МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____202___ р.

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

ВЕБПЛАТФОРМА ДЛЯ ПЛАНУВАННЯ РОЗВИТКУ УНІВЕРСИТЕТСЬКОЇ СИСТЕМИ ОСВІТИ

Спеціальність 124 «Системний аналіз»

124 – КРМ – 607.21810225

Виконав студент 6-го курсу, групи 607
_____ В.О. Степанюк
« » _____2024 р.

Керівник: д-р пед. наук, професор
_____ О.П. Мещанінов
« » _____2024 р.

Миколаїв – 2024

Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Освітньо-кваліфікаційний рівень **магістр**
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)
Спеціальність **124 «Системний аналіз»**
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____20___ р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

Степанюку Владиславу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи магістра «Вебплатформа для планування розвитку університетської системи освіти».
Керівник роботи Мещанінов Олександр Павлович, д-р пед. наук, професор.
Затв. наказом Ректора ЧНУ ім. Петра Могили від «03» жовтня 2023 р. № 202
2. Строк подання студентом роботи 21 лютого 2024 р.
3. Вхідні (початкові) дані до роботи: аналіз потреб та вимог до університетської системи освіти з погляду адміністрації, викладачів та студентів; огляд існуючих вебплатформ для управління освітніми процесами та планування розвитку освіти. Очікуваний результат: розробка і впровадження вебплатформи, яка дозволяє ефективно планувати, аналізувати та адаптувати стратегії розвитку університетської освіти з урахуванням змінних потреб і тенденцій в освітньому середовищі. Платформа повинна надавати інструменти для збору та аналізу даних, планування освітніх програм і курсів, а також для моніторингу ефективності запроваджених змін.

4. Зміст пояснювальної записки (перелік питань, які потрібно розглянути):

– університетська система освіти;

– впровадження вебплатформи для планування та оптимізації управління розвитком університетської системи освіти;

– проектування та розробка вебплатформи, яка б дозволила університетському персоналу та студентам ефективно управляти освітніми ресурсами та планувати розвиток навчальних програм.

5. Перелік графічного матеріалу: презентація

6. Завдання до спеціальної частини: Захист від іонізуючих випромінювань.

7. Консультанти:

Розділ	**Прізвище, ініціали та посада консультанта**	**Підпис**
Спеціальна частина з охорони праці	д-р.біол.наук.,професор Л. І. Григор'єва	
Методична частина	д-р пед. наук, професор О.П. Мещанінов	

Керівник роботи __д-р пед. наук, професор. Мещанінов О.П.

(наук. ступінь, вчене звання, прізвище та ініціали)

_____

(підпис)

Завдання прийнято до виконання ___Степанюк В.О.

(прізвище та ініціали)

_____

(підпис)

Дата видачі завдання «__31__» _____ жовтня____ 2023 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи магістра

Тема: Вебплатформа для планування розвитку університецької системи освіти

№	Найменування роботи	Початок	Закінчення	Примітки
1	Визначення керівника і теми КРМ.	01.092023	30.09.2023	
2	Подання заяви на затвердження теми КРМ	03.10.2023	12.10.2023	
3	Отримання завдання на виконання КРМ	13.10.2023	10.11.2023	
4	Складання календарного плану на період виконання КРМ	11.11.2023	15.11.2023	
5	Огляд літератури за темою дослідження	16.11.2023	04.12.2023	
6	Проходження переддипломної практики, збір та аналіз матеріалів до КРМ	05.12.2023	25.12.2023	
7	Аналіз предметної області та розробка технічного завдання. Моделювання результатів	26.12.2023	12.01.2024	
8	Опис фахової частини КРМ	13.01.2024	25.01.2024	
9	Розробка спеціальної частини з охорони праці та методичної частини	26.01.2024	02.02.2024	
10	Попередній захист КРМ на засіданні комісії кафедри	03.03.2024	03.03.2024	
11	Коригування роботи за результатами	04.02.2024	06.02.2024	
12	Остаточне оформлення пояснювальної записки та слайдів доповіді для захисту	07.02.2024	09.02.2024	
13	Подання КРМ рецензенту	09.02.2024	10.02.2024	
14	Рецензування КРМ	11.02.2024	12.02.2024	
15	Подання КРМ, її електронної копії та інших документів (відгуку, рецензії) до захисту	16.02.2024	17.02.2024	
16	Захист КРМ перед екзаменаційною комісією (ЕК)	28.02.2024	28.02.2024	

Розробив студент Степанюк В.О.

(прізвище та ініціали) (*підпис*)

Керівник роботи д-р пед. наук, професор. Мещанінов О.П

(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«09» листопада 202 р

АНОТАЦІЯ

до кваліфікаційної роботи магістра

студента групи 607 ЧНУ ім. Петра Могили

Степанюка Владислава Олександровича

на тему: **“ВЕБПЛАТФОРМА ДЛЯ ПЛАНУВАННЯ РОЗВИТКУ УНІВЕРСИТЕТСЬКОЇ СИСТЕМИ ОСВІТИ”**

Актуальність даної роботи обумовлена потребою у сучасних інструментах для ефективного планування та адаптації університетських освітніх програм до швидко змінювальних вимог ринку праці та освітнього середовища. Розробка вебплатформи для планування розвитку університетської системи освіти дозволить підвищити якість навчання та адаптацію освітніх програм до потреб студентів та викладачів.

Обєкт дослідження – процес планування розвитку університетської системи освіти.

