

Сутність «Спеціальність» відображає конкретний освітній напрямок, що забезпечує певний набір знань і компетенцій. Ці дані включають назву спеціальності, відповідний факультет та перелік обов’язкових дисциплін, що входять до освітньої програми (рисунок 3.6).

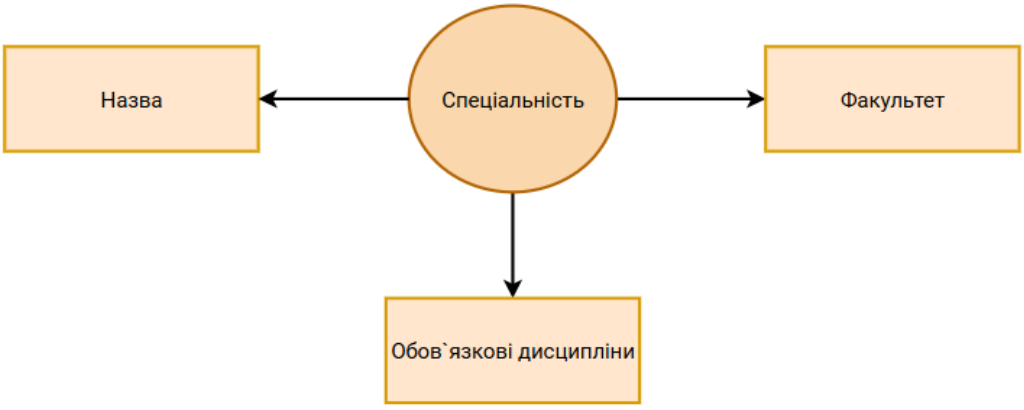


Рисунок 3.6 – Атрибути спеціальності

Сутність навчальної дисципліни містить інформацію про курси, що входять до навчальних планів. Кожна дисципліна описується унікальною назвою, обсягом в кредитах ECTS, закріпленим викладачем, приналежністю до відповідної спеціальності, списком необхідних пререквізитів, а також переліком компетентностей, які студент здобуває після вивчення курсу (рисунок 3.7).

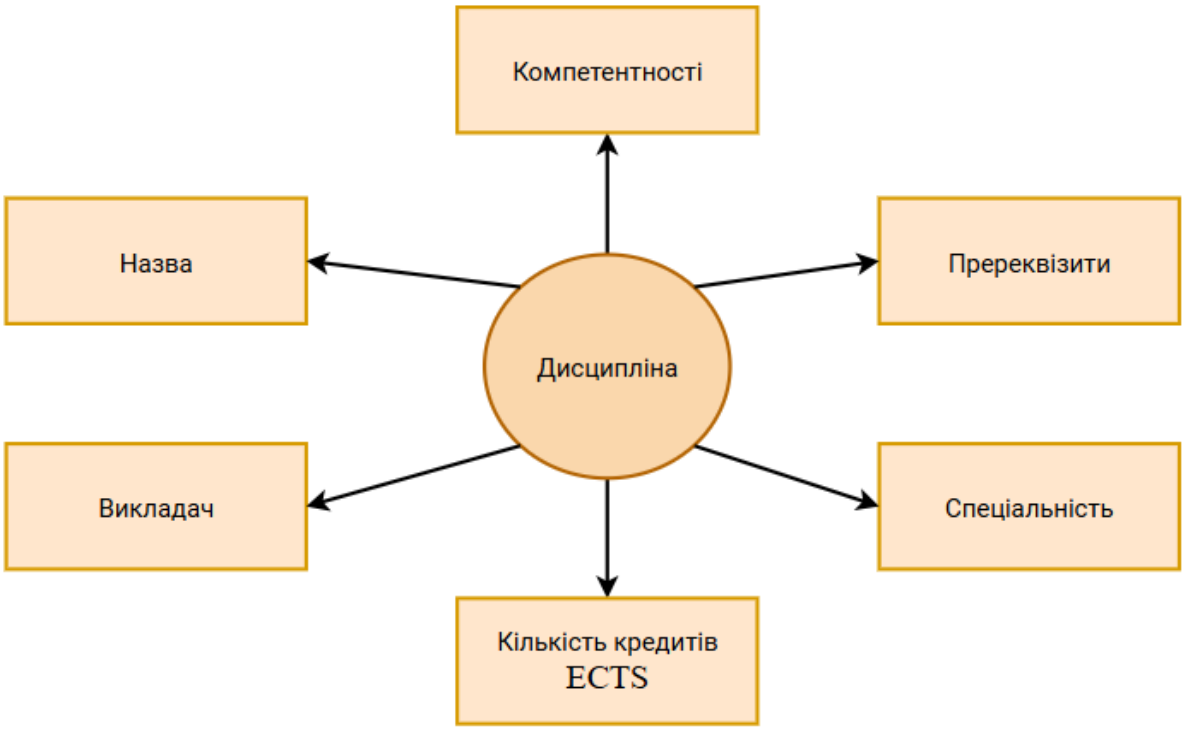


Рисунок 3.7 – Атрибути навчальної дисципліни

Сутність «Компетентність» зберігає дані про конкретні знання, уміння та навички, які студент отримує під час проходження навчального курсу. Включає в себе опис компетентності та позначку, що визначає обов’язковість компетентності (рисунок 3.8).

Сутність «Викладач» описує дані викладача навчального закладу. Кожен викладач має такі характеристики, як: ім’я ти список дисциплін в яких спеціалізується (рисунок 3.9).

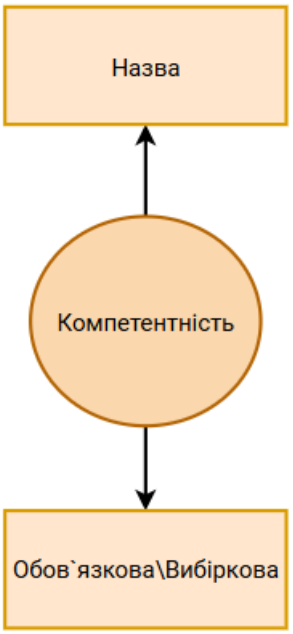


Рисунок 3.8 – Атрибути сутності «Компетентність»

Рисунок 3.9 – Атрибути викладача

Розробка та впровадження цих сутностей у БД є ключовим для створення комплексної інформаційної системи, здатної ефективно керувати освітнім процесом. Перейдемо тепер до розгляду конкретних прикладів таблиць БД, які інкапсулюють структуру описаних вище сутностей (таблиця 3.1 – 3.10). Це дозволить глибше

зрозуміти, як інформаційна система організує та керує даними для оптимізації освітнього процесу.

Таблиця 3.1 – Атрибути таблиці «Users»

№ п/п	Назва атрибуту	Тип даних	Опис
1.	Id	int	Первинний ключ для однозначної ідентифікації записів у таблиці
2.	Name	nvarchar	Ім’я користувача
3.	Email	nvarchar	Електрона пошта користувача
4.	Password	nvarchar	Пароль користувача
5.	ChairId	int	Вторинний ключ (кафедра)

Таблиця 3.2 – Атрибути таблиці «Roles»

№ п/п	Назва атрибуту	Тип даних	Опис
1.	Id	int	Первинний ключ для однозначної ідентифікації записів у таблиці
2.	Name	nvarchar	Назва системної ролі

Таблиця 3.3 – Атрибути таблиці «UserRoles»

№ п/п	Назва атрибуту	Тип даних	Опис
1.	UserId	int	Вторинний ключ (користувач)
2.	RoleId	int	Вторинний ключ (роль)

Таблиця 3.4 – Атрибути таблиці «Faculty»

№ п/п	Назва атрибуту	Тип даних	Опис
1.	Id	int	Первинний ключ для однозначної ідентифікації записів у таблиці
2.	Title	nvarchar	Назва факультету

Таблиця 3.5 – Атрибути таблиці «Chairs»

№ п/п	Назва атрибуту	Тип даних	Опис
1.	Id	int	Первинний ключ для однозначної ідентифікації записів у таблиці
2.	Title	nvarchar	Назва кафедри
3.	FacultyId	int	Вторинний ключ (факультет)

Таблиця 3.6 – Атрибути таблиці «Teachers»

№ п/п	Назва атрибуту	Тип даних	Опис
1.	Id	int	Первинний ключ для однозначної ідентифікації записів у таблиці
2.	Name	nvarchar	Ім’я викладача

Таблиця 3.7 – Атрибути таблиці «Courses»

№ п/п	Назва атрибуту	Тип даних	Опис
1.	Id	int	Первинний ключ для однозначної ідентифікації записів у таблиці
2.	Title	nvarchar	Назва дисципліни
3.	ECTS	int	Кількість кредитів ECTS
4.	ChairId	int	Вторинний ключ (кафедра)
5.	TeachersId	int	Вторинний ключ (викладач)

Таблиця 3.8 – Атрибути таблиці «Competences»

№ п/п	Назва атрибуту	Тип даних	Опис
1.	Id	int	Первинний ключ для однозначної ідентифікації записів у таблиці
2.	Name	nvarchar	Назва компетентності
3.	IsMandatory	bool	Обов’язковість компетентності

Таблиця 3.9 – Атрибути таблиці «CourseCompetence»

№ п/п	Назва атрибуту	Тип даних	Опис
1.	CourseId	int	Вторинний ключ (дисципліна)
2.	CompetenceId	int	Вторинний ключ (компетентність)

Таблиця 3.10 – Атрибути таблиці «CoursePrerequisite»

№ п/п	Назва атрибуту	Тип даних	Опис
1.	CourseId	int	Вторинний ключ (дисципліна)
2.	PrerequisiteId	int	Вторинний ключ (пререквізит)

З врахуванням вище зазначеної архітектури, потрібно спроектувати директиви для розробки функціоналу застосунку, що спрямований на ефективне керування освітнім циклом через веб-інтерфейс. Така система відкриває питання розробки специфічних програмних модулів, що відповідатимуть потребам користувачів.

Перш за все, вхід в систему потребує точної ідентифікації користувачів з відповідними привілеями, що в свою чергу наводить на думку розробки модуля автентифікації та авторизації. Серед основних завдань цього модуля буде перевірка користувачів з адміністративними правами у базі даних, що гарантуватиме отримання доступу до системи лише верифікованим користувачам. Також, з огляду на відповідальність адміністратора за створення та управління навчальними компонентами, очевидним стає формування модуля управління контентом. Цей модуль об’єднує інструменти для налаштування факультетів, спеціальностей, дисциплін та прив’язки до них відповідних компетентностей. Модуль повинен надати зручний інтерфейс для додавання нових елементів, оновлення існуючих даних або ж видалення застарілих записів. Студенти, у свою чергу, потребують механізму для огляду навчальних компонентів, що робить необхідним розробку модуля перегляду освітніх пропозицій. Це має забезпечити легкий доступ до описів курсів, списку компетентностей, які можна здобути, і формування попереднього навчального плану. Модуль генерації навчальних планів стане завершальною ланкою в цьому ланцюгу, дозволяючи студентам не тільки оглядати потенційні дисципліни, але й ефективно

планувати особисті освітні траєкторії, обираючи навчальні елементи, що відповідають інтересам, здобувачів освіти (рисунок 3.10).

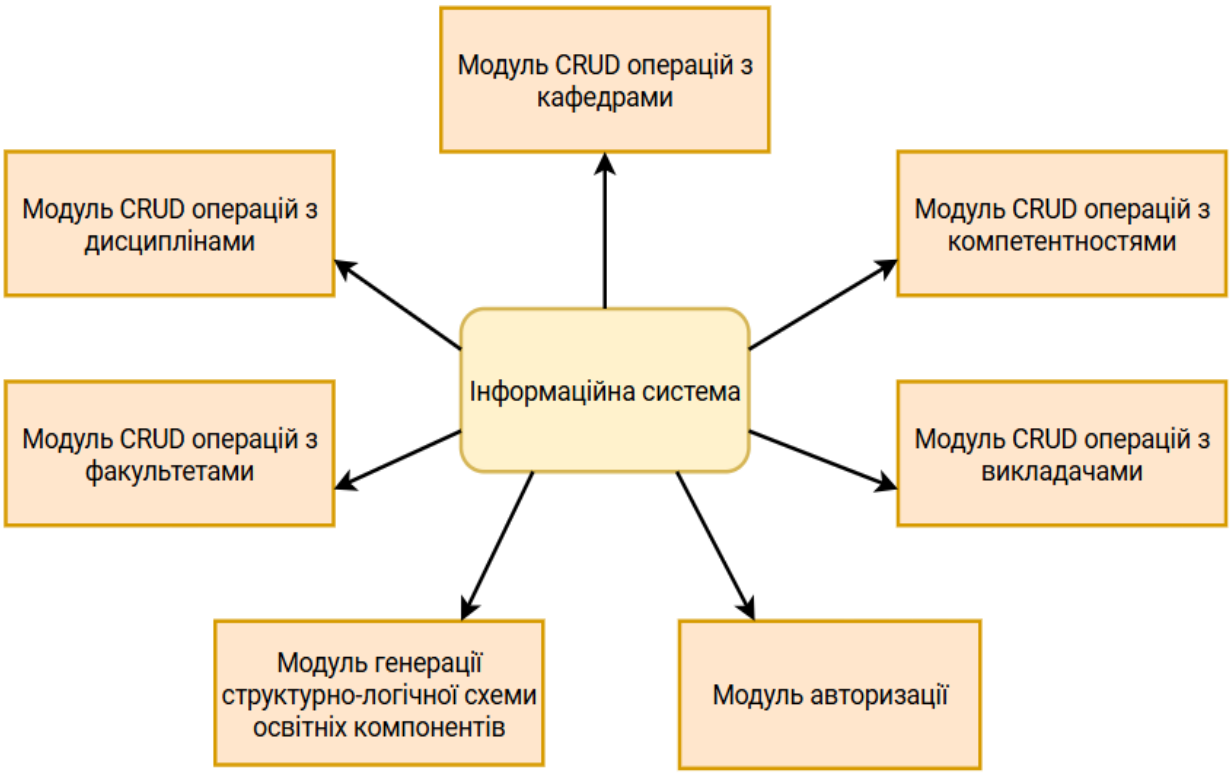


Рисунок 3.10 – Модулі інформаційної системи

Отже, розглянувши базові компоненти та сутності інформаційної системи, постає питання створення низки програмних модулів, котрі задовольняють основні вимоги до функціональності веб-додатку. Розробка такої системи вимагає уваги до деталей авторизації, контент-менеджменту, інтерфейсу для користувачів та алгоритму побудови структурно-логічної схеми освітніх компонентів.

3.2 Структура і функціональне призначення модулів програмної реалізації

Почнемо з модуля авторизації як основного у нашій системі. Модуль авторизації відповідає за верифікацію даних користувачів системи, визначаючи їхні права доступу до різних частин веб-додатку в залежності від їхньої ролі

(адміністратор, студент тощо). Це критично важливо для забезпечення безпеки системи та правильного розподілу функцій між користувачами (рисунок 3.11).

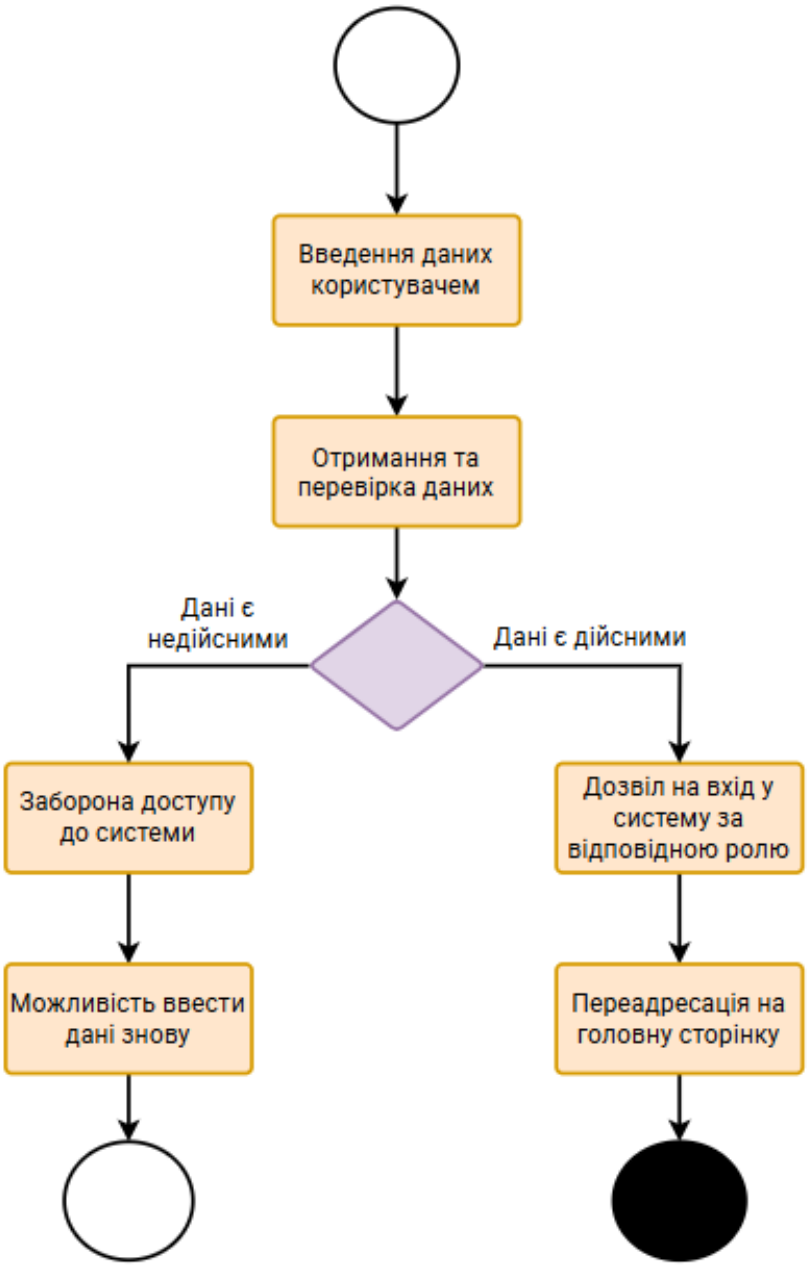


Рисунок 3.11 – Діаграма модуля авторизації

Модулі CRUD є одними з ключових компонентів інформаційної системи, призначені для керування даними факультетів, кафедр, викладачів, дисциплін та компетентностей в рамках вищих навчальних закладів. Цей модуль дозволяє адміністраторам системи виконувати базові операції створення, читання, оновлення

та видалення (CRUD) інформації, що сприяє гнучкому управлінню структурою застосунку.

Створення (Create) – дозволяє адміністратору додавати нову інформацію до системи, вказуючи базову інформацію про сутності. Цей процес сприяє розширенню можливостей застосунку завдяки створенню нових структурних одиниць (рисунок 3.12).

Рисунок 3.12 – Діаграма створення нових записів сутностей

Читання (Read) – забезпечує перегляд списків сутностей, які наявні в базі даних. Адміністратори мають змогу переглядати детальну інформацію про кожен запис, включаючи дані всіх залежних сутностей (рисунок 3.13).

Рисунок 3.13 – Діаграма перегляду даних сутностей

Оновлення (Update) – надає можливість редагування даних існуючих сутностей. Завдяки цьому, адміністратор може оновлювати інформацію записів у відповідності до організаційних змін (рисунок 3.14).

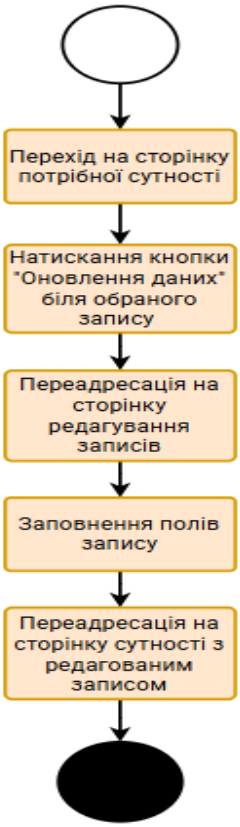


Рисунок 3.14 – Діаграма редагування записів сутностей

Видалення (Delete) – дозволяє видаляти записи з системи, що стає необхідним у випадках ліквідації факультетів, кафедр, дисциплін, або можливого їх об’єднання з іншими структурними одиницями (рисунок 3.15).

Рисунок 3.15 – Діаграма видалення записів сутностей

Модулі CRUD тісно інтегровані один з одним, та з іншими компонентами системи, забезпечуючи єдину взаємодію між різними рівнями ієрархії. Така інтеграція спрощує адміністрування навчального процесу та забезпечує актуальність інформації в системі.

