БАКАЛАВРСЬКА РОБОТА

БР. ІП - 53.00.00.000 ПЗ

Група ІП-21-3

Шпирка Тарас

2025

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Шпирка Тарас Романович

(прізвище, ім'я, по батькові)

УДК 004

(індекс)

БАКАЛАВРСЬКА РОБОТА

Веб-застосунок для системи рекомендацій навчальних курсів

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач освітнього рівня Шпирка Т.Р.

(підпис, ініціали та прізвище здобувача)

Науковий керівник Яцишин Микола Миколайович, к.т.н., доцент

(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2025

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ ____ ” _____2025 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Шпирці Тарасу Романовичу

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) “ Веб-застосунок для системи рекомендацій навчальних курсів”

керівник проекту (роботи) Яцишин М.М., доцент

затверджені наказом закладу вищої освіти від “ 28 ” квітня 2025 р. № 264/7

2. Строк подання студентом проекту (роботи) 10 червня 2025 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

4. Зміст розрахунково - пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної області побудови рекомендаційних систем для навчання
2. Огляд використаного програмного забезпечення та архітектури бази даних
3. Моделювання архітектури системи за допомогою DFD та UML діаграм
4. Розробка алгоритму генерації рекомендацій курсів і представлення інтерфейсу системи
5. Тестування веб-застосунку для системи рекомендацій навчальних курсів

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Діаграма потоку даних для функціональності входу та реєстрації (рис. 3.1)
2. Діаграма потоку даних для операцій, що виконуються студентом (рис. 3.2)
3. Діаграма потоку даних для операцій, що виконуються консультантом (рис. 3.3)
4. Діаграма потоку даних для операцій, що виконуються адміністратором (рис. 3.4)
5. Діаграма випадків використання для студента (рис. 3.5)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання **28 квітня 2025 р.**

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Аналіз предметної області побудови рекомендаційних систем для навчання	04.05.2025	виконано
2	Огляд використаного програмного забезпечення та архітектури бази даних	15.05.2025	виконано
3	Моделювання архітектури системи за допомогою DFD та UML діаграм	21.05.2025	виконано
4	Розробка алгоритму генерації рекомендацій курсів і представлення інтерфейсу системи	28.05.2025	виконано
5	Тестування веб-застосунку для системи рекомендацій навчальних курсів	03.06.2025	виконано
6	Оформлення пояснювальної записки дипломної роботи завідувачем кафедри	10.06.2025	виконано

Студент – дипломник _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Бакалаврська робота містить 79 сторінок, 44 рисунки, список використаних джерел із 34 найменуваннями.

Метою роботи є розробка веб-застосунку, який реалізує систему рекомендацій навчальних курсів із використанням алгоритмів персоналізації, що забезпечує індивідуалізовану побудову освітнього маршруту.

Об'єкт дослідження - процеси підтримки користувачів у виборі навчального контенту в онлайн-середовищі.

Предмет дослідження - методи, алгоритми та програмні засоби реалізації рекомендаційної системи у вигляді веб-застосунку для індивідуального підбору навчальних курсів.

В першому розділі проведено аналіз предметної області, визначено мету проєкту, вимоги до системи та обґрунтовано архітектурний підхід.

В другому розділі обґрунтовано вибір технологічного стеку та реалізовано базу даних, що підтримує масштабовану логіку зберігання даних.

В третьому розділі виконано моделювання архітектури системи за допомогою DFD та UML, що забезпечує формалізацію бізнес-логіки.

В четвертому розділі реалізовано алгоритм генерації рекомендацій та візуалізовано шляхи користувацької взаємодії з системою.

В п'ятому розділі проведено тестування веб-застосунку, проаналізовано результати й підтверджено ефективність роботи системи на практиці.

Висновок: запропоновано та реалізовано комплексний підхід до побудови системи рекомендацій навчальних курсів, що поєднує контентно-орієнтовану фільтрацію з модульною архітектурою веб-застосунку

КЛЮЧОВІ СЛОВА: РЕКОМЕНДАЦІЙНА СИСТЕМА, НАВЧАЛЬНІ КУРСИ, ПЕРСОНАЛІЗАЦІЯ, ВЕБ-ЗАСТОСУНОК, КОНТЕНТНА ФІЛЬТРАЦІЯ, БАЗА ДАНИХ, АДАПТИВНЕ НАВЧАННЯ.

ANNOTATION

The bachelor's thesis contains 79 pages, 44 figures, a list of used sources with 34 names.

The purpose of the work is to develop a web application that implements a system of recommendations for educational courses using personalization algorithms, which provides an individualized construction of an educational route.

The object of the study is the processes of supporting users in choosing educational content in an online environment.

The subject of the study is methods, algorithms and software tools for implementing a recommendation system in the form of a web application for individual selection of educational courses.

In the first section, an analysis of the subject area is conducted, the goal of the project is determined, the requirements for the system are determined, and the architectural approach is justified.

In the second section, the choice of a technological stack is justified and a database that supports scalable data storage logic is implemented.

In the third section, the system architecture is modeled using DFD and UML, which provides formalization of business logic.

In the fourth section, the algorithm for generating recommendations is implemented and the ways of user interaction with the system are visualized.

In the fifth section, the web application is tested, the results are analyzed and the effectiveness of the system in practice is confirmed.

Conclusion: a comprehensive approach to building a system of recommendations for training courses is proposed and implemented, combining content-oriented filtering with a modular architecture of the web application

KEYWORDS: RECOMMENDATION SYSTEM, TRAINING COURSES, PERSONALIZATION, WEB APPLICATION, CONTENT FILTERING, DATABASE, ADAPTIVE LEARNING.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

E – Elective Course – Вибірковий курс

C – Core Course – Основний курс

P – Prerequisite Course – Курс попередніх вимог

DB – Database – База даних

UML – Unified Modeling Language – Уніфікована мова моделювання

DFD – Data Flow Diagram – Діаграма потоку даних

App – Application – Додаток

ID – Identifier – Ідентифікатор (використовується в "student ID", "courseID", "C_ID")

ВСТУП

Сучасна цифрова епоха спричинила кардинальні зміни в усіх сферах суспільного життя, зокрема в галузі освіти. Широке впровадження онлайн-курсів, інтерактивних навчальних платформ та масових відкритих онлайн-курсів (MOOC) створило нові можливості для здобуття знань, водночас поставивши перед користувачами низку нових викликів. Одним із таких викликів є проблема ефективного орієнтування у великому обсязі освітнього контенту, що постійно зростає. За таких умов виникає потреба в інструментах, які б допомагали користувачам не просто знаходити інформацію, а отримувати індивідуалізовані, цілеспрямовані рекомендації, що відповідають їхнім потребам, рівню знань, інтересам та навчальним цілям.

Рекомендаційні системи стали ефективним інструментом підтримки прийняття рішень у багатьох сферах, зокрема в електронній комерції, стримінгових сервісах, соціальних мережах та освіті. У контексті навчання такі системи дозволяють формувати персоналізовані освітні маршрути, адаптовані до кожного користувача. Це особливо актуально для початківців, які не мають чіткого уявлення про послідовність вивчення матеріалу, або для професіоналів, які прагнуть підвищити кваліфікацію в конкретній галузі.

Водночас розробка ефективної системи рекомендацій у сфері освіти є складним завданням, що вимагає врахування не лише технічних аспектів (механізмів фільтрації, структури даних, алгоритмів тощо), а й педагогічних, зокрема принципів побудови навчального контенту, когнітивних характеристик користувачів та вимог до освітніх стандартів. У цьому контексті важливим є поєднання сучасних підходів до розробки програмного забезпечення з методами персоналізації, а також побудова зручного інтерфейсу для взаємодії користувача з системою.

Розроблений веб-застосунок має потенціал до масштабування, подальшого вдосконалення та інтеграції з наявними освітніми платформами. Його впровадження може суттєво підвищити якість навчального процесу, зменшити когнітивне навантаження на користувача при виборі курсів і сприяти побудові індивідуалізованих освітніх траєкторій.

Актуальність теми

В умовах стрімкого розвитку онлайн-освіти виникає проблема надмірної інформаційної насиченості освітнього простору, що ускладнює процес вибору ефективної навчальної програми для конкретного користувача. Сучасні освітні платформи часто обмежуються статичними або малоперсоналізованими рекомендаціями, що не враховують контекстуальні та поведінкові особливості користувача. Саме тому побудова системи інтелектуальної підтримки вибору навчальних курсів з використанням гнучких алгоритмів фільтрації та адаптації є актуальним напрямом як з наукової, так і з практичної точки зору.

Метою дипломної роботи є розробка веб-застосунку, який реалізує систему рекомендацій навчальних курсів із використанням алгоритмів персоналізації, що забезпечує індивідуалізовану побудову освітнього маршруту.

Завдання дослідження

1. Провести аналіз предметної області та обґрунтувати доцільність використання рекомендаційних систем у сфері освіти.

2. Розробити архітектуру та функціональну модель системи.

3. Реалізувати веб-застосунок з використанням сучасного технологічного стеку.

4. Спроєктувати та реалізувати базу даних для зберігання інформації про користувачів, курси та рекомендації.

5. Реалізувати алгоритм персоналізованих рекомендацій курсів.

6. Провести тестування системи та оцінити її ефективність.

Об’єкт дослідження - процеси підтримки користувачів у виборі навчального контенту в онлайн-середовищі.

Предмет дослідження - методи, алгоритми та програмні засоби реалізації рекомендаційної системи у вигляді веб-застосунку для індивідуального підбору навчальних курсів.

Методи дослідження

- Аналіз предметної області та порівняння існуючих рішень.

- Методології об’єктно-орієнтованого моделювання (DFD, UML).

- Методи фільтрації та ранжування даних (контентно-орієнтована фільтрація).

- Засоби розробки веб-застосунків (Node.js, Express, PostgreSQL).

- Методи функціонального та наскрізного тестування (Nightwatch.js).

Наукова новизна

Запропоновано та реалізовано комплексний підхід до побудови системи рекомендацій навчальних курсів, що поєднує контентно-орієнтовану фільтрацію з модульною архітектурою веб-застосунку. В роботі представлено деталізовану модель взаємодії користувача із системою на основі різних сценаріїв, що дозволяє підвищити рівень персоналізації освітнього процесу.

Практичне значення

Розроблений веб-застосунок може бути інтегрований у реальні освітні платформи з метою підвищення якості навчального обслуговування, зменшення часу на пошук релевантного контенту та створення індивідуальних траєкторій навчання. Рішення є гнучким і масштабованим, що дозволяє адаптувати його до різних навчальних доменів.

Бакалаврська робота містить 79 сторінок, 44 рисунків, 5 розділів, список використаних джерел із 34 найменуваннями.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПОБУДОВИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ ДЛЯ НАВЧАННЯ

1.1. Мета та особливості розробки веб-застосунку для системи рекомендацій навчальних курсів

Основною метою розробки веб-застосунку для системи рекомендацій навчальних курсів є формування індивідуалізованих траєкторій навчання для студентів, що сприятимуть успішному та своєчасному завершенню освітньої програми. Зазначені траєкторії являють собою послідовні списки курсів, рекомендованих для проходження студентом у кожному академічному семестрі, починаючи з першого семестру після зарахування і до моменту випуску. Відбір курсів здійснюється на основі фільтрації відповідно до інтересів студента, отриманих шляхом аналізу даних, наданих у попередньо запропонованій анкеті.

Ключовим елементом бізнес-логіки системи є розробка та імплементація складного алгоритму рекомендацій. Цей алгоритм враховує різноманітні параметри, такі як академічна успішність студента, його попередній досвід навчання, обрана спеціалізація, а також вищезгадані інтереси. Він спрямований на прогнозування найбільш релевантних та корисних курсів, які допоможуть студенту досягти бажаних результатів.

Ефективність та стабільність функціонування розробленого веб-застосунку забезпечується шляхом проведення ретельного наскрізного функціонального тестування (end-to-end testing). Для цього використовується потужний фреймворк Nightwatch.js, який функціонує на платформі Node.js. Цей інструмент дозволяє симулювати взаємодію користувача з додатком, перевіряючи коректність роботи всіх його компонентів. Тестові випадки розробляються для кожного окремого модуля системи, починаючи від інтерфейсу користувача і закінчуючи взаємодією з базою даних. Ці тестові

сценарії інтегруються безпосередньо в процес розробки додатка, що дозволяє виявляти та усувати потенційні помилки на ранніх стадіях життєвого циклу проекту. Такий підхід гарантує високу якість кінцевого продукту та його надійність у реальних умовах експлуатації.

1.2. Обґрунтування розробки системи рекомендацій навчальних курсів

1.2.1. Мотивація збору даних та ідентифікація проблем

Мотивацією для збору даних про зарахування на курси є подвійне прагнення: з одного боку, ідентифікація попиту на навчальні курси серед студентів, а з іншого – систематизація академічних записів студентів. Інформація про зарахування є критично важливою для адміністрації програми при плануванні розкладу курсів.

Традиційний процес консультування студентів перед початком кожного академічного семестру є значною мірою часозатратним. Цей процес зазвичай починається з реєстрації студента на консультацію за принципом "першим прийшов – першим обслужений".

Недотримання вимоги щодо консультації може призвести до накладення адміністративної заборони на реєстрацію, що перешкоджає студенту записатися на бажані курси.

Існуючий підхід до вибору курсів нерідко призводить до низки проблем, зокрема:

1. Зарахування на курси без дотримання попередніх вимог - студенти можуть обирати курси, для яких вони не мають необхідних академічних передумов.

2. Вибір недоступних курсів - обрання курсів, які не пропонуються у поточному семестрі.

3. Невідповідний вибір вибіркових курсів - студенти можуть обирати курси, які не відповідають їхнім академічним інтересам або майбутнім кар'єрним планам.

4. Проблеми з академічною успішністю - студенти можуть обирати курси, що пропонуються в інших семестрах, не дотримуватися мінімальної кількості курсових одиниць, або ж мати низький середній бал через вибір курсів, які не викликають у них інтересу, що знижує їхню мотивацію до навчання.

1.2.2. Мета проекту

Основною метою даного проекту є розробка та впровадження алгоритму, який забезпечить формування індивідуалізованих траєкторій рекомендованих курсів. Цей алгоритм має враховувати низку критичних параметрів, включаючи:

- Дані про пропозиції курсів: актуальний перелік доступних курсів.

- Обмеження студента:індивідуальні графіки роботи, особисті інтереси щодо вибіркових курсів та бажаний розклад занять.

Результатом роботи алгоритму буде представлення оптимального шляху навчання, який студент повинен пройти кожен семестр для своєчасного завершення освітньої програми. Своєчасний випуск є ключовим фактором для успіху студента та сприяє досягненню цілей Ініціативи випуску університету, що спрямована на підвищення ефективності та якості освітнього процесу.

Крім того, аналіз агрегованих даних рекомендованих шляхів навчання для всіх поточних студентів інженерії програмного забезпечення Івано-Франківського національного технічного університету нафти і газу (ІФНТУНГ) може надати безцінну інформацію для академічного планування. Ці дані дозволять:

1. Прогнозувати попит на курси і визначити кількість студентів, які потенційно потребуватимуть певний курс.

2. Оптимізувати складання розкладу курсів, забезпечити адекватну пропозицію курсів відповідно до прогнозованого попиту.

3. Планувати необхідні ресурси, можливість оцінити потребу у викладацькому складі та матеріально-технічних ресурсах для підтримки освітнього процесу.

Такий комплексний підхід дозволить не лише покращити індивідуальний освітній досвід студентів, а й оптимізувати адміністративні та ресурсні аспекти управління навчальними програмами.

1.3. Архітектура та функціональні можливості запропонованої системи рекомендацій навчальних курсів

Пропонований веб-додаток являє собою інтегровану систему консультування з курсів, призначену для формування оптимізованих траєкторій навчання студентів. Основною метою системи є підтримка студентів у плануванні їхньої академічної траєкторії для забезпечення успішного та своєчасного завершення освітньої програми. Рекомендований шлях навчання відображає послідовний перелік курсів, які студент може проходити в кожному академічному семестрі, починаючи з першого семестру після вступу і до запланованого семестру випуску.

1.3.1. Механізми персоналізації та фільтрації

Відбір курсів здійснюється на основі багатофакторного аналізу відповідей, отриманих від студента через спеціально розроблену анкету. Ключовими критеріями фільтрації є:

- Бажаний варіант випускної роботи: Вибір між проектною роботою, дисертацією або комплексним іспитом.

- Особисті переваги щодо розкладу: Бажані дні та час тижня для проходження курсів.

- Академічні інтереси: Визначення сфер інтересів студента для формування релевантних рекомендацій щодо вибіркових курсів.

1.3.2. Бізнес-логіка та забезпечення якості

Центральним елементом бізнес-логіки системи є розробка та реалізація комплексного алгоритму рекомендацій. Цей алгоритм відповідає за генерацію індивідуалізованих навчальних планів.

Функціональність веб-додатку підлягає ретельному наскрізному тестуванню (end-to-end testing) за допомогою фреймворку Nightwatch.js, що функціонує на базі Node.js. Nightwatch.js пропонує простий, але потужний синтаксис, що дозволяє швидко писати тестові випадки, використовуючи CSS або XPath селектори для взаємодії з елементами DOM. Для кожного модуля системи розробляються та інтегруються тестові випадки на етапі побудови додатка. Це дозволяє забезпечити його стабільну роботу та запобігти виникненню збоїв.

1.3.3. Система аутентифікації користувачів

Аутентифікація користувачів реалізована за допомогою форми реєстрації. Кожен студент може зареєструватися, використовуючи свій унікальний ID, наданий університетом, як ідентифікатор користувача, та індивідуально обраний пароль. Після успішної реєстрації студент отримує доступ до системи за допомогою своїх облікових даних.

Індивідуалізація рекомендованих шляхів навчання залежить від декількох факторів, включаючи статус студента (денна або заочна форма навчання) та його особисті переваги, визначені на основі даних, отриманих з анкети. Це гарантує, що кожен студент отримує унікальний та оптимальний для нього шлях навчання.

1.4. Специфікація вимог до системи рекомендацій навчальних курсів

Розробка системи рекомендацій навчальних курсів вимагає дотримання певних апаратних та програмних вимог, а також застосування відповідних архітектурних рішень.

1.4.1. Апаратні вимоги

Для ефективного функціонування та тестування системи необхідні наступні апаратні ресурси:

- Клієнтська сторона: ноутбук або персональний комп'ютер, оснащений веб-браузером Google Chrome для проведення тестування.

- Середовище розробки: персональний комп'ютер з достатньою обчислювальною потужністю для виконання завдань розробки.