Предмет дослідження – розширення функціональних можливостей вебплатформи для планування розвитку університетської системи освіти.

Мета – підвищення ефективності процесу планування розвитку університетської освіти та дослідження використання веб технологій для планування університетської освіти.

Результатом дослідження стало створення вебплатформи, яка дозволяє адміністрації, викладачам та студентам ефективно планувати розвиток освітніх програм, курсів та проектів.

Робота включає аналіз потреб і вимог до системи, опис архітектури та технологій розробки, а також методики тестування та впровадження розробленого рішення. Робота складається з пяти розділів Загальний обсягроботи 105 сторінок. Магістерська кваліфікаційна робота містить один додаток, 20 рисунків, одну таблицю і посилання на 45 літературних джерел.

Ключові слова: вебплатформа, планування розвитку, університетська система освіти, управління освітніми ресурсами, адаптація освітніх програм.

ABSTRACT

to the master's qualification work by the student of the group 607 of Petro Mohyla Black Sea National University

Vladyslav Stepanyuk

“**WEB PLATFORM FOR PLANNING THE DEVELOPMENT OF UNIVERSITY EDUCATION SYSTEM**”

A relevance of this work is driven by the need for modern tools for effective planning and adaptation of university educational programs to the rapidly chЦanging demands of the labor market and educational environment. The development of a web platform for planning the development of the university education system will improve the quality of education and adapt educational programs to the needs of students and teachers.

An object of study is the process of planning the development of the university education system

A subject of study is expanding the functionality of a web platform for planning the development of the university education system.

A propose is to investigate the peculiarities of development and implementation of a web platform for planning the development of the university education system.

The result of the research was the creation of a web platform that allows the administration, faculty, and students to effectively plan the development of educational programs, courses, and projects.

Whis work includes an analysis of the needs and requirements of the system, a description of the architecture and development technologies, as well as testing methodologies and the implementation of the developed solution. The work consists of five sections. The total volume of the work is 105 pages. The master's qualification work contains one appendix, 20 figures, one tables, and references to 45 literary sources.

Key words: web platform, development planning, university education system, management of educational resources, adaptation of educational programs.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 3
ВСТУП 4
1. АНАЛІЗ ЗАДАЧІ РОЗВИТКУ УНІВЕРСИТЕТСЬКОЇ СИСТЕМИ ОСВІТИ 5
1.1 Вебплатформи 5
1.2 Обгрунтування розробки вебплатформ для університетів 8
1.3 Аналіз існуючих підходів до розвитку освітніх вебплатформ 10
1.4 Формулювання вимог до вебплатформи 14
1.5 Ідеї для Університету сьогодення 15
2. АРХІТЕКТУРА ТА ФУНКЦІОНАЛ ВЕБПЛАТФОРМИ 17
2.1 Огляд архітектурних рішень 17
2.2 Функціональні вимоги та модулі платформи 25
2.3 Системні та безпекові аспекти 29
2.4 Інтеграція з існуючими системами університету 33
2.5 Переваги та виклики у застосуванні вебплатформи 35
3.МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ВЕБПЛАТФОРМИ 36
3.1 Розробка інтерфейсу користувача 36
3.2 Принцип роботи клієнтської частини вебплатформи 41
3.3 Принцип роботи серверної частини вебплатформи 43
3.4 Тестування вебплатформи 45
3.5 Методи та концепції вдосконалення вебплатформи 48
ВИСНОВКИ 53
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 54
ДОДАТОК А Лістинг коду вебплатформи 59

ПЕРЕЛІК СКОРОЧЕНЬ

РРФСР – Російська Радянська Федеративна Соціалістична Республіка

ВЦВК – Всеросійський центральний виконавчий комітет

РНК – Рада народних комісарів

LMS – Learning Management System

AJAX – Asynchronous Javascript And XML

MOOCs – Massive Open Online Course

API – Application Programming Interface

HTTPS – Hypertext Transfer Protocol Secure

ВСТУП

У швидкоплинному світі цифрових технологій, інформація стала не лише основним ресурсом розвитку суспільства, а й ключовим елементом у системі освіти. Як і в економіці, де актуальна та достовірна інформація надає переваги на ринку, у сфері освіти, своєчасні та точні дані можуть значно покращити процес навчання та адміністрування. Сучасні університети стикаються з викликом інтегрування інформаційних потоків у свої освітні та управлінські процеси, що вимагає розробки ефективних інструментів для планування розвитку освітніх програм та управління ресурсами.

Розвиток вебтехнологій відкриває нові можливості для університетів, зокрема через створення спеціалізованих платформ, що дозволяють оптимізувати процеси планування та аналізу, забезпечуючи оперативне рішення задач та адаптацію до змінюваних умов. В контексті цих потреб та можливостей, виникає завдання створення вебплатформи для планування розвитку університетської системи освіти – основна тема дослідження цієї роботи.

Кваліфікаційна робота складається з вступу, п'яти основних розділів, що охоплюють теоретичні та практичні аспекти створення вебплатформи, і завершується висновками та перспективами її подальшого розвитку. У першому розділі висвітлено актуальність та необхідність розробки вебплатформи для освітнього сектору, у другому – теоретичні основи планування розвитку освіти, у третьому – аналізується технічна реалізація і інструменти, у четвертому – методичні напрацювання, а п'ятий розділ присвячений охороні праці.