Модуль побудови структурно-логічної схеми освітніх компонентів являє собою найважливішу частину інформаційної системи, яка спрямована на планування та оптимізацію освітнього процесу. Цей модуль надає користувачам, засоби для ефективного проектування індивідуалізованих освітніх траєкторій та програм з урахуванням заданих компетентностей. В основу планувальної логіки модуля закладено використання генетичних алгоритмів для пошуку оптимальних рішень в

просторі можливостей, забезпечуючи ефективну адаптацію навчальних планів до індивідуальних потреб студентів.

Модуль автоматизує процес формування освітніх планів, дозволяючи студентам вибирати компетентності, які вони прагнуть розвинути. На основі цього вибору, система пропонує оптимальний набір дисциплін, що відповідають запитам користувача. Використовуючи інтелектуальний аналіз даних дисциплін, факультетів та спеціальностей, модуль визначає і стратегічно розподіляє курси згідно з пререквізитами та навчальними цілями, максимізуючи ефект від навчання та збалансовуючи навчальне навантаження (рисунок 3.16). Основним викликом для алгоритму є знайти баланс між наданням студентам гнучкості у формуванні освітнього шляху та забезпеченням дотримання академічних вимог та стандартів.

Рисунок 3.16 – Діаграма діяльності модуля генерації структурно-логічної схеми освітніх компонентів

Кожен із цих модулів відіграє знакову роль у загальній структурі та функціональності системи, сприяючи її ефективному використанню задля задоволення освітніх потреб користувачів та оптимізації процесу навчання.

3.3 Розробка програмної реалізації

Розвиток сучасної освітньої інформаційної системи вимагає ретельного підходу до розробки спеціалізованих програмних модулів, кожен з яких відповідає за визначену частину функціоналу та взаємодію з користувачем. Враховуючи попередньо описані характеристики модулів CRUD, модуля авторизації та модуля побудови структурно-логічної схеми, процес розробки повинен відповідати високим вимогам до якості, безпеки та масштабованості.

Модуль авторизації є фундаментальною частиною будь-якої інформаційної системи, забезпечуючи перевірку ідентичності користувачів і надаючи доступ до ресурсів системи. В контексті розробки на мові програмування C# з використанням патерну MVC (Model-View-Controller), модуль авторизації акцентує увагу на безпеці, зручності у використанні та інтеграції з загальною структурою веб-додатку (рисунок 3.17). Основні компоненти модуля представлені наступним чином:

1. Створення моделей для управління даними користувача та сесій. Моделі включають класи, що зберігають необхідну інформацію про користувачів, їхні ролі в системі та статус поточної сесії.
2. Розробка користувацького інтерфейсу за допомогою HTML5 та CSS, що містить текстові поля для введення логіну та паролю, а також кнопку для надсилання даних.
3. Логіка обробки даних введення та виконання процедури авторизації. Контролер приймає дані від форми авторизації, виконує валідацію введених даних і, у разі збігу з записами в базі даних, ініціює сесію користувача в системі, що надає доступ до відповідних ресурсів та функцій системи. При невдалій аутентифікації користувачу відображається повідомлення про помилку через

веб-інтерфейс, пропонуючи повторити спробу введення або відновлення доступу.

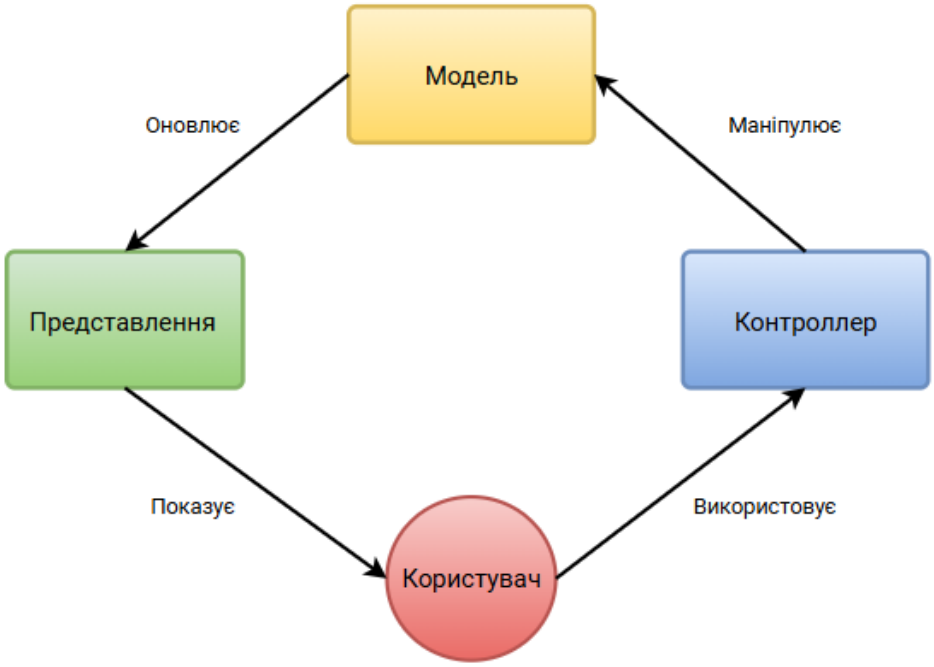


Рисунок 3.17 – Схема патерну MVS

Модулі CRUD дозволяють керувати даними за допомогою відповідного інтерфейсу. В контексті нашої інформаційної системи на базі платформи Microsoft .NET із використанням патерну MVC, ці модулі забезпечують адміністративні можливості управління основними освітніми сутностями, такими як факультети, кафедри, викладачі, дисципліни та компетентності. Кожна сутність представлена у табличному форматі з наступними характеристиками:

1. Назва сутності. Зазначення конкретної категорії, наприклад, «Факультети» або «Дисципліни».
2. Дані записів. Відповідні поля, що містять інформацію про кожен запис у сутності – такі як назва факультету, назва дисципліни, прізвище викладача тощо.
3. Поле дії. Додаткова колонка у таблиці, що дозволяє виконувати специфічні дії над кожним окремим записом, включаючи редагування, видалення, або перегляд деталей.

Функціонал модулів передбачає такі можливості:

1. Імплементація кнопки «Додати новий», що відкриває форму для введення даних нового запису у відповідну сутність.
2. Табличне представлення всіх наявних записів з можливістю фільтрації і сортування за ключовими полями для зручного пошуку і перегляду інформації.
3. Кнопка «Редагувати» біля кожного запису, яка дозволяє змінювати існуючі дані через окрему сторінку редагування.
4. Опція «Видалити» для кожного запису, з додатковим запитом на підтвердження видалення, для запобігання випадкової втрати даних.

Для реалізації модуля CRUD операцій використовувався Razor для динамічного генерування HTML-таблиць в представленнях, включаючи поля для дій. Розробка контролерів, які обробляють запити на CRUD операції, взаємодіючих з моделями і базою даних через Entity Framework. Також, застосовувались атрибути безпеки для контролю доступу до CRUD операцій, гарантуючи, що лише авторизовані користувачі мають відповідні права на виконання дій.

Модуль побудови структурно-логічної схеми освітніх компонентів слугує центральним елементом інформаційної системи, спрямований на оптимізацію освітнього процесу через планування індивідуалізованих навчальних траєкторій. Цей модуль інтегрується з іншими компонентами системи, забезпечуючи можливість адміністраторам та студентам генерувати, аналізувати та кастомізувати навчальні плани за обраними дисциплінами та компетентностями. Наступні кроки описують детально технічну реалізацію цього процесу:

1. Шляхом інтерактивного опитування, студенти подають перелік бажаних до розвитку компетенцій, на основі яких система визначає оптимальний навчальний шлях.
2. Формується початковий набір навчальних планів з обов’язковими дисциплінами та їхніми пререквізитами, враховуючи ліміт кредитів ECTS.
3. Розподіляється навчальне навантаження для забезпечення збалансованості за семестрами, після чого розраховується ранг кожної програми на основі її відповідності заданим компетентностям і балансу.

4. Застосовуються операції селекції, схрещування та мутації для генерації нового пулу навчальних програм, де додатково здійснюється балансування навантаження та перевірка на дотримання пререквізитів.
5. На основі ранжування, формується нова популяція, що складається з найбільш адаптованих навчальних планів, готових до подальшого аналізу та оптимізації.
6. Процес повторюється до досягнення заданого порогу стабільності або до точки виродження популяції, гарантуючи формування оптимальної та комплексної навчальної програми.

Висновки до третього розділу

В третьому розділі виконана програмна реалізація методу побудови структурно-логічної схеми освітніх компонентів. Представлена концепція та детальний опис розробки ключових модулів застосунку для побудови структурно-логічної схеми освітніх компонентів. Створений веб-застосунок, здатний ефективно адаптуватися до динамічних освітніх потреб та адміністративних вимог вищих навчальних закладів, використовуючи сучасні технології та методики програмування.

Основні програмні модулі системи включають:

1. Модуль авторизації, що забезпечує безпечний доступ до системи для адміністраторів та студентів, гарантуючи валідацію користувацьких прав та ідентифікацію користувачів.
2. Модулі CRUD операцій для керування даними факультетів, кафедр, викладачів, дисциплін і компетентностей, забезпечуючи основу для гнучкого маніпулювання освітнім контентом.
3. Модуль побудови структурно-логічної схеми освітніх компонентів, що дозволяє студентам формувати індивідуальні навчальні траєкторії, оптимізуючи вибір дисциплін і сприяючи розвитку бажаних компетентностей.

Кожен з модулів був спроектований та розроблений, виходячи з принципів патерну MVC на мові програмування C#.

РОЗДІЛ 4 Експериментальне дослідження методу побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом

4.1 Порядок використання програмної реалізації

Початок використання застосунку розпочинається зі сторінки аутентифікації, де адміністратор вводить логін і пароль для входу до системи (рисунок 4.1). У випадку успішної авторизації, адміністратор отримує доступ до інструментів управління системою.

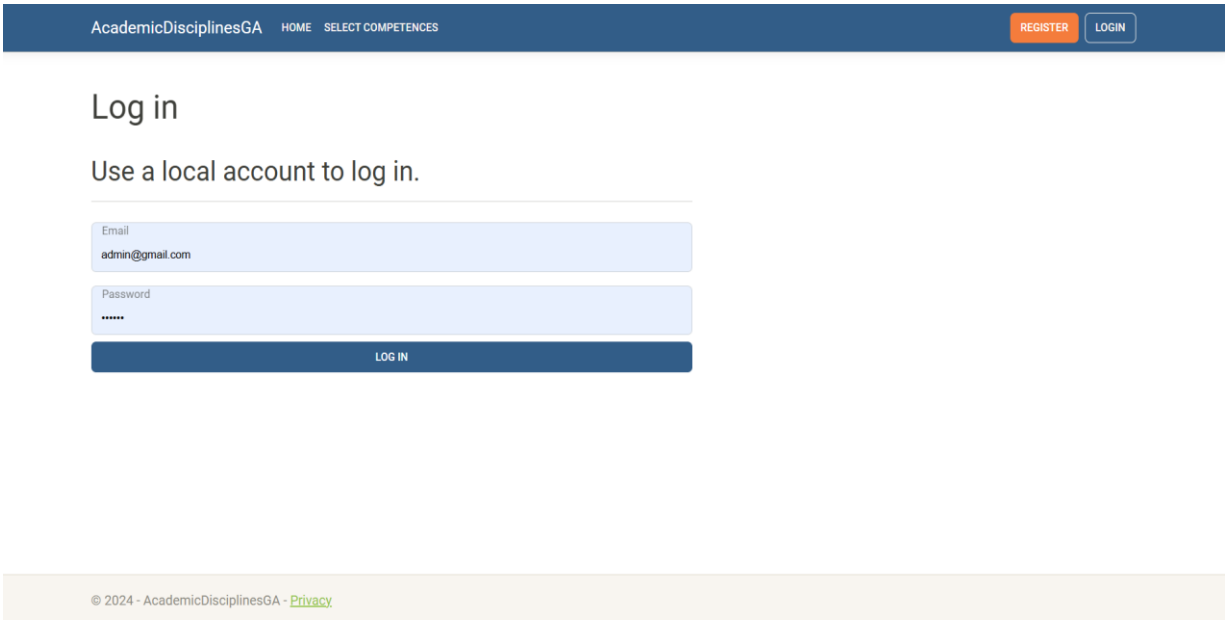


Рисунок 4.1 – Сторінка авторизації

Першими у списку на створення ідуть факультети, для їх адміністрування потрібно перейти на відповідну сторінку до таблиці факультетів, де можна переглядати, видаляти чи редагувати існуючі записи (рисунок 4.2). При натисканні на кнопку «Додати новий» відкривається окрема сторінка, де адміністратор може внести дані для створення нового факультету (рисунок 4.3).

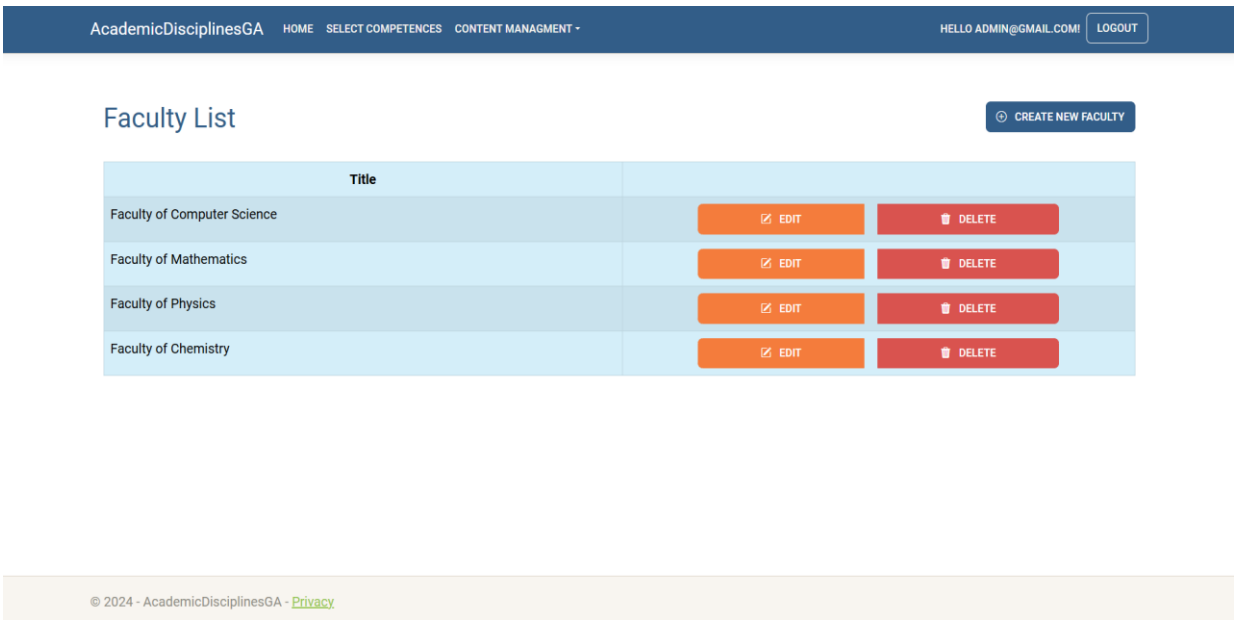


Рисунок 4.2 – Адміністративна сторінка керування факультетами

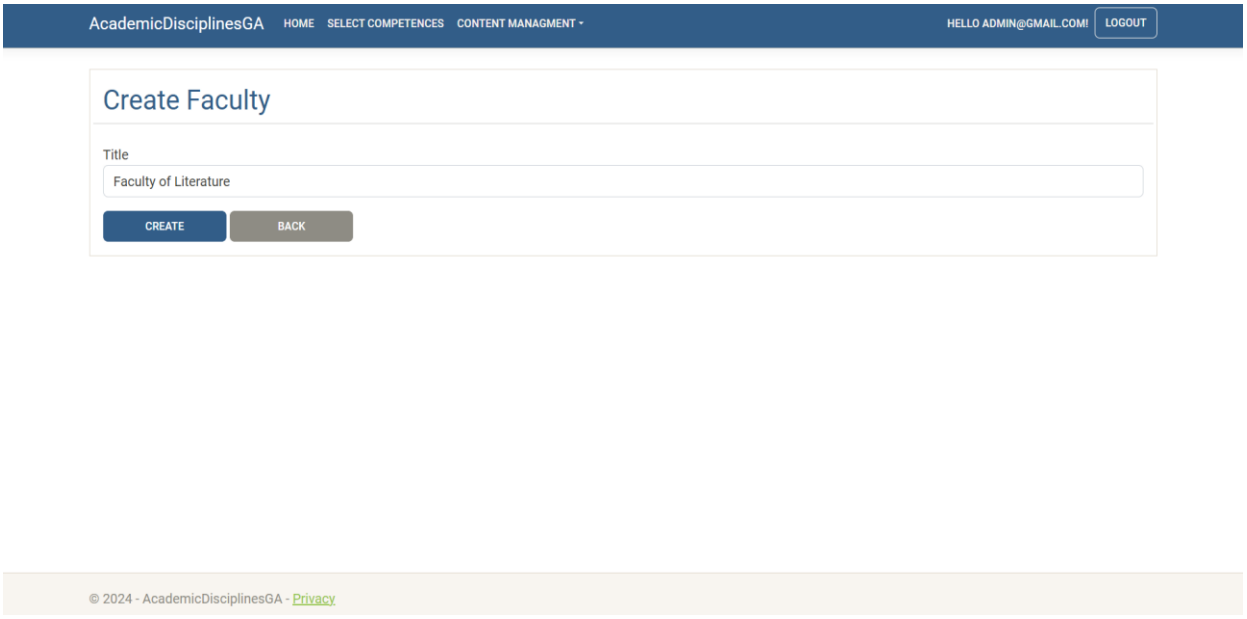


Рисунок 4.3 – Сторінка створення нових записів факультетів

Після створення даних факультетів, наступний крок полягає у адмініструванні кафедр, використовуючи таблицю для перегляду всіх наявних записів та можливості їх редагувати чи видаляти (рисунок 4.4). Також є і окрема форма для створення нових записів (рисунок 4.5).