1.4.2. Програмні вимоги

Програмне забезпечення, що використовується для реалізації системи, включає:

- Тестування: Nightwatch.js для автоматизованого наскрізного тестування.

- Серверна частина (Backend): Express.js, що є фреймворком для розробки веб-додатків на Node.js.

- Клієнтська частина (Frontend): AngularJS для створення інтерактивного користувацького інтерфейсу та HTML5 для структурування веб-сторінок.

- Операційні системи: Mac OS X та Windows 10, що забезпечують сумісність та можливість розгортання в різних середовищах.

- Система управління базами даних (СУБД): PostgreSQL для зберігання та управління даними.

- Інтегроване середовище розробки (IDE): WebStorm, що надає повний набір інструментів для розробки.

1.4.3. Архітектурне рішення: MVC та REST

Для архітектурної організації системи застосовано шаблон Model-View-Controller (MVC). MVC є парадигмою програмної архітектури, що розділяє додаток на три взаємопов'язані компоненти для ізоляції внутрішніх представлень інформації від способів її представлення користувачеві.

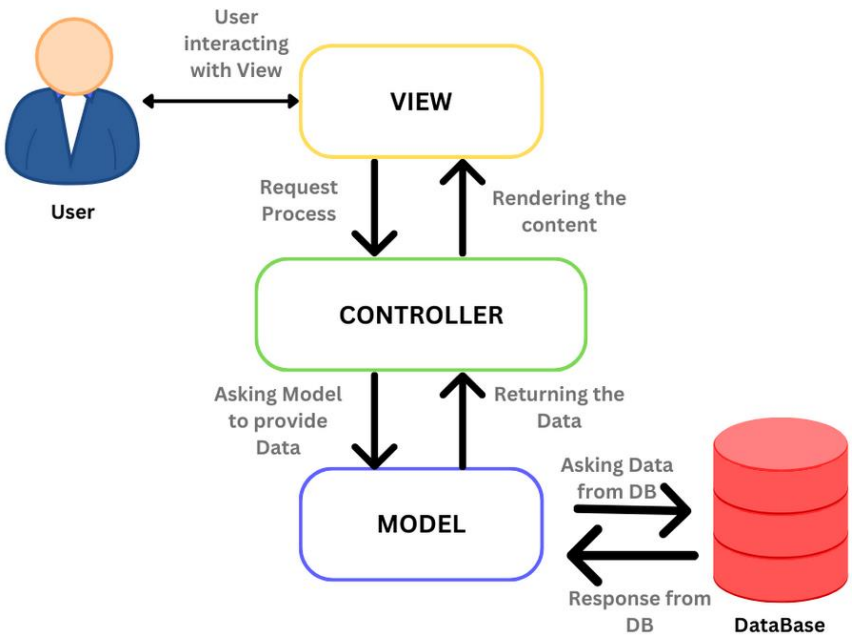

Рисунок 1.1 – Архітектура шаблону Model-View-Controller

У контексті даного проекту:

- Модель (Model): реалізована за допомогою Express.js, відповідає за бізнес-логіку та взаємодію з базою даних.

- Контролер (Controller): представлений RESTful-контролером (Representational State Transfer), що забезпечує логіку обробки запитів та взаємодію між моделлю та представленням. REST є архітектурним стилем для розподілених систем, що визначає обмеження, такі як уніфікований інтерфейс, які сприяють підвищенню продуктивності, масштабованості та

модифікованості веб-сервісів, що є критично важливим для їх ефективного функціонування в мережевому середовищі.

- Представлення (View): створене за допомогою фабрики Angular (Angular factory), відповідає за відображення даних користувачеві.

Така архітектура сприяє модульності, спрощує розробку та підтримку системи, а також забезпечує високий рівень масштабованості та адаптивності до майбутніх змін.

РОЗДІЛ 2. ОГЛЯД ВИКОРИСТАНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА АРХІТЕКТУРИ БАЗИ ДАНИХ

Для реалізації системи рекомендацій навчальних курсів було обрано комплексний набір програмних засобів та архітектурних рішень, що забезпечують її функціональність, масштабованість та надійність.

2.1. Фреймворк Node.js. Залежності проекту

Node.js є відкритою серверною платформою, що дозволяє виконувати JavaScript-код на стороні сервера. Розроблена у 2009 році, Node.js була задумана як програмне забезпечення, орієнтоване на динамічне оновлення стану замість традиційних запитів до сервера. У даному проекті Node.js застосовується для розробки backend-частини веб-додатку, забезпечуючи встановлення та підтримку з'єднання з сервером.

Ключові особливості Node.js включають:

- Розробка серверів: Надає потужні інструменти для створення високопродуктивних та масштабованих серверних рішень.

- Колекція вбудованих модулів: Містить основні функціональні модулі для криптографії, вводу/виводу файлової системи, мережевих технологій та потоків даних.

- Екосистема NPM (Node Package Manager): Найбільша у світі колекція публічних та приватних бібліотек з відкритим вихідним кодом, що дозволяє повторно використовувати код та ефективно управляти залежностями проекту.

У рамках цього проекту пакетний менеджер npm використовується для встановлення всіх необхідних залежностей, перелік яких наведено нижче на рисунку 2.1.

Залежність	Опис
body-parser	Парсер тіла запитів для Node.js.
consolidate	Бібліотека для уніфікації шаблонізаторів.
cookie-parser	Парсер HTTP-куків.
cryptr	Легка бібліотека для шифрування/дешифрування рядків.
express	Фреймворк для розробки веб-додатків на Node.js.
http	Вбудований модуль Node.js для роботи з HTTP.
jade	Шаблонізатор
morgan	HTTP-логер запитів.
nightwatch	Фреймворк для наскрізного тестування.
nodemailer	Модуль для відправлення електронних листів.
pg	Драйвер PostgreSQL для Node.js.
stylus	Динамічний препроцесор CSS.
webdriver-manager	Інструмент для управління та оновлення бінарних файлів Selenium WebDriver.

Рисунок 2.1 - Залежності проекту, встановлені за допомогою Npm

2.2. Фреймворк для наскрізного тестування Nightwatch.js

Nightwatch.js – це фреймворк для наскрізного тестування (end-to-end testing) веб-додатків, розроблений на базі Node.js. Він використовує Selenium WebDriver для автоматизації взаємодії з веб-браузерами та виконання тестових випадків. Однією з ключових переваг Nightwatch.js є підтримка повторного використання коду через функціональність експорту та імпорту модулів. Фреймворк використовує стандартизований WebDriver API, що забезпечує надійну та послідовну взаємодію з різними браузерами (Chrome, Firefox, Safari, Edge тощо). Це дозволяє проводити крос-браузерне тестування.

У поточному проекті Nightwatch.js застосовується для розробки тестових випадків для кожного функціонального розділу системи. Приклад тестового випадку для сторінки входу, що демонструє перевірку перенаправлення та успішної аутентифікації, наведено на рисунку 2.2.

```
module.exports = {
    'Does redirect to login page': function (client) {
        var login = client.page.loginPage();
        login
            .navigate()
            .waitForElementVisible('@Header', 1000)
            .assert.visible('@CID')
    },
    'Does login successful and redirects to student home page': function (client) ·
        var login = client.page.loginPage();
        var loginProps = login.props;
        login
            .click('@CID')
            .setValue('@CID', loginProps.StudentID)
            .click('@Password')
            .setValue('@Password', loginProps.Studentpassword)
            .waitForElementVisible('@LoginButton', 1000)
            .assert.visible('@LoginButton')
            .click('@LoginButton');
        client
            .pause(1000)
            .assert.urlEquals(loginProps.StudentURL)
    }
}
```

Рисунок 2.2 - Тестові випадки для сторінки входу, реалізовані за допомогою Nightwatch.js.

2.3 Розробка та опис таблиць бази даних

PostgreSQL обрано як систему управління базами даних (СУБД) для даного веб-додатку. Ця крос-платформна, відкрита, об'єктно-реляційна СУБД надає низку значних переваг:

- Висока продуктивність: Забезпечує ефективне оброблення великих обсягів даних та запитів.

- Підтримка численних типів даних: Гнучкість у роботі з різноманітними форматами даних.

- Сильна підтримка спільноти та третіх сторін: Наявність широкої спільноти та розвиненої екосистеми інструментів та розширень.

У базі даних було створено п'ятнадцять таблиць для зберігання та управління інформацією системи. Детальний опис кожної таблиці,

включаючи атрибути, типи даних та примітки (з використанням ЛЕГЕНДИ: PK - Первинний ключ, FK - Зовнішній ключ, NN - NOT NULL, CK - Складений первинний ключ), наведено нижче.

1. Таблиця 'АДМІНІСТРАТОР'

Спеціалізація (підклас) суперкласу 'КОРИСТУВАЧ'

Атрибут	Опис	Тип даних	Примітки
c_id	10-значний унікальний ID, призначений університетом.	numeric	PK, FK, що посилається на PK таблиці USER.

Рисунок 2.3 - Структура таблиці бази даних 'АДМІНІСТРАТОР'

2. Таблиця 'КОНСУЛЬТАНТ'

Спеціалізація (підклас) суперкласу 'КОРИСТУВАЧ'

Атрибут	Опис	Тип даних	Примітки
c_id	10-значний унікальний ID, призначений університетом.	numeric	PK, FK, що посилається на PK таблиці USER.

Рисунок 2.4 - Структура таблиці бази даних 'КОНСУЛЬТАНТ'

3. Таблиця 'БАЗОВИЙ_КУРС'

Спеціалізація (підклас) суперкласу 'КУРС'

Атрибут	Опис	Тип даних	Примітки
course_id	3-значний унікальний ID	numeric	PK

Рисунок 2.5 - Структура таблиці бази даних 'БАЗОВИЙ_КУРС'

4. Таблиця 'КУРС'

Узагальнення (суперклас), що описує деталі курсу, пропонованого в програмі.

Атрибут	Опис	Тип даних	Примітки
course_id	3-значний унікальний ID	numeric	PK
has_lab	1 - курс має лабораторний компонент; 0 - курс не має лабораторного компонента.	boolean	NN
course_level	Аспірантура або бакалаврат.	змінної довжини символ	NN
units	Загальна кількість одиниць, призначених курсу.	integer	NN
course_dept	Абревіатура відділу, який пропонує курс.	змінної довжини символ	NN
name	Назва курсу.	змінної довжини символ	NN

Рисунок 2.6 - Структура таблиці бази даних 'КУРС'

5. Таблиця 'ПОПЕРЕДНІ_ВИМОГИ_КУРСУ'

Спеціалізація (підклас) суперкласу 'КУРС', що містить інформацію про курси, які є попередніми вимогами до інших курсів.

Атрибут	Опис	Тип даних	Примітки
course_id	3-значний унікальний ID,	numeric	CK, FK, що посилається на PK таблиці КУРС.
prerequisite_id	3-значний унікальний ID, призначений попередній вимозі.	numeric	CK, FK, що посилається на PK таблиці КУРС.

Рисунок 2.7 - Структура таблиці бази даних 'ПОПЕРЕДНІ_ВИМОГИ_КУРСУ'

6. Таблиця 'РОЗКЛАД_КУРСУ'

Надає інформацію про те, коли (семестр/рік, день тижня, час) пропонується курс і хто його викладає.

Атрибут	Опис	Тип даних	Примітки
course_id	3-значний унікальний ID	numeric	CK, FK, що посилається на PK таблиці КУРС.
year	4-значний рік, коли пропонується курс.	numeric	CK
quarter	Квартал (осінь, зима, весна, літо), коли пропонується курс.	змінної довжини символ	CK
session_no	Розрізняє між різними пропозиціями одного і того ж курсу в одному кварталі; наприклад, якщо курс пропонується двічі в кварталі, то розклад S1 матиме session_no = 1, а розклад S2 матиме session_no = 2.	numeric	NN
instructor	Ім'я викладача, який викладає курс.	змінної довжини символ	NN
course_day	День тижня (понеділок та середа, вівторок та четвер, п'ятниця), коли пропонується лекційна частина курсу.	змінної довжини символ	
course_start_time	Час початку (у 24-годинному форматі), коли пропонується лекційна частина курсу (приклад: 19:50).	час	
course_end_time	Час закінчення (у 24-годинному форматі), коли пропонується лекційна частина курсу (приклад: 17:20).	час	
lab_day	День тижня (понеділок та середа, вівторок та четвер, п'ятниця), коли пропонується лабораторна частина курсу.	змінної довжини символ	
lab_start_time	Час початку (у 24-годинному форматі), коли пропонується лабораторна частина курсу (приклад: 9:50).	час	
lab_end_time	Час закінчення (у 24-годинному форматі), коли пропонується лабораторна частина курсу (приклад: 10:50).	час	

Рисунок 2.8 - Структура таблиці бази даних 'РОЗКЛАД_КУРСУ'

7. Таблиця 'ПРОЙДЕНІ_КУРСИ'

Вказує, які курси та коли були пройдені студентом, а також отримані ним оцінки.

Атрибут	Опис	Тип даних	Примітки
c_id	10-значний унікальний ID	numeric	CK, FK, що посилається на PK таблиці СТУДЕНТ.
course_id	3-значний унікальний ID	numeric	CK, FK, що посилається на PK таблиці КУРС.
quarter_year	Рік і квартал (наприклад, Осінь2017), коли курс був пройдений.	змінної довжини символ	CK
day_time	День і час, коли курс був пройдений. Час у 24-годинному форматі (наприклад, {"Day": "MondayAndWednesday", "time": "9:50:00-07"}).	json	NN
instructor	Ім'я викладача, який викладав курс у тому кварталі.	змінної довжини символ	NN
grade	Літерна оцінка (A, A-, B+, B, B-, C+, C, C-, D, F, I, RP), отримана студентом.	змінної довжини символ	

Рисунок 2.9 - Структура таблиці бази даних 'ПРОЙДЕНІ_КУРСИ'

8. Таблиця 'ПЕРЕВАГИ_ДНЯ'

Вказує, які дні тижня студент віддає перевагу для проходження курсу.

Атрибут	Опис	Тип даних	Примітки
c_id	10-значний унікальний ID	numeric	CK, FK, що посилається на PK таблиці ПЕРЕВАГИ_СТУДЕНТА.
day_preference	День тижня (понеділок та середа, вівторок та четвер, п'ятниця), який студент віддає перевагу для проходження курсу, як зазначено в анкеті.	змінної довжини символ	CK

Рисунок 2.10 - Структура таблиці бази даних 'ПЕРЕВАГИ_ДНЯ'

9. Таблиця 'ВИБІРКОВИЙ_КУРС'

Атрибут	Опис	Тип даних	Примітки
course_id	3-значний унікальний ID	numeric	PK, FK, що посилається на PK таблиці КУРС.

Рисунок 2.11 - Структура таблиці бази даних 'ВИБІРКОВИЙ_КУРС'

10. Таблиця 'СЕМЕСТР_ПРОПОНУВАННЯ'

Вказує, в яких семестрах пропонується курс.

Атрибут	Опис	Тип даних	Примітки
course_id	3-значний унікальний ID	numeric	СК, FK, що посилається на РК таблиці КУРС.
quarter	Квартал, коли пропонується курс.	змінної довжини символ	СК

Рисунок 2.12 - Структура таблиці бази даних 'СЕМЕСТР_ ПРОПОНУВАННЯ'

11. Таблиця 'СТУДЕНТ'

Вказує, чи є студент міжнародним/внутрішнім, повний/неповний робочий день та перелічує рекомендовані курси для проходження.

Атрибут	Опис	Тип даних	Примітки
c_id	10-значний унікальний ID	numeric	PK, FK, що посилається на PK таблиці КОРИСТУВАЧ.
residency	1 - міжнародний студент; 0 - внутрішній студент.	boolean	NN
ft_or_pt	1 - неповний робочий день; 0 - повний робочий день.	boolean	NN
current_recommendation_path	Шлях рекомендацій, згенерований системою.	json	

Рисунок 2.13 - Структура таблиці бази даних 'СТУДЕНТ'

12. Таблиця 'ПЕРЕВАГИ_СТУДЕНТА'

Вказує всі переваги, які студент вибрав в анкеті.

Атрибут	Опис	Тип даних	Примітки
c_id	10-значний унікальний ID	numeric	PK, FK, що посилається на PK таблиці СТУДЕНТ.
degree_preference	Перевага ступеня (Проект, Дисертація, Комплексний іспит), з яким студент хоче закінчити.	змінної довжини символ	NN
course_count_preference	Загальна кількість курсів, які студент хоче пройти в кварталі.	numeric	NN
lecture_preference	1 - студент хоче пройти курси з лабораторією; 0 - студент хоче пройти курси без лабораторії.	boolean	NN
summer_course_preference	1 - студент хоче пройти літні курси; 0 - студент не хоче пройти літні курси.	boolean	NN
course_overload_preference	1 - студент хоче пройти більше ніж 16 одиниць; 0 - студент хоче пройти максимум 16 одиниць.	boolean	NN
independent_study_preference	1 - студент хоче пройти незалежне навчання; 0 - студент не хоче пройти незалежне навчання.	boolean	NN

Рисунок 2.14 - Структура таблиці бази даних 'ПЕРЕВАГИ_СТУДЕНТА'

13. Таблиця 'ПОПЕРЕДНІ_ВИМОГИ_СТУДЕНТА'

Перелічує всі попередні курси, які студент повинен пройти, як зазначено у Формі рішення аспіранта.

Атрибут	Опис	Тип даних	Примітки
c_id	10-значний унікальний ID	numeric	CK, FK, що посилається на PK таблиці СТУДЕНТ.
course_id	3-значний унікальний ID	numeric	CK, FK, що посилається на PK таблиці КУРС.

Рисунок 2.15 - Структура таблиці бази даних 'ПОПЕРЕДНІ_ВИМОГИ_СТУДЕНТА'

14. Таблиця 'ПЕРЕВАГИ_ЧАСУ'

Описує, який час дня студент віддає перевагу для проходження курсів.

Атрибут	Опис	Тип даних	Примітки
c_id	10-значний унікальний ID	numeric	CK, FK, що посилається на PK таблиці ПЕРЕВАГИ_СТУДЕНТА.
time_preference	Час дня (ранок, день, вечір), який студент віддає перевагу для проходження курсу, як зазначено в анкеті.	змінної довжини символ	CK

Рисунок 2.16 - Структура таблиці бази даних 'ПЕРЕВАГИ_ЧАСУ'

15. Таблиця 'КОРИСТУВАЧ'

Містить інформацію про кожного зареєстрованого користувача в системі.