1 АНАЛІЗ ЗАДАЧІ РОЗВИТКУ УНІВЕРСИТЕТСЬКОЇ СИСТЕМИ ОСВІТИ

1.1 Вебплатформи

Вебплатформи — це комплексні інтернет-рішення, які надають користувачам можливість виконувати різноманітні завдання через вебінтерфейс, використовуючи браузер як клієнтську частину. Ці системи зазвичай розробляються на основі клієнт-серверної архітектури і можуть включати інтеграцію з базами даних, використання хмарних технологій та надання доступу до різноманітних онлайн-ресурсів та сервісів [6].

У контексті університетської системи освіти, вебплатформи можуть виконувати широкий спектр функцій, від управління навчальним процесом до координації дослідницької роботи та співпраці з індустрією та громадськістю. Це може включати модулі для розкладу занять, електронного навчання (e-learning), студентських портфоліо, форумів, блогів, систем управління ресурсами, фінансового планування, аналізу даних та багато іншого.

Розробка ефективної вебплатформи для університетів передбачає глибоке розуміння не тільки технологічних аспектів, а й освітнього процесу, потреб та очікувань усіх зацікавлених сторін, включаючи студентів, викладачів, адміністративний персонал та зовнішніх партнерів.

Сучасний розвиток вебплатформ для університетів також вимагає врахування глобальних тенденцій в освіті, таких як цифрова трансформація, персоналізація навчального процесу, інтеграція штучного інтелекту та великих даних для кращого розуміння та прогнозування потреб університету та його студентів [40].

Вебплатформи для університетської освіти є інноваційними рішеннями, які трансформують традиційні підходи до навчання та управління. Вони мають свої переваги та недоліки, які важливо враховувати при розробці та впровадженні.

Переваги вебплатформ:

– доступність: вебплатформи забезпечують легкий доступ до навчальних матеріалів та адміністративних функцій з будь-якого місця та в будь-який час, де є інтернет-з'єднання;

– гнучкість: вони дозволяють студентам працювати за власним графіком, що може підвищити ефективність навчання;

– інтеграція: можливість інтеграції з іншими системами та інструментами, наприклад, з електронними бібліотеками, системами управління навчальним процесом тощо;

– масштабованість: вебплатформи легко масштабуються, що дозволяє університетам розширювати свої навчальні програми та курси без значних додаткових витрат;

– персоналізація: системи можуть бути налаштовані для забезпечення персоналізованого досвіду для кожного користувача.

Недоліки вебплатформ:

– залежність від технологій: висока залежність від стабільності інтернет-з'єднання та технічного обладнання;

– безпека даних: потреба в сильних заходах безпеки для захисту конфіденційності та інтегральності даних користувачів;

– відсутність особистого контакту: вебплатформи можуть зменшити безпосереднє спілкування між студентами та викладачами, що є важливим для деяких аспектів навчання;

– технічні проблеми: проблеми з сумісністю, баги в програмному забезпеченні, або помилки у використанні можуть перешкоджати навчальному процесу;

– опір змінам: не всі учасники освітнього процесу можуть бути готовими або відкритими до впровадження нових технологій.

Розуміння цих плюсів і мінусів є критично важливим для розробки вебплатформи, яка б максимізувала потенціал інтерактивного та інтегрованогонавчання, одночасно мінімізуючи ризики та перешкоди для користувачів.

Історія вебплатформ для університетської освіти бере свій початок з ранніх днів інтернету, коли основними ресурсами були статичні вебсторінки. Протягом останніх десятиліть, вони пройшли шлях від простих вебсайтів до складних інтегрованих систем, що підтримують широкий спектр академічних і адміністративних процесів.

У 1990-х роках, з появою перших систем управління навчанням (LMS) [7], університети почали активно використовувати технології для підтримки навчального процесу. Платформи, такі як Blackboard та Moodle, стали одними з перших, що надали викладачам інструменти для створення онлайн-курсів, управління контентом і взаємодії зі студентами.

Рисунок 1.1 – Схема LMS

З розвитком вебтехнологій в епоху 2000-х, з'явились більш динамічні та взаємодійні платформи, які дозволили більш глибоку персоналізацію навчального досвіду та підтримку колаборативної роботи. Розвиток AJAX та інших технологій веб-2.0 сприяв створенню більш респонсивних та інтуїтивно зрозумілих інтерфейсів [8].

В останнє десятиліття, розвиток хмарних технологій та великих даних відкрив нові можливості для аналізу та управління навчальними процесами. Сучасні платформи включають штучний інтелект та машинне навчання для покращення аналітики навчання, автоматизації процесів та персоналізації навчального досвіду.

Еволюція вебплатформ також відбувалася у відповідь на зміни в академічному світі, такі як зростання відкритих онлайн-курсів (MOOCs) та перехід до гібридних або повністю онлайн-програм. Це вимагало від систем освіти більшої гнучкості та адаптивності [23].

Останніми роками, пандемія COVID-19 спровокувала черговий стрибок у розвитку та впровадженні вебплатформ, змусивши університети шукати швидкі та ефективні рішення для підтримки віддаленого навчання. Це підтвердило необхідність розвитку вебплатформ як стійких та гнучких інструментів для забезпечення освітньої діяльності у будь-яких умовах.

Розглянувши історію та еволюцію вебплатформ, можна зрозуміти, як технологічний прогрес та зовнішні виклики формують потреби та можливості в галузі університетської освіти. Це знання є важливим для розуміння поточного стану справ та для прогнозування майбутнього розвитку вебплатформ.