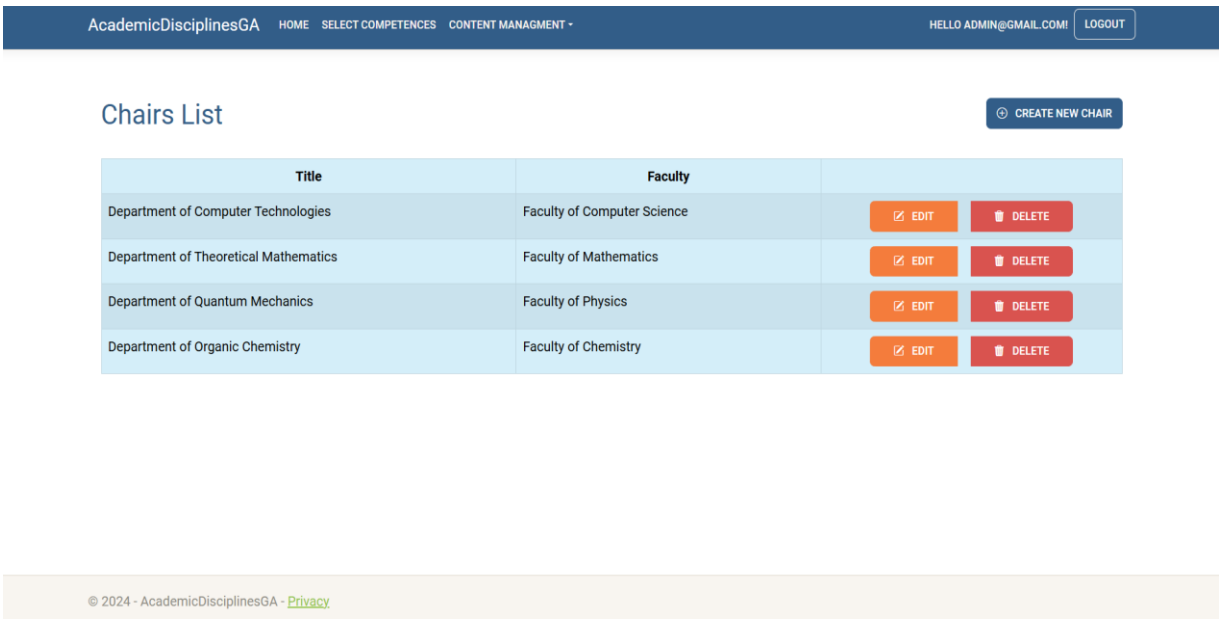


Рисунок 4.4 – Адміністративна сторінка керування кафедрами

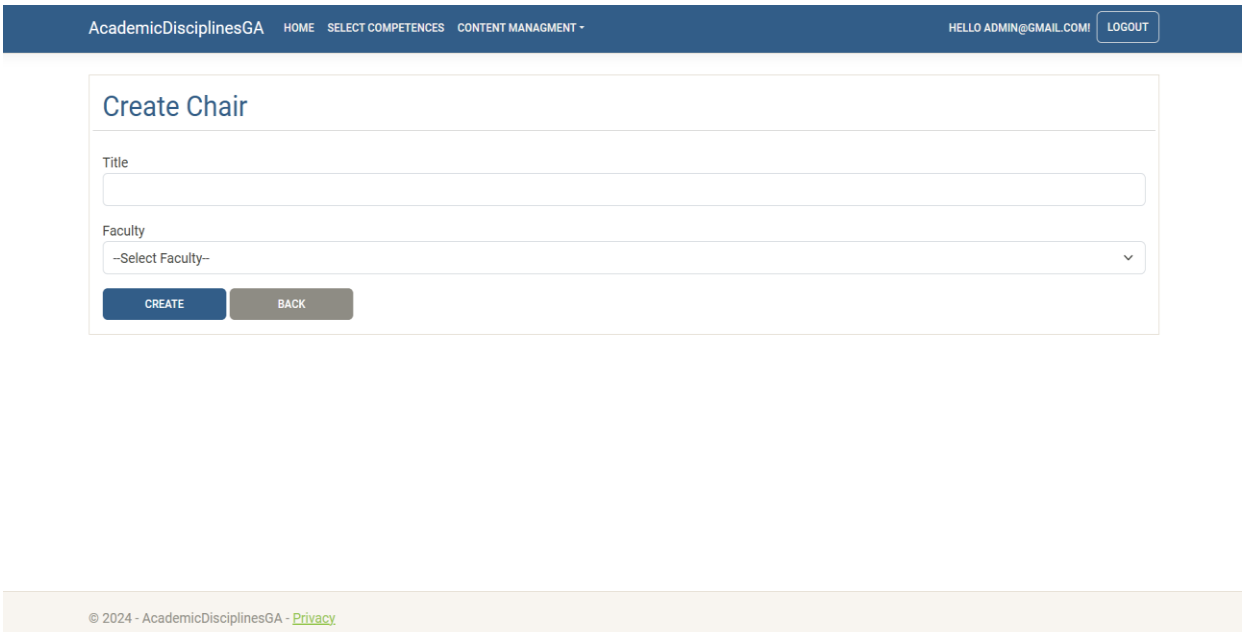


Рисунок 4.5 – Сторінка створення нових записів кафедр

Далі, адміністратор переходить до управління викладачами. На відповідній сторінці реалізований схожий функціонал для роботи з викладачами (рисунок 4.6-4.7).

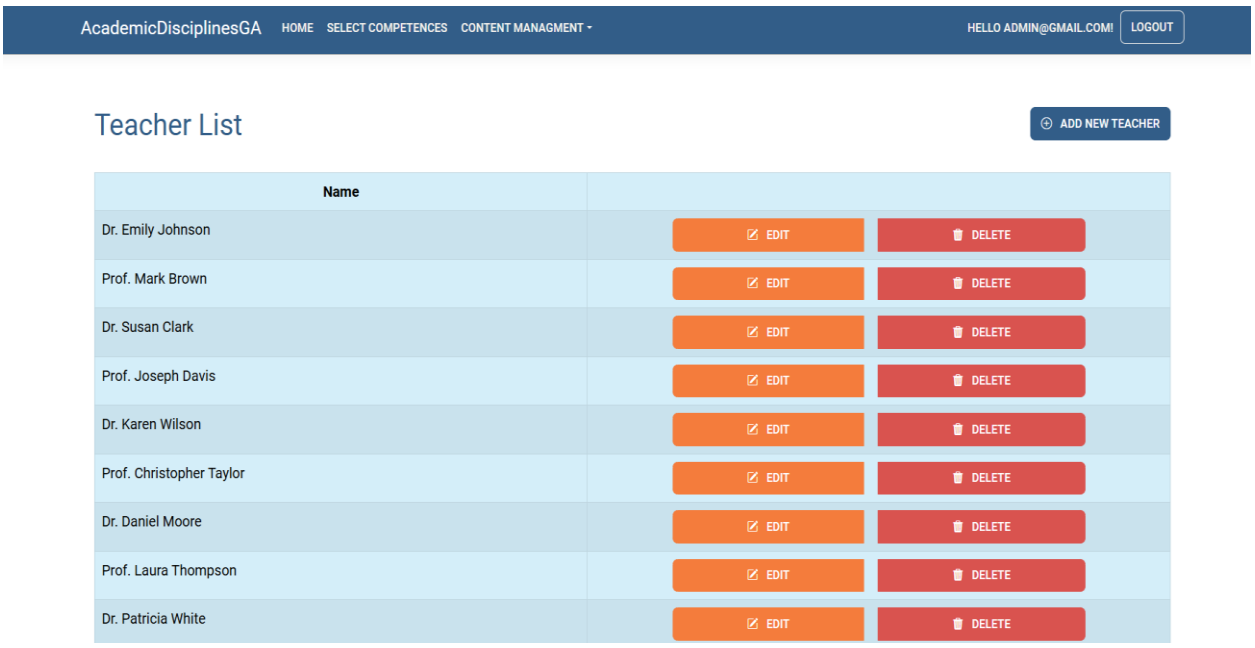


Рисунок 4.6 – Адміністративна сторінка керування викладачами

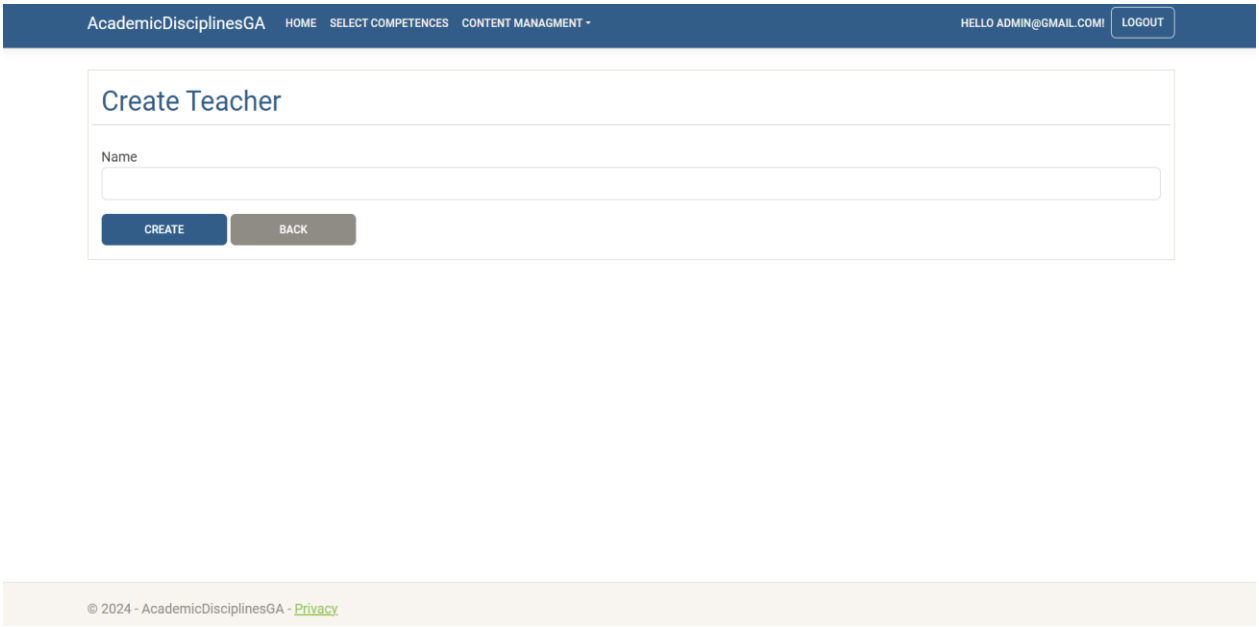


Рисунок 4.7 – Сторінка створення нових записів викладачів

Наступним кроком для адміністратора є робота з компетентностями. Використовуючи відповідну таблицю для огляду переліку компетентностей, з можливістю редагування, видалення та створення (рисунок 4.8-4.9).

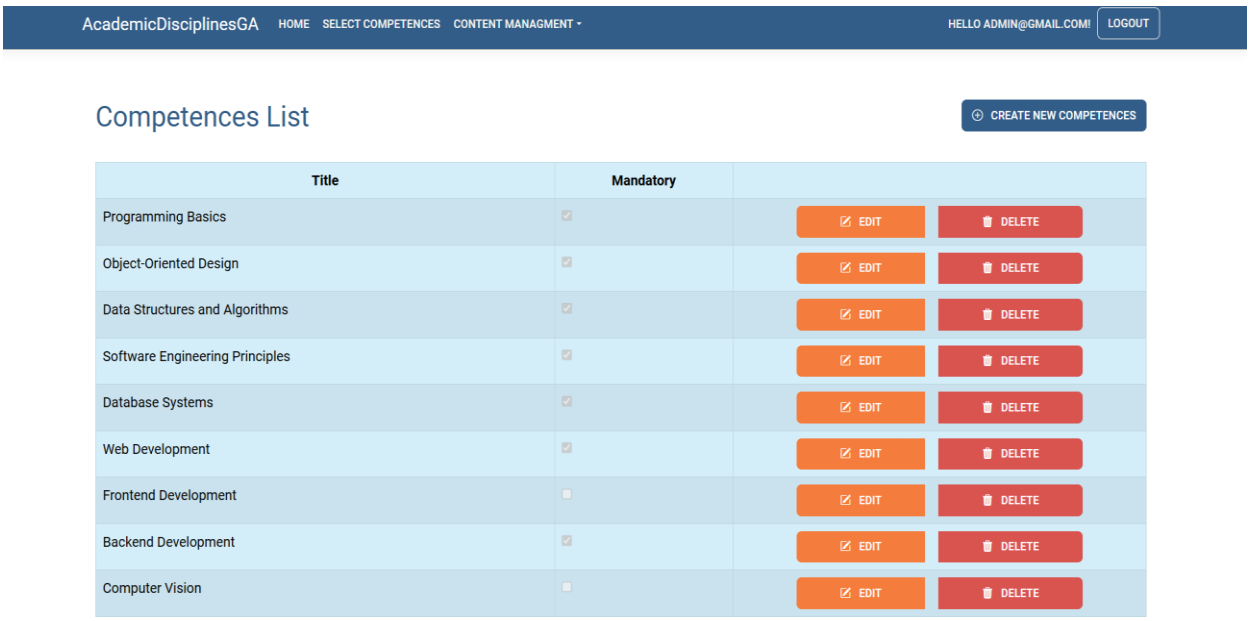


Рисунок 4.8 – Адміністративна сторінка керування компетентностями

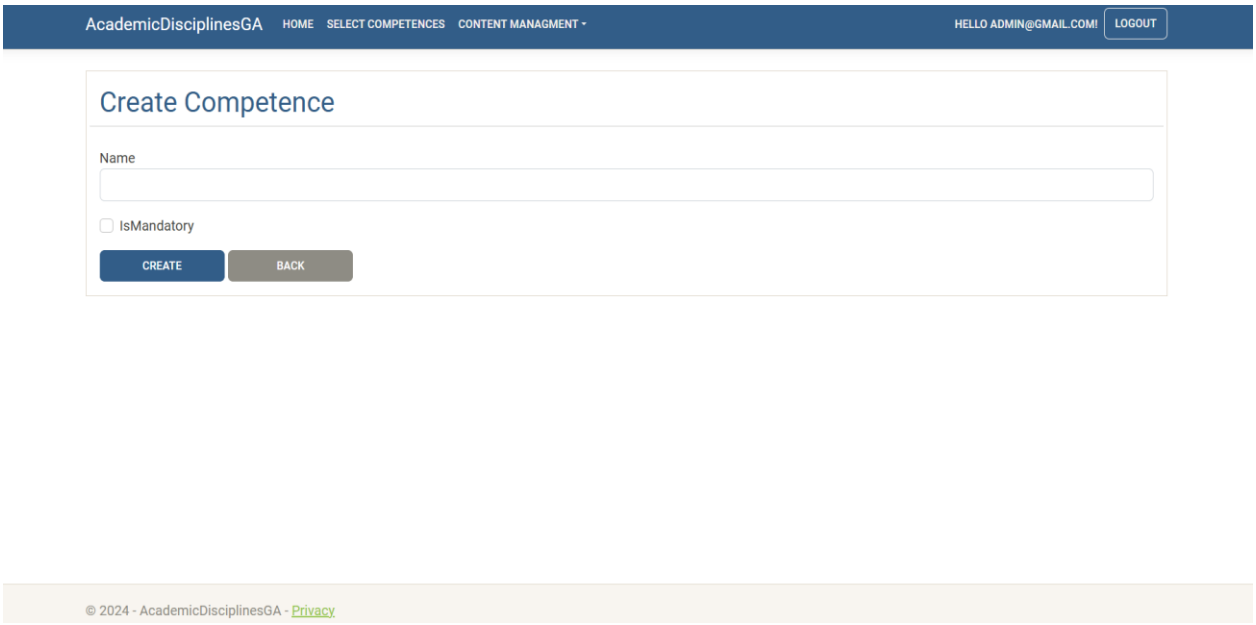


Рисунок 4.9 – Сторінка створення нових записів компетентностей

Наприкінці, адміністратору необхідно налаштувати навчальні дисципліни. Використовуючи відповідну таблицю, адміністратор може переглядати повний список курсів з детальною інформацією, вносити зміни до існуючих записів або ж видаляти їх. Функція «Створити запис» дозволяє адміністратору створювати нові курси, вказуючи їх назву, обсяг кредитів ECTS, відповідні компетентності, що розвиваються в рамках курсу, та призначити викладача (рисунок 4.10-4.11). Такий

інструментарій допомагає у формуванні збалансованого та гнучкого навчального плану, адаптованого до потреб студентів.

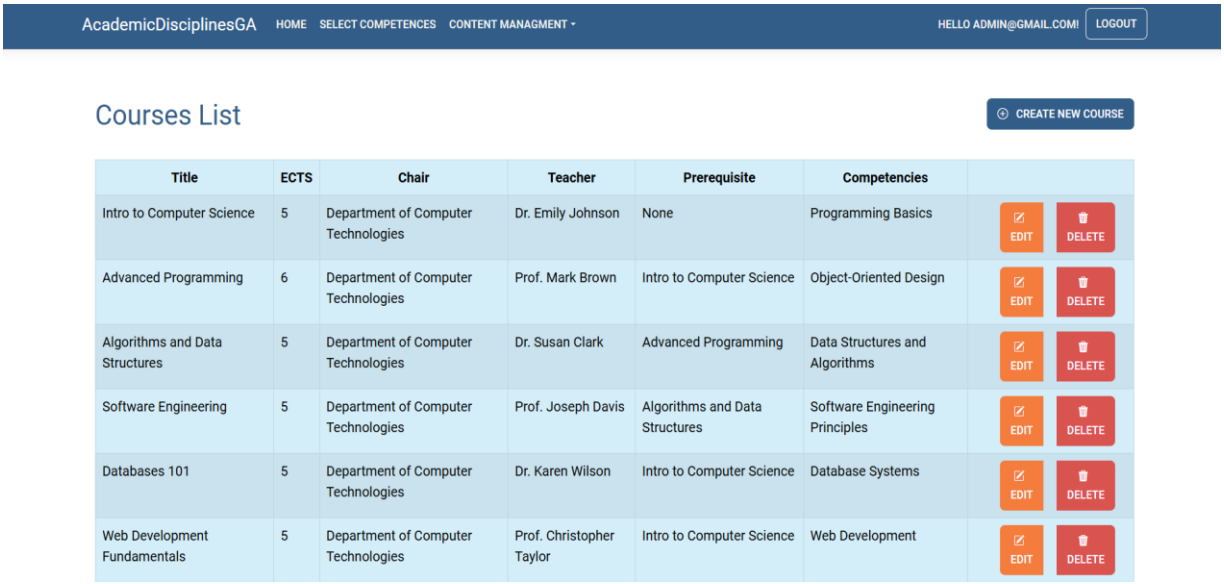


Рисунок 4.10 – Адміністративна сторінка керування компетентностями

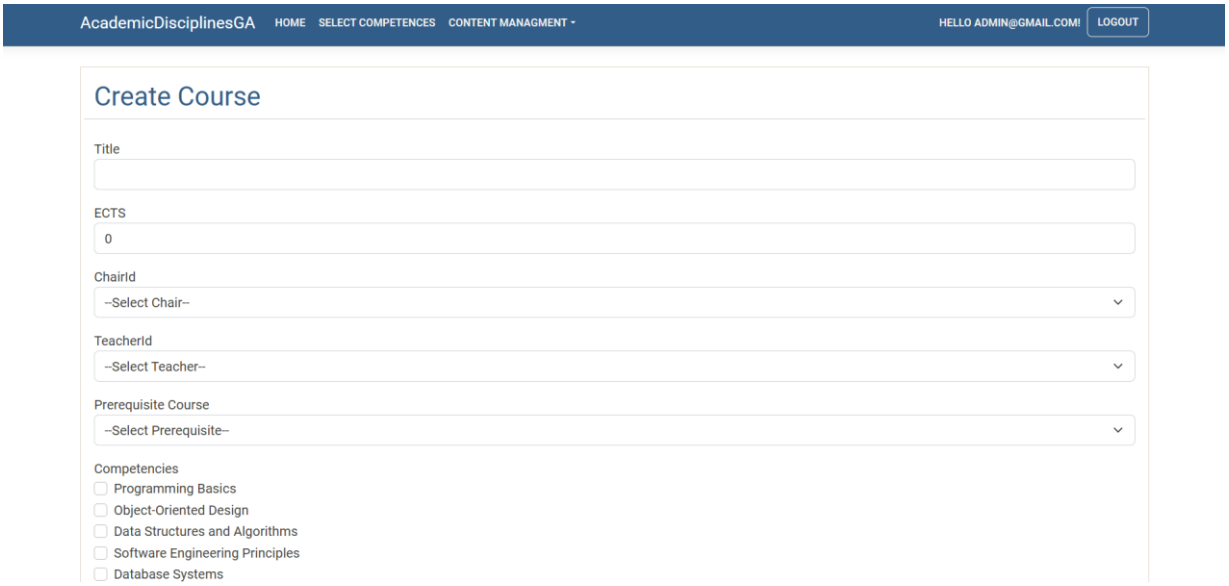


Рисунок 4.11 – Сторінка створення нових записів дисциплін

Після налаштування, адміністратором, середовища застосунку студенти можуть перейти до складання власної ІОТ. З головної сторінки вони можуть перейти на сторінку з вибірковими компетентностями та обрати бажані (рисунок 4.12). На їх основі буде згенеровано оптимальне ІОТ для користувача (рисунок 4.13).