Атрибут	Опис	Тип даних	Примітки
c_id	10-значний унікальний ID	numeric	PK
name	Ім'я, по батькові та прізвище студента.	змінної довжини символ	NN
email_id	email ID студента.	змінної довжини символ	NN
phone	Телефонний контактний номер студента.	numeric	NN
address	Адреса (вулиця, місто, штат, поштовий індекс) студента.	json	NN
date_of_birth	Дата народження (MM/DD/YYYY) користувача.	date	NN
password	Зашифрований пароль користувача.	змінної довжини символ	NN

Рисунок 2.17 - Структура таблиці бази даних 'КОРИСТУВАЧ'

2.4. Визначення та застосування програмних пакетів у проекті

Розробка програмного забезпечення для даного проекту передбачала використання спеціалізованих програмних пакетів, кожен з яких надає

унікальну функціональність та набір класів, що є інтегральними компонентами архітектури застосунку.

2.4.1. Пакет pg

Пакет pg функціонує як чистий JavaScript-клієнт для PostgreSQL, забезпечуючи механізм встановлення з'єднання з базою даних. Він підтримує ключові можливості PostgreSQL, такі як параметризовані запити, асинхронні повідомлення, а також функціональність імпорту та експорту даних.

У даному проекті pg використовується для виконання операцій з базою даних. Наприклад, фрагмент коду, представлений на рисунку 2.18, демонструє застосування pg для видалення запису користувача з таблиці user на основі c_id.

```
function deleteUser(req, res, next) {
    const c_id = req.query.c_id;
    console.log("inside");
    pg.connect(connectionString, function(err, client, done){
        // Обробка помилок з'єднання
        if(err) {
            done();
            console.log(err);
            return res.status(500).json({success: false, data: err});
        }
        const query = client.query('DELETE FROM "user" WHERE c_id =($1);',
            [c_id]);
        query.on('end', function () {
            done();
            return res.json("success");
        });
    });
}
```

Рисунок 2.18 - Приклад використання пакета pg для взаємодії з базою даних

2.4.2. Фреймворк Express

Express.js є мінімалістичним та гнучким веб-фреймворком для Node.js, який спрощує розробку веб-додатків та API. Його основні особливості

включають потужний маршрутизатор, високу продуктивність, можливості перенаправлення HTTP-запитів та кешування.

У контексті цього проекту, Express.js було використано для швидкого створення серверної частини та ефективної обробки HTTP-запитів. Рисунок 2.19 ілюструє його застосування для визначення маршрутів та обслуговування статичних файлів, що є типовим для веб-додатків на Node.js.

```
const express = require('express');
const router = express.Router();
const pg = require('pg');
const path = require('path');
const loginQueries = require('../queries/login.queries');
const questionnaireQueries = require('../queries/questionnaire.queries');
const userQueries = require('../queries/user.queries');
const courseQueries = require('../queries/course.queries');
router.get('/', function(req, res, next){
    console.log(path.join(
        __dirname, '..','..','index.html'))
    res.sendFile(path.join(
    __dirname, '..','..','index.html'));
});
router.get('/login', loginQueries.getUser);
module.exports = router;
```

Рисунок 2.19 - Приклад визначення маршрутів та обслуговування файлів за допомогою Express.js.

2.4.3. Пакет cookie-parser

Пакет cookie-parser призначений для розбору заголовків HTTP-кукі та представлення їх у вигляді об'єкта, де ключами є назви кукі. Він також підтримує функціональність підпису кукі за умови надання секретного параметра, що підвищує безпеку.

У даному проекті cookie-parser інтегрований у функції аутентифікації (login та logout). Зберігання таких даних, як ідентифікатор користувача (Coyote ID) та його роль у кукі, дозволяє відновити стан сесії навіть після перезавантаження сторінки, забезпечуючи безперервність взаємодії користувача з системою. Рисунок 2.20 демонструє процес встановлення кукі після успішного входу.

```
UserService.getUserRole(vm.userID).then(
    function success(response) {
        vm.role = response.data[0];
        cookieData = {
            auth: vm.data.userID,
            role: vm.role
        };
        $cookies.put('usedata', cookieData);
        $state.go('root', {userID : vm.data.c_id, userRole: vm.role});
    }, function error(response) {
        console.log(response)
    }
)
```

Рисунок 2.20 - Механізм встановлення кукі при вході користувача

2.4.4. Пакет Nodemailer

Nodemailer є модулем Node.js, розробленим для надсилання електронних листів. Він підтримує надсилання повідомлень з HTML-вмістом та звичайним текстом, а також прикріплення файлів.

```
function resetPassword(req, res, next) {
    var email = req.body.email;
    var code = req.body.code;
    var text = 'Код для скидання пароля: ' + code;
    var mailOptions = {
        from: '######@gmail.com', // Використання заглушки для конфіденційності
        to: email,
        subject: 'Скидання пароля',
        text: text
    };
    var transporter = nodemailer.createTransport({
        service: 'gmail',
        auth: {
            user: '######@gmail.com', // Використання заглушки для конфіденційності
            pass: '##########'       // Використання заглушки для конфіденційності
        }
    });
    transporter.sendMail(mailOptions, function(error, info){
        if(error){
            console.log(error);
            res.json({yo: 'error'});
        }else{
            console.log('Повідомлення надіслано: ' + info.response);
            res.json({yo: info.response});
        }
    });
}
```

Рисунок 2.21 - Надсилання коду скидання пароля за допомогою Nodemailer

У цьому проекті Nodemailer використовується для реалізації функціональності відновлення пароля. Зокрема, він застосовується для надсилання випадково згенерованого коду на електронну пошту користувача, який потім може бути використаний для скидання облікових даних. Рисунок 2.21 ілюструє процес надсилання такого коду.

2.4.5. Пакет Cryptr

Модуль Cryptr для Node.js реалізує симетричне шифрування та дешифрування рядків з використанням алгоритму AES (Advanced Encryption Standard).

```
function changePassword(req, res, next) {
    const coyote_id = req.body.coyote_id;
    const password = cryptr.encrypt(req.body.password);
    pg.connect(connectionString, function(err, client, done){
        if(err) {
            done();
            console.log(err);
            return res.status(500).json({success: false, data: err});
        }
        const query = client.query('UPDATE "user" SET password=($1) WHERE c
        query.on('end', function(){
            done();
            return res.json('Success');
        });
    });
}
```

Рисунок 2.22 - Застосування Cryptr для шифрування пароля перед збереженням у базі даних

У цьому проекті Cryptr застосовується в модулях входу (login) та реєстрації (registration) для забезпечення безпеки облікових даних. Він шифрує рядок пароля перед його збереженням у базі даних та дешифрує його під час процесу аутентифікації для порівняння з введеним паролем. Рисунок 2.22 демонструє використання Cryptr для шифрування пароля перед оновленням запису користувача.

РОЗДІЛ 3. МОДЕЛЮВАННЯ АРХІТЕКТУРИ СИСТЕМИ ЗА ДОПОМОГОЮ DFD ТА UML ДІАГРАМ

Для ефективного розуміння та аналізу функціонування розробленої системи було застосовано стандартизовані методи візуалізації архітектури, зокрема діаграми потоку даних (DFD) та діаграми уніфікованої мови моделювання (UML). Ці діаграми надають структуроване представлення взаємодії компонентів, потоків інформації та поведінки системи в різних сценаріях.

3.1. Розробка діаграм потоку даних (DFD)

Діаграма потоку даних служить графічним представленням переміщення даних у межах системи, одночасно надаючи детальний опис кожного її компонента. У цьому проекті DFD використовувалися для ілюстрації інформаційних потоків у ключових модулях системи:

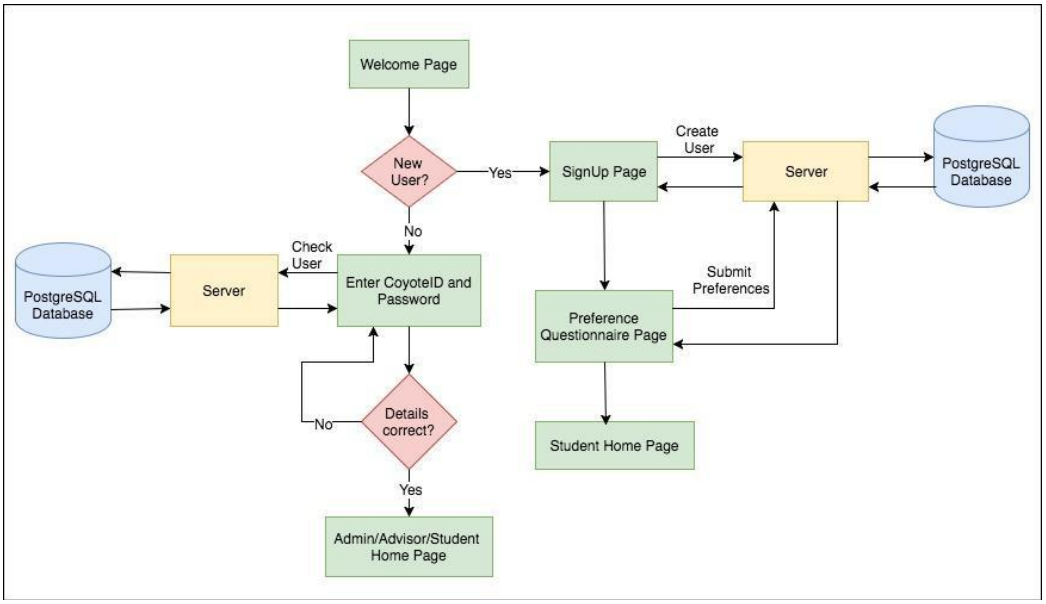

Рисунок 3.1 - Діаграма потоку даних для функціональності входу та реєстрації

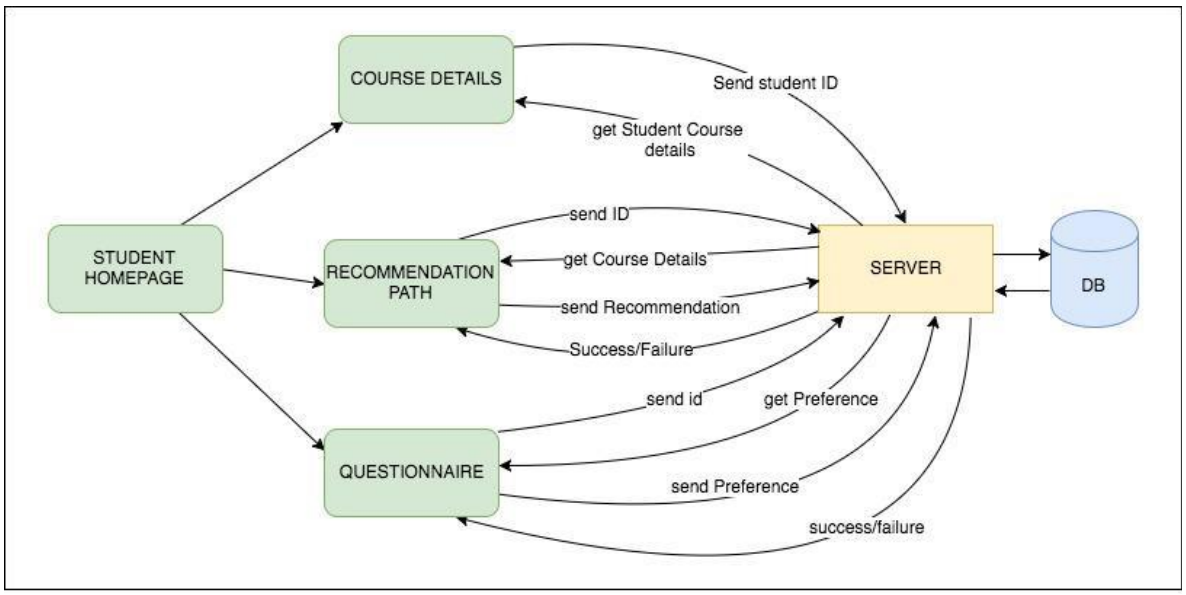

Рисунок 3.2 - Діаграма потоку даних для операцій, що виконуються студентом

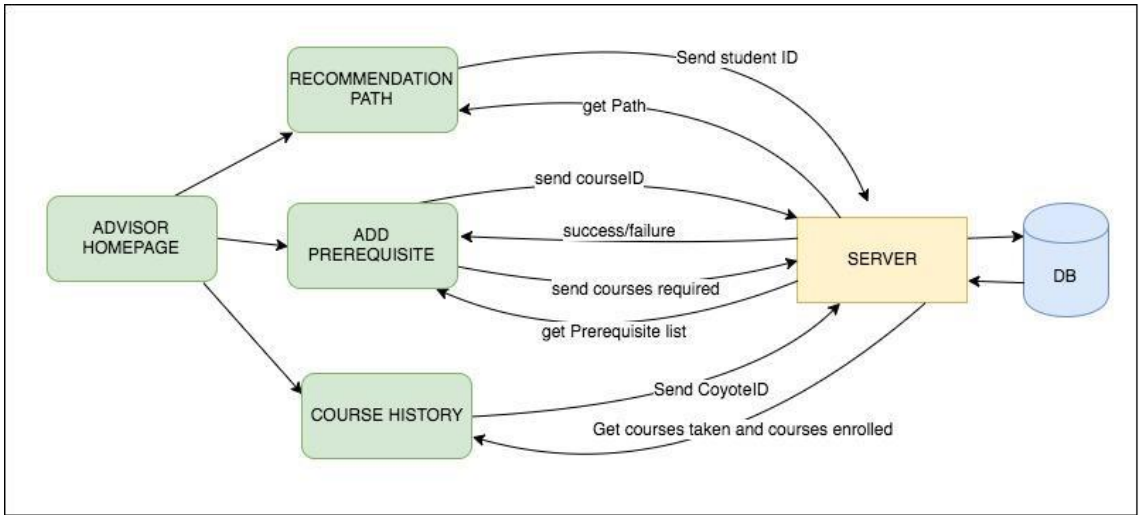

Рисунок 3.3 - Діаграма потоку даних для операцій, що виконуються консультантом

Діаграма демонструє, як консультант, використовуючи домашню сторінку, може взаємодіяти з системою для отримання рекомендованих навчальних планів для студентів, керування попередніми вимогами до курсів та перегляду академічної історії студентів. Усі ці операції виконуються через центральний сервер, який, у свою чергу, взаємодіє з базою даних для доступу та маніпулювання інформацією.

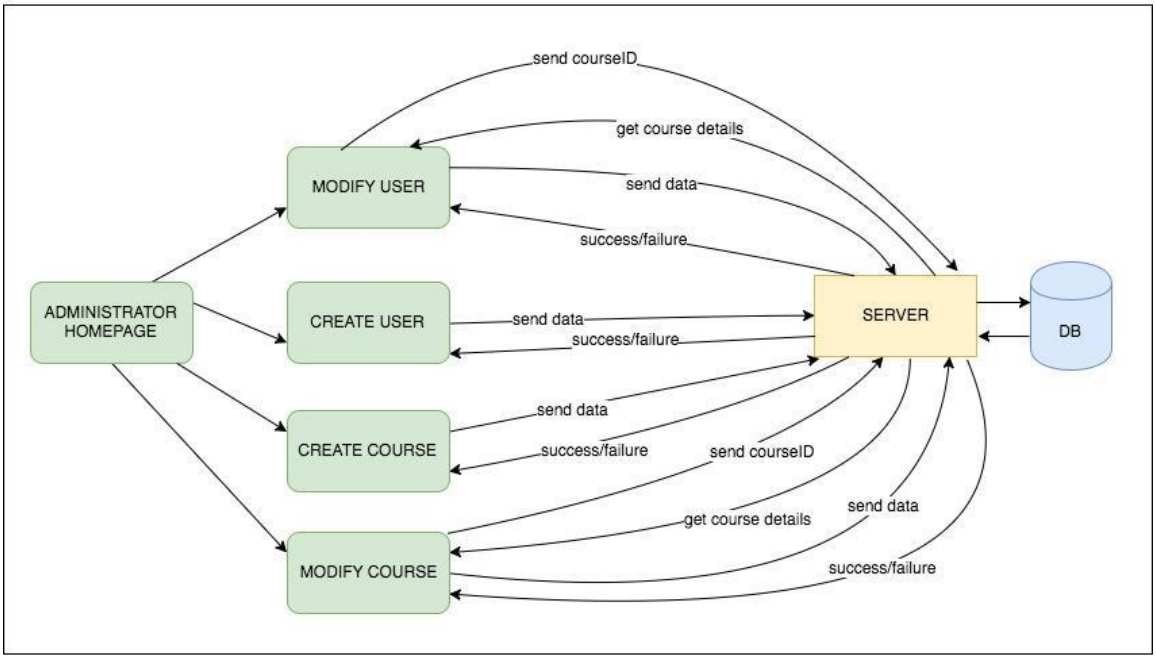

Рисунок 3.4 - Діаграма потоку даних для операцій, що виконуються адміністратором

UML-діаграми є стандартним інструментом для моделювання програмних систем. У рамках цього проекту було використано два основних типи UML-діаграм: діаграми випадків використання та діаграми послідовності.

3.2. Побудова діаграм випадків використання

Діаграма випадків використання візуалізує взаємодію користувача (актора) з системою, а також показує зв'язки між користувачем та різними випадками використання, в яких він бере участь. Ці діаграми є цінними для визначення функціональних вимог системи з точки зору користувача. Для даної системи розроблено наступні діаграми випадків використання.

Діаграма варіантів використання (Use Case Diagram), представлена на рисунку 3.5 ілюструє взаємодію актора (користувача) із системою. Вона показує, які функції або "варіанти використання" може виконувати студент у даній системі.

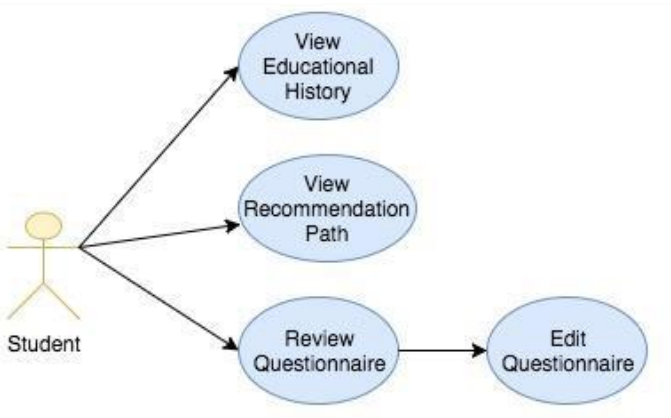

Рисунок 3.5 - Діаграма випадків використання для студента

Проведемо опис елементів діаграми.