1.2 Обгрунтування розробки вебплатформ для університетів

У сучасному освітньому просторі вебплатформи для університетів вже не є просто нововведенням; вони стали необхідністю. Ця потреба в цифровізації освіти є результатом постійного пошуку більш ефективних і доступних способів

навчання. Завдяки технологічному прогресу, університети мають унікальну можливість розширити свої освітні послуги за межі традиційного класу, пропонуючи студентам гнучкість і варіативність у виборі навчальних траєкторій.

Вебплатформи сприяють оптимізації адміністративних та навчальних процесів. З їх допомогою, університети можуть значно підвищити ефективність управління ресурсами, планування навчальних програм та оцінювання студентських досягнень. Ці системи також відіграють ключову роль у забезпеченні адекватної взаємодії між викладачами та студентами, підтримуючи неперервний обмін інформацією та ресурсами.

Інтеграція інноваційних методик навчання є ще однією значущою перевагою вебплатформ. Вони дозволяють впроваджувати новітні освітні підходи, включаючи змішане навчання, гейміфікацію та масові відкриті онлайн-курси, що збагачують навчальний досвід та сприяють кращому засвоєнню матеріалу студентами.

Міжнародна співпраця та обмін знаннями, які стають можливими через вебплатформи, відкривають двері до глобального освітнього співтовариства. Це не тільки збільшує можливості для студентів і викладачів, але й сприяє крос-культурному обміну та розвитку міжнародних наукових проектів. Така інтеграція є особливо важливою в контексті підготовки студентів до роботи в інтернаціональному середовищі.

Проте, розробка та впровадження вебплатформ несуть у собі певні виклики. Висока вартість розробки, потреба у постійному оновленні та технічній підтримці, а також ризики, пов'язані з безпекою даних, вимагають ретельного планування та управління. Важливо розуміти, що ці виклики не зменшують актуальності вебплатформ, але вони вказують на необхідність комплексного підходу до їх реалізації.

В кінцевому підсумку, актуальність розробки вебплатформ для університетів не може бути оспорена. Сучасна освітня дійсність вимагає адаптації та інновацій, а вебплатформи є ключовим інструментом для задоволення цих потреб. Вони

відіграють вирішальну роль у формуванні майбутнього освітнього середовища, що базується на доступності, гнучкості та інтерактивності [25].

1.3 Аналіз існуючих підходів до розвитку освітніх вебплатформ

1.3.1 Порівняльний аналіз існуючих рішень

Порівняльний аналіз існуючих вебплатформ для університетів виявляє ключові тенденції та відмінності між різними системами, даючи змогу зрозуміти, які функції та характеристики найкраще відповідають потребам освітніх закладів. При аналізі таких платформ, як Blackboard, Moodle, Canvas і Google Classroom, розглядаються такі аспекти, як користувацький інтерфейс, функціональність, інтеграція з іншими інструментами, масштабованість, вартість впровадження та підтримки, а також здатність системи адаптуватися до специфічних вимог університету [8].

На прикладі Blackboard [26], ця платформа часто вибирається за її широкий функціонал та інтеграцію з різноманітними інструментами та службами, але може бути відносно дороговизною з точки зору ліцензування та необхідності спеціалізованого персоналу для її підтримки. Moodle [27], з іншого боку, є відкрито-джерельною платформою, яка пропонує гнучкість та велику кількість плагінів, розроблених спільнотою, але може потребувати більше зусиль для налаштування та управління.

Canvas, як сучасна платформа, забезпечує інтуїтивно зрозумілий користувацький інтерфейс та міцну підтримку мобільних пристроїв, але її функціональність може бути обмеженою у порівнянні з більш естаблішментними системами [28]. Google Classroom відрізняється своєю простотою та тісною інтеграцією з Google Suite, пропонуючи низькі витрати на впровадження, однак вона може не відповідати всім потребам університетів, особливо коли мова йде про складніші освітні та адміністративні функції[29,30].

Крім того, важливо розглядати відгуки користувачів та випадки використання вебплатформ університетами, що дозволяє ідентифікувати як сильні сторони, так і потенційні слабкості кожного рішення. Враховуючи швидкість технологічних змін, регулярне оновлення аналізу є критично важливим для підтримання актуальності інформації, що сприяє обґрунтованому вибору вебплатформи для конкретних потреб університету.