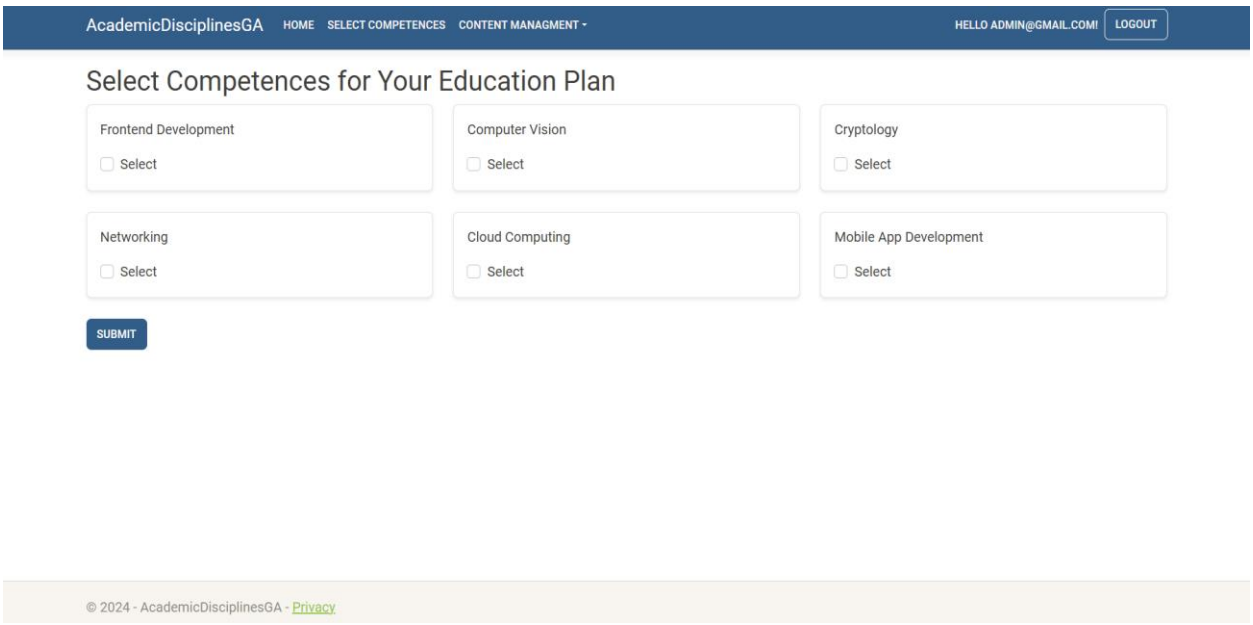


Рисунок 4.12 – Сторінка з вибором бажаних компетентностей

Title	ECTS	Chair	Teacher	Prerequisite	Competencies	Year	Semester
Intro to Computer Science	5	Department of Computer Technologies	Dr. Emily Johnson	None	Programming Basics	1	1
Frontend Web Development	5	Department of Computer Technologies	Dr. Daniel Moore	None	Frontend Development	1	1
Databases 101	5	Department of Computer Technologies	Dr. Karen Wilson	Intro to Computer Science	Database Systems	1	2
Advanced Programming	6	Department of Computer Technologies	Prof. Mark Brown	Intro to Computer Science	Object-Oriented Design	1	2
Web Development Fundamentals	5	Department of Computer Technologies	Prof. Christopher Taylor	Intro to Computer Science	Web Development	2	1
Computer Vision Basics	5	Department of Computer Technologies	Dr. Patricia White	Intro to Computer Science	Computer Vision	2	1
Introduction to Cryptology	5	Department of Computer Technologies	Dr. Emily Johnson	Databases 101	Cryptology	2	2
Algorithms and Data Structures	5	Department of Computer Technologies	Dr. Susan Clark	Advanced Programming	Data Structures and Algorithms	2	2
Cloud Computing Basics	5	Department of Computer Technologies	Dr. Karen Wilson	Algorithms and Data Structures	Cloud Computing	3	1
Introduction to Ethical Hacking	5	Department of Computer Technologies	Prof. Joseph Davis	Web Development Fundamentals	Ethical Hacking	3	1

Рисунок 4.13 – Сторінка згенерованого індивідуального навчального плану

Після огляду функціоналу інформаційної системи наступним кроком буде її тестування з використанням освітніх даних. Це передбачає формування списків

курсів із відповідними кредитами ECTS та набором компетентностей, типових для вищих навчальних закладів.

4.2 Дослідження методу побудови структурно-логічної схеми освітніх компонентів

У рамках дослідження методу побудови структурно-логічної схеми освітніх компонентів було вирішено здійснити експеримент, що базується на освітніх компонентах «Комп'ютерних технологій». В рамках експерименту було задіяно перелік дисциплін, для детального випробування методу, та, виходячи з академічних критеріїв, були встановлені умови щодо сумарного обсягу кредитів ECTS. Для кожної дисципліни були визначені асоційовані компетентності, що дозволило відобразити зв'язки між навчальними предметами та цільовими освітніми результатами. Переглянути тестові дані можна у таблицях 4.1-4.3.

Таблиця 4.1 – Перелік обов'язкових дисциплін

Назва дисципліни	**Кредити**	**Пререквізити**	**Компетентності**
Intro to Computer Science	5	Немає	Programming Basics
Advanced Programming	6	Intro to Computer Science	Object-Oriented Design
Algorithms and Data Structures	5	Advanced Programming	Data Structures and Algorithms
Software Engineering	5	Algorithms and Data Structures	Software Engineering Principles
Databases	5	Intro to Computer Science	Database Systems
Web Development Fundamentals	5	Intro to Computer Science	Web Development
System Analysis	5	Немає	Systems Analysis

Таблиця 4.2 – Перелік вибіркових дисциплін

Назва дисципліни	**Кредити**	**Пререквізити**	**Компетентності**
Frontend Web Development	5	Немає	Frontend Development
Backend Web Development	6	Frontend Web Development	Backend Development
Computer Vision	5	Intro to Computer Science	Computer Vision
Biostatistics	5	Intro to Computer Science	Biostatistics
Introduction to Cryptology	5	Databases	Cryptology
Quantum Computing	6	Algorithms and Data Structures	Quantum Computing
Networking Fundamentals	4	Intro to Computer Science	Networking
Introduction to Ethical Hacking	5	Web Development	Ethical Hacking
Cloud Computing	5	Algorithms and Data Structures	Cloud Computing
Game Development	6	Introduction to AI	Game Development
Introduction to AI	5	Backend Web Development	Artificial Intelligence
Mobile App Development	5	Frontend Web Development	Mobile App Development
UI/UX Design	4	Web Development Fundamentals	UI/UX Principles
Machine Learning	6	Introduction to AI	Machine Learning

Кінець таблиці 4.2 – Перелік вибіркових дисциплін

Cybersecurity	5	Networking Fundamentals	Cybersecurity
Robotics Introduction	5	Algorithms and Data Structures	Robotics
Internet of Things	6	Немає	Robotics, IoT Development
Data Visualization Techniques	5	UI/UX Design	Data Visualization
Blockchain	6	Introduction to Cryptology	Blockchain Technology
Augmented and Virtual Reality	6	Frontend Web Development	AR/VR Development

Таблиця 4.3 – Перелік компетентностей

Назва	**Обов'язкова**
Programming Basics	Так
Object-Oriented Design	Так
Data Structures and Algorithms	Так
Software Engineering Principles	Так
Database Systems	Так
Web Development	Так
Frontend Development	Ні
Backend Development	Ні
Computer Vision	Ні
Biostatistics	Ні

Кінець таблиці 4.3 – Перелік компетентностей

Cryptology	Hi
Quantum Computing	Hi
Networking	Hi
Ethical Hacking	Hi
Cloud Computing	Hi
Game Development	Hi
Artificial Intelligence	Hi
Mobile App Development	Hi
Systems Analysis	Так
UI/UX Principles	Hi
Machine Learning	Hi
Cybersecurity	Hi
Robotics	Hi
IoT Development	Hi
Data Visualization	Hi
Blockchain Technology	Hi
AR/VR Development	Hi

Надалі ці тестові дані будуть використовуватись для перевірки методу підвищення якості формування структурно-логічної схеми освітніх компонентів за генетичним алгоритмом.

4.3 Демонстрація та аналітичний огляд результатів дослідження методу

Дослідження методу побудови індивідуальної освітньої траєкторії була здійснена з використанням створеного у розділі 3 застосунку. У нашому випадку, розглядається студент, що бажає працювати у галузі кібербезпеки. Інтереси та освітні потреби студента були такими, як зображено на рисунку 4.14.

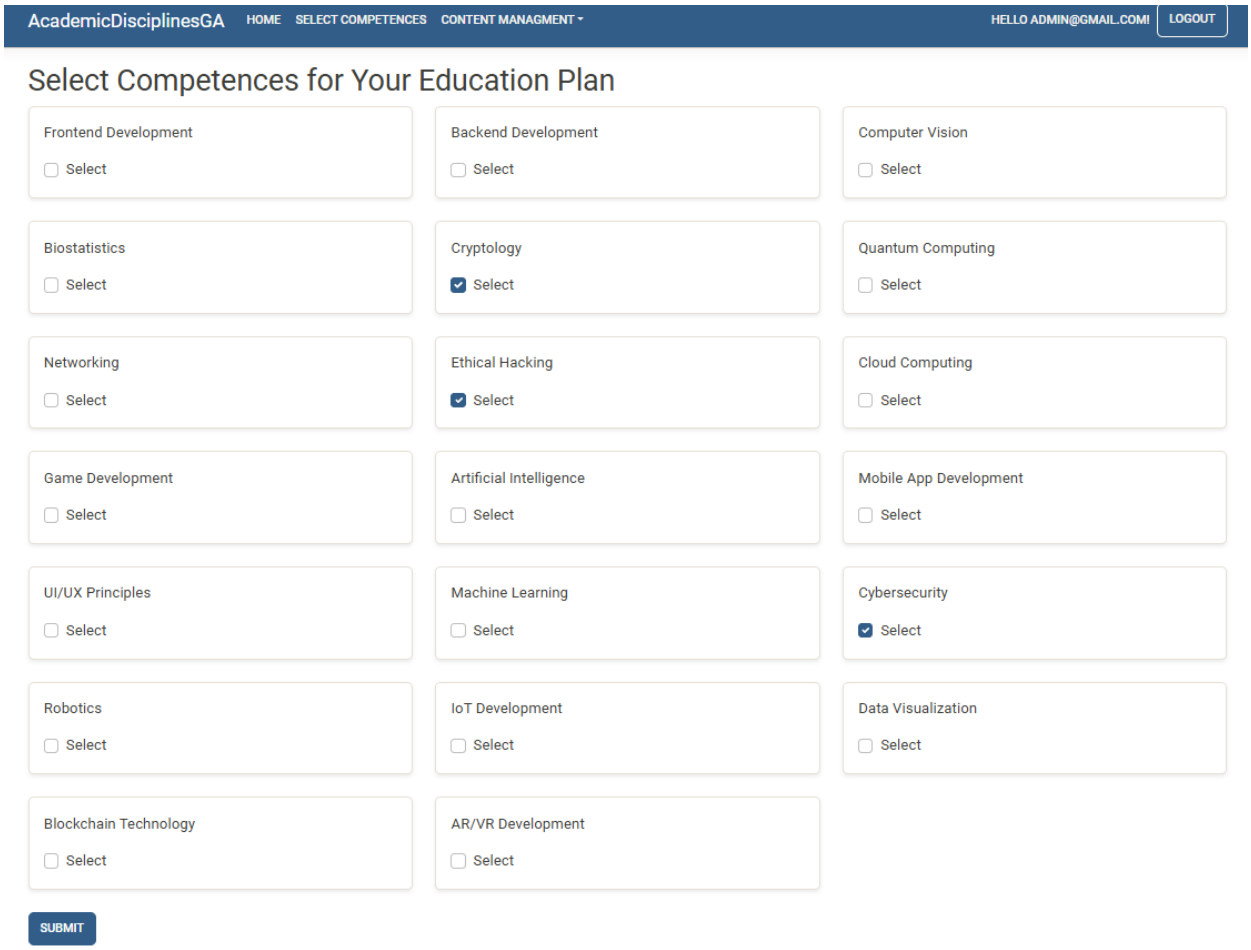


Рисунок 4.14 – Обрані користувачем компетентності

Після генерації IOT було проведено детальне порівняння цього списку із обраними компетентностями. Було підтверджено, що всі обов’язкові дисципліни присутні в плані, а вибіркові дисципліни були адаптовані згідно з індивідуальними цілями студента, зокрема: «Cybersecurity», «Introduction to Ethical Hacking», «Introduction to Cryptology».

Особливу увагу було приділено встановленню правильної послідовності дисциплін, зокрема дотриманню необхідних пререквізитів. Метод забезпечив логічну послідовність навчання, де більш складні курси слідують після того, як були засвоєні базові дисципліни (рисунок 4.15). Для прикладу розглянемо пререквізити дисципліни «Introduction to Cryptology», що вивчається на 1-му семестрі 4-го року навчання, основним пререквізитом для неї є предмет «Databases», що у свою чергу вивчається на 2-му навчальному році та 2-му семестрі і має пререквізитом дисципліну «Intro to Computer Science», яка є обов'язковою та вивчається однією з перших.

Індивідуальний навчальний план

Title	ECTS	Chair	Teacher	Prerequisite	Competencies	Year	Semester
Frontend Web Development	5	Department of Computer Technologies	Dr. Daniel Moore	None	Frontend Development	1	1
Intro to Computer Science	5	Department of Computer Technologies	Dr. Emily Johnson	None	Programming Basics	1	1
Augmented and Virtual Reality	6	Department of Computer Technologies	Prof. Christopher Taylor	Frontend Web Development	AR/VR Development	1	2
System Analysis	5	Department of Computer Technologies	Prof. Christopher Taylor	None	Systems Analysis	1	2
Advanced Programming	6	Department of Computer Technologies	Prof. Mark Brown	Intro to Computer Science	Object-Oriented Design	2	1
Backend Web Development	6	Department of Computer Technologies	Prof. Laura Thompson	Frontend Web Development	Backend Development	2	1
Web Development Fundamentals	5	Department of Computer Technologies	Prof. Christopher Taylor	Intro to Computer Science	Web Development	2	2
Networking Fundamentals	4	Department of Computer Technologies	Dr. Susan Clark	Intro to Computer Science	Networking	2	2
Databases	5	Department of Computer Technologies	Dr. Karen Wilson	Intro to Computer Science	Database Systems	2	2
Algorithms and Data Structures	5	Department of Computer Technologies	Dr. Susan Clark	Advanced Programming	Data Structures and Algorithms	3	1
Introduction to AI	5	Department of Computer Technologies	Dr. Daniel Moore	Backend Web Development	Artificial Intelligence	3	1
Introduction to Ethical Hacking	5	Department of Computer Technologies	Prof. Joseph Davis	Web Development Fundamentals	Ethical Hacking	3	2
Cybersecurity	5	Department of Computer Technologies	Prof. Mark Brown	Networking Fundamentals	Cybersecurity	3	2
Cloud Computing Basics	5	Department of Computer Technologies	Dr. Karen Wilson	Algorithms and Data Structures	Cloud Computing	4	1
Introduction to Cryptology	5	Department of Computer Technologies	Dr. Emily Johnson	Databases	Cryptology	4	1

Рисунок 4.15 – Згенерований індивідуальний навчальний план студента

Огляд результатів розробленого методу підтвердила його здатність ефективно формувати структурно-логічні схеми освітніх компонентів, які чітко відповідають

індивідуальним потребам студентів. Ключовим аспектом цього підтвердження стало те, що реалізований метод відповідає всім визначеним у розділі 2 критеріям, забезпечуючи адаптацію навчального процесу до специфічних освітніх вимог та інтересів здобувачів освіти.

Висновки до четвертого розділу

У рамках четвертого розділу було проведено дослідження методу побудови структурно-логічної схеми освітніх компонентів, та досліджено порядок її використання користувачами.

Дослідження показало, що запропонований метод ефективно сприяє створенню індивідуальних освітніх траєкторій та забезпечує глибокий аналіз вибраних компетентностей, пропонуючи найбільш релевантні дисципліни, а також враховує пререквізити та баланс навчального навантаження, підтверджуючи адаптивність та ефективність підходу.

Висновки

У рамках виконання кваліфікаційної роботи магістра розроблено метод для підвищення якості формування структурно-логічної схеми освітніх компонентів за генетичним алгоритмом, що забезпечує відповідність освітнього процесу до потреб і побажань здобувачів вищої освіти з урахуванням академічних правил та стандартів.

Проведено детальний аналіз методів побудови структурно-логічної схеми освітніх компонентів. Аналіз дозволив ідентифікувати прогалини в існуючих методах та визначити потенціал для їх покращення з використанням генетичних алгоритмів.

Спроектовано метод побудови структурно-логічної схеми освітніх компонентів. Алгоритмічна структура методу забезпечує ефективне генерування індивідуалізованих освітніх траєкторій, адаптованих до кожного конкретного користувача.

Визначено набір критеріїв для оцінки підвищення якості формування побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом.

Програмно реалізовано метод побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом, та проведено експериментальне дослідження за тестовими наборами даних, що підтвердило високу ефективність методу та його здатність забезпечувати значні покращення в освітньому процесі.

У підсумку, розроблений метод разом з його програмною реалізацією відповідає всім передбаченим критеріям, зокрема, оптимізацію обсягу кредитів ECTS, впорядкованість дисциплін у відповідності до необхідних пререквізитів, а також врахування індивідуалізованих навчальних побажань студента.

Спроектований метод ефективно сприяє створенню індивідуальних освітніх траєкторій та забезпечує глибокий аналіз вибраних компетентностей, пропонуючи найбільш релевантні дисципліни, а також враховує пререквізити та баланс навчального навантаження, підтверджуючи адаптивність та ефективність підходу.

Перелік посилань

1. OECD Handbook for Internationally Comparative Education Statistics. С. 57-76. URL: https://www.oecd-ilibrary.org/docserver/9789264279889-8-en.pdf?expires=1731705436&id=id&accname=guest&checksum=2715380F3579AB3C816915FC48015BA1

2. Про повну загальну та середню освіту. URL: https://osvita.ua/legislation/law/2232/

3. Рівні вищої освіти та наукові ступені. URL: https://mon.gov.ua/osvita-2/vishcha-osvita-ta-osvita-doroslikh/rivni-vishchoi-osviti-ta-naukovi-stupeni

4. Професійна освіта. URL: https://mon.gov.ua/tag/profesiyna-osvita?&type=all&tag=profesiyna-osvita

5. Програми підвищення кваліфікації та професійного розвитку. URL: https://wiki.legalaid.gov.ua/index.php/Підвищення_кваліфікації_педагогічних_працівників

6. Про акредитацію освітніх програм за якими здійснюється підготовка здобувачів вищої освіти. URL: https://zakon.rada.gov.ua/laws/show/z1013-24#Text

7. Вибіркові дисципліни. URL: https://khmnu.edu.ua/wp-content/uploads/normatyvni-dokumenty/polozhennya/pro-poryadok-realizacziyi-prava-na-vilnyj-vybir-navchalnyh-dysczyplinu-chynne-z-01.09.2020.pdf

8. Top trends in education technology. URL: https://www.educate-me.co/blog/trends-in-educational-technology

9. Teaching Excellence in Higher Education: Theory and Practice. С. 64-79. URL: https://ihed.org.ua/wp-content/uploads/2023/03/Vdoskonal_vykladan_u_VO-IVO-2023-255p.pdf

10. Рекомендації щодо застосування критеріїв оцінювання якості освітньої програми. URL: https://naqa.gov.ua/wp-content/uploads/2020/12/Рекомендації-щодо-застосування-критеріїв-оцінювання-якості-ОП.pdf

11. Why electives are essential to student growth. URL: https://brancheslearning.org/unlocking-potential-why-electives-are-essential-to-student-growth/

12. Impact of Teachers' qualifications and teaching methods. URL: https://www.researchgate.net/publication/372891203_Impact_of_Teachers'_qualifications_and_teaching_methods

13. Factors influencing pupils performance in education. URL: https://www.researchgate.net/publication/334678107_FACTORS_INFLUENCING_PUPILS_PERFORMANCE_IN_EDUCATION

14. Study on factors influencing employment decisions in graduate students. URL: https://www.researchgate.net/publication/370608917_STUDY_ON_FACTORS_INFLUENCING_EMPLOYMENT_DECISIONS_IN_GRADUATE_STUDENTS

15. Feedback and reporting. URL: https://www.education.vic.gov.au/school/teachers/teachingresources/practice/Pages/insight-feedback.aspx

16. The use of innovative technologies in education. URL: https://www.researchgate.net/publication/376302849_The_Use_of_Innovative_Technologies_in_Education_analysis_of_effectiveness_and_implementation_at_different_levels_of_education

17. Role of AI in education. URL: https://www.researchgate.net/publication/369545925_Role_of_AI_in_Education

18. Залучення бізнес-партнерів для підвищення якості підготовки фахівців. URL: https://www.kmu.gov.ua/news/zaluchennya-biznes-partneriv-pidvishchennya-yakosti-pidgotovki-fahivciv-rannya-adaptaciya-pershi-rezultati-vprovadzhennya-dualnoyi-formi-u-vishchij-osviti

19. Evaluation model of education programs. URL: https://novateurpublication.com/index.php/np/catalog/download/4/3/94-1?inline=1

20. Міждисциплінарність. URL: https://en.wikipedia.org/wiki/Interdisciplinarity

21. Інтегративне навчання як перспективний напрямок розвитку сучасної вищої освіти. URL: https://pedpsy.duan.edu.ua/images/PDF/2014/2/32.pdf