Student (Студент) - зовнішній користувач системи, який взаємодіє з нею для досягнення своїх цілей.

Варіанти використання (Use Cases):

- View Educational History (Перегляд освітньої історії): Цей варіант використання дозволяє студенту отримати доступ до своєї академічної історії, яка може включати пройдені курси, оцінки, академічний прогрес тощо.

- View Recommendation Path (Перегляд рекомендованого шляху): Цей варіант використання дозволяє студенту переглядати згенерований системою рекомендований навчальний план або шлях курсів.

- Review Questionnaire (Перегляд анкети): Цей варіант використання дає студенту можливість переглянути раніше заповнену анкету, яка, ймовірно, містить його переваги, інтереси та інші дані, що використовуються для генерації рекомендацій.

- Edit Questionnaire (Редагування анкети): Цей варіант використання дозволяє студенту вносити зміни до своєї анкети. Зверніть увагу, що цей варіант використання пов'язаний з "Review Questionnaire" відношенням "include" (включення). Це означає, що функціональність "Edit Questionnaire"

включається в функціональність "Review Questionnaire". Іншими словами, для редагування анкети студент спочатку повинен її переглянути.

Ця діаграма показує основні функції системи рекомендацій курсів з точки зору студента. Вона демонструє, що система дозволяє студентам отримувати інформацію про свій прогрес, бачити рекомендовані курси та керувати своїми вподобаннями через анкету.

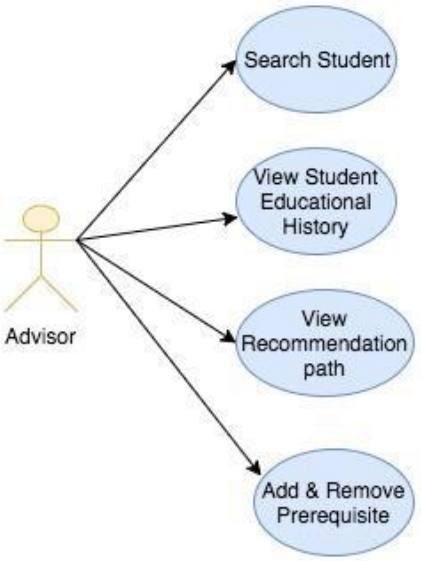

Рисунок 3.6 - Діаграма випадків використання для консультанта

Діаграма (рис. 3.6) ілюструє взаємодію актора — "Advisor" (Радник/Консультант) — із системою рекомендацій курсів. Вона показує, які функції або "варіанти використання" може виконувати консультант.

Advisor (Радник/Консультант) - зовнішній користувач системи, відповідальний за академічне консультування студентів.

Варіанти використання (Use Cases):

- Search Student (Пошук студента): Цей варіант використання дозволяє консультанту знаходити конкретних студентів у системі, ймовірно, за іменем, ідентифікатором або іншими критеріями.

- View Student Educational History (Перегляд освітньої історії студента): Цей варіант використання дозволяє консультанту отримати доступ до

академічної історії конкретного студента, включаючи пройдені курси, оцінки, академічний прогрес та іншу відповідну інформацію.

- View Recommendation Path (Перегляд рекомендованого шляху): Цей варіант використання дає консультанту можливість переглядати згенерований системою рекомендований навчальний план або шлях курсів для певного студента. Це дозволяє консультанту оцінювати та обговорювати запропонований план зі студентом.

- Add & Remove Prerequisite (Додати та видалити попередню вимогу): Цей варіант використання дозволяє консультанту керувати попередніми вимогами до курсів. Це може включати додавання нових попередніх вимог або видалення існуючих, що може бути необхідно для оновлення навчальних програм або індивідуальних коригувань для студентів.

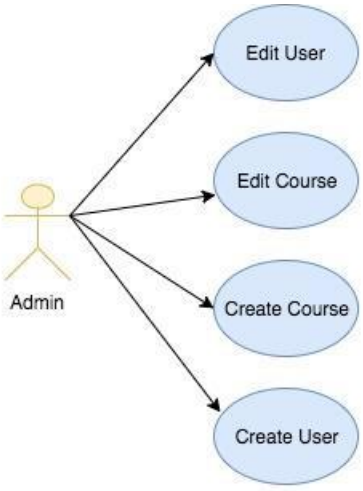

Рисунок 3.7 - Діаграма випадків використання для адміністратора

Діаграма на рисунку 3.7 ілюструє взаємодію третього актора — "Admin" (Адміністратор) — із системою рекомендацій курсів. Вона показує, які адміністративні функції може виконувати адміністратор.

Admin (Адміністратор) - це зовнішній користувач системи, відповідальний за її управління та підтримку, включаючи керування даними користувачів та курсів.

Варіанти використання (Use Cases):

- Edit User (Редагувати користувача): Цей варіант використання дозволяє адміністратору змінювати інформацію про існуючих користувачів системи. Це може включати оновлення профілів студентів, консультантів або інших адміністраторів.

- Edit Course (Редагувати курс): Цей варіант використання дає адміністратору можливість змінювати деталі існуючих курсів, такі як назва, опис, кількість кредитів, семестри пропозиції, попередні вимоги тощо.

- Create Course (Створити курс): Цей варіант використання дозволяє адміністратору додавати нові курси до системи.

- Create User (Створити користувача): Цей варіант використання дає адміністратору можливість додавати нових користувачів до системи, призначаючи їм відповідні ролі (студент, консультант, інший адміністратор).

Ця діаграма завершує загальну картину функціональності системи рекомендацій курсів, показуючи можливості управління на адміністративному рівні. Вона демонструє, що система має комплексні можливості для підтримки повного життєвого циклу даних: від керування користувачами (студентами, консультантами) до управління навчальними курсами. Це забезпечує гнучкість та можливість адаптації системи до змін у навчальних програмах та контингенті користувачів.

3.3. Побудова діаграм послідовності

Діаграма послідовності є типом діаграми взаємодії, яка демонструє, як об'єкти або процеси взаємодіють між собою з плином часу. Вона чітко показує порядок викликів методів та обмін повідомленнями між різними компонентами системи. У цьому проекті діаграми послідовності були використані для деталізації логіки ключових функціональних можливостей.

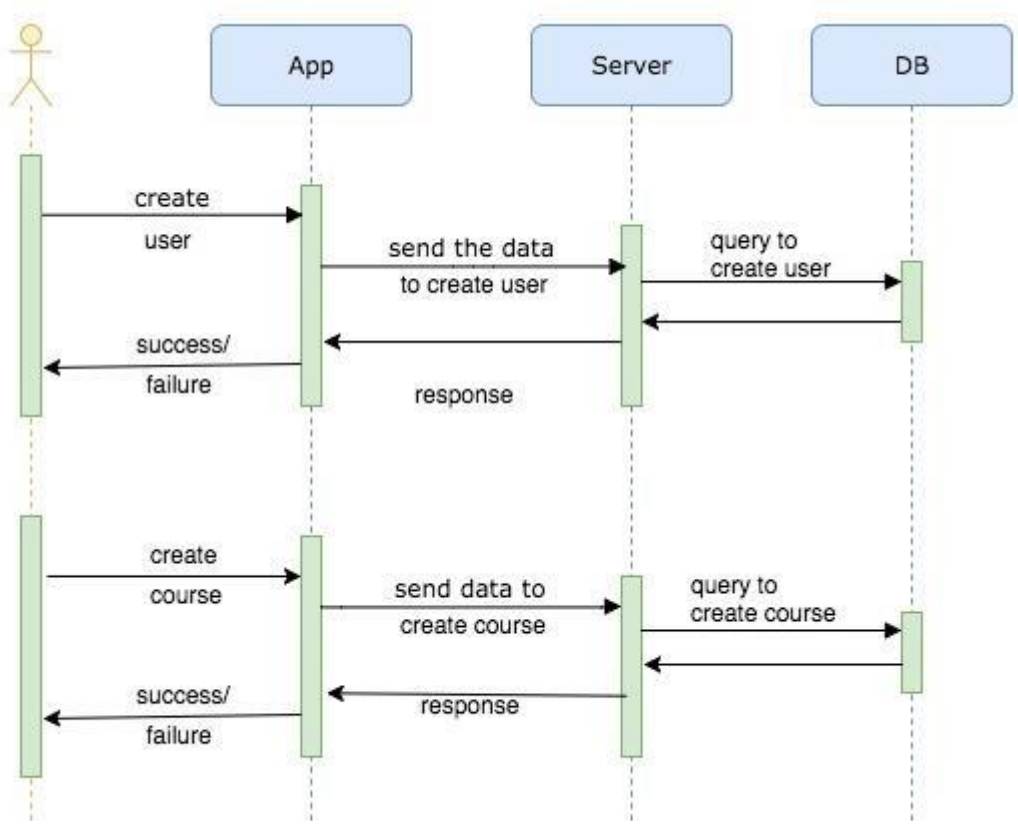

Рисунок 3.8 - Діаграма послідовності для створення користувача та курсу

Дана діаграма ілюструє процес створення нового користувача або курсу. Адміністратор ініціює операцію, обираючи відповідну функцію. Система перенаправляє користувача на форму введення даних, які потім відправляються на сервер. Сервер обробляє запит, взаємодіє з базою даних для збереження інформації та повертає статус відповіді (успіх/помилка) до клієнтського застосунку. Застосунок, у свою чергу, відображає отриманий статус користувачеві. Аналогічна послідовність дій застосовується для створення курсів.

Сценарій "Створення курсу":

1. Людина ініціює дію create course (створити курс), надсилаючи повідомлення до App.

2. App отримує запит і відправляє повідомлення send data to create course (відправити дані для створення курсу) до Server.

3. Server отримує дані і виконує query to create course (запит на створення курсу) до DB.

4. DB обробляє запит і надсилає відповідь (response) назад до Server.

5. Server пересилає цю відповідь (response) до App.

6. App обробляє відповідь і відображає success/failure (успіх/невдача) для Людини.

Ця діаграма (рис. 3.8) послідовності наочно демонструє потік даних та взаємодію між рівнями (клієнтський додаток, сервер, база даних) при виконанні типових операцій додавання даних у системі. Вона показує, що обидві операції (створення користувача та створення курсу) слідують однаковому трирівневому архітектурному патерну: користувач взаємодіє з додатком, додаток звертається до сервера, сервер взаємодіє з базою даних, а потім відповідь повертається у зворотному порядку. Це дозволяє розробникам чітко розуміти логіку та порядок викликів функцій у системі

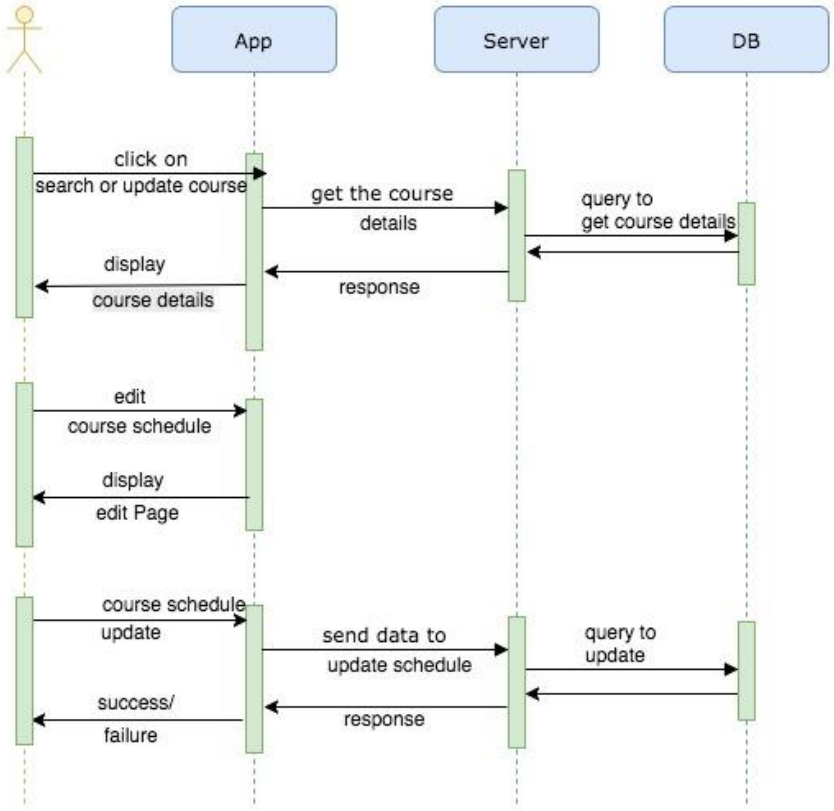

Рисунок 3.9 - Діаграма послідовності для оновлення курсу

Ця діаграма відображає процедуру оновлення інформації про курс. Спочатку адміністратор здійснює пошук курсу за його ідентифікатором або назвою. Система отримує відповідні деталі курсу з бази даних та відображає їх. Після вибору конкретного розкладу курсу для редагування, адміністратор перенаправляється на сторінку редагування, вносить необхідні зміни та підтверджує їх збереження. Оновлені дані передаються на сервер, який відповідає за їх модифікацію в базі даних та інформування застосунку про результат операції.

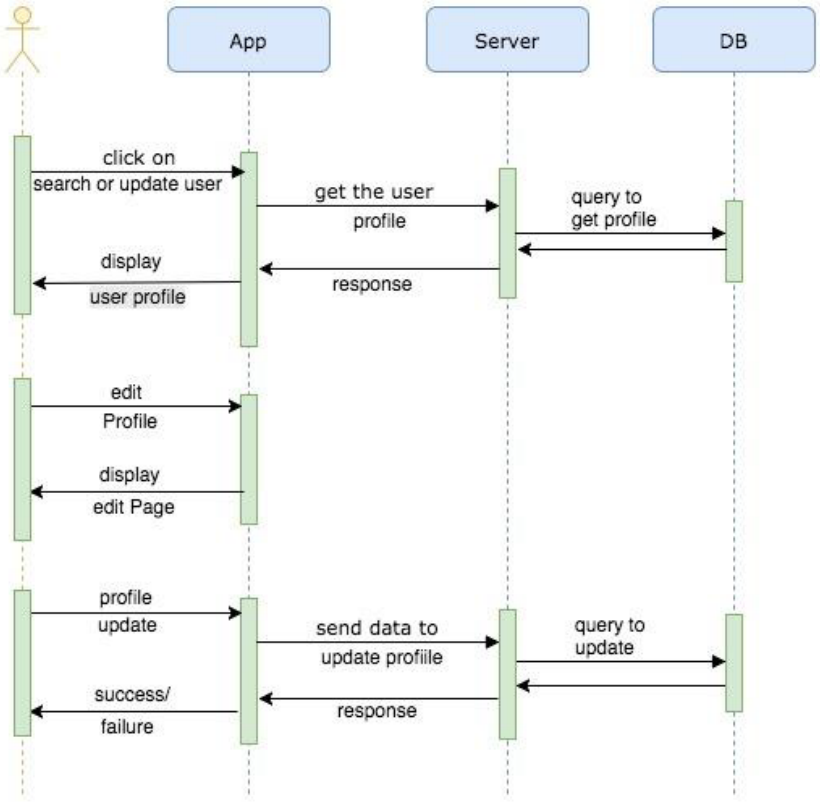

Рисунок 3.10 - Діаграма послідовності для оновлення профілю користувача

Дана діаграма деталізує процес оновлення профілю користувача. Адміністратор ініціює пошук користувача, ввівши частину його імені або C_ID. Після ідентифікації користувача адміністратор може переглянути його профіль. Застосунок надає функціональність редагування даних, дозволяючи

адміністратору вносити зміни та зберігати їх на сервері. Сервер, у свою чергу, оновлює інформацію в базі даних та надсилає підтвердження про успішне завершення операції.

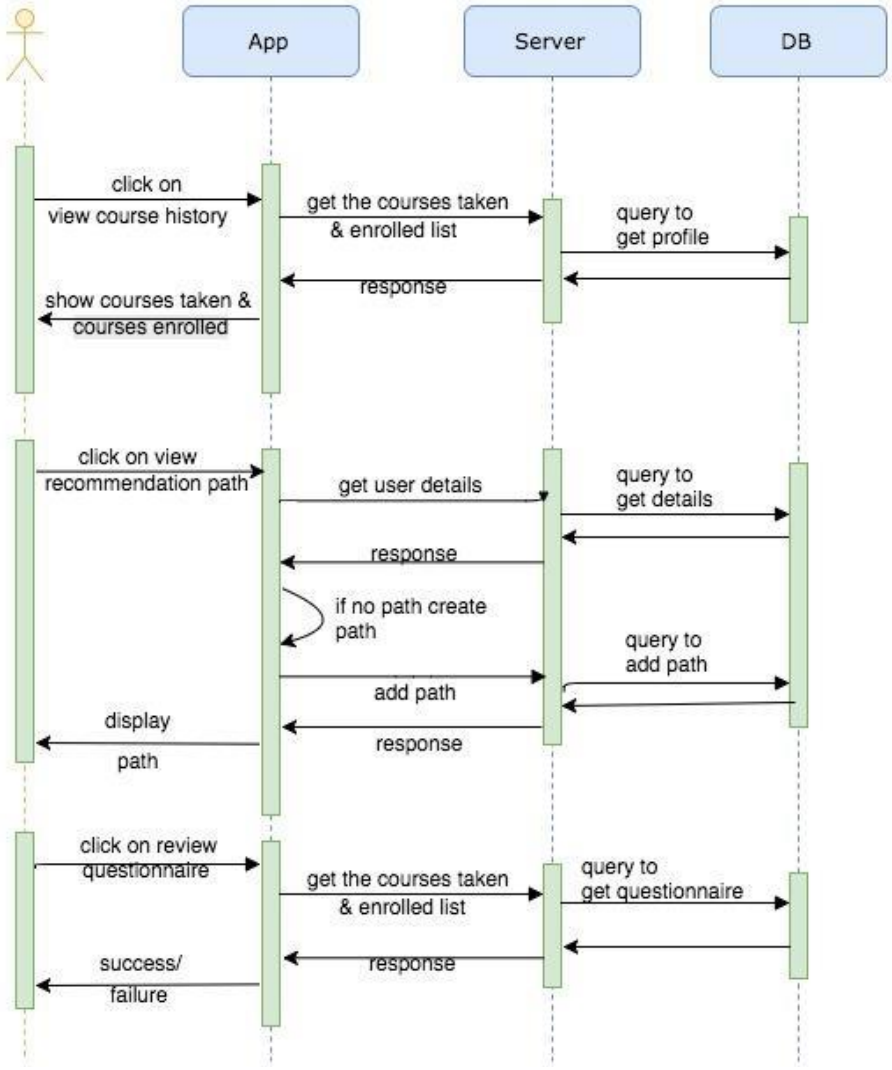

Рисунок 3.11 - Діаграма послідовності для ролі студента

Домашня сторінка студента інтегрує три ключові функціональні блоки: "Історія курсів", "Шлях рекомендацій" та "Анкета". При виборі "Історії курсів" студент отримує доступ до своєї академічної історії, включаючи зареєстровані та пройдені курси, а також відповідний середній бал. "Шлях рекомендацій" надає індивідуалізований план курсових робіт від поточного

до випускного семестру. Функція "Анкета" дозволяє студенту переглядати та коригувати раніше встановлені переваги.