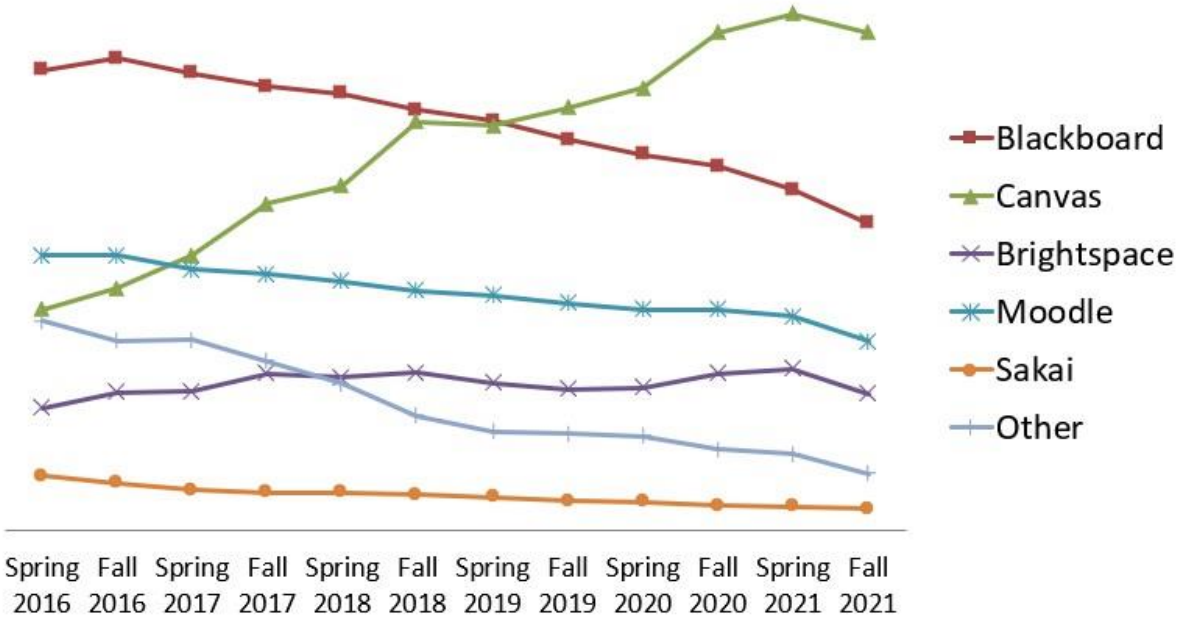

Рисунок 1.2 – Дані про частку ринку LMS

Такий порівняльний аналіз є необхідним етапом у процесі прийняття рішення про впровадження або оновлення вебплатформи, гарантуючи, що університет зможе забезпечити своїм студентам та персоналу доступ до найбільш ефективних та інноваційних освітніх інструментів.

1.3.2 Вимоги до сучасних вебплатформ

Сучасні вебплатформи для університетів повинні відповідати ряду вимог, щоб задовольняти потреби сучасної освіти та забезпечувати ефективне управління навчальними та адміністративними процесами. Перш за все, платформа має бути

інтуїтивно зрозумілою та легкою в обігу, щоб користувачі могли швидко адаптуватися до нового середовища без потреби в тривалому навчанні.

Масштабованість також є критично важливою, оскільки університети змінюються та розширюються, вебплатформа має можливість витримувати зростаючу кількість користувачів та розширювати свої функціональні можливості. Це включає не лише збільшення числа студентів і курсів, але й інтеграцію нових інструментів та сервісів.

Вебплатформа має бути гнучкою, щоб можна було легко інтегрувати з іншими системами, такими як електронні бібліотеки, системи відеоконференцій, інструменти співпраці та аналітики. Ця інтеграція повинна бути безшовною та не створювати додаткових труднощів для користувачів.

Щоб відповідати сучасним тенденціям, вебплатформи повинні бути орієнтовані на мобільність, надаючи користувачам змогу легко доступати ресурси та виконувати завдання через мобільні пристрої, такі як смартфони та планшети. Це особливо важливо для студентів, які все більше покладаються на мобільність у своєму навчанні.

Нарешті, з огляду на бюджетні обмеження багатьох університетів, важливо, щоб рішення були економічно ефективними, з прозорими вартісними моделями та низькими загальними витратами на впровадження та підтримку. Це дозволить університетам інвестувати в освіту, не перевантажуючи свої фінансові ресурси.

Врахування цих вимог під час вибору або розробки вебплатформи для університетів є ключовим для забезпечення того, що вони відповідатимуть поточним і майбутнім потребам освітнього середовища та забезпечуватимуть студентам та персоналу найкращі можливості для навчання та викладання [31].

1.3.3 Ключові функції та інструменти

Розробка вебплатформ для університетів вимагає інтеграції низки ключових функцій та інструментів, що забезпечують комплексне та ефективне управління

освітніми процесами. Однією з основних функцій є система управління навчальними матеріалами (LMS), яка дозволяє викладачам креативно та ефективно організовувати курси, розподіляти завдання та слідкувати за виконанням робіт студентами [41].

Ключовим інструментом є також система віртуальних класів, яка підтримує відеоконференції та онлайн-семінари, забезпечуючи можливість для викладачів та студентів проводити заняття в режимі реального часу, незалежно від їх географічного розташування. Це сприяє гнучкості навчального процесу та підтримує високий рівень взаємодії.

Для оцінювання студентських досягнень вебплатформа має включати інструменти для тестування та відстеження прогресу, що дозволяють викладачам створювати різноманітні форми контролю, від онлайн-квізів до комплексних іспитів. Важливим є також наявність аналітичних інструментів, які надають дані для аналізу ефективності навчальних методик та студентського залучення.

Для сприяння колаборації та спільної роботи, вебплатформа повинна включати інструменти соціального взаємодії, такі як форуми, групи та чати, які дозволяють студентам та викладачам легко спілкуватися та обмінюватися ідеями. Це також включає можливості для спільного редагування документів та управління проектами, що є важливим для групових завдань та дослідницьких проектів.

Нарешті, необхідною складовою є адаптивний дизайн, який забезпечує оптимальний перегляд та взаємодію з платформою на різних пристроях, включаючи настільні комп'ютери, ноутбуки, планшети та смартфони. Такий підхід забезпечує зручність користування і доступність, що є критично важливим для сучасних студентів, які все частіше покладаються на мобільні пристрої для освітніх потреб.

Інтеграція цих ключових функцій та інструментів є необхідною для створення вебплатформи, яка відповідає високим стандартам сучасної університетської освіти та задовольняє потреби викладачів та студентів в динамічному та інтерактивному навчальному середовищі.