22. Модульний підхід до навчання. URL: https://osvita.ua/vnz/reports/pedagog/14235/

23. The role of the competence approach in pedagogical education. URL: https://www.idpublications.org/wp-content/uploads/2019/10/Full-Paper-THE-ROLE-OF-THE-COMPETENCE-APPROACH-IN-PEDAGOGICAL-EDUCATION.pdf

24. The project approach to teaching and learning. URL: https://www.communityplaythings.com/resources/articles/the-project-approach-to-teaching-and-learning

25. Інтеграція інноваційних технологій у викладанні освітніх компонент. URL: http://www.innovpedagogy.od.ua/archives/2024/71/part_1/29.pdf

26. Modern approaches to teaching and assessment of higher education seekers: a conceptual vision. . URL: https://www.researchgate.net/publication/376828561_Modern_Approaches_to_Teaching_and_Assessment_of_Higher_Education_Seekers_A_Conceptual_Vision

27. Education’s role in fostering environmental awareness and advancing sustainable development within a holistic framework. URL: https://www.researchgate.net/publication/382040507_Education’s_role_in_fostering_environmental_awareness_and_advancing_sustainable_development_within_a_holistic_framework/fulltext/66895a5bf3b61c4e2cb73a34/Educations-role-in-fostering-environmental-awareness-and-advancing-sustainable-development-within-a-holistic-framework.pdf

28. Supporting improvements in classroom climate for students and teachers with the four pillars of wellbeing curriculum. URL: https://www.researchgate.net/publication/327563612_Supporting_improvements_in_classroom_climate_for_students_and_teachers_with_the_four_pillars_of_wellbeing_curriculum

29. The Determinants of Lifelong Learning. URL: https://www.researchgate.net/publication/298215074_The_Determinants_of_Lifelong_Learning

30. Coursera. URL: https://www.coursera.org/

31. Khan Academy. URL: https://www.khanacademy.org/

32. Генетичні алгоритми. Ключові поняття і методи реалізації. URL: http://www.znannya.org/?view=ga_general

33. An improved genetic algorithm with initial population strategy and self-adaptive member grouping. URL: https://www.sciencedirect.com/science/article/abs/pii/S004579490700301X

34. Кросинговер. URL: https://uk.wikipedia.org/wiki/Кросинговер_(генетичний_алгоритм)

35. Мутація. URL: https://en.wikipedia.org/wiki/Mutation_(genetic_algorithm)

36. A genetic algorithm based global search strategy for population pharmacokinetic/pharmacodynamic model selection. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC4294074/

37. NP-hardness. URL: https://en.wikipedia.org/wiki/NP-hardness

38. Multi-objective optimization. URL: https://en.wikipedia.org/wiki/Multi-objective_optimization

39. Convergent evolution. URL: https://en.wikipedia.org/wiki/Convergent_evolution

40. Огляд сучасного використання генетичних та еволюційних алгоритмів. стратегії, можливості. URL: http://misapr.khpi.edu.ua/article/view/261793

ДОДАТКИ

Додаток А

Програмні коди

Лістинг CourseController.cs:

```
namespace
AcademicDisciplinesGA.Areas.Admin.Controllers
{
    [Area("Admin")]
    [Authorize(Roles = "Admin")]
    public class CourseController : Controller
    {
        private    readonly    ApplicationDbContext
_context;

        public
CourseController(ApplicationDbContext context)
        {
            _context = context;
        }

        public IActionResult Index()
        {
            IEnumerable<Course>    courseList    =
_context.Courses.Include(x => x.Teacher).Include(y
=> y.Chair).ToList();
            return View(courseList);
        }

        public IActionResult Upsert(int? id)
        {
            CourseVM courseVM = new()
            {
                Course = new(),
                TeacherList                        =
_context.Teachers.Select(x => new SelectListItem {
Text = x.Name, Value = x.Id.ToString() }),
                ChairList                          =
_context.Chairs.Select(x  =>  new  SelectListItem  {
Text = x.Title, Value = x.Id.ToString() })
            };

            if (id == 0 || id == null)
            {
                return View(courseVM);
            }
            else
            {
                courseVM.Course                    =
_context.Courses.FirstOrDefault(x => x.Id == id);
                return View(courseVM);
            }
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Upsert(CourseVM obj)
        {
            if (ModelState.IsValid)
            {
                if (obj.Course.Id == 0)
                {
                    //add

_context.Courses.Add(obj.Course);
                }
                else
                {
                    //update

_context.Courses.Update(obj.Course);
                }
                _context.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(obj);
        }

        [HttpDelete]
        public IActionResult Delete(int id)
        {
            var               course               =
_context.Courses.FirstOrDefault(x => x.Id == id);

            if (course == null)
            {
                return NotFound();
            }

            _context.Courses.Remove(course);
            _context.SaveChanges();
            return RedirectToAction("Index");
        }
    }
}
```

Лістинг FacultyController.cs:

```
namespace
AcademicDisciplinesGA.Areas.Admin.Controllers
{
    [Area("Admin")]
    [Authorize(Roles = "Admin")]
    public class FacultyController : Controller
    {
```

```
        private readonly ApplicationDbContext
_context;

        public
FacultyController(ApplicationDbContext context)
        {
            _context = context;
        }

        public IActionResult Index()
        {
            IEnumerable<Faculty> facultyList =
_context.Faculties.ToList();
            return View(facultyList);
        }

        public IActionResult Upsert(int? id)
        {
            Faculty faculty = new();

            if (id == 0 || id == null)
            {
                return View(faculty);
            }
            else
            {
                faculty =
_context.Faculties.FirstOrDefault(x => x.Id == id);
                return View(faculty);
            }
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Upsert(Faculty obj)
        {
            if (ModelState.IsValid)
            {
                if (obj.Id == 0)
                {
                    _context.Faculties.Add(obj);
                }
                else
                {

_context.Faculties.Update(obj);
                }
                _context.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(obj);
        }

        [HttpDelete]
        public IActionResult Delete(int id)
        {
            var faculty =
_context.Faculties.Find(id);

            if (faculty == null)
            {
                return NotFound();
            }

            _context.Faculties.Remove(faculty);
            _context.SaveChanges();
            return RedirectToAction("Index");
        }
    }
}
```

Лістинг TeacherController.cs:

```
namespace
AcademicDisciplinesGA.Areas.Admin.Controllers
{
    [Area("Admin")]
    [Authorize(Roles = "Admin")]
    public class TeacherController : Controller
    {
        private readonly ApplicationDbContext
_context;

        public
TeacherController(ApplicationDbContext context)
        {
            _context = context;
        }

        public IActionResult Index()
        {
            IEnumerable<Teacher> teacherList =
_context.Teachers.ToList();
            return View(teacherList);
        }

        public IActionResult Upsert(int? id)
        {
            Teacher teacher = new();

            if (id == 0 || id == null)
            {
                return View(teacher);
            }
            else
            {
                teacher =
_context.Teachers.FirstOrDefault(x => x.Id == id);
                return View(teacher);
            }
```

```
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Upsert(Teacher obj)
        {
            if (ModelState.IsValid)
            {
                if (obj.Id == 0)
                {
                    _context.Teachers.Add(obj);
                }
                else
                {
                    _context.Teachers.Update(obj);
                }
                _context.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(obj);
        }

        [HttpDelete]
        public IActionResult Delete(int id)
        {
            var              teacher              =
_context.Teachers.FirstOrDefault(x => x.Id == id);

            if (teacher == null)
            {
                return NotFound();
            }

            _context.Teachers.Remove(teacher);
            _context.SaveChanges();
            return RedirectToAction("Index");
        }
    }
}
```

Лістинг CourseVM.cs:

```
namespace AcademicDisciplinesGA.Areas.Admin.Models
{
    public class CourseVM
    {
        public Course Course { get; set; }

        [ValidateNever]
        public         IEnumerable<SelectListItem>
TeacherList { get; set; }

        [ValidateNever]
        public         IEnumerable<SelectListItem>
ChairList { get; set; }
    }
```

Лістинг CompetenceController.cs:

```
    [Area("Admin")]
    [Authorize(Roles = "Admin")]
    public class CompetenceController : Controller
    {
        private    readonly    ApplicationDbContext
_context;

        public
CompetenceController(ApplicationDbContext context)
        {
            _context = context;
        }

        public IActionResult Index()
        {
            var            competences            =
_context.Competences.ToList();
            return View(competences);
        }

        public IActionResult Upsert(int? id)
        {
            CompetenceVM    competenceVM    =    new
CompetenceVM()
            {
                Competence = new Competence()
            };
            if (id == null)
            {
                return View(competenceVM);
            }
            else
            {
                competenceVM.Competence          =
_context.Competences.FirstOrDefault(x  =>  x.Id  ==
id);
                if    (competenceVM.Competence    ==
null)
                {
                    return NotFound();
                }
                return View(competenceVM);
            }
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public  IActionResult  Upsert(CompetenceVM
competenceVM)
        {
            if (ModelState.IsValid)
            {
```

```
                if (competenceVM.Competence.Id == 0)
                {
                    _context.Competences.Add(competenceVM.Competence);
                }
                else
                {
                    _context.Competences.Update(competenceVM.Competence);
                }
                _context.SaveChanges();
                return RedirectToAction(nameof(Index));
            }
            return View(competenceVM);
        }

        public IActionResult Delete(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }
            var competence = _context.Competences.Find(id);
            if (competence == null)
            {
                return NotFound();
            }

            _context.Competences.Remove(competence);
            _context.SaveChanges();
            return RedirectToAction(nameof(Index));
        }
    }
```

Лістинг DisciplinesChromosome.cs:

```
namespace AcademicDisciplinesGA.GA
{
    public class DisciplinesChromosome
    {
        private readonly ApplicationDbContext _dataContext;
        public List<CourseChromosome> Sequence { get; set; }
        public int Rank { get; set; }
        public int ECTSCount { get; set; }
        public int TeacherFitness { get; set; }
        public int ChairFitness { get; set; }
        public int CompetenceFitness { get; set; }
        public int LoadFitness { get; set; }
        public List<Competence> StudentCompetences { get; set; }
        public List<Teacher> Teachers { get; set; }
        public List<Chair> Chairs { get; set; }

        public static int _length = 10;

        public static int ECTSKap = 80;

        static readonly Random Random = new Random();

        public DisciplinesChromosome(ApplicationDbContext dataContext, List<Competence> studentInterests)
        {
            _dataContext = dataContext;
            StudentCompetences = studentInterests;
            Generate();
            AllocateCoursesWithPrerequisites();
            LoadFitness = CalculateLoadDistribution();
            ECTSCount = GetTotalECTS();
            CompetenceFitness = CalculateCompetenceFitness();
        }

        public DisciplinesChromosome(List<CourseChromosome> courses, List<Competence> studentInterests)
        {
            Sequence = courses.ToList();
            StudentCompetences = studentInterests;
            AllocateCoursesWithPrerequisites();
            LoadFitness = CalculateLoadDistribution();
            ECTSCount = GetTotalECTS();
            CompetenceFitness = CalculateCompetenceFitness();
        }

        private int CalculateCompetenceFitness()
        {
            int totalFitness = 0;

            foreach (var course in Sequence)
            {
                var matchedCompetences = course.RequiredCompetences.Intersect(StudentCompetences).Count();
                totalFitness += matchedCompetences * 5;
            }
```

```
            return totalFitness;
        }

        public int CalculateLoadDistribution()
        {
            var      semesterECTS      =      new
Dictionary<string, int>();
            foreach (var course in Sequence)
            {
                string   key   =   $"{course.Year}-
{course.Semester}";
                if
(!semesterECTS.ContainsKey(key))
                    semesterECTS[key] = 0;

                semesterECTS[key] += course.ECTS;
            }

            int             idealECTS             =
semesterECTS.Values.Sum() / semesterECTS.Count;
            int       loadVariancePenalty       =
semesterECTS.Values.Select(x   =>   Math.Abs(x   -
idealECTS)).Sum();

            return -loadVariancePenalty;
        }

        public                                  void
AllocateCoursesWithPrerequisites()
        {
            var   graph   =   new   Dictionary<int,
List<int>>();
            var  inDegree  =  new  Dictionary<int,
int>();
            var      courseIndexMap      =      new
Dictionary<int, int>();
            var     semesterCredits     =     new
Dictionary<int, int>();

            int totalECTS = Sequence.Sum(course =>
course.ECTS);
            int idealECTSPerSemester = totalECTS /
8;

            for   (int   index   =   0;   index   <
Sequence.Count; index++)
            {
                var course = Sequence[index];
                courseIndexMap[course.Id] = index;

                if (!graph.ContainsKey(course.Id))
                {
                    graph[course.Id]     =     new
List<int>();
                    inDegree[course.Id] = 0;
                }

                foreach     (var     prereq     in
course.Prerequisites)
                {
                    if
(!graph.ContainsKey(prereq))
                    {
                        graph[prereq]     =     new
List<int>();
                        inDegree[prereq] = 0;
                    }
                    graph[prereq].Add(course.Id);
                    inDegree[course.Id]++;
                }
            }

            for (int i = 1; i <= 8; i++)
            {
                semesterCredits[i] = 0;
            }

            var queue = new Queue<int>();
            foreach (var item in inDegree)
            {
                if (item.Value == 0)
                    queue.Enqueue(item.Key);
            }

            int semester = 1;
            while (queue.Count > 0)
            {
                int size = queue.Count;
                for (int i = 0; i < size; i++)
                {
                    var         courseId         =
queue.Dequeue();
                    var          course          =
Sequence[courseIndexMap[courseId]];

                    while
(semesterCredits[semester]    +    course.ECTS    >
idealECTSPerSemester + 5 && semester <= 8)
                    {
                        semester++;
                    }

                    if (semester <= 8)
                    {
                        course.Year = (semester -
1) / 2 + 1;
```

```
                        course.Semester          =
(semester % 2 == 0) ? 2 : 1;
                        semesterCredits[semester]
+= course.ECTS;

Sequence[courseIndexMap[courseId]] = course;
                    }

                    foreach   (var   neighbor   in
graph[courseId])
                    {
                        inDegree[neighbor]--;
                        if  (inDegree[neighbor]  ==
0)

queue.Enqueue(neighbor);
                    }
                }

                if (semester > 8)
                {
                    break;
                }
            }
        }

        public int GetTotalECTS()
        {
            var total = 0;

            for (int i = 0; i < Sequence.Count();
i++)
            {
                var fromCourse = Sequence[i];

                total += fromCourse.ECTS;
            }

            return total;
        }

        public int IsTeacherSelected(List<Teacher>
teachers)
        {
            var totalSelected = 0;
            for (int i = 0; i < Sequence.Count();
i++)
            {
                for (int j = 0; j < teachers.Count;
j++)
                {
                    if
(Sequence[i].TeacherId.Equals(teachers[j].Id))
                    {
                        totalSelected++;
                    }
                }
            }
            return totalSelected;
        }

        public   int   IsChairSelected(List<Chair>
chairs)
        {
            var totalSelected = 0;
            for (int i = 0; i < Sequence.Count();
i++)
            {
                for (int j = 0; j < chairs.Count;
j++)
                {
                    if
(Sequence[i].ChairId.Equals(chairs[j].Id))
                    {
                        totalSelected++;
                    }

                }
            }
            return totalSelected;
        }

        public                                  int
AddCourseWithPrerequisites(Course          course,
HashSet<int>                   selectedCoursesIds,
List<CourseChromosome> result)
        {
            if
(selectedCoursesIds.Contains(course.Id))
                return 0;

            int totalECTS = 0;

            foreach    (var    prerequisite    in
course.Prerequisites)
            {
                Course       prereqCourse       =
_dataContext.Courses.Find(prerequisite.Prerequisit
eId);
                if (prereqCourse != null)
                    totalECTS                   +=
AddCourseWithPrerequisites(prereqCourse,
selectedCoursesIds, result);
            }

            selectedCoursesIds.Add(course.Id);
            result.Add(new CourseChromosome
            {
```

```
                Id = course.Id,
                Title = course.Title,
                ECTS = course.ECTS,
                ChairId = course.ChairId,
                TeacherId = course.TeacherId,
                RequiredCompetences = course.Competences.Select(cc => cc.Competence).ToList(),
                Prerequisites = course.Prerequisites.Select(p => p.PrerequisiteId).ToList()
            });

            return totalECTS + course.ECTS;
        }

        public void Generate()
        {
            var allCourses = _dataContext.Courses
                .Include(course => course.Teacher)
                .Include(course => course.Chair)
                .Include(course => course.Competences).ThenInclude(courseCompetence => courseCompetence.Competence)
                .Include(course => course.Prerequisites)
                .ToList();

            var selectedCoursesIds = new HashSet<int>();
            var result = new List<CourseChromosome>();
            int currentECTS = 0;
            HashSet<int> triedIndices = new HashSet<int>();

            //var mandatoryCourses = allCourses.Where(c => c.Competences.Any(cc => cc.Competence.IsMandatory)).ToList();
            //foreach (var mandatoryCourse in mandatoryCourses)
            //{
            //    if (!selectedCoursesIds.Contains(mandatoryCourse.Id) && (currentECTS + mandatoryCourse.ECTS <= ECTSKap))
            //    {
            //        AddCourseWithPrerequisites(mandatoryCourse, selectedCoursesIds, result);
            //        currentECTS += mandatoryCourse.ECTS;
            //    }
            //}

            while (currentECTS < ECTSKap && triedIndices.Count < allCourses.Count)
            {
                int index = Random.Next(0, allCourses.Count);
                if (!triedIndices.Add(index))
                {
                    continue;
                }

                Course selectedCourse = allCourses[index];
                if (!selectedCoursesIds.Contains(selectedCourse.Id))
                {
                    int possibleECTS = AddCourseWithPrerequisites(selectedCourse, selectedCoursesIds, result);
                    if (currentECTS + possibleECTS <= ECTSKap)
                    {
                        currentECTS += possibleECTS;
                    }
                    else
                    {
                        result.RemoveAll(c => c.Id == selectedCourse.Id);
                        selectedCoursesIds.Remove(selectedCourse.Id);
                        break;
                    }
                }
            }

            Sequence = result;
        }
    }
}
```

Лістинг DisciplinesPopulation.cs:

```
public class DisciplinesPopulation
{
    private readonly ApplicationDbContext _dataContext;
    public List<DisciplinesChromosome> Population { get; set; }
    public List<Teacher> Teachers { get; set; }
    public List<Chair> Chairs { get; set; }
    public List<Competence> Competences { get; set; }
    public int GenerationCount { get; set; } = 0;
    public List<int> FitnessOverTime { get; private set; }
```

```
    public int NoImprovementCount { get; private set; } = 0;
    public bool HasConverged =>
          GenerationCount                          >
GAConfig.MaxGenerations
        ||          NoImprovementCount          >
GAConfig.MaxNoImprovementCount;

    private   float   previousConvergenceArea   =
float.MaxValue;

    public
DisciplinesPopulation(ApplicationDbContext
dbContext, List<Competence> competences)
    {
        _dataContext = dbContext;
        Competences = competences;
        FitnessOverTime = new List<int>();
        Spawn();
    }

    public void Spawn()
    {
        var                result                =
PopulationHelper.SpawnPopulation(_dataContext,
Competences);
        Population = result;
    }

    public void DoGeneration()
    {
        GenerationCount++;

        var         offspring         =         new
List<DisciplinesChromosome>();

        while          (offspring.Count          <
GAConfig.PopulationCount)
        {
            var mother = GetParent();
            var father = GetParent();

            while (mother == father)
            {
                father = GetParent();
            }

            var    (offspringA,    offspringB)    =
GetOffspring(mother, father);

            (offspringA,       offspringB)       =
Mutate(offspringA, offspringB);

            offspring.Add(offspringA);
            offspring.Add(offspringB);
        }

        Population.AddRange(offspring);

        MultiObjectiveHelper.UpdatePopulationFitness(Popul
ation);

        var       newPopulation       =       new
List<DisciplinesChromosome>();

        foreach      (var      individual      in
Population.OrderBy(i => i.Rank))
        {
            if
(!newPopulation.Contains(individual))
            {
                newPopulation.Add(individual);
            }
        }

        newPopulation                              =
newPopulation.Take(GAConfig.PopulationCount).ToLis
t();

        Population.Clear();

        newPopulation.ForEach(i                    =>
Population.Add(i));

        var  firstRank  =  Population.OrderBy(c  =>
c.CompetenceFitness).ThenBy(t => t.LoadFitness);
        var              currentArea              =
MultiObjectiveHelper.CalculateArea(firstRank);

        if   (Math.Abs(previousConvergenceArea   -
currentArea) < 0.1)
        {
            NoImprovementCount++;
        }
        else
        {
            NoImprovementCount = 0;
            previousConvergenceArea = currentArea;
        }
    }
    public                    DisciplinesChromosome
GetBestIndividual()
    {
        DisciplinesChromosome   bestChromosome   =
null;
        int maxFitness = int.MinValue;
```

```
        foreach (var chromosome in Population)
        {
            if (chromosome.CompetenceFitness + chromosome.LoadFitness > maxFitness)
            {
                maxFitness = chromosome.CompetenceFitness + chromosome.LoadFitness;
                bestChromosome = chromosome;
            }
        }

        return bestChromosome;
    }

    private (DisciplinesChromosome, DisciplinesChromosome) Mutate(DisciplinesChromosome individualA, DisciplinesChromosome individualB)
    {
        return PopulationHelper.Mutate(individualA, individualB, _dataContext, Competences);
    }

    private (DisciplinesChromosome, DisciplinesChromosome) GetOffspring(DisciplinesChromosome individualA, DisciplinesChromosome individualB)
    {
        var offspringA = DoCrossover(individualA, individualB);
        var offspringB = DoCrossover(individualB, individualA);

        return (offspringA, offspringB);
    }

    private DisciplinesChromosome DoCrossover(DisciplinesChromosome individualA, DisciplinesChromosome individualB)
    {
        return PopulationHelper.DoCrossover(individualA, individualB, Competences);
    }

    private DisciplinesChromosome GetParent()
    {
        var (candidate1, candidate2) = PopulationHelper.GetCandidateParents(Population);
        return PopulationHelper.TournamentSelection(candidate1, candidate2);
    }

}
```