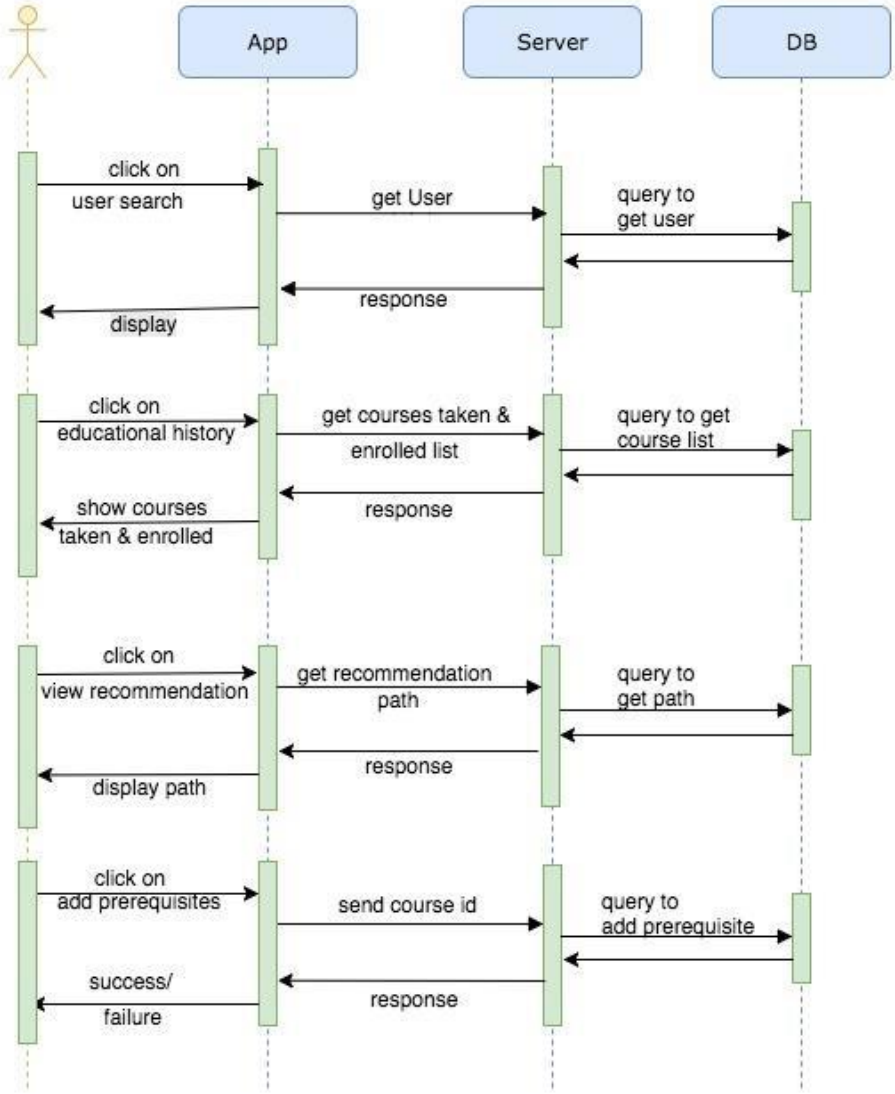

Рисунок 3.12 - Діаграма послідовності для ролі консультанта

Дана діаграма відображає основні функціональні можливості, доступні консультанту. Консультант може здійснювати пошук користувачів за ім'ям або C_ID, що активує відповідну серверну службу для отримання даних про користувача. Для перегляду академічної історії студента використовується серверна служба "отримати пройдені курси". Окрім того, консультант може переглядати шлях рекомендацій для студента, що також передбачає запит до сервера. Функція "додати попередні вимоги" дозволяє консультанту вводити

рекомендовані курси попередніх вимог для студента, використовуючи POST-запит для збереження даних у базі даних. Успішне виконання цієї транзакції підтверджується відповідним повідомленням.

РОЗДІЛ 4. РОЗРОБКА АЛГОРИТМУ ГЕНЕРАЦІЇ РЕКОМЕНДАЦІЙ КУРСІВ І ПРЕДСТАВЛЕННЯ ІНТЕРФЕЙСУ СИСТЕМИ

4.1. Алгоритм генерації рекомендацій курсів

Алгоритм рекомендацій розроблено для формування індивідуалізованого навчального плану для студентів, який охоплює період від першого до останнього семестру навчання. Процес генерації шляху враховує декілька ключових факторів: доступність курсів у конкретних семестрах, попередні вимоги, визначені у формі рішення аспіранта при вступі, та індивідуальні переваги студента, отримані з поданої анкети. До цих переваг належать: обраний варіант ступеня (проект, дисертація, комплексний іспит), бажаний розклад (дні/час занять) та предметні інтереси для вибіркових курсів. Сформований шлях включає всі необхідні попередні вимоги, основні та вибіркові курси, рекомендовані для проходження у кожному семестрі.

4.1.1. Принципи пріоритетизації курсів

Рисунок 4.1 представляє алгоритм, що рекомендацій послідовність дій та пріоритетів. Найвищий пріоритет надається курсам попередніх вимог. Це обумовлено тим, що студент не може бути зарахований на основний курс, доки не будуть виконані всі його попередні вимоги. Наприклад, курс С 430 має бути пройдений до зарахування на курс С 630. Після задоволення попередніх вимог, до шляху додаються основні курси, які ще не були пройдені студентом.

Семестр для включення основного курсу визначається на основі завершення його попередньої вимоги. Нарешті, вибіркові курси додаються до шляху, їхній вибір базується на даних, отриманих з анкети переваг студента.

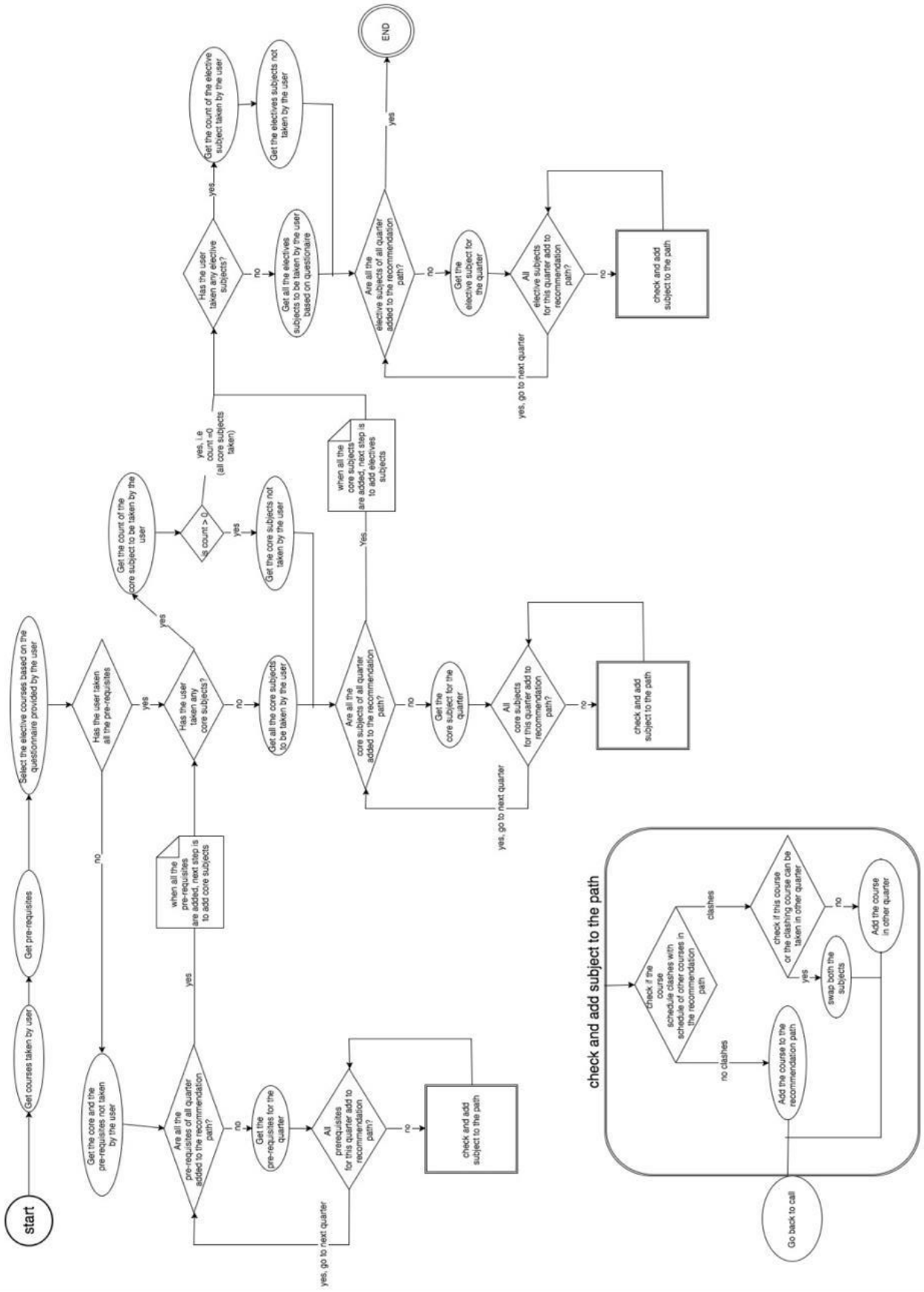

Рисунок 4.1 – Рекомендаційний алгоритм

4.1.2. Деталізований опис алгоритму рекомендацій

Алгоритм складається з чотирьох послідовних кроків (A, B, C, D), кожен з яких відповідає за додавання певного типу курсів до навчального шляху.

Крок A. Додавання курсів попередніх вимог

1. Крок 1: Здійснити запит до таблиць `student_prerequisite` та `course_taken` для отримання списку всіх попередніх вимог, які студент ще не виконав.

2. Крок 2: Обрати одну невиконану попередню вимогу (далі — P_1) зі списку, отриманого на Кроці 1.

3. Крок 3: Визначити оптимальний семестр (Q_1) для P_1. Цей семестр обирається з таблиці `quarter_offered` і повинен передувати семестру, в якому пропонується пов'язаний основний курс. Також необхідно перевірити відповідність Q_1 перевазі студента щодо кількості курсів:

- Якщо умова щодо кількості курсів не задовольняється, повторити Крок 3 для вибору іншого семестру для P_1.

- Якщо умова задовольняється, перейти до Кроку 4.

4. Крок 4: Перевірити наявність конфліктів у розкладі. Порівняти розклад P_1 (отриманий з таблиці `course_schedule`) з розкладом будь-якого іншого курсу (P_2), який вже включений до шляху рекомендацій для семестру Q_1:

- Якщо P_1 конфліктує з P_2, перейти до Кроку 6.

- Якщо конфліктів немає, перейти до Кроку 5.

5. Крок 5: Додати P_1 до шляху рекомендацій:

- Якщо є інші невиконані попередні вимоги для додавання, повернутися до Кроку 2.

- Якщо всі попередні вимоги додані, перейти до Кроку B.

6. Крок 6: Усунення конфліктів розкладу. Перевірити можливість перенесення P_2 в інший семестр (Q_2):

- Якщо P_2 може бути перенесений до Q_2, оновити шлях рекомендацій, перемістивши P_2 до Q_2, а потім повернутися до Кроку 5, щоб додати P_1 до Q_1.

- Якщо перенесення неможливе, додати P_1 до наступного доступного семестру, в якому він пропонується.

Крок B. Додавання основних курсів

1. Крок 1: Отримати список усіх основних курсів, які ще не були пройдені студентом, шляхом запиту до таблиць `core_course` та `course_taken` бази даних.

2. Крок 2: Обрати один основний курс (далі — C_1) зі списку, отриманого на Кроці 1.

3. Крок 3: Визначити оптимальний семестр (Q_1) для C_1. Це повинен бути найближчий семестр, в якому C_1 пропонується, або найближчий семестр після того, як його попередня вимога (P) була (або буде) пройдена.

4. Крок 4: Перевірити наявність конфліктів у розкладі. Порівняти розклад C_1 (з `course_schedule`) з розкладом будь-якого іншого курсу (C_2), який вже є у шляху рекомендацій для семестру Q_1:

- Якщо C_1 конфліктує з C_2, перейти до Кроку 6.

- Якщо конфліктів немає, перейти до Кроку 5.

5. Крок 5: Додати C_1 до шляху рекомендацій:

- Якщо є інші основні курси для додавання, повернутися до Кроку 2.

- Якщо всі основні курси додані, перейти до Кроку C.

6. Крок 6: Усунення конфліктів розкладу. Перевірити можливість перенесення C_2 в інший семестр (Q_2):

- Якщо C_2 може бути перенесений до Q_2, оновити шлях рекомендацій, перемістивши C_2 до Q_2, а потім повернутися до Кроку 5, щоб додати C_1 до Q_1.

- Якщо перенесення неможливе, додати C_1 до наступного доступного семестру, в якому він пропонується.

Крок C. Додавання вибіркових курсів

Вибіркові курси фільтруються відповідно до переваг студента, вказаних в анкеті (варіант ступеня, наявність/відсутність лабораторій, бажані дні та час занять, інтерес до літніх курсів, інтерес до незалежного навчання).

1. Крок 1: Визначити загальну кількість вибіркових курсів, які необхідно пройти, виходячи з обраного варіанта ступеня студента (проект: 5 курсів, дисертація: 4 курси, комплексний іспит: 6 курсів).

2. Крок 2: Обчислити залишкову кількість вибіркових курсів, які ще потрібно додати. Обрати один вибірковий курс (далі — E_1) зі списку доступних вибіркових курсів, що відповідають перевагам студента. Цей вибір здійснюється шляхом запиту до таблиць `elective_course`, `course_taken`, `student_preference`, `day_preference` та `time_preference` бази даних.

3. Крок 3: Визначити оптимальний семестр (Q_1) для E_1.

4. Крок 4: Перевірити наявність конфліктів у розкладі. Порівняти розклад E_1 з розкладом будь-якого іншого курсу (E_2), який вже включений до шляху рекомендацій для семестру Q_1:

- Якщо E_1 конфліктує з E_2, перейти до Кроку 6.

- Якщо конфліктів немає, перейти до Кроку 5.

5. Крок 5: Додати E_1 до шляху рекомендацій:

- Якщо є інші вибіркові курси для додавання, повернутися до Кроку 2.

- Якщо всі вибіркові курси додані, перейти до Кроку D.

6. Крок 6: Усунення конфліктів розкладу. Перевірити можливість перенесення E_2 в інший семестр (Q_2):

- Якщо E_2 може бути перенесений до Q_2, оновити шлях рекомендацій, перемістивши E_2 до Q_2, а потім повернутися до Кроку 5, щоб додати E_1 до Q_1.

- Якщо перенесення неможливе, додати E_1 до наступного доступного семестру, в якому він пропонується.

Крок D. Додавання курсу варіанта ступеня

1. Крок 1: Визначити семестр (Q), в якому був доданий останній основний курс до шляху рекомендацій.

2. Крок 2: Додати відповідний курс варіанта ступеня до шляху в семестрі, що слідує за Q. Наприклад, якщо останній основний курс був пройдений в зимовому семестрі 2025 року, курс варіанта ступеня (C 690 для проекту, C 699 для дисертації або C 689 для комплексного іспиту) додається до весняного семестру 2025 року.

4.2. Візуалізація шляху отримання рекомендацій для ролі студента

Розроблений програмний проєкт пройшов комплексну валідацію за допомогою різних методів тестування, включно з модульним тестуванням (unit tests), тестуванням компонентів (module tests), інтеграційним тестуванням (integration tests), системним тестуванням (system tests) та тестуванням користувацького прийняття (user acceptance tests). Усі етапи тестування були успішно завершені, підтвердивши відсутність виявлених дефектів. Візуалізація ключових етапів тестування та функціональності системи представлена на наступних рисунках.

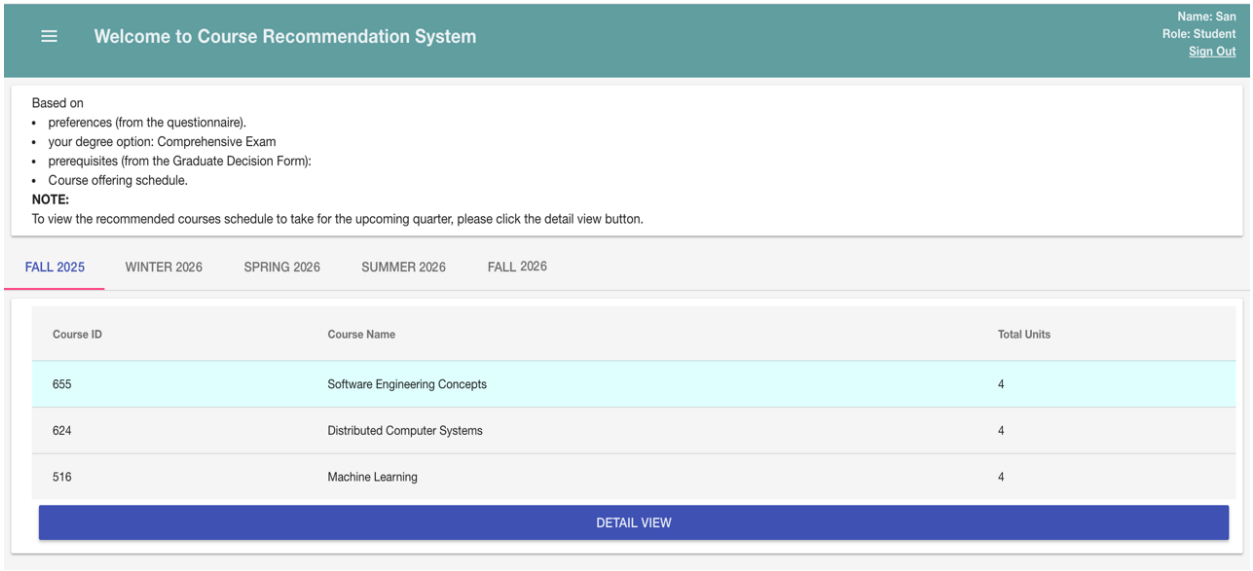

Рисунок 4.2 - Шлях рекомендацій для ролі студента – Частина 1

Даний рисунок демонструє згенерований шлях рекомендацій курсів для студента, представлений у хронологічній послідовності за семестрами та роками.