1.4 Формулювання вимог до вебплатформи

Формулювання вимог до вебплатформи є вирішальним етапом у процесі її розробки та впровадження. Ці вимоги визначають, як система повинна функціонувати, які функціональні можливості вона повинна надавати, та яким чином вона взаємодіятиме з користувачами та іншими системами. Наступні критерії є ключовими при формулюванні вимог до сучасної вебплатформи для університетської освіти:

– функціональність: перелічіть та описайте конкретні функції та задачі, які вебплатформа повинна виконувати. це може включати управління курсами, оцінювання, колаборацію, обмін ресурсами, віртуальні класи тощо;

– зручність користування: вебплатформа повинна мати інтуїтивно зрозумілий та привабливий інтерфейс, що спрощує навігацію та забезпечує зручність користування для всіх категорій користувачів;

– масштабованість та продуктивність: система повинна ефективно працювати з великою кількістю користувачів та великим обсягом даних, забезпечуючи швидкий доступ до ресурсів та стабільну роботу під час пікових навантажень;

– безпека: вимоги до безпеки повинні включати захист даних, аутентифікацію користувачів, контроль доступу та шифрування, щоб забезпечити конфіденційність та цілісність інформації;

– інтеграція: вебплатформа повинна легко інтегруватися з іншими системами та інструментами, які вже використовуються в університеті, включаючи бази даних, засоби електронної пошти, програми для відеоконференцій тощо;

– адаптивність та доступність: система має підтримувати адаптивний дизайн, що забезпечує оптимальне відображення на різних пристроях, включаючи настільні комп'ютери, ноутбуки, планшети та смартфони, та бути доступною для користувачів з особливими потребами;

– підтримка та оновлення: визначте вимоги до технічної підтримки, плану оновлення системи та регулярних оглядів безпеки, щоб забезпечити її актуальність та відповідність сучасним вимогам.

Формулювання чітких, детальних та зрозумілих вимог є критично важливим для успішної розробки та впровадження вебплатформи, яка відповідає потребам університетської системи освіти та забезпечує високий рівень задоволення користувачів [39].

1.5 Ідеї для Університету сьогодення

Багато наукових праць присвячені вивченню основ організації університетською освітою та шляхів її поліпшення. Основними проблемами є відсутність інтересу до навчання серед значної частини студентів та збільшення тиску на них з боку адміністрації на різних рівнях. Важливим завданням є розробка нових методів об'єднання студентських поколінь в університетському середовищі для передачі не тільки знань, але й цінностей освіти, спадкоємності, прагнення до нового і поваги до пошуку невідомого.

Гарним прикладом ситуацій у кафедрах університетів, які детально описано в творі І. Грекової, що насправді є псевдонімом О.С. Вентцель (1907-2002 роки), у її повісті "Кафедра". Цей твір я настійливо раджу молодим викладачам як вартий уваги [1]. Також хочу зазначити про відсутність будь-яких дискусій щодо перейменування діючих інститутів в Україні на університети, що відбувалося без значних вимог чи обговорень. Це породжує питання про різницю між новими та традиційними університетами, і чому думка вчених у цьому питанні знову залишилася поза увагою. Подібно до історичних моментів, коли, незважаючи на обговорення в Москві під керівництвом Н.К. Крупської та А.В. Луначарського, університети в РРФСР були збережені у 20-ті роки XX століття, а п'ять університетів України були перейменовані в інститути за рішенням Я.П. Ряппо.

«Декретом ВЦВК і РНК РРФСР була заснована Державна комісія з народної освіти. Згідно з декретом, вона повинна була не управляти навчальними та освітніми установами, а «служити зв'язком і помічницею організувати джерела матеріальної, ідейної та моральної підтримки муніципальних і приватних, особливо ж трудовим і класовим освітнім установам в державному масштабі» [2]

У сучасних реаліях викладачі опиняються у складній ситуації, стикаючись з різними підходами до навчання та виховання студентів. Деякі з них намагаються стимулювати студентів, інші зосереджують увагу на роботі з кращими студентами, а дехто, на жаль, поділяє загальну байдужість. В таких умовах постає питання про якість університетської освіти.

Нагадаю слова Теофраста Парацельса, справжнє ім'я Філіп Ауреол Теофраст Бомбаст фон Гогенгайм (1493-1541 рр.), який стверджував, що дія речовини як отрути чи ліки залежить від дози. Це допомагає зрозуміти важливість балансу в будь-якому процесі, включно з освітою. Також варто згадати біологічний закон Арндта-Шульца [3] або феномен гормезису, який підкреслює, що реакція живих систем залежить від сили подразника, підтверджуючи значення міри та балансу у всьому, включаючи підходи до навчання та виховання.

Отже, кожен із викладачів обирає власний алгоритм протидії, поширенню академічній не доброчесності, як прояв реальної, а не декларативної академічної свободи[4].

За словами О.П. Мещанінова, вдосконалення університетської освіти є ключовою стратегічною задачею, важливою для державної політики та національної безпеки. Створення нового типу держави можливе лише за умови наявності висококваліфікованих національних кадрів, здатних працювати на благо національних інтересів, та які володіють новим світоглядом і готові до життя у свободі. Основними напрямками розвитку української університетської освіти стають виховання та формування глибокого світогляду, демократизація особистості та її адаптація до умов вільного суспільства [5].