Лістинг GAConfig.cs:

```
namespace AcademicDisciplinesGA.GA
{
    public static class GAConfig
    {
        public static int MaxGenerations => 1000;
        public static double MutationChance => 0.05;
        public static int PopulationCount => 20;
        public static int MaxNoImprovementCount => 20;
    }
}
```

Лістинг MultiObjectiveHelper.cs:

```
public static class MultiObjectiveHelper
{
    public static void UpdatePopulationFitness(List<DisciplinesChromosome> population)
    {
        foreach (var individual in population)
        {
            individual.Rank = -1;
        }
        CalculateRank(population);
    }

    private static void CalculateRank(List<DisciplinesChromosome> population)
    {
        var currentFront = new List<DisciplinesChromosome>();
        var individualsDominated = new Dictionary<DisciplinesChromosome, List<DisciplinesChromosome>>();
        var individualDominationCount = new Dictionary<DisciplinesChromosome, int>();

        foreach (var individualA in population)
        {
            individualsDominated.Add(individualA, new List<DisciplinesChromosome>());
            individualDominationCount.Add(individualA, 0);

            foreach (var individualB in population)
```

```
            {
                if (individualA == individualB)
                {
                    continue;
                }

                if          (Dominates(individualB,
individualA))
                {

individualDominationCount[individualA]++;
                }
                else   if   (Dominates(individualA,
individualB))
                {

individualsDominated[individualA].Add(individualB)
;
                }
            }

            if
(individualDominationCount[individualA] == 0)
            {
                individualA.Rank = 0;
                currentFront.Add(individualA);
            }
        }

        var i = 0;
        while (currentFront.Any())
        {
            var        nextFront        =        new
List<DisciplinesChromosome>();
            foreach     (var     individualA     in
currentFront)
            {
                foreach    (var    individualB    in
individualsDominated[individualA])
                {

individualDominationCount[individualB]--;
                    if
(individualDominationCount[individualB] == 0)
                    {
                        individualB.Rank = i + 1;

nextFront.Add(individualB);
                    }
                }
            }
            i++;

            currentFront = nextFront;
        }
    }

    public               static               bool
Dominates(DisciplinesChromosome                  a,
DisciplinesChromosome b)
    {
        int ECTSKap = 50;
        bool   withinECTSCapA   =   a.ECTSCount   <=
ECTSKap;
        bool   withinECTSCapB   =   b.ECTSCount   <=
ECTSKap;

        if  (!withinECTSCapA  &&  !withinECTSCapB)
return false;

        bool betterOrEqualOnAll =
            a.TeacherFitness  >=  b.TeacherFitness
&&
            a.ChairFitness >= b.ChairFitness &&
            a.CompetenceFitness                 >=
b.CompetenceFitness &&
            a.LoadFitness >= b.LoadFitness &&
            withinECTSCapA && !withinECTSCapB;

        bool betterOnAtLeastOne =
            a.TeacherFitness > b.TeacherFitness ||
            a.ChairFitness > b.ChairFitness ||
            a.CompetenceFitness                  >
b.CompetenceFitness ||
            a.LoadFitness > b.LoadFitness ||
            (withinECTSCapA && !withinECTSCapB);

        return        betterOrEqualOnAll        &&
betterOnAtLeastOne;
    }

    internal             static             float
CalculateArea(IOrderedEnumerable<DisciplinesChromo
some> firstRank)
    {
        var slices = GetSlices(firstRank);

        return slices.Sum(s => s.Area);
    }

    private       static       IEnumerable<Slice>
GetSlices(IOrderedEnumerable<DisciplinesChromosome
> firstRank)
    {
        var previousSlice = new Slice(0, 0, 0, 0);

        foreach(var individual in firstRank)
```

```
        {
            previousSlice = new Slice(previousSlice.XUpper, 0, individual.TeacherFitness, individual.ChairFitness);
            yield return previousSlice;
        }
    }
}
```

Лістинг PopulationHelper.cs:

```
public class PopulationHelper
 {
    private static readonly Random random = new Random();

    public static List<DisciplinesChromosome> SpawnPopulation(
        ApplicationDbContext dataContext, List<Competence> competences)
    {
        var population = new HashSet<DisciplinesChromosome>();

        int remainingCount = GAConfig.PopulationCount;
        while (remainingCount > 0)
        {
            var individuals = Enumerable.Range(0, remainingCount)
                                        .Select(i => new DisciplinesChromosome(dataContext, competences))

.ToList();

            foreach (var individual in individuals)
            {
                population.Add(individual);
            }

            remainingCount = GAConfig.PopulationCount - population.Count;
        }
        return population.ToList();
    }

    public static (DisciplinesChromosome, DisciplinesChromosome) GetCandidateParents(List<DisciplinesChromosome> population)
    {
        var candidateA = population[random.Next(GAConfig.PopulationCount)];
        var candidateB = population[random.Next(GAConfig.PopulationCount)];

        while (candidateA == candidateB)
        {
            candidateB = population[random.Next(GAConfig.PopulationCount)];
        }

        return (candidateA, candidateB);
    }

    public static DisciplinesChromosome TournamentSelection(DisciplinesChromosome candidateA, DisciplinesChromosome candidateB)
    {
        if (candidateA.Rank <= candidateB.Rank)
        {
            return candidateA;
        }
        else
        {
            return candidateB;
        }

    }

    public static DisciplinesChromosome DoCrossover(DisciplinesChromosome individualA, DisciplinesChromosome individualB, List<Competence> competences)
    {
        var offspringSequence = new List<CourseChromosome>();
        var appeared = new HashSet<int>();
        int currentECTS = 0;
        int maxECTS = 50;

        int minSequenceLength = Math.Min(individualA.Sequence.Count, individualB.Sequence.Count);

        List<int> indices = Enumerable.Range(0, minSequenceLength).OrderBy(x => random.Next()).ToList();

        foreach (int index in indices)
        {
            if (currentECTS >= maxECTS)
                break;

            CourseChromosome chosenCourse = random.NextDouble() > 0.5 ?
```

```
individualA.Sequence[index] : individualB.Sequence[index];

            if (!appeared.Contains(chosenCourse.Id) && currentECTS + chosenCourse.ECTS <= maxECTS)
            {
                if (ArePrerequisitesMet(chosenCourse, offspringSequence))
                {
                    offspringSequence.Add(chosenCourse);
                    appeared.Add(chosenCourse.Id);
                    currentECTS += chosenCourse.ECTS;
                }
            }
        }

        return new DisciplinesChromosome(offspringSequence, competences);
    }


    private static void AddMandatoryCourses(List<CourseChromosome> offspringSequence, DisciplinesChromosome individualA, DisciplinesChromosome individualB, HashSet<int> appeared, ref int currentECTS, int maxECTS)
    {
        foreach (var individual in new[] { individualA, individualB })
        {
            foreach (var course in individual.Sequence)
            {
                if (course.RequiredCompetences.Any(c => c.IsMandatory) && !appeared.Contains(course.Id) && currentECTS + course.ECTS <= maxECTS)
                {
                    if (ArePrerequisitesMet(course, offspringSequence))
                    {
                        offspringSequence.Add(course);
                        appeared.Add(course.Id);
                        currentECTS += course.ECTS;
                    }
                }
            }
        }

        private static bool ArePrerequisitesMet(CourseChromosome course, List<CourseChromosome> offspringSequence)
        {
            foreach (var prereqId in course.Prerequisites)
            {
                if (!offspringSequence.Any(c => c.Id == prereqId))
                {
                    return false;
                }
            }
            return true;
        }

        public static DisciplinesChromosome DoMutate(DisciplinesChromosome individual, ApplicationDbContext dataContext, List<Competence> competences)
        {
            var courses = dataContext.Courses
                .Include(course => course.Teacher)
                .Include(course => course.Chair)
                .Include(course => course.Competences).ThenInclude(courseCompetence => courseCompetence.Competence)
                .Include(course => course.Prerequisites)
                .ToList();

            var sequence = individual.Sequence;
            int randomIndex = random.Next(0, sequence.Count);
            int newCourseId;
            CourseChromosome newCourse;

            do
            {
                newCourseId = random.Next(courses.First().Id, courses.Last().Id + 1);
                var selectedCourse = courses.Find(c => c.Id.Equals(newCourseId));
                newCourse = new CourseChromosome()
                {
                    Id = selectedCourse.Id,
                    Title = selectedCourse.Title,
                    ECTS = selectedCourse.ECTS,
```

```
                ChairId = selectedCourse.ChairId,
                TeacherId = selectedCourse.TeacherId,
                RequiredCompetences = selectedCourse.Competences.Select(cc => cc.Competence).ToList(),
                Prerequisites = selectedCourse.Prerequisites.Select(p => p.PrerequisiteId).ToList()
            };
        }
        while (sequence.Contains(newCourse));

        sequence[randomIndex] = newCourse;

        return new DisciplinesChromosome(sequence, competences);
    }

    public static (DisciplinesChromosome, DisciplinesChromosome) Mutate(DisciplinesChromosome individualA, DisciplinesChromosome individualB, ApplicationDbContext dataContext, List<Competence> competences)
    {
        var newIndividualA = new DisciplinesChromosome(individualA.Sequence, competences);
        var newindividualB = new DisciplinesChromosome(individualB.Sequence, competences);

        if (random.NextDouble() < GAConfig.MutationChance)
        {
            newIndividualA = DoMutate(individualA, dataContext, competences);
        }

        if (random.NextDouble() < GAConfig.MutationChance)
        {
            newindividualB = DoMutate(individualB, dataContext, competences);
        }

        return (newIndividualA, newindividualB);
    }

}
```

Лістинг ApplicationUser.cs:

```
namespace AcademicDisciplinesGA.Models
{
    public class ApplicationUser : IdentityUser
    {
        [Required]
        public string Name { get; set; }

        [Display(Name = "Chair")]
        public int? ChairId { get; set; }
        [ForeignKey("ChairId")]
        [ValidateNever]
        public Chair Chair { get; set; }
    }
}
```

Лістинг Chair.cs:

```
namespace AcademicDisciplinesGA.Models
{
    public class Chair
    {
        public int Id { get; set; }
        public string Title { get; set; }
        [Display(Name ="Faculty")]
        public int FacultyId { get; set; }
        [ValidateNever]
        public Faculty Faculty { get; set; }
    }
}
```

Лістинг Course.cs:

```
    public class Course
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public int ECTS { get; set; }

        public int TeacherId { get; set; }

        [ValidateNever]
        public Teacher Teacher { get; set; }

        public int ChairId { get; set; }

        [ValidateNever]
        public Chair Chair { get; set; }

        [ValidateNever]
        public List<CourseCompetence> Competences { get; set; }

        [ValidateNever]
        public List<CoursePrerequisite> Prerequisites { get; set; }
    }
```

Лістинг CourseChromosome.cs:

```
    public struct CourseChromosome
    {
        public int Id { get; set; }
```

```
        public string Title { get; set; }
        public int ECTS { get; set; }
        public int ChairId { get; set; }
        public int TeacherId { get; set; }
        public List<Competence> RequiredCompetences { get; set; }
        public List<int> Prerequisites { get; set; }
        public int Year { get; set; }
        public int Semester { get; set; }
    }
```

Лістинг Faculty.cs:

```
namespace AcademicDisciplinesGA.Models
{
    public class Faculty
    {
        public int Id { get; set; }
        public string Title { get; set; }
        [ValidateNever]
        public List<Chair> Chairs { get; set; }
    }
}
```

Лістинг Teacher.cs:

```
namespace AcademicDisciplinesGA.Models
{
    public class Teacher
    {
        public int Id { get; set; }
        public string Name { get; set; }
        [ValidateNever]
        public List<Course> Courses { get; set; }
    }
}
```

Лістинг CourseCompetence.cs:

```
    public class CourseCompetence
    {
        public int CourseId { get; set; }
        public Course Course { get; set; }

        public int CompetenceId { get; set; }
        public Competence Competence { get; set; }
    }
```

Лістинг CoursePrerequisite.cs:

```
    public class CoursePrerequisite
    {
        public int CourseId { get; set; }
        public Course Course { get; set; }

        public int PrerequisiteId { get; set; }
        public Course Prerequisite { get; set; }
    }
```

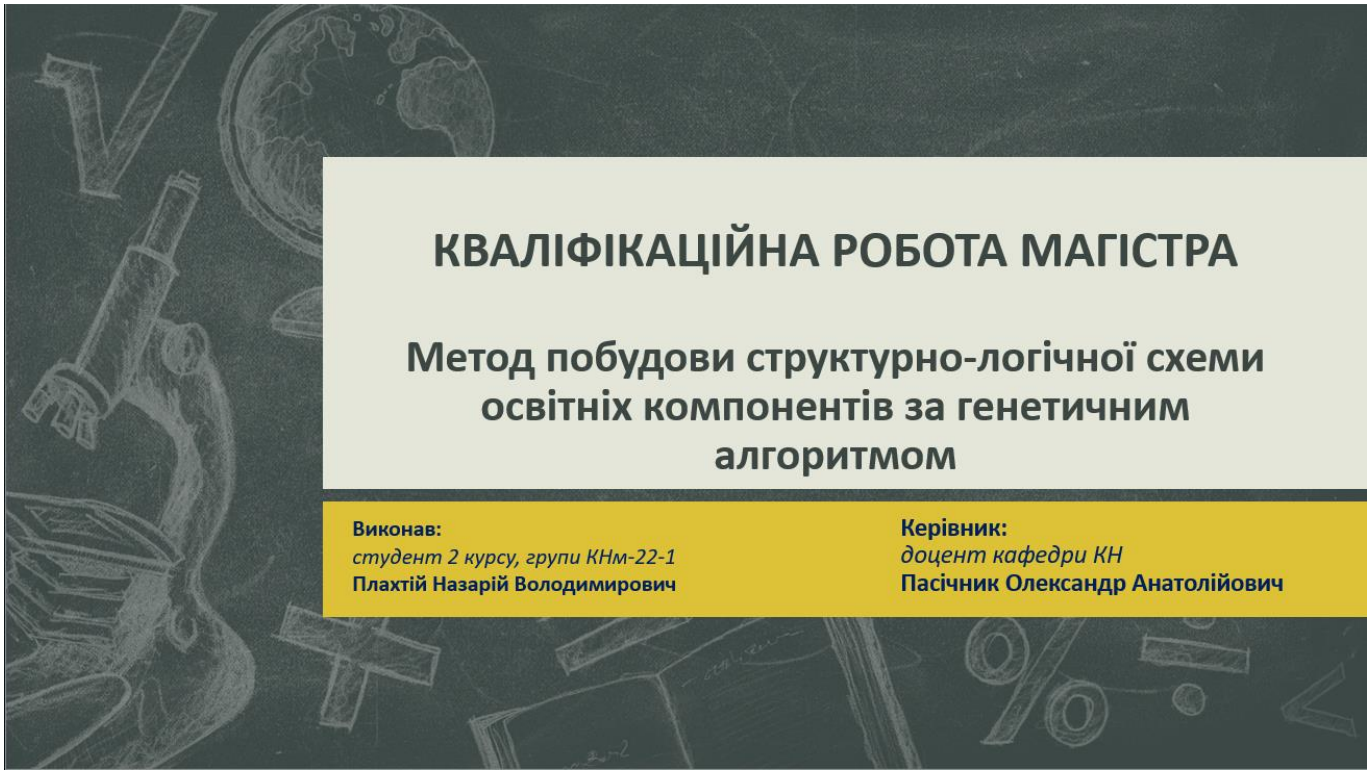
Лістинг EducationPlanController.cs:

```
    [Area("User")]
    [Authorize]
    public class EducationPlanController : Controller
    {
        private readonly ApplicationDbContext _context;
        DisciplinesPopulation population;
        DisciplinesChromosome disciplinesChromosomes;

        public EducationPlanController(ApplicationDbContext context)
        {
            _context = context;
        }

        public IActionResult Index()
        {
            var model = new CompetenceSelectionVM
            {
                Competences = _context.Competences
                    .Where(c => !c.IsMandatory)
                    .Select(c => new SelectListItem
                    {
                        Text = c.Name,
                        Value = c.Id.ToString()
                    }).ToList()
            };

            return View(model);
        }

        [HttpPost]
        public IActionResult SelectCompetences(CompetenceSelectionVM model)
        {
            if (model.SelectedCompetenceIds != null && model.SelectedCompetenceIds.Count > 0)
            {
                var user = _context.ApplicationUsers.FirstOrDefault(x => x.Id == model.User.Id);

                return RedirectToAction("GenerateEducationPlan", new { compIds = model.SelectedCompetenceIds });
            }

            return View(model);
        }
```

```
        public IActionResult GenerateEducationPlan(List<int> compIds)
        {
            var mandatoryCompIds = _context.Competences
                .Where(c => c.IsMandatory)
                .Select(c => c.Id)
                .ToList();

            compIds.AddRange(mandatoryCompIds);

            var competences = _context.Competences.Where(c => compIds.Contains(c.Id)).ToList();
            population = new DisciplinesPopulation(_context, competences);

            disciplinesChromosomes = Run();

            ViewData["Courses"] = ConvertToCourses(disciplinesChromosomes.Sequence);

            return View("Result");
        }

        private DisciplinesChromosome Run()
        {
            while (population.GenerationCount < GAConfig.MaxGenerations || population.NoImprovementCount < GAConfig.MaxNoImprovementCount)
            {
                population.DoGeneration();
            }

            return population.GetBestIndividual();
        }

        public List<CourseVM> ConvertToCourses(List<CourseChromosome> courseChromosomes)
        {
            var courseIds = courseChromosomes.Select(c => c.Id).ToList();
            var coursesDetails = _context.Courses
                .Where(c => courseIds.Contains(c.Id))
                .Include(c => c.Teacher)
                .Include(c => c.Chair)
                .Include(c => c.Competences).ThenInclude(cc => cc.Competence)
                .Include(c => c.Prerequisites).ThenInclude(pr => pr.Prerequisite)
                .ToList();

            var courses = courseChromosomes.Select(cc =>
            {
                var courseData = coursesDetails.FirstOrDefault(c => c.Id == cc.Id);
                if (courseData == null)
                    return null;

                return new CourseVM
                {
                    Id = cc.Id,
                    Title = cc.Title,
                    ECTS = cc.ECTS,
                    Teacher = courseData.Teacher,
                    Chair = courseData.Chair,
                    Competences = courseData.Competences,
                    Prerequisites = courseData.Prerequisites,
                    Year = cc.Year,
                    Semester = cc.Semester
                };
            }).Where(c => c != null).ToList();

            courses.Sort((x, y) =>
            {
                int yearComparison = x.Year.CompareTo(y.Year);
                if (yearComparison == 0)
                {
                    return x.Semester.CompareTo(y.Semester);
                }
                return yearComparison;
            });

            return courses;
        }
    }
```

Додаток Б

Презентаційний матеріал



Актуальність

В контексті змін сучасного світу, особливо у сфері технологій та знань, освіта стикається з величезними викликами у питанні своєї актуалізації та оптимізації. Питання покращення освітніх процесів стає все актуальнішим, оскільки і державні, і приватні освітні заклади прагнуть до прогресивних змін, які забезпечать ефективне та якісне навчання.