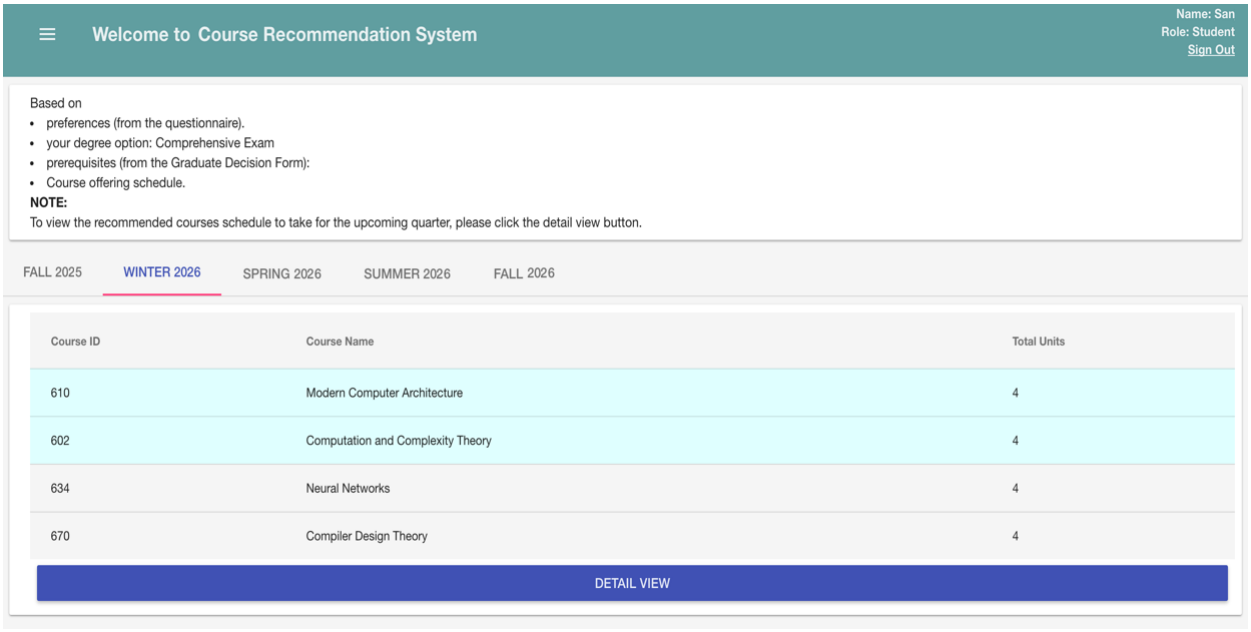

Рисунок 4.3 - Шлях рекомендацій для ролі студента – Частина 2

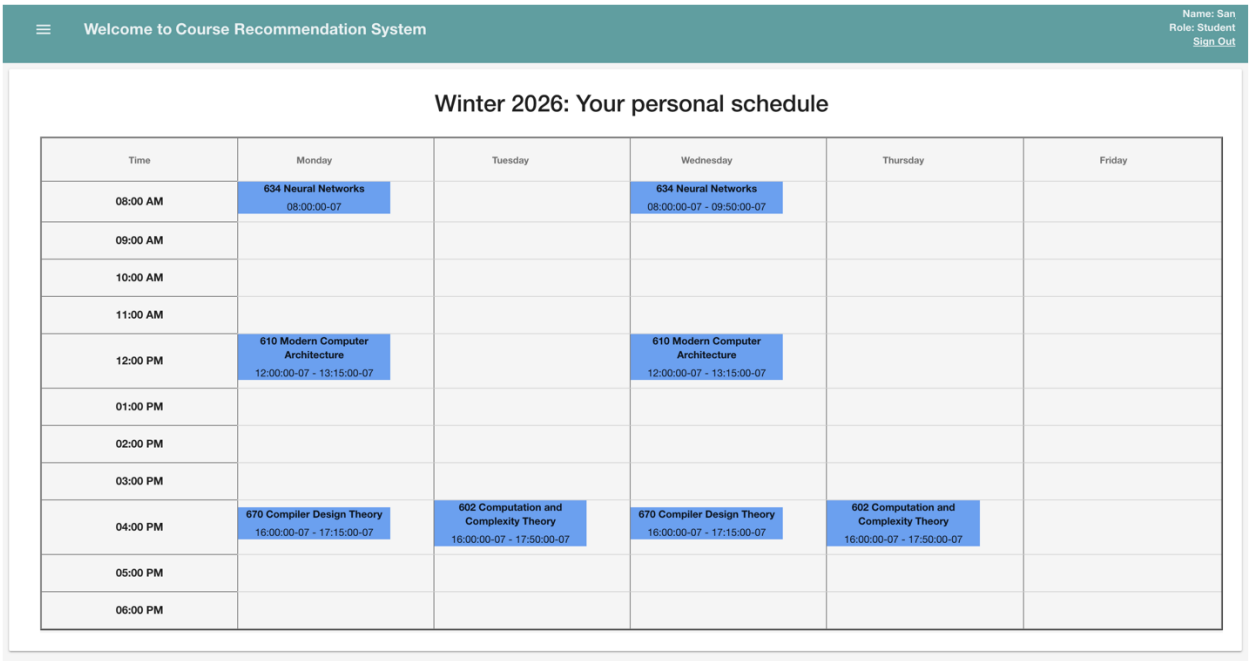

Рисунок 4.4 - Деталізований перегляд

На рисунку 4.4 наведено запланований розклад занять студента на конкретний семестр, сформований на основі курсів, рекомендованих алгоритмом.

4.3. Візуалізація шляху отримання рекомендацій для ролі консультанта

Рисунок 4.5 ілюструє інтерфейс сторінки шляху рекомендацій студента з перспективи консультанта. Шлях представлено в послідовності семестрів та років.

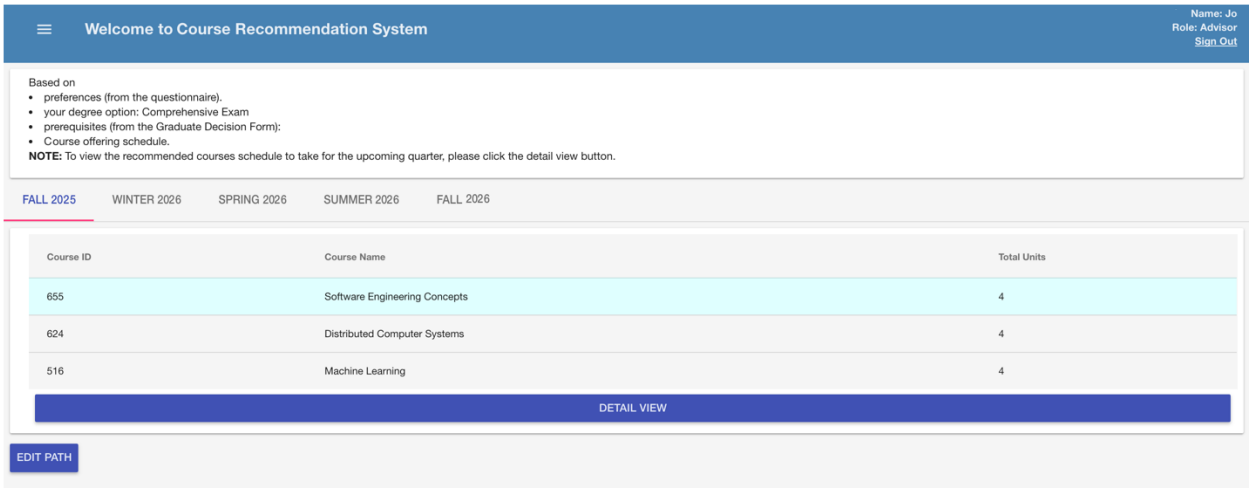

Рисунок 4.5 - Шлях рекомендацій для ролі консультанта

Активація кнопки "ДЕТАЛЬНИЙ ПЕРЕГЛЯД" здійснює перенаправлення консультанта на сторінку деталізованого перегляду, зображену на рисунку 4.6 Натискання кнопки "РЕДАГУВАТИ ШЛЯХ" призводить до переходу в інтерфейс редагування шляху рекомендацій, представлений на рисунку 4.7.

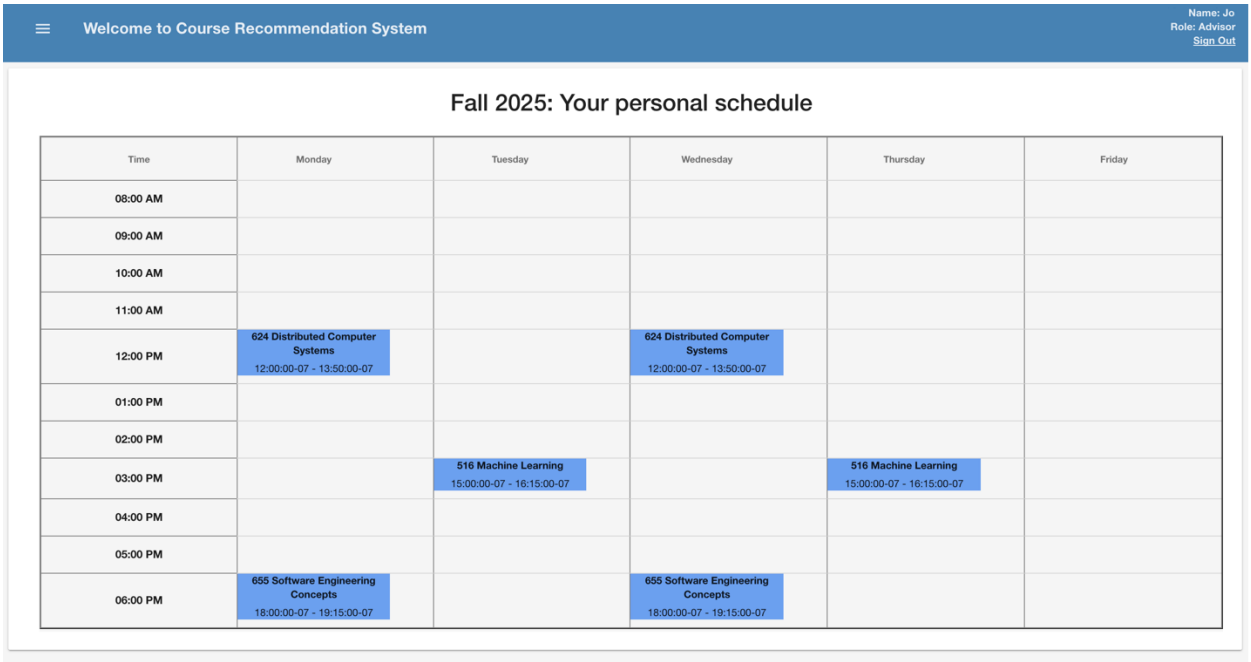

Рисунок 4.6 - Деталізований перегляд курсів за семестр

Рисунок 4.6 відображає деталізований перегляд курсів, запланованих на певний семестр. Ця сторінка містить розклад занять студента, які будуть взяті відповідно до рекомендацій.

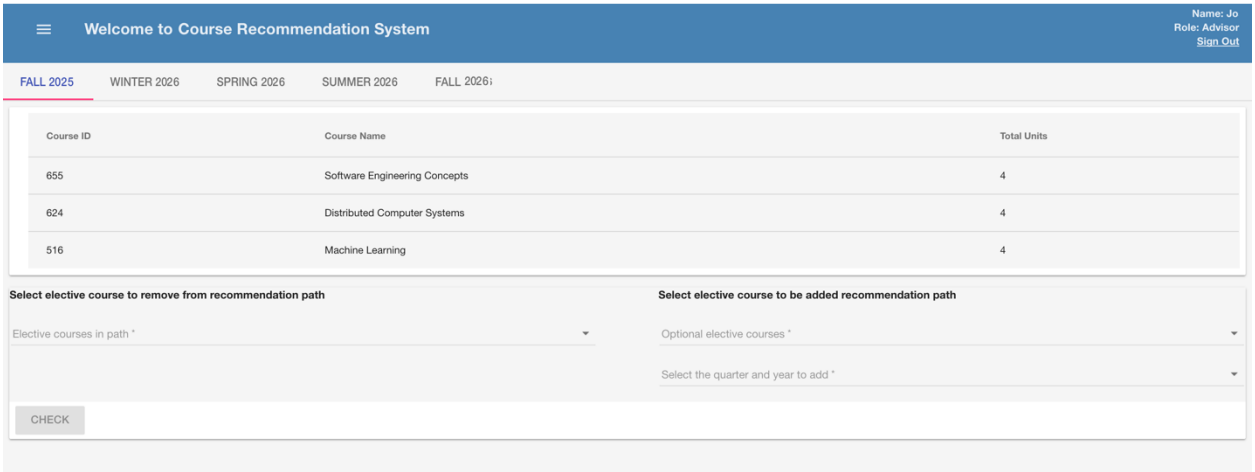

Рисунок 4.7 - Редагування шляху рекомендацій для ролі консультанта

На рисунку 4.7 показано інтерфейс редагування шляху рекомендацій, який дозволяє консультанту модифікувати послідовність курсів. Процедура редагування передбачає вибір двох курсів: курсу для видалення з поточного

шляху та нового курсу для додавання. Також необхідно вказати семестр та рік, до якого буде додано новий курс.

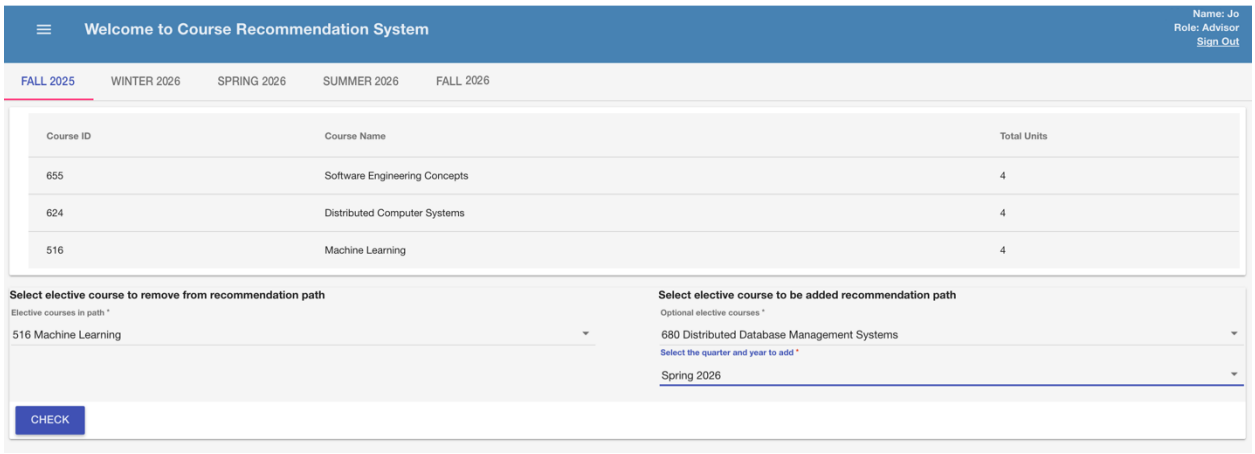

Рисунок 4.8 - Редагування шляху рекомендацій із заповненою інформацією

Рисунок 4.8 демонструє інтерфейс редагування шляху рекомендацій із заповненими полями, що ілюструє готовність системи до внесення змін до навчального плану.

РОЗДІЛ 5. ТЕСТУВАННЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ СИСТЕМИ РЕКОМЕНДАЦІЙ НАВЧАЛЬНИХ КУРСІВ

5.1. Методологія тестування програмного забезпечення

Основною метою тестування програмного забезпечення є ідентифікація та оцінка дефектів і помилок у системі, а також перевірка нормального функціонування усього вихідного коду. Цей процес дозволяє отримати всебічне розуміння поточного стану та показників продуктивності системи. Продуктивність, окрім безпосередніх помилок, може бути скомпрометована проблемами сумісності при розгортанні на різних програмно-апаратних платформах, що може призвести до аномальної поведінки. Глибокий інтроспективний аналіз системи через тестування часто сприяє оптимізації коду та ефективнішому використанню ресурсів.

Для валідації даної програмної реалізації було застосовано комплексний підхід до тестування, що включає декілька методологій:

- Модульне тестування (Unit Testing)
- Тестування компонентів (Module Testing)
- Інтеграційне тестування (Integration Testing)
- Тестування функціональності (Output Testing)
- Тестування прийняття користувачем (User Acceptance Testing)

5.1.1. Деталізація методів тестування

Модульне тестування (Unit Testing)

Модульне тестування спрямоване на оцінку коректності логіки та функціональності найменших, незалежно тестованих компонентів системи, які зазвичай відповідають окремим методам або функціям. Це тестування виконується на ранніх етапах життєвого циклу розробки програмного забезпечення з використанням методу "білої скриньки" (white-box testing), що

передбачає знання внутрішньої структури коду. Модульне тестування сприяє адаптивності системи до змін, прискорює процес розробки та підвищує надійність кодової бази.

Тестування компонентів (Module Testing)

Аналогічно модульному тестуванню, тестування компонентів (модулів) зосереджується на перевірці функціональності окремих модулів системи, де модуль визначається як сукупність декількох одиниць. Цей тип тестування також належить до тестування "білої скриньки". Основна мета тестування модулів — виявлення дефектів у конкретних програмних модулях, причому акцент робиться на виявленні помилок, а не лише на перевірці функціональності. Одночасне тестування кількох модулів підвищує ефективність процесу розробки. Модулі можуть тестуватися як незалежно, так і інкрементально.

Інтеграційне тестування (Integration Testing)

Інтеграційне тестування проводиться для виявлення дефектів, що виникають при взаємодії та об'єднанні окремих протестованих модулів або компонентів. Воно є логічним продовженням модульного тестування, оскільки не може бути виконано без попередньо протестованих одиниць. Інтеграційне тестування ефективно виявляє дефекти в інтерфейсах між компонентами системи. Існують два основні підходи до інтеграційного тестування: "зверху-вниз" (top-down), при якому спочатку тестуються модулі вищого рівня з подальшим переходом до нижчих, та "знизу-вгору" (bottom-up), де тестування починається з компонентів нижчого рівня та поступово просувається до вищих.