2 АРХІТЕКТУРА ТА ФУНКЦІОНАЛ ВЕБПЛАТФОРМИ

2.1 Огляд архітектурних рішень

2.1.1 Модель клієнт-сервер

Модель клієнт-сервер є основним архітектурним рішенням для вебплатформи, що розробляється для планування розвитку університетської системи освіти. Ця модель визначає розподілену структуру системи, де клієнт та сервер взаємодіють для забезпечення функціональності.

Основні аспекти моделі клієнт-сервер.

Клієнтська частина:

– вебінтерфейс: забезпечує зручний та інтуїтивний інтерфейс для користувачів, де вони можуть здійснювати взаємодію з системою;

– аутентифікація та авторизація: відповідає за перевірку ідентифікації користувачів та надання їм необхідних прав доступу.

Серверна частина:

– бізнес-логіка: містить логіку обробки запитів, виконання бізнес-процесів та забезпечення функціональних можливостей платформи;

– база даних: зберігає та управляє даними, необхідними для функціонування системи.

Переваги моделі клієнт-сервер:

– розподіленість: забезпечує розділення відповідальностей між клієнтом та сервером, що полегшує розширення та підтримку системи;

– масштабованість: дозволяє гнучко масштабувати окремі компоненти системи в залежності від потреб.

Використані Технології:

– Frontend: використання сучасних вебтехнологій, таких як HTML, CSS, та JavaScript (зокрема, фреймворки типу React або Vue.js);

– Backend: використання мов програмування, наприклад Python, та фреймворків, таких як Django або Flask.

Модель клієнт-сервер забезпечує ефективну взаємодію між користувачами та системою, дозволяючи забезпечити високу продуктивність та зручність використання вебплатформи для університетської освіти [24].

2.1.2 Компонентна архітектура

Компонентна архітектура є ключовим аспектом розробки вебплатформи для планування розвитку університетської системи освіти. Цей підхід передбачає розгляд системи як набору взаємодіючих компонентів, кожен із яких відповідає за конкретну функціональність. Розглянемо основні аспекти компонентної архітектури:

Модульність та Розширюваність:

Кожен компонент в системі виконує конкретну роль і має чітко визначені інтерфейси для взаємодії з іншими компонентами. Це дозволяє легко розширювати та модифікувати систему, додаючи або змінюючи окремі компоненти без впливу на решту системи.

Розподілена Навантаженість:

Компоненти можуть бути розподілені на різних серверах або областях, що сприяє ефективній роботі системи при великому навантаженні. Ця розподіленість дозволяє оптимізувати роботу кожного компонента та підтримувати високу продуктивність.

Незалежність Інтеграції:

Кожен компонент може бути розроблений та підтримуватися незалежно. Це дозволяє використовувати різні технології та мови програмування для кожного компонента, що є важливим аспектом для гнучкості системи.

Підтримка Легкості Тестування:

Компонентна архітектура сприяє проведенню тестів окремих компонентів незалежно один від одного. Це полегшує виявлення та виправлення помилок, а також забезпечує високу якість та стабільність системи.

Компонентна архітектура визначає структуру системи, де кожен компонент виконує конкретну функцію та взаємодіє з іншими компонентами для досягнення загальної мети вебплатформи.

2.1.3 Вибір технологій для вебплатформи

В процесі розробки вебплатформи для планування розвитку університетської системи освіти були обрані ключові технології, що визначають якість та функціональність системи. Розглянемо основні аспекти вибору технологій:

Клієнтська частина додатку, розроблена з використанням **JavaScript** та фреймворку **React.js**, становить основу для інтерфейсу користувача (UI).

JavaScript і React.js.

JavaScript є мовою програмування, яка використовується для створення інтерактивних елементів на вебсторінках. Вона дозволяє розробникам створювати динамічні вебдодатки, де користувач може взаємодіяти з інтерфейсом без необхідності перезавантажувати сторінку. React.js — це бібліотека, розроблена Facebook, для побудови інтерфейсів користувача, особливо великих застосунків, де необхідна швидка взаємодія без перезавантаження сторінки [11].

React використовує компонентний підхід, де UI розбивається на незалежні, повторно використовувані частини, які управляються їхнім власним станом. Це дозволяє розробникам легше управляти інтерфейсом та розробляти складні інтерфейси, роблячи код більш читабельним і легким для підтримки.

React використовує JSX, синтаксис, який дозволяє писати елементи HTML у JavaScript. JSX робить код компонентів більш зрозумілим та спрощує процес створення інтерфейсу користувача, дозволяючи розробникам вставляти HTML код безпосередньо в JavaScript.

У React, стан (state) та властивості (props) використовуються для управління даними компонентів. Стан дозволяє компонентам реагувати на зміни даних та користувальницькі дії, оновлюючи UI відповідно. Властивості дозволяють передавати дані між компонентами, роблячи компоненти більш гнучкими та повторно використовуваними.

Рисунок 2.1 – Основні технології UI

Клієнтська частина часто взаємодіє з сервером для отримання або відправлення даних. React може використовувати такі API, як Fetch API або Axios, для створення HTTP-запитів до серверної частини. Це дозволяє додаткам завантажувати дані асинхронно без перезавантаження сторінки, забезпечуючи швидку та плавну взаємодію користувача [9,10].

Для організації навігації у великих односторінкових застосунках (SPA) використовується маршрутизація. React Router є стандартною бібліотекою для маршрутизації у React додатках, що дозволяє управляти переходами між різними частинами додатку без перезавантаження сторінки.