Традиційно, оптимізація в освіті часто використовувалась у контексті економії ресурсів, що нерідко призводило до скорочення чисельності кадрів, зменшення бюджетного фінансування, або навіть до закриття навчальних закладів. Однак, сучасне розуміння та застосування оптимізації в освіті має на меті досягнення оптимальних результатів з використанням існуючих ресурсів, що веде до підвищення якості освіти та забезпечення можливості для студентів розвивати свої потенціали.

Актуальність дослідження полягає в розробці та впровадженні інноваційних підходів до побудови структурно-логічних схем освітніх компонентів з використанням генетичних алгоритмів. Оптимізація у цьому контексті означає не лише покращення існуючих методів навчання, а й створення систем, здатних адаптуватися та відповідати індивідуальним потребам студентів.

Мета і задачі роботи

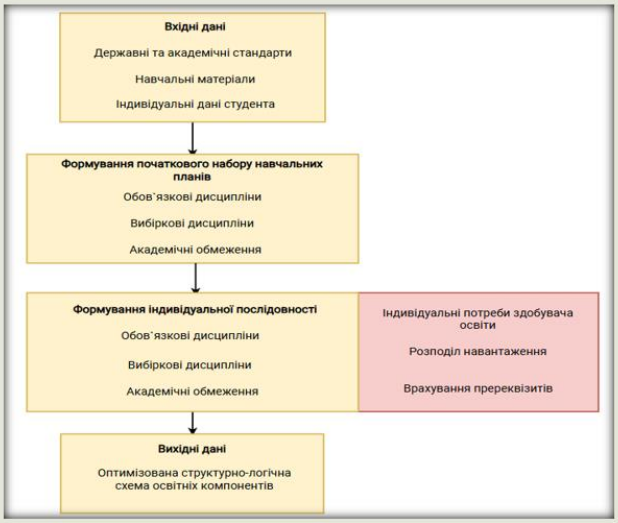
Метою кваліфікаційної роботи магістра є полягає у підвищенні якості формування структурно-логічної схеми освітніх компонентів за генетичним алгоритмом. Для досягнення цієї мети були визначені наступні дослідницькі задачі:

- Провести аналіз методів побудови структурно-логічної схеми освітніх компонентів, можливостей, переваг та недоліків генетичного алгоритму.
- Спроектувати метод побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом.
- Визначити набір критеріїв для оцінки підвищення якості формування побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом.
- Виконати програмну реалізацію методу побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом.
- Провести експериментальне дослідження реалізованого методу за тестовими наборами даних шляхом оцінки якості побудови структурно-логічної схеми освітніх компонентів.

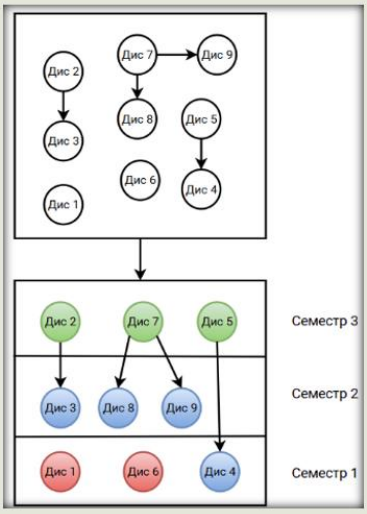
Об'єкт дослідження – процес формування структурно-логічної схеми освітніх компонентів за генетичним алгоритмом.

Предмет дослідження – моделі, алгоритми та засоби побудови структурно-логічної схеми освітніх компонентів.

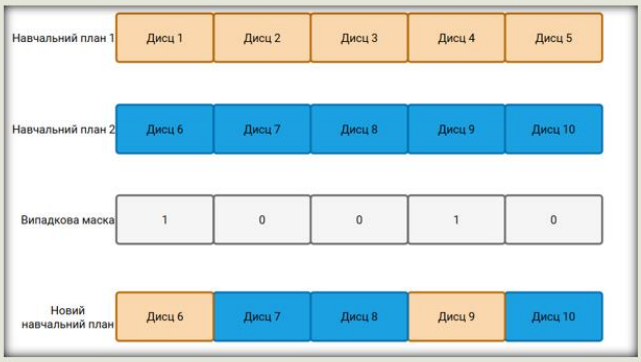
Метод побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом

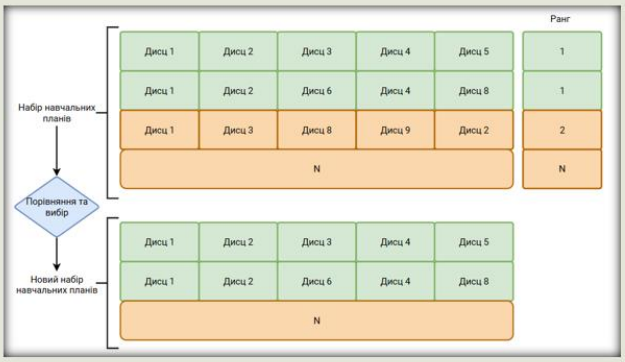


Етапи роботи методу

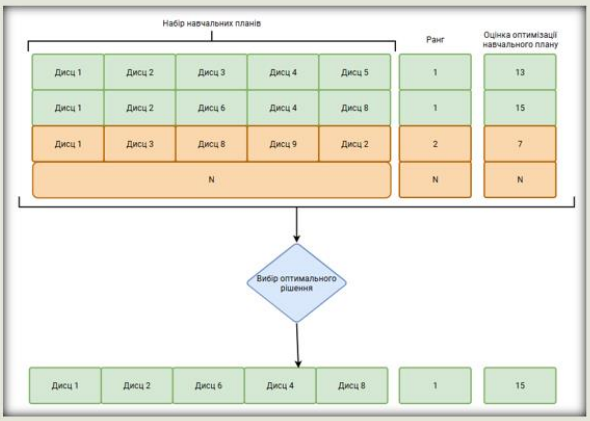


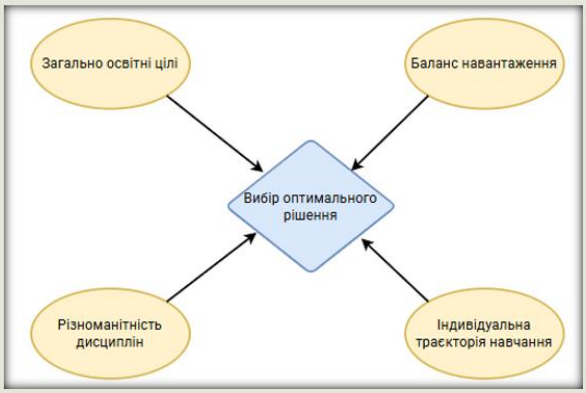
Етапи роботи методу





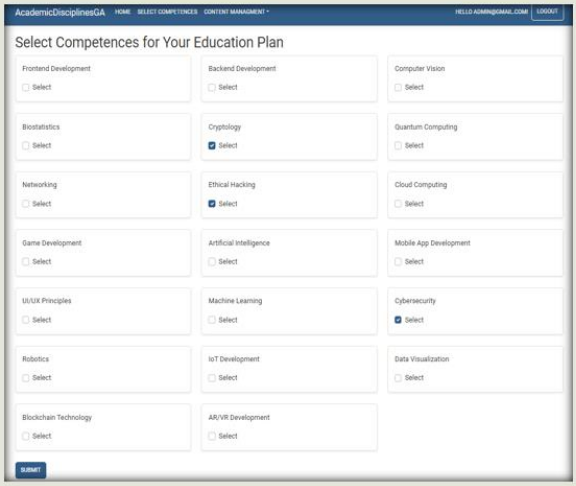
Етапи роботи методу





Програмна реалізація

Експериментальне дослідження методу побудови структурно-логічної схеми освітніх компонентів



Індивідуальний навчальний план

Title	ECTS	Chair	Teacher	Prerequisite	Competencies	Year	Semester
Frontend Web Development	5	Department of Computer Technologies	Dr. Daniel Moore	None	Frontend Development	1	1
Intro to Computer Science	5	Department of Computer Technologies	Dr. Emily Johnson	None	Programming Basics	1	1
Augmented and Virtual Reality	6	Department of Computer Technologies	Prof. Christopher Taylor	Frontend Web Development	AR/VR Development	1	2
System Analysis	5	Department of Computer Technologies	Prof. Christopher Taylor	None	Systems Analysis	1	2
Advanced Programming	6	Department of Computer Technologies	Prof. Mark Brown	Intro to Computer Science	Object-Oriented Design	2	1
Backend Web Development	6	Department of Computer Technologies	Prof. Laura Thompson	Frontend Web Development	Backend Development	2	1
Web Development Fundamentals	5	Department of Computer Technologies	Prof. Christopher Taylor	Intro to Computer Science	Web Development	2	2
Networking Fundamentals	4	Department of Computer Technologies	Dr. Susan Clark	Intro to Computer Science	Networking	2	2
Databases	5	Department of Computer Technologies	Dr. Karen Wilson	Intro to Computer Science	Database Systems	2	2
Algorithms and Data Structures	5	Department of Computer Technologies	Dr. Susan Clark	Advanced Programming	Data Structures and Algorithms	3	1
Introduction to AI	5	Department of Computer Technologies	Dr. Daniel Moore	Backend Web Development	Artificial Intelligence	3	1
Introduction to Ethical Hacking	5	Department of Computer Technologies	Prof. Joseph Davis	Web Development Fundamentals	Ethical Hacking	3	2
Cybersecurity	5	Department of Computer Technologies	Prof. Mark Brown	Networking Fundamentals	Cybersecurity	3	2
Cloud Computing Basics	5	Department of Computer Technologies	Dr. Karen Wilson	Algorithms and Data Structures	Cloud Computing	4	1
Introduction to Cryptology	5	Department of Computer Technologies	Dr. Emily Johnson	Databases	Cryptology	4	1

Висновки

У рамках виконання кваліфікаційної роботи магістра було розроблено метод для підвищення якості формування структурно-логічної схеми освітніх компонентів за генетичним алгоритмом.

Було проведено детальний аналіз методів побудови структурно-логічної схеми освітніх компонентів. Аналіз дозволив ідентифікувати прогалини в існуючих методах та визначити потенціал для їх покращення з використанням генетичних алгоритмів.

Спроектовано метод побудови структурно-логічної схеми освітніх компонентів. Було визначено набір критеріїв для оцінки підвищення якості формування побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом.

Програмно реалізовано метод побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом, та проведено експериментальне дослідження за тестовими наборами даних.

У підсумку, розроблений метод разом з його програмною реалізацією відповідає всім передбаченим критеріям, зокрема, оптимізацію обсягу кредитів ECTS, впорядкованість дисциплін у відповідності до необхідних пререквізитів, а також врахування індивідуалізованих навчальних побажань студента.

Дякую за увагу

Додаток В

Наукова публікація

DOI 10.31891/2307-5732-2024-335-3-26
УДК 004.8+519.7+37.09+378

ПЛАХТІЙ НАЗАРІЙ
Хмельницький національний університет
https://orcid.org/0009-0005-3927-6965
e-mail:nazarplahtij@gmail.com
ПАСІЧНИК ОЛЕКСАНДР
Хмельницький національний університет
https://orcid.org/0000-0002-8760-4688
e-mail:o.a.pasichnyk@gmail.com
МАНЗЮК ЕДУАРД
Хмельницький національний університет
https://orcid.org/0000-0002-7310-2126
e-mail: eduard.em.km@gmail.com
СКРИПНИК ТЕТЯНА
Хмельницький національний університет
https://orcid.org/0000-0002-8531-5348
e-mail: tkskripnik1970@gmail.com
ПЕТРОВСЬКИЙ СЕРГІЙ
Хмельницький національний університет
https://orcid.org/0000-0002-0590-0484
e-mail: petrovskijs69@gmail.com

МЕТОД ФОРМУВАННЯ ПУЛУ ВИБІРКОВИХ НАВЧАЛЬНИХ ДИСЦИПЛІН НА ОСНОВІ ГЕНЕТИЧНОГО АЛГОРИТМУ

У роботі розглядається створення методу формування пулу вибіркових навчальних дисциплін на основі генетичного алгоритму та його реалізація як інформаційної системи. Актуальність дослідження полягає в зростаючому інтересі до впровадження ефективних методів управління навчальними програмами в закладах вищої освіти. Генетичні алгоритми, які вже успішно використовуються в різних галузях, надають можливість автоматизованого вибору дисциплін, що відповідають потребам студентів та меті навчання.

Ключові слова: генетичний алгоритм, спосіб формування пулу, вибіркова дисципліна.

PLAHTII NAZARII, PASICHNYK OLEKSANDR, MANZIUK EDUARD, SKRYPNYK TETIANA, PETROVSKYI SERGII
Khmelnytskyi National University

METHOD OF FORMING A POOL OF ELECTIVE COURSES BASED ON THE GENETIC ALGORITHM

This paper gives an in-depth analysis on the development and application of a method for creating a comprehensive elective course pool, which is fundamentally based on a genetic algorithm, and efficiently incorporated within an information system. The significance and extent of this research becomes prominent due to the surging interest and increased demand in employing efficacious management techniques to streamline educational curricula in higher education institutions, thereby catering to students' particular educational needs and predetermined objectives.

Genetic algorithms, which have been extensively applied and have achieved noteworthy success in a wide array of fields, have shown significant potential for an automated selection of disciplines, ultimately leading to a precise alignment with a student's individual objectives and enhancing their overall educational experience.

The study dedicates a significant portion of its focus to the implementation phases of the method. It illustrates an exhaustive analysis of the pre-existing methodologies applied in curriculum development. Furthermore, it offers a well-rounded assessment of the potentials and inherent limitations of genetic algorithms in the context of curriculum development. The research elaborates on the design and deployment of a specialized information system that efficaciously supports the automated compilation of an elective discipline pool.

A substantial part of the study also delves into conducting an empirical risk analysis, and identifies an array of key factors that weigh in on the selection of disciplines. Some of these influential factors include specific characteristics of students, availability of faculty members, accessibility and allocation of institutional resources, and various budget constraints.

The research results demonstrate that employing a genetic algorithm for curriculum formation can enhance the quality of education by providing greater individualization and alignment with students' educational needs.

Key words: genetic algorithm, method of forming a pool, elective course.

Постановка проблеми у загальному вигляді та її зв'язок із важливими науковими чи практичними завданнями

Базовою тенденцією розвитку сьогодення є широке, практично всеохоплююче запровадження інформаційних технологій зі стійким трендом до їх поглибленої інтелектуалізації. Гнучким та ефективним кібернетичним інструментом вирішення достатнього широкого кола практичних задач є генетичні алгоритми.

Іншою панівною тенденцією є широка модернізація системи освіти, переосмислення її місця та ролі у суспільному житті. Це супроводжується запровадженням нової парадигми та новітніх підходів щодо формування змісту освітнього процесу. Основним принципом сучасної вищої освіти є студентоцентрований підхід, який передбачає, з одного боку, безпосередню участь здобувача вищої освіти у формуванні власної освітньої траєкторії, а, з іншого боку, надання йому широких можливостей щодо напрямку та рівня отримуваних знань та вмінь. Практична реалізація зазначеного підходу передбачає наявність в освітній

програмі обов'язкових та вибіркових начальних дисциплін. Для задоволення потреб здобувачів вищої освіти навчальні заклади пропонують широкий та різноманітний перелік вибіркових навчальних дисциплін, до прикладу, в Хмельницькому національному університеті їх кількість перевищує 1000. Остання обставина робить нетривіальною задачу визначення переліку вибіркових дисциплін, що додатково ускладняється різноманіттям прагнень здобувачів вищої освіти щодо результатів навчання.

Аналіз останніх досліджень і публікацій

Генетичний алгоритм є одним з ключових методів оптимізації, що застосовується в широкому спектрі наукових та технічних областей [1, 2], у тому числі й в освітній галузі, зокрема в частині рішення управлінських задач. Так, одним з прикладів використання генетичного алгоритму в освітянській діяльності пов'язаний з побудовою розкладів занять [3]. Це дослідження пропонує комплексний підхід до вирішення задачі з використанням генетичного алгоритму, що дозволяє з урахуванням відповідних вимог автоматизовано формувати розклади занять для закладів вищої.

У дослідженні [4] розглянуто використання генетичного алгоритму для формування індивідуальної освітньої траєкторії студентів вищих навчальних закладів. Основна ціль даного наукового дослідження полягає у формуванні освітньої траєкторії, яка включала б тільки ті дисципліни, які розвивають унікальні компетенції. Результати цієї роботи можуть бути корисними гарантам освітніх програм в їх практичній діяльності при формування навчальних планів, безпосередньому формуванні освітніх програм та їх вдосконаленні.

Формулювання цілей статті

Робота присвячена реалізації студентоцентрованого підходу в частині вибіркової складової освітньої програми та полягає у створенні методу формування пулу вибіркових навчальних дисциплін на основі генетичного алгоритму. Цей метод орієнтований на здобувачів вищої освіти та дозволяє покращити навчання у закладах вищої освіти та передбачає формування пулу вибіркових дисциплін з точки зору власних прагнень та потреб здобувачів вищої освіти.

Виклад основного матеріалу

Склад вибіркових дисциплін у закладах вищої освіти визначається навчальним планом спеціальності та вимогами до її здобуття. Вибіркові дисципліни мають бути пов'язані з основними напрямами та спеціалізаціями навчання, а також відповідати профілю та вимогам ринку праці.

У загальному розумінні, склад вибіркових дисциплін може бути доволі різноманітним, але важливо дотримуватись прагнень здобувача вищої освіти що до результатів навчання, які визначаються спеціалізованими дисциплінами, що доповнюють обов'язкові. Загальноосвітні дисципліни допомагають студентам отримати широке загальне освітнє підґрунтя, вивчити загальні принципи та закономірності, розвинути критичне мислення та аналітичні навички. Спеціалізовані дисципліни допомагають студентам засвоїти конкретні знання та навички, необхідні для роботи в певних професійних галузях.

Заклад вищої освіти розробляє освітню програму для кожної спеціальності, яка визначає загальний перелік обов'язкових і вибіркових дисциплін, що необхідні для отримання кваліфікації. Освітня програма повинна відповідати вимогам державних стандартів вищої освіти та реальним потребам ринку праці.

Формування загального переліку вибіркових дисциплін здійснюється на рівні закладу вищої освіти та містить пропозиції всіх факультетів, кафедр та викладачів.

Перш ніж почати опис роботи генетичного алгоритму, важливо ознайомитися з деякою базовою термінологією, що буде використовуватись надалі.

Популяція – це множина хромосом, що використовується для пошуку оптимального розв'язку задачі. Популяція може складатися з кількох десятків або сотень індивідуумів. У даному випадку популяція складається з певної кількості пулів вибіркових дисциплін. У свою чергу хромосома – це послідовність генетичних елементів, які кодують інформацію про характеристики індивідуума. У генетичному алгоритмі хромосома представляє кандидата на рішення задачі. В контексті задачі хромосома – це пул вибіркових дисциплін, а ген – це окрема одиниця у складі хромосоми, яка кодує конкретну характеристику індивідуума, тобто геном. В контексті даної задачі, геном є унікальна для хромосоми навчальна дисципліна, що має наступний список властивостей: факультет, кафедру, викладача та кількість кредитів ЄКТС.

Генетичний алгоритм може бути використаним для розв'язання задачі побудови бажаного пулу вибіркових навчальних дисциплін для здобувача вищої освіти. Основна ідея полягає в тому, щоб створити популяцію потенційних пулів навчальних дисциплін і використовувати еволюційний підхід, щоб знайти оптимальний пул для поданого здобувача вищої освіти.

Створення початкової популяції у такому випадку можна здійснити за допомогою випадкового вибору дисциплін з наявного пулу. Отримана популяція може містити дублікати дисциплін або ж бути недостатньо різноманітною. У такому випадку можна застосувати додаткові методи селекції, наприклад, метод турніру.

Турнірний відбір є одним з методів відбору особин у генетичному алгоритмі, який полягає у тому, щоб обрати кращих особин з декількох випадково вибраних підмножин популяції. Процес виконання турніру може бути описаний таким чином:

- випадковим чином обирається певна кількість особин з популяції;
- серед вибраних особин обирається найкраща;
- вибрана особина стає батьківською для наступного покоління.