Тестування функціональності (Output Testing)

Тестування функціональності (вихідних даних), як випливає з назви, орієнтоване на перевірку коректності вихідних даних вибраних функцій. В рамках цього тестування розробляються тестові випадки, виконання яких повинно призвести до відомих, передбачуваних результатів. Ці тестові

випадки виконуються, і будь-які відхилення отриманих результатів від очікуваних вказують на наявність дефектів у системі. Незважаючи на потенційну трудомісткість генерації тестових випадків, ці зусилля є виправданими, оскільки вони дозволяють оцінити продуктивність системи та передбачити її поведінку в реальному середовищі експлуатації.

Тестування прийняття користувачем (User Acceptance Testing – UAT)

Тестування прийняття користувачем є критично важливим етапом, що забезпечує розуміння взаємодії кінцевого користувача із системою. Воно зосереджено на оцінці зручності використання (usability) та відповідності системи потребам кінцевих користувачів. На цьому етапі до процесу тестування залучається група реальних користувачів, які взаємодіють із системою та надають зворотний зв'язок щодо її функціональності та ергономіки. Цей підхід подібний до випуску альфа-версії програмного забезпечення. Результати UAT є важливими для визначення інтересів користувачів та планування подальших оновлень системи, що робить його невід'ємною частиною підтримки програмного забезпечення.

5.1.2. Результати та висновки тестування

Кожен з вищезазначених видів тестування був виконаний на різних етапах розробки додатка. Модульні тести сприяли глибокому розумінню алгоритмічної поведінки компонентів. Тестування на валідацію підтвердило відповідність системи вимогам, визначеним зацікавленими сторонами. Інтеграційне тестування значно підвищило ефективність процесу збірки системи.

Для виявлення дефектів було використано значну кількість тестових випадків, що дозволило покращити механізми обробки помилок. Для отримання зворотного зв'язку та подальшого розвитку додатка, він був наданий деяким кінцевим користувачам. Висловлюється подяка всім

користувачам, які взяли участь у тестуванні системи, сприяли виявленню помилок та надали цінні пропозиції щодо її вдосконалення.

Таблиця 5.1 - Застосування методів тестування до модулів

Тип тестування	Використовується в модулі
Модульне тестування	1. Модуль додавання курсів попередніх вимог 2. Модуль додавання основних курсів 3. Модуль сортування вибіркових курсів 4. Модуль додавання вибіркових курсів 5. Модуль додавання переваги ступеня
Тестування модулів	Модуль рекомендацій
Інтеграційне тестування	Модуль генерації/оновлення рекомендацій

5.2. Оцінка системи за допомогою користувацького зворотного зв'язку

Для валідації функціональності та зручності використання розробленої системи було проведено тестування із залученням реальних користувачів. Отриманий зворотний зв'язок дозволив ідентифікувати потенційні недоліки та підтвердити ефективність реалізованих рішень. Деталізований звіт про взаємодію з користувачами та отримані коментарі представлено в таблиці 5.2.

Результати взаємодії з користувачами та аналіз зворотного зв'язку

Таблиця 5.2 - Зведений звіт про користувачів та їхній зворотний зв'язок

	Статус	Коментар/зворотний зв'язок користувача	Виявлені проблеми та їх вирішення
User_1	Умовно прийнята	Шлях рекомендацій згенеровано правильно відповідно до моїх	**Проблема:** Користувач не зміг відновити доступ після забутого пароля, що вимагало ручного скидання через запит на оновлення

		переваг, які я надав в анкеті.	бази даних. **Вирішення:** Впроваджено функціональність для самостійного скидання пароля користувачем, що підвищило автономність та зручність використання.
User_2	Умовно прийнята	Відповідно до моїх поточних курсів шлях рекомендацій згенерував курси точно. Вибіркові курси були вибрані з урахуванням моїх переваг.	**Проблема:** Система припиняла функціонування після оновлення сторінки у браузері. Це було пов'язано з втратою системних даних (таких як C_ID та роль користувача), які очищалися під час перезавантаження. **Вирішення:** Інтегровано використання файлів cookie для постійного зберігання C_ID та ролі користувача. Ці дані встановлюються під час входу в систему та автоматично очищаються при виході, забезпечуючи стабільну роботу застосунку навіть при оновленні сторінки.

5.3. Test Case 1: Користувач має попередні вимоги та ще не пройшов жодних курсів

Цей випадок розглядає нового студента, який має попередні вимоги та ще не пройшов жодних курсів. Шлях буде згенеровано з першого семестру до останнього семестру. Спочатку всі попередні вимоги додаються до шляху, і залежно від семестру, в якому вони додаються, відповідний основний курс також додається до шляху. Останніми будуть вибіркові курси, які будуть вибрані на основі переваг, вказаних студентом у поданій анкеті. Приклад буде використаний для чіткої ілюстрації того, як генерується шлях рекомендацій, дотримуючись кроків A-D, описаних нижче.

Таблиця 5.3 - Інформація про тестовий випадок 1

Варіант ступеня	Проект
Початковий семестр	Осінь 2025
Потрібні попередні вимоги	P1 - пропонується в зимовому; -- потрібні для основного курсу C1 P2 - пропонується в зимовому, весняному; -- потрібні для основного курсу C2
Потрібні вибіркові курси Вибрані зі списку всіх можливих вибіркових курсів та переваг студента	E1 - пропонується в осінньому, весняному - на основі переваг: переваги днів: понеділок, вівторок, середа та четвер; переваги часу: ранок, вечір. E2 - пропонується в зимовому - на основі переваг: переваги днів: понеділок, вівторок, середа та четвер; переваги часу: ранок, вечір. E3 - пропонується в весняному - на основі переваг: переваги днів: понеділок, вівторок, середа та четвер; переваги часу: ранок, вечір. E4 - пропонується в осінньому, зимовому, весняному, літньому - на основі переваг: переваги днів: понеділок, вівторок, середа та четвер; переваги часу: ранок, вечір; переваги незалежного навчання: так E5 - пропонується в осінньому - на основі переваг: переваги днів: понеділок, вівторок, середа та четвер; переваги часу: ранок, вечір.
Потрібні основні курси Використовує щорічну пропозицію курсів для визначення, в яких семестрах викладається основний курс, та схематичний малюнок залежностей курсів для попередніх вимог	C1 - пропонується в осінньому, весняному; потребує P1 C2 - пропонується в осінньому; потребує P2 C3 - пропонується в зимовому; потребує P3 C4 - пропонується в зимовому; потребує P4 C5 - пропонується в весняному; потребує P5
Курс варіанту проекту Можна пройти лише після того, як всі основні курси пройдено, усний іспит здано, а пропозиція проекту представлена та затверджена	C690

ПРИМІТКА: E - вибірковий курс, C - основний курс, P - курс попередніх вимог.

5.3.1. Методологія генерації навчального плану

Розглянутий сценарій моделює процес побудови академічного плану для студента, який має невиконані курси попередніх вимог (P), але ще не

розпочинав навчання за основними (C) або вибірковими (E) курсами. Метою є створення інтегрованого навчального шляху від початкового до кінцевого академічного семестру, що враховує послідовність курсів, їхню доступність та індивідуальні переваги студента. Процес генерації плану деталізовано в наступних кроках A-D.

Крок A: Розподіл курсів попередніх вимог

Згідно з вихідними даними, студенту необхідно пройти два курси попередніх вимог: P1 та P2. Інформація про доступність цих курсів за семестрами отримана шляхом запиту до баз даних `course_schedule` та `quarter_offered`. Вибір конкретних курсів попередніх вимог зі списку необхідних здійснюється випадковим чином.

1. Розподіл P1: Курс P1 випадково обирається зі списку необхідних попередніх вимог. P1 є обов'язковою попередньою умовою для основного курсу C1. Таким чином, P1 повинен бути завершений до початку C1. З огляду на те, що P1 пропонується в зимовому семестрі, а C1 — в осінньому та весняному семестрах, оптимальним розподілом є включення P1 до зимового семестру 2026 року, а C1 — до весняного семестру 2026 року. Оскільки P1 є першим курсом, інтегрованим у рекомендований навчальний план, потенційні конфлікти розкладу відсутні, що дозволяє успішно додати P1 до зимового семестру 2026 року.

2. Розподіл P2: Наступна попередня вимога, P2, також обирається випадковим чином. P2 є попередньою вимогою для основного курсу C2, що зумовлює його проходження до C2. Враховуючи, що P2 пропонується в зимовому та весняному семестрах, а C2 — виключно в осінньому, оптимальним періодом для P2 є зимовий/весняний семестр 2026 року, з наступним проходженням C2 в осінньому семестрі 2026 року. У випадку, коли попередня вимога пропонується в кількох семестрах, обирається найраніший доступний семестр. Для P2 це зимовий семестр 2026 року. Розклад P2 на зимовий семестр 2026 року перевіряється на наявність

конфліктів з іншими курсами, вже включеними до навчального плану на цей семестр. Відсутність конфліктів дозволяє успішно додати P2 до рекомендованого плану. Після завершення розподілу всіх необхідних курсів попередніх вимог, процес переходить до Кроку B.

Крок B: Розподіл основних курсів

Основні курси випадково обираються зі списку "курсів, які ще потрібно пройти". Інформація про їхню доступність за семестрами також отримується з розкладу занять шляхом запиту до таблиць `course_schedule` та `quarter_offered`.

1. Розподіл C1: Основний курс C1 випадково обирається зі списку незавершених основних курсів. Оскільки P1 було додано до зимового семестру 2026 року, C1 планується на весняний семестр 2026 року. У поточному навчальному плані на весняний семестр 2026 року відсутні інші курси, тому C1 є першим доданим курсом, що виключає конфлікти розкладу. Таким чином, C1 успішно інтегрується в план.

2. Розподіл C3: Наступний основний курс, C3, обирається зі списку незавершених. C3 пропонується в зимовому семестрі, що робить зимовий семестр 2026 року оптимальним для його проходження. Розклад C3 на зимовий семестр 2026 року верифікується на відповідність існуючим розкладам у тому ж семестрі, і конфлікти не виявляються. Отже, C3 успішно додається до рекомендованого плану.

3. Розподіл C4: Основний курс C4 випадково обирається зі списку незавершених. C4 також пропонується в зимовому семестрі. Розклад C4 на зимовий семестр 2026 року перевіряється на наявність конфліктів з іншими курсами у тому ж семестрі, і конфлікти відсутні. Відповідно, C4 додається до плану.

4. Розподіл C5: Останній основний курс, C5, випадково обирається. C5 пропонується у весняному семестрі. Оптимальним семестром для його проходження є весняний семестр 2026 року. Розклад C5 на весняний семестр

2026 року верифікується на наявність конфліктів з іншими курсами у тому ж семестрі, і конфлікти не виявляються. Відповідно, C5 додається до плану.

Після успішного розподілу всіх основних курсів процес переходить до Кроку C.

Крок C: Розподіл вибіркових курсів

Для даного сценарію, оскільки варіант ступеня студента — "Проект", необхідно пройти п'ять вибіркових курсів. Вибіркові курси фільтруються відповідно до наданих студентом переваг, які включають: бажані дні тижня, бажаний час занять, переваги щодо літніх курсів та переваги щодо незалежного навчання.

1. Розподіл E1: Курс E1 випадково обирається зі списку вибіркових курсів, які необхідно пройти. E1 пропонується в осінньому та весняному семестрах. Розклад E1 на осінній семестр 2025 року перевіряється на наявність конфліктів з іншими курсами, вже включеними в план на цей семестр. Відсутність конфліктів дозволяє додати E1 до плану.

2. Розподіл E2: Наступний вибірковий курс, E2, випадково обирається. E2 пропонується в зимовому семестрі. Розклад E2 на зимовий семестр 2026 року перевіряється на наявність конфліктів з іншими курсами в тому ж семестрі. Конфлікти відсутні, тому E2 додається до плану.

3. Розподіл E3: Вибірковий курс E3 випадково обирається. E3 пропонується у весняному семестрі. Розклад E3 на весняний семестр 2026 року верифікується на відповідність існуючим розкладам у тому ж семестрі, і конфлікти не виявляються. Отже, E3 додається до плану.

4. Розподіл E4: Вибірковий курс E4 випадково обирається. E4 пропонується в осінньому, зимовому, весняному та літньому семестрах. Розклад E4 на осінній семестр 2025 року перевіряється на наявність конфліктів з іншими курсами, вже включеними в план на цей семестр. Конфлікти відсутні, тому E4 додається до плану.

5. Розподіл E5: Останній вибірковий курс, E5, випадково обирається. E5 пропонується в осінньому семестрі. Розклад E5 на осінній семестр 2025 року перевіряється на наявність конфліктів з іншими курсами, вже включеними в план на цей семестр. Виявлено конфлікт: час початку E5 (18:00) збігається з часом початку C2 (18:00). Оскільки C2 є основним курсом і доступний лише в осінньому семестрі, його перепланування неможливе. У зв'язку з цим, розклад E5 перевіряється на наступний доступний осінній семестр (Осінь 2026 року) на наявність конфліктів з курсами, вже внесеними до рекомендованого плану на цей період. Конфлікти відсутні, тому E5 додається до плану на осінній семестр 2026 року.

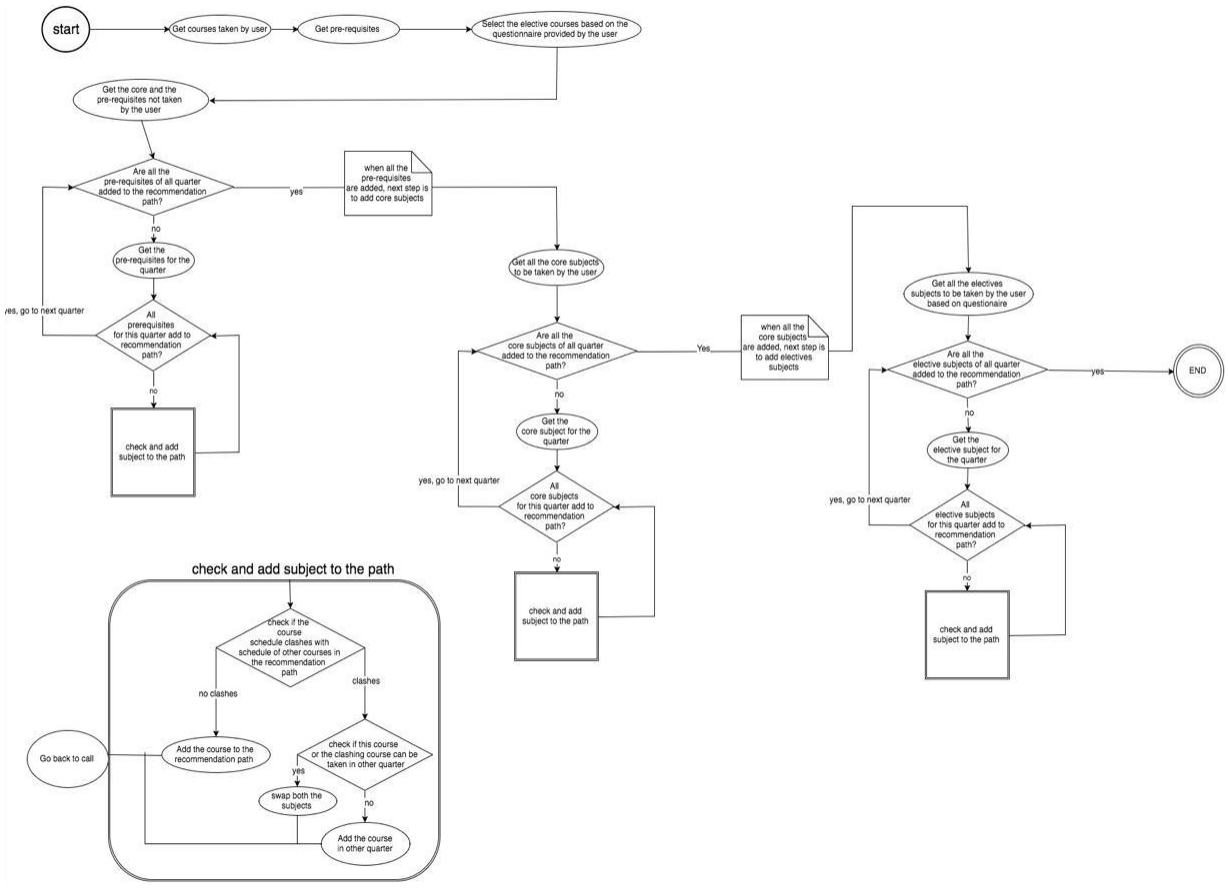

Рисунок 5.1 - Ілюстрація рекомендованого навчального плану для студента з попередніми вимогами без пройдених курсів

Після успішного розподілу всіх вибіркових курсів, процес переходить до Кроку D.

Крок D: Додавання курсу варіанта ступеня

Осінній семестр 2026 року є кінцевим семестром, до якого додається останній основний курс, C2. Оскільки обраний студентом варіант ступеня — "Проект", курс C690 має бути включений до плану. Цей курс планується на зимовий семестр 2027 року, що забезпечує його проходження після успішного завершення C2.

5.4. Test Case 2: користувач має попередні вимоги та пройшов деякі курси

Цей випадок розглядає студента, якому потрібно пройти попередні вимоги та який вже пройшов деякі курси. Шлях буде згенеровано з поточного семестру до останнього семестру. Спочатку всі попередні вимоги додаються до шляху, і залежно від семестру, в якому вони додаються, відповідний основний курс також додається до шляху. Останніми будуть вибіркові курси, які будуть вибрані на основі переваг у анкеті, поданій студентом. У цьому випадку до шляху додаються лише ті курси, які ще не пройдені студентом. Приклад буде використаний для чіткої ілюстрації того, як генерується шлях рекомендацій, дотримуючись кроків A-D, описаних нижче.

Таблиця 5.4 - Інформація про тестовий випадок 2

Варіант ступеня	**Дисертація**
Початковий семестр	Весна 2025
Пройдені курси	E1 - пройдено у весняному 2025
	E2 - пройдено у весняному 2025
Потрібні попередні вимоги	P3 - пропонується в зимовому;
	-- потрібні для основного курсу C3
Потрібні вибіркові курси	E3 - пропонується в зимовому,
Вибрані зі списку всіх	- на основі переваг: переваги днів:

можливих вибіркових курсів	понеділок, вівторок, середа,
та переваг студента	четвер та п'ятниця; переваги часу:
	ранок, вечір; переваги лекцій:
	лекція та лабораторія.
	E4 - пропонується в осінньому,
	- на основі переваг: переваги днів:
	понеділок, вівторок, середа,
	четвер та п'ятниця; переваги часу:
	ранок, вечір; переваги лекцій:
	лекція та лабораторія.
Потрібні основні курси	C1 - пропонується в осінньому, весняному; потребує P1
Використовує щорічну пропозицію курсів для визначення, в яких семестрах викладається основний курс, та схематичний малюнок залежностей курсів для попередніх вимог, необхідних для основного курсу	C2 - пропонується в осінньому; потребує P2 C3 - пропонується в зимовому; потребує P3 C4 - пропонується в зимовому; потребує P4 C5 - пропонується в весняному; потребує P5
Курс варіанту дисертації Можна пройти лише після того, як всі основні курси пройдено, пропозиція дисертації представлена та затверджена	C699

5.4.1. Методологія генерації навчального плану

Розглянутий сценарій демонструє процес формування академічного плану для студента, який має невиконану одну попередню вимогу (P) та вже пройшов деякі основні (C) та вибіркові (E) курси. Метою є побудова інтегрованого навчального шляху, що враховує послідовність курсів, їхню доступність, індивідуальні переваги студента та вже пройдені дисципліни. Детальний алгоритм генерації плану описано в кроках A-D.

Крок A: Розподіл курсів попередніх вимог

Згідно з вихідними даними, студенту необхідно пройти лише одну попередню вимогу — P3. Інформація про доступність цього курсу за семестрами отримується шляхом запиту до баз даних `course_schedule` та `quarter_offered`.

Курс Р3 є обов'язковою попередньою умовою для основного курсу С3. Відповідно, Р3 повинен бути завершений до початку С3. Оскільки Р3 пропонується в зимовому семестрі, а С3 також доступний у зимовому семестрі, оптимальним розподілом є включення Р3 до зимового семестру 2026 року, а С3 — до зимового семестру 2027 року. Оскільки Р3 є першим курсом, що додається до рекомендованого навчального плану, потенційні конфлікти розкладу відсутні, що дозволяє успішно інтегрувати Р3 до зимового семестру 2026 року. Після завершення розподілу всіх необхідних курсів попередніх вимог, процес переходить до Кроку В.

Крок В: Розподіл основних курсів

Основні курси обираються зі списку "курсів, які ще потрібно пройти". Інформація про їхню доступність за семестрами також отримується з розкладу занять шляхом запиту до таблиць `course_schedule` та `quarter_offered`.

1. Розподіл С1: Курс С1 пропонується в осінньому та весняному семестрах. Розклад С1 на осінній семестр 2025 року перевіряється на наявність конфліктів з іншими курсами, вже включеними в план на цей семестр. Відсутність конфліктів дозволяє успішно додати С1 до плану.

2. Розподіл С2: Наступний основний курс, С2, випадково обирається зі списку незавершених. С2 пропонується в осінньому семестрі. Розклад С2 на осінній семестр 2025 року верифікується на відповідність існуючим розкладам у тому ж семестрі. Конфлікти не виявлені, тому С2 успішно додається до плану.

3. Розподіл С3: Основний курс С3 випадково обирається зі списку незавершених. С3 пропонується в зимовому семестрі. Оскільки Р3 було додано до зимового семестру 2026 року, зимовий семестр 2027 року є оптимальним для проходження С3. Розклад С3 на зимовий семестр 2027 року перевіряється на наявність конфліктів з іншими курсами, вже включеними в

план на цей семестр. Відсутність конфліктів дозволяє успішно додати С3 до рекомендованого плану.

4. Розподіл С4: Наступний основний курс, С4, випадково обирається. С4 пропонується в зимовому семестрі. Розклад С4 на зимовий семестр 2026 року перевіряється на наявність конфліктів з іншими курсами, вже включеними в план на цей семестр. Конфлікти відсутні, тому С4 успішно додається до плану.

5. Розподіл С5: Останній основний курс, С5, випадково обирається. С5 пропонується у весняному семестрі. Оптимальним семестром для його проходження є весняний семестр 2026 року. Розклад С5 на весняний семестр 2026 року верифікується на наявність конфліктів з іншими курсами, вже включеними в план на цей семестр. Конфлікти не виявлені, тому С5 успішно додається до плану.

Після успішного розподілу всіх основних курсів, процес переходить до Кроку С.

Крок С: Розподіл вибіркових курсів

Для даного сценарію, оскільки варіант ступеня студента — "Дисертація", необхідно пройти чотири вибіркові курси. Згідно з наданою інформацією, студент вже пройшов два вибіркові курси, отже, залишається додати ще два вибіркові курси. Е3 та Е4 є вибірковими курсами, які необхідно включити до рекомендованого плану.

Вибіркові курси фільтруються на основі наданих студентом переваг. Спочатку застосовується фільтрація за перевагами днів. Якщо кількість відфільтрованого списку (L1) перевищує необхідну кількість вибіркових курсів (2), то до L1 застосовується додаткова фільтрація за перевагами часу. Якщо й після цього кількість відфільтрованого списку (L2) залишається більшою за 2, то до L2 застосовується фільтрація за перевагами типу лекцій.

1. Розподіл Е3: Курс Е3 пропонується в зимовому семестрі. Розклад Е3 на зимовий семестр 2026 року перевіряється на наявність конфліктів з

іншими курсами, вже включеними в план на цей семестр. Конфлікти відсутні, тому E3 успішно додається до плану.

2. Розподіл E4: Наступний вибірковий курс, E4, випадково обирається зі списку незавершених вибіркових курсів. E4 пропонується в осінньому семестрі. Розклад E4 на осінній семестр 2025 року перевіряється на наявність конфліктів з іншими курсами, вже включеними в план на цей семестр. Конфлікти відсутні, тому E4 успішно додається до плану.

Після успішного розподілу всіх вибіркових курсів, процес переходить до Кроку D.

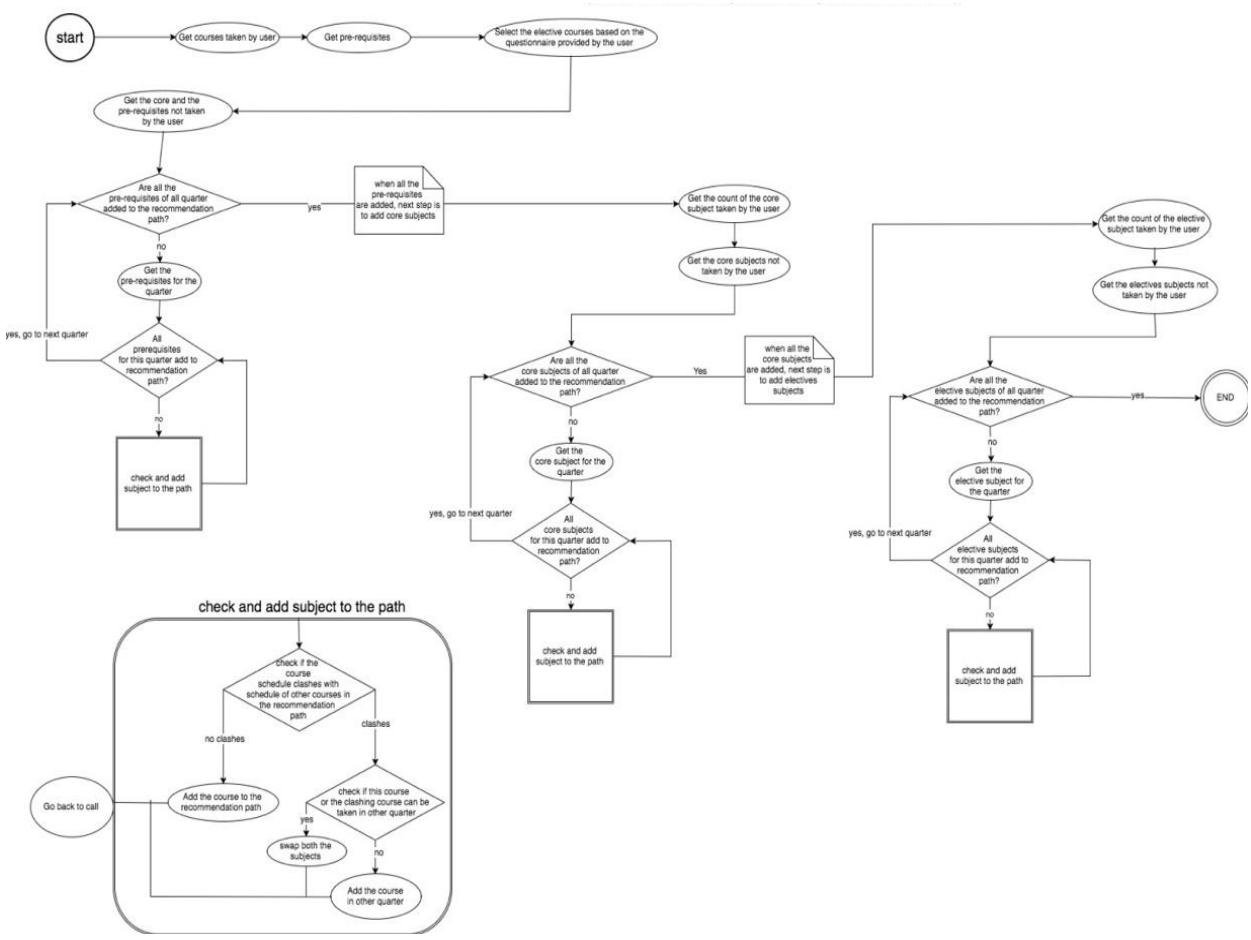

Рисунок 5.2 - Ілюстрація рекомендованого навчального плану для студента з попередніми вимогами та частково пройденими курсами (основні + вибіркові)

Крок D: Додавання курсу варіанта ступеня

Зимовий семестр 2027 року є кінцевим семестром, до якого додається останній основний курс, С3. Оскільки обраний студентом варіант ступеня — "Дисертація", курс С 699 має бути включений до плану. Цей курс планується на весняний семестр 2027 року, що забезпечує його проходження після успішного завершення С3.

Представлена система рекомендацій призначена для оптимізації процесу формування індивідуальних навчальних планів для студентів вищих навчальних закладів. Вона автоматично генерує послідовність курсів, які необхідно пройти, базуючись на таких ключових параметрах:

- Успішно завершені курси: Враховується академічна історія студента.

- Невиконані попередні вимоги: Ідентифікуються необхідні курси-передумови для подальшого навчання.

- Індивідуальні переваги студента: Аналізуються дані, надані студентом у спеціально розробленій анкеті (наприклад, бажаний розклад, формат навчання, інтереси до певних галузей).

- Згенерований навчальний план слугує дорожньою картою для студента, спрощуючи процес планування навчання до моменту випуску.

Традиційно, академічні консультанти витрачають значний обсяг часу на ручне визначення оптимальних курсів для кожного студента на наступний навчальний квартал. Дана система покликана автоматизувати цей трудомісткий процес, забезпечуючи швидку та точну генерацію рекомендованих навчальних траєкторій. Впровадження цієї автоматизації дозволяє істотно економити час як для студентів, так і для консультантів, одночасно знижуючи рівень стресу та робочого навантаження для обох сторін.

ВИСНОВКИ

У результаті виконання дипломної роботи на тему «Веб-застосунок для системи рекомендацій навчальних курсів» було вирішено низку теоретичних, аналітичних та практичних завдань, спрямованих на створення функціональної, персоналізованої та масштабованої системи підтримки вибору навчальних курсів для користувачів з різними освітніми цілями.

У першому розділі проведено аналіз предметної області та обґрунтовано актуальність розробки системи рекомендацій курсів в умовах зростаючого обсягу цифрового освітнього контенту. Визначено основні проблеми, зокрема складність у виборі релевантного навчального контенту та недостатню персоналізацію сучасних платформ. Окреслено мету проєкту — створення веб-застосунку, здатного забезпечити адаптивне формування індивідуального навчального шляху для користувача.

У другому розділі охарактеризовано вибір технологічного стеку, що включає Node.js, Express, PostgreSQL, а також пакети, необхідні для реалізації бізнес-логіки, безпеки, комунікацій та тестування. Детально описано структуру бази даних і принципи організації її таблиць, що забезпечують цілісність, узгодженість і масштабованість системи.

Третій розділ було присвячено моделюванню архітектури системи засобами DFD та UML. Створено діаграми потоків даних, варіантів використання та послідовності, що дозволило формалізувати вимоги до функціонування системи, забезпечити чітке розмежування ролей користувачів (студент, консультант, адміністратор) та відобразити логіку обробки запитів.

У четвертому розділі розроблено алгоритм генерації рекомендацій курсів, що базується на поєднанні принципів контентно-орієнтованої та фільтраційної логіки. Запропонований підхід враховує попередній досвід користувача, його освітні потреби та індивідуальні преференції. Здійснено

візуалізацію сценаріїв взаємодії користувачів із системою, що сприяє підвищенню юзабіліті веб-застосунку.

У п’ятому розділі проведено тестування програмного забезпечення із застосуванням наскрізного тестування, юзер-тестування та аналізу зворотного зв’язку від користувачів. Оцінено ефективність рекомендацій на прикладі різних сценаріїв використання системи. Результати тестування підтвердили правильність реалованої логіки рекомендацій, стабільність функціонування системи та позитивну реакцію цільової аудиторії.

Таким чином, поставлену мету дипломної роботи досягнуто. Було розроблено веб-застосунок, який відповідає сучасним вимогам до адаптивних освітніх платформ, забезпечує персоналізовані рекомендації навчальних курсів, підтримує автентифікацію користувачів, інтегрує ефективну бізнес-логіку та гарантує безперебійну роботу в умовах реального використання. Перспективами подальших досліджень і розробок є впровадження гібридних моделей машинного навчання, розширення системи аналітики успішності користувачів та інтеграція з зовнішніми освітніми платформами.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 734-749.
2. Al-Badarneh, R. A., & Al-Ta'ani, A. M. (2018). An intelligent recommender system for academic course selection. International Journal of Advanced Computer Science and Applications, 9(9), 118-124.
3. Al-Hmouz, A., & Al-Tarawneh, A. (2013). A framework for a university course recommendation system based on data mining. International Journal of Computer Science and Network Security, 13(8), 101-106.
4. Beck, K., & Andres, C. (2004). Extreme Programming Explained: Embrace Change (2nd ed.). Addison-Wesley Professional.
5. Bruegge, B., & Dutoit, A. H. (2010). Object-Oriented Software Engineering: Using UML, Patterns, and Java (3rd ed.). Prentice Hall.
6. Choudhury, N., & Das, S. (2020). Course recommendation system using machine learning. International Journal of Recent Technology and Engineering, 8(5), 1830-1834.
7. Dix, A., Finlay, J., Abowd, G. D., & Beale, R. (2004). Human-Computer Interaction (3rd ed.). Prentice Hall.
8. Fowler, M. (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd ed.). Addison-Wesley Professional.
9. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.
10. Gasevic, D., Djuric, D., & Devedzic, V. (2007). Model-Driven Architecture and Ontology Development. Springer.

11. Gorla, S., & Al-Obaidi, H. (2021). A recommender system for academic advising: A survey. Journal of Applied Computer Science & Mathematics, 15(2), 5-14.
12. Guttman, R. H., & Maes, P. (1998). Agent-mediated integrative negotiation for retail electronic commerce. Journal of Retailing, 74(3), 431-448.
13. Han, J., Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques (3rd ed.). Morgan Kaufmann.
14. IEEE Computer Society. (2008). IEEE Standard for Extensible Markup Language (XML) Model Exchange File Format for Unified Modeling Language (UML). IEEE Std 1901-2008.
15. IBM. (n.d.). UML basics: An introduction to the Unified Modeling Language.
16. Jacobson, I., Booch, G., & Rumbaugh, J. (1999). The Unified Software Development Process. Addison-Wesley.
17. Jung, J., & Kim, H. (2012). A personalized course recommendation system based on learning styles. Journal of Educational Technology & Society, 15(1), 108-117.
18. Kumar, A., & Goyal, S. (2017). An intelligent course recommendation system for e-learning using collaborative filtering. International Journal of Computer Science and Engineering, 5(6), 51-57.
19. Lussier, A. (2019). Academic Advising: A Comprehensive Handbook (2nd ed.). Jossey-Bass.
20. Maalej, W., & Thurimella, A. (2011). Requirements engineering for recommender systems: A survey. Requirements Engineering, 16(3), 195-212.
21. McGregor, J. D., & Sykes, D. (2009). A Practical Guide to Feature-Driven Development. Addison-Wesley Professional.

22. Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), The Adaptive Web (pp. 325-341). Springer.
23. Pressman, R. S., & Maxim, B. R. (2019). Software Engineering: A Practitioner's Approach (9th ed.). McGraw-Hill Education.
24. Ramakrishnan, R., & Gehrke, J. (2003). Database Management Systems (3rd ed.). McGraw-Hill.
25. Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth." In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 210-217). ACM.
26. Suseendran, G., & Kumar, S. N. (2021). A comprehensive review on course recommendation systems. Journal of Ambient Intelligence and Humanized Computing, 12(7), 7055-7073.
27. Wang, Y., & Dong, R. (2010). Design and implementation of a course recommendation system based on association rules. In Proceedings of the 2010 International Conference on Computer and Communication Technologies in Agriculture Engineering (Vol. 1, pp. 297-300). IEEE.
28. Weng, C., & Lee, M. (2007). A personalized course recommendation system for e-learning environments. In Proceedings of the 2007 International Conference on Machine Learning and Cybernetics (Vol. 5, pp. 2736-2741). IEEE.
29. Рікер А. Рекомендаційні системи: повний посібник. – К.: Вид. дім «Освіта», 2020. – 416 с.
30. Швед В.І., Кириченко К.О. Методи побудови рекомендаційних систем у сфері онлайн-освіти // Інформаційні технології і засоби навчання. – 2021. – Т. 83, №1. – С. 98–112.
31. Ricci F., Rokach L., Shapira B. Recommender Systems Handbook. – 2nd ed. – New York: Springer, 2015. – 1003 p.

32. Adomavicius G., Tuzhilin A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions // IEEE Transactions on Knowledge and Data Engineering. – 2005. – Vol. 17(6). – P. 734–749.
33. ISO/IEC 25010:2011. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models.
34. Gulla J. A., Zhang L., Liu P. et al. The ICT-based recommender system for learning environments // Computers in Human Behavior. – 2017. – Vol. 67. – P. 657–669.

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи: “ Веб-застосунок для системи рекомендацій навчальних курсів ”

Обсяг пояснювальної записки: __79__ аркушів.

Дата закінчення роботи: __9 червня__ 2025 р.

Підпис студента _____