Кількість особин, які беруть участь у турнірі, може бути встановлена заздалегідь. Зазвичай використовують значення від 2 до 10. Збільшення кількості учасників турніру збільшує ймовірність вибору кращої особини, але при цьому збільшується обчислювальна складність.

Наступним кроком після формування початкової популяції, виконується кросинговер. Схрещування або кросинговер – це операція, яка здійснюється між двома батьківськими хромосомами і призводить до створення нової хромосоми-потомка. Цей процес полягає у випадковому виборі двох батьківських особин, з яких будуть обрані певні відрізки геномів. Далі, випадково обирається точка перетину (точка ділення геному), і генетична інформація до цієї точки береться з однієї батьківської особини, а від заданої точки і до кінця геному, з іншої батьківської особини. Таким чином, створюються нові особини, що містять комбінації генів батьків.

Далі необхідно створити можливість мутації для нових особин. Мутація – це процес, в результаті якого змінюється значення одного або кількох генів на хромосомі. Мутація використовується для додавання різноманітності до популяції і забезпечення можливості виявлення оптимального розв'язку, який не був би доступний за умови використання лише схрещування. Цей процес здійснюється випадковою заміною гена на інший, що не співпадає з уже існуючими у хромосомі.

Імовірність мутації зазвичай є значно меншою, ніж імовірність кросинговеру, оскільки мутації повинні бути достатньо рідкісними, щоб зберегти різноманітність популяції, але й не дозволяти відхилятись від оптимального розв'язку.

Функція-пристосованості – це функція, яка оцінює якість кожної хромосоми у популяції. Функція пристосованості для задачі формування пулу вибіркових дисциплін може бути доволі складною, оскільки повинна відображати багато різних факторів, що впливають на ефективність кожної дисципліни. Один з можливих підходів до формування функції пристосованості полягає в поєднанні різних факторів, таких як:

- вимоги законодавства та стратегії університету, щодо вибіркових дисциплін;
- рівень зацікавленості, у співпраці, здобувача освіти із конкретним викладачем;
- рівень інтересу кандидата до дисципліни, що може впливати на його мотивацію та продуктивність;
- загальноуніверситетський рейтинг дисципліни;
- належність до певного факультету та кафедри.

Функція пристосованості розраховується як сума оцінок кожного з цих факторів. Також слід зазначити, що формування функції пристосованості є ітеративним процесом, тому покращується на основі аналізу результатів попередніх ітерацій та корегування ваг факторів відповідно до отриманих даних.

Розглядаючи обмеження, що обумовлені законодавством важливо відзначити те, що для отримання освітнього ступеня бакалавра необхідно успішно виконати та засвоїти навчальну програму, обсягом 240 кредитів ЄКТС, 25% яких складають вибіркові дисципліни. Отже кількість годин, сформованого пулу вибіркових дисциплін, повинна дорівнювати 1800 або 60 кредитам ЄКТС.

Модуль генерації пулу вибіркових дисциплін є ключовим модулем системи. Цей модуль відповідає за генерацію пулу вибіркових дисциплін для студента на основі його вподобань.

Для генерації пулу вибіркових дисциплін модуль має такий функціонал:

- отримання даних опитування студентів - модуль отримує дані опитування, що містять інформацію про вподобання студентів,
- аналіз даних опитування - на основі даних опитування модуль аналізує вподобання студентів та формує список дисциплін, які можуть відповідати заданим вимогам.
- процес роботи генетичного алгоритму - модуль використовує функції пристосованості, кросинговеру, мутації та селекції для оцінки того, наскільки добре дисципліна відповідає вимогам студента.
- збереження результатів - модуль зберігає результати формування пулу вибіркових дисциплін.

Опитування акцентує увагу саме на визначенні основних вподобань користувача, таких як: опанування нових чи поглиблення вже існуючих компетенцій, навчальний напрям, улюблений викладач, тощо.

Для прикладу розглянемо користувача, що здобуває вищу освіту за спеціальністю для якої випусковою є кафедра філології, гуманітарно-педагогічного факультету, а сам здобувач прагне набути якомога більше компетентностей у різноманітних галузях із нахилом до напряму комп'ютерних наук. Крім того, здобувачу освіти до вподоби навчальні дисципліни вподобаного викладача. Відповідно до зазначених користувачем вимог, формуються відповіді до опитування (рис. 1). Після проходження вище зазначеного опитування користувача буде переадресовано на сторінку з результатами, тобто згенерованим пулом вибіркових дисциплін (рис. 1).

Висновки з даного дослідження і перспективи подальших розвідок у даному напрямку

Розроблений спосіб формування пулу вибіркових навчальних дисциплін на основі генетичного алгоритму дозволяє ефективно оптимізувати процес формування пулу, враховуючи особисті вподобання користувачів. Це сприяє підвищенню задоволеності студентів навчальним процесом та покращенню результатів їхньої освіти.

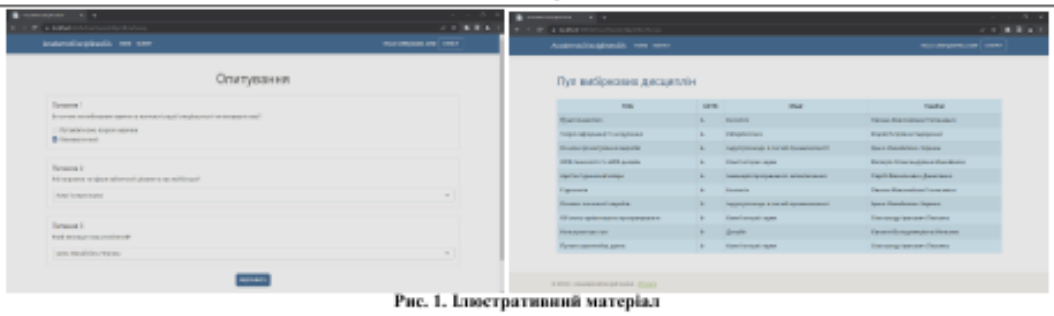


Рис. 1. Ілюстративний матеріал

Розроблена програмна система, яка імплементує запропонований спосіб формування пулу, демонструє ефективність та точність у вирішенні оптимізаційної задачі. Вона може бути використана в навчальних закладах для полегшення процесу вибору вибіркових дисциплін студентами.

Подальші перспективи дослідження можуть полягати в урахуванні низки впливу додаткових чинників, як, до прикладу, рейтинг викладачів, важливість дисциплін тощо. Також можливим є розширення функціональності системи задля забезпечення максимально персоніфікованого підходу до вибору навчальних дисциплін, а також впровадження інформаційної системи на основі запропонованого підходу в практичну діяльність по формуванню індивідуальних навчальних планів здобувачів вищої освіти відповідними навчальними закладами. Пропонується запровадження реалізованої інформаційної системи або принципів та підходів, покладених в її основу, в автоматизовані системи управління закладами вищої освіти.

Література

1. Троцько В.В. Методи штучного інтелекту : навчально-методичний і практичний посібник. Київ : Університет економіки та права «КРОК», 2020. 86 с.

2. Кононюк А.Ю. Нейроні мережі і генетичні алгоритми. К. : «Корнійчук», 2008. 446 с.

3. Богач В. В., Шамрелюк В. В., Шпичко А. В., Мазурець О. В. Метод побудови розкладів занять за генетичним алгоритмом. Збірник наукових праць за матеріалами XIII Всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2021». Хмельницький, 2021. с. 291-297.

4. Малайко А., Пасічник О., Скрипник Т. Метод побудови оптимальної освітньої траєкторії здобувачів вищої освіти. Вісник Хмельницького національного університету, 2022, № 6, Том 1 (315), С. 208–212. DOI: 10.31891/2307-5732-2022-315-6-208-212

5. Manziuk E. A., Barmak O. V., Krak Iu. V., Pasichnyk O. A., Radiuk P. M., Mazurets O. V. Semantic alignment of ontologies meaningful categories with the generalization of descriptive structures. Problems in programming. 2022. Vol. 3, No. 4. P. 355-363. DOI: https://doi.org/10.15407/pp2022.03-04.355

6. Суприган О.І., Ваховська Л.М. Комбінування генетичних алгоритмів в елементах штучної нейронної мережі. Оптико-електронні інформаційно-енергетичні технології, 2019, Том 37, № 1, С. 5–10. DOI: 10.31649/1681-7893-2019-37-1-5-10

7. Kovalyshyn, O. S. Neuro fuzzy genetic algorithm of optimization of rehabilitation procedures. Scientific Papers (Ukrainian Academy of Printing) 2, no. 57 (2018): 72–81. DOI: 10.32403/1998-6912-2018-2-57-72-81.

References

1. Trotsko V.V. Metody shtuchnoho intelektu : navchalno-metodychnyi i praktychnyi posibnyk. Kyiv : Universytet ekonomiky ta prava «KROK», 2020. 86 s.

2. Kononiuk A.Iu. Neironi merezhi i henetychni alhorytmy. K. : «Korniichuk», 2008. 446 s.

3. Bohach V. V., Shamreliuk V. V., Shpychko A. V., Mazurets O. V. Metod pobudovy rozkladiv zaniat za henetychnym alhorytmom. Zbirnyk naukovykh prats za materialamy XIII Vseukrainskoi naukovo-praktychnoi konferentsii «Aktualni problemy kompiuternykh nauk APKN-2021». Khmelnytskyi, 2021. s. 291-297.

4. Malaiko A., Pasichnyk O., Skrypnyk T. Metod pobudovy optymalnoi osvitnoi traiektorii zdobuvachiv vyshchoi osvity. Visnyk Khmelnytskoho natsionalnoho universytetu, 2022, № 6, Tom 1 (315), S. 208–212. DOI: 10.31891/2307-5732-2022-315-6-208-212

5. Manziuk E. A., Barmak O. V., Krak Iu. V., Pasichnyk O. A., Radiuk P. M., Mazurets O. V. Semantic alignment of ontologies meaningful categories with the generalization of descriptive structures. Problems in programming. 2022. Vol. 3, No. 4. P. 355-363. DOI: https://doi.org/10.15407/pp2022.03-04.355

6. Supryhan O.I., Vakhovska L.M. Kombinuvannia henetychnykh alhorytmiv v elementakh shtuchnoi neironnoi merezhi. Optyko-elektronni informatsiino-enerhetychni tekhnolohii, 2019, Tom 37, № 1, S. 5–10. DOI: 10.31649/1681-7893-2019-37-1-5-10

7. Kovalyshyn, O. S. Neuro fuzzy genetic algorithm of optimization of rehabilitation procedures. Scientific Papers (Ukrainian Academy of Printing) 2, no. 57 (2018): 72–81. DOI: 10.32403/1998-6912-2018-2-57-72-81.

Fri Dec 13 07:53:21 EET 2024, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.258 Educational

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 8%**

	Документ		Сумарний збіг по Базі Даних	
ID: 158533 Назва: КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА на тему Метод побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом Додано в БД: 2024-12-13 Автора: Назарій ПЛАХТІЙ Керівники: Олександр ПАСІЧНИК Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	82554	1209	2217 (3%)	38 (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: Назарій ПЛАХТІЙ

Співавтор:

Назва: Метод побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом

Науковий керівник: Олександр ПАСІЧНИК, к.т.н., доцент

Підрозділ: Кафедра комп'ютерних наук

Коефіцієнт подібності 1:0.9%

Коефіцієнт подібності 2:0.4%

Мікропробіли: 0

Заміна букв: 1

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2024-12-13 07:48:38.0

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 13.12.2024 експерт Петрівський Р.С.

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНИХ НАУК

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Метод побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом

Автор: Плахтій Назарій Володимирович

Спеціальність: 122 – Комп'ютерні науки

Освітня програма: освітньо-професійна

Науковий керівник: к.т.н., доцент Пасічник Олександр Анатолійович

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
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.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) за програмою Anti-Plagiarism виявлені 1,0 %, схожість виявлена зі звітом автора з науково-дослідної практики.

2) за програмою StrikePlagiarism КП1 0,85%, КЦ 0,48%,

які містить матеріали огляду предметної області; інші схожості є фрагментарними – містять поширені конструкції, загальновідомі терміни, скорочення та визначення, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи. Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається

Керівник роботи Олександр ПАСІЧНИК

Гарант ОП Руслан БАГРІЙ

Завідувач кафедри КН Олександр БАРМАК

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ
кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу магістра

гр. КНм-23-1 Плахтія Назарія за темою: *Метод побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом*

1. Актуальність теми

В контексті змін сучасного світу, особливо у сфері технологій та знань, освіта стикається з величезними викликами у питанні своєї актуалізації та оптимізації. Питання покращення освітніх процесів стає все актуальнішим, оскільки і державні, і приватні освітні заклади прагнуть до прогресивних змін, які забезпечать ефективне та якісне навчання. Актуальність дослідження полягає в розробці та впровадженні інноваційних підходів до побудови структурно-логічних схем освітніх компонентів з використанням генетичних алгоритмів. Оптимізація у цьому контексті означає не лише покращення існуючих методів навчання, а й створення систем, здатних адаптуватися та відповідати індивідуальним потребам студентів.

2. Відповідність роботи предметній області спеціальності 122 Комп'ютерні науки та загальним вимогам до наукових робіт

Кваліфікаційна робота магістра КНм-23-1 Назарія Плахтія за ступенем обґрунтованості наукових положень, новизни, а також обсягом, структурою та змістом викладеного матеріалу відповідає вимогам щодо наукових робіт. У роботі використані методи, що повністю відповідає предметній області спеціальності 122 Комп'ютерні науки

3. Професійні та особистісні якості магістранта

В період виконання кваліфікаційної роботи магістра Назарій Плахтій виявив себе кваліфікованим фахівцем здатним на високому рівні виконувати поставлені завдання. Володіє необхідними професійними навичками та загальними компетентостями.

4. Ступінь самостійності під час виконання кваліфікаційної роботи

Кваліфікаційна робота виконана студентом особисто. Визначення мети та постановка задач виконувалося спільно з науковим керівником.

5. Наукова новизна та оригінальність запропонованих підходів

Удосконалено метод побудови структурно-логічної схеми освітніх компонентів, який використовує генетичний алгоритмів та дозволяє врахувати послідовність та

повноту опанування освітніх компонент. Результати роботи були опубліковані у Віснику Хмельницького національного університету, № 3, 2024, (335), С. 182 – 185.

6. Ступінь оволодіння методами дослідження

Продемонстровано високий рівень володіння методами дослідження, які були використанні у роботі.

7. Повнота та якість розкриття теми роботи

Тема роботи розкрита якісно на високому рівні, задачі дослідження виконані в повному обсязі.

8. Логічність, послідовність, аргументованість, літературна грамотність викладу матеріалу

Позитивними рисами кваліфікаційної роботи є системність та послідовність викладення матеріалу. Продемонстрована здатність збирати і аналізувати дані, для забезпечення якості прийняття рішень. У кваліфікаційній роботі магістра формалізовані та систематизовані вимоги до розробленої комп'ютерної системи. Робота відповідає всім граматичним нормам та демонструє зрозумілий, виважений стиль подання інформації.

9. Можливість практичного застосування кваліфікаційної роботи, окремих її частин

Результати роботи можуть бути використані в закладах освіти різних форм власності в частині реалізації студентоцентрованого підходу у формуваннях індивідуальних навчальних планів.

10. Висновок про можливість допуску кваліфікаційної роботи до захисту, на яку оцінку заслуговує робота

Кваліфікаційна робота магістра Назарія Плахтія виконана повністю у відповідності із представленими вимогами та є завершеною науковою працею. Вона містить рішення наукової задачі, яка по суті полягає у реалізації методу побудови структурно-логічної схеми освітніх компонентів за генетичним. З огляду на вище сказане, робота рекомендується до захисту та заслуговує на оцінку «відмінно».

Науковий керівник _____ к.т.н., доц.. Олександр ПАСІЧНИК

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ
кафедра комп'ютерних наук

ВІДГУК ОПОНЕНТА

на кваліфікаційну роботу магістра

гр. КНм-23-1 ПЛАХТІЯ Назарія за темою*: Метод побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом.*

1. Актуальність обраної теми

Питання покращення освітніх процесів стає все актуальнішим, оскільки і державні, і приватні освітні заклади прагнуть до прогресивних змін, які забезпечать ефективне та якісне навчання. Актуальність дослідження полягає в розробці та впровадженні інноваційних підходів до побудови структурно-логічних схем освітніх компонентів з використанням генетичних алгоритмів. Оптимізація у цьому контексті означає не лише покращення існуючих методів навчання, а й створення систем, здатних адаптуватися та відповідати індивідуальним потребам студентів. Традиційно, оптимізація в освіті часто використовувалась у контексті економії ресурсів, що нерідко призводило до скорочення чисельності кадрів, зменшення бюджетного фінансування, або навіть до закриття навчальних закладів. Однак, сучасне розуміння та застосування оптимізації в освіті має на меті досягнення оптимальних результатів з використанням існуючих ресурсів, що веде до підвищення якості освіти та забезпечення можливості для студентів розвивати свої потенціали. В площині оптимізаційних процесів часто йдеться про мінімізацію витрат та максимізацію результативності. З цією метою важливо не лише наявність чітко сформульованих цілей, але й глибоке розуміння того, що є найкращим результатом в конкретних умовах та для конкретної групи здобувачів освіти.

2. Відповідність роботи предметній області спеціальності 122 Комп'ютерні науки та загальним вимогам до наукових робіт

Тема кваліфікаційної роботи у повній мірі відповідає предметній області спеціальності 122 Комп'ютерні науки та вимогам до кваліфікаційної роботи магістра згідно Стандарту освіти.

3. Повнота розкриття мети та завдань дослідження

Було проведено порівняння та аналіз можливих методів розв'язання поставленої задачі та обрано сучасний підхід щодо побудови структурно-логічної схеми освітніх компонентів, а саме через використання генетичного алгоритму, як потужного інструменту для отримання бажаного результату. Виходячи з наведених положень, мета є повністю розкритою, а завдання дослідження виконані в повному обсязі.

4. Наявність наукової новизни

В результаті проведення кваліфікаційної роботи були отриманні наступні результати, а саме - Удосконалено метод побудови структурно-логічної схеми освітніх компонентів, який використ-

товує генетичний алгоритмів та дозволяє врахувати послідовність та повноту опанування освітніх компонент. Результати роботи були опубліковані у Віснику Хмельницького національного університету, № 3, 2024, (335), С. 182 – 185.

5. Зміст кожного розділу роботи

В першому розділі наведено характеристику предметної області з оглядом підходів до метод побудови структурно-логічної схеми освітніх компонентів. Визначено мету та задачі дослідження. В другому розділі реалізовано метод побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом. В третьому розділі виконана програмна реалізація метод побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом та виконано тестування програмної реалізації. В четвертому розділі виконано дослідження реалізованого метод побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом.

6. Ступінь розкриття теми роботи

Кваліфікаційна робота магістра присвячена метод побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом. В рамках роботи було проаналізовано предметну область, програмні системи для вирішення аналогічних завдань, створено відповідний метод та його програмна реалізація із проведенням необхідного тестування та досліджень. Тема роботи розкрита якісно на високому рівні, задачі дослідження виконані в повному обсязі.

7. Якість оформлення кваліфікаційної роботи

Кваліфікаційна робота магістра відповідає всім вимогам до оформлення таких робіт. Стиль подання інформації є фаховим та зрозумілим. Робота не містить стилістичних відхилень та відповідає всім нормам граматики. Робота виконана логічно, послідовно та аргументовано. Матеріал викладено якісно із дотриманням вимог до професійного літературного стилю.

8. Недоліки кваліфікаційної роботи

В роботі не зазначено, які, за рівнем підготовки, заклади освіти можуть використовувати запропонований метод.

9. Загальний висновок (допускається чи не допускається до захисту), якої оцінки заслуговує кваліфікаційна робота.

Беручи до уваги новизну, актуальність, важливість отриманих результатів, їх достовірність та обґрунтованість, вважаю, що кваліфікаційна робота магістра Назарія Плахтія «Метод побудови структурно-логічної схеми освітніх компонентів за генетичним алгоритмом» є оригінальним та завершеним науковим дослідженням. Кваліфікаційна робота магістра Назарія Плахтія рекомендується до захисту, рекомендована оцінка «відмінно».

Опонент д.т.н. професор кафедри КІС Лисенко С.М.