React та його екосистема включають інструменти та методи для зведення до мінімуму та оптимізації додатків, що дозволяє покращити швидкодію та зменшити час завантаження, особливо важливо для мобільних пристроїв і повільних інтернет-з'єднань.

Клієнтська частина додатку, створена на основі цих принципів та технологій, забезпечує багатий інтерфейс користувача, який є інтуїтивно зрозумілим, швидким та зручним для взаємодії.

Серверна частина додатку, розроблена з використанням C# та платформи .NET 6 [12], є ключовою для створення бекенду, який відповідає за логіку обробки даних, безпеку, взаємодію з базами даних та іншими сервісами. Ось детальніший опис того, як ці технології використовуються у серверній частині:

.NET 6 — це остання версія крос-платформеного фреймворку від Microsoft, яка підтримує розробку високопродуктивних вебдодатків та сервісів. .NET 6 включає підтримку розробки веб-API за допомогою ASP.NET Core, яке є сучасною, високопродуктивною та модульною платформою для будівництва вебдодатків та сервісів.

ASP.NET Core є основою для створення веб-API та вебдодатків у .NET 6. Воно надає розробникам набір інструментів та бібліотек, які дозволяють легко створювати масштабовані, безпечні вебдодатки. ASP.NET Core підтримує розробку MVC (Model-View-Controller), RESTful API, реалізацію аутентифікації та авторизації, а також взаємодію з базами даних [13].

У ASP.NET Core, middleware використовується для побудови конвеєра обробки HTTP-запитів, де кожен компонент middleware може виконувати операції перед або після наступного компонента в конвеєрі. Це дозволяє реалізовувати різноманітну функціональність, включаючи обробку помилок, логування, аутентифікацію та багато іншого.

Entity Framework Core — це ORM (Object-Relational Mapping) фреймворк, який дозволяє розробникам працювати з базами даних, використовуючи .NET об'єкти, без необхідності писати прямий SQL-код. EF Core спрощує доступ до даних та управління ними, автоматизуючи багато аспектів взаємодії з базою даних [14].

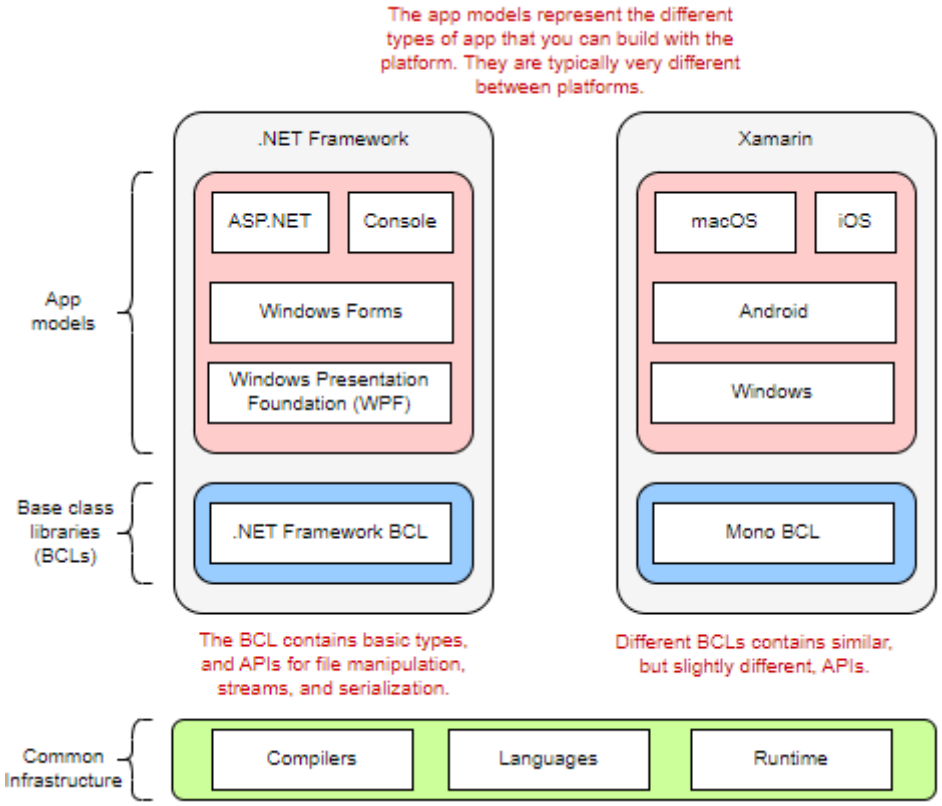

Рисунок 2.2 – Шари, що складають .NET Framework

ASP.NET Core надає потужні та гнучкі можливості для реалізації аутентифікації та авторизації, дозволяючи захищати доступ до вебдодатку або API. Розробники можуть використовувати різні схеми аутентифікації, включаючи JWT (JSON Web Tokens), OAuth та інші.

Для документації **API** та спрощення процесу тестування, .NET 6 дозволяє легко інтегрувати Swagger / OpenAPI, який автоматично генерує інтерактивну документацію для веб-API. Це дозволяє розробникам та зацікавленим сторонам легко взаємодіяти з API, тестувати його можливості та розуміти його специфікації.

Серверна частина додатку на .NET 6 та C# є міцною основою, яка забезпечує не тільки обробку даних та логіку додатку, але й гарантує безпеку, ефективність та масштабованість вебсервісів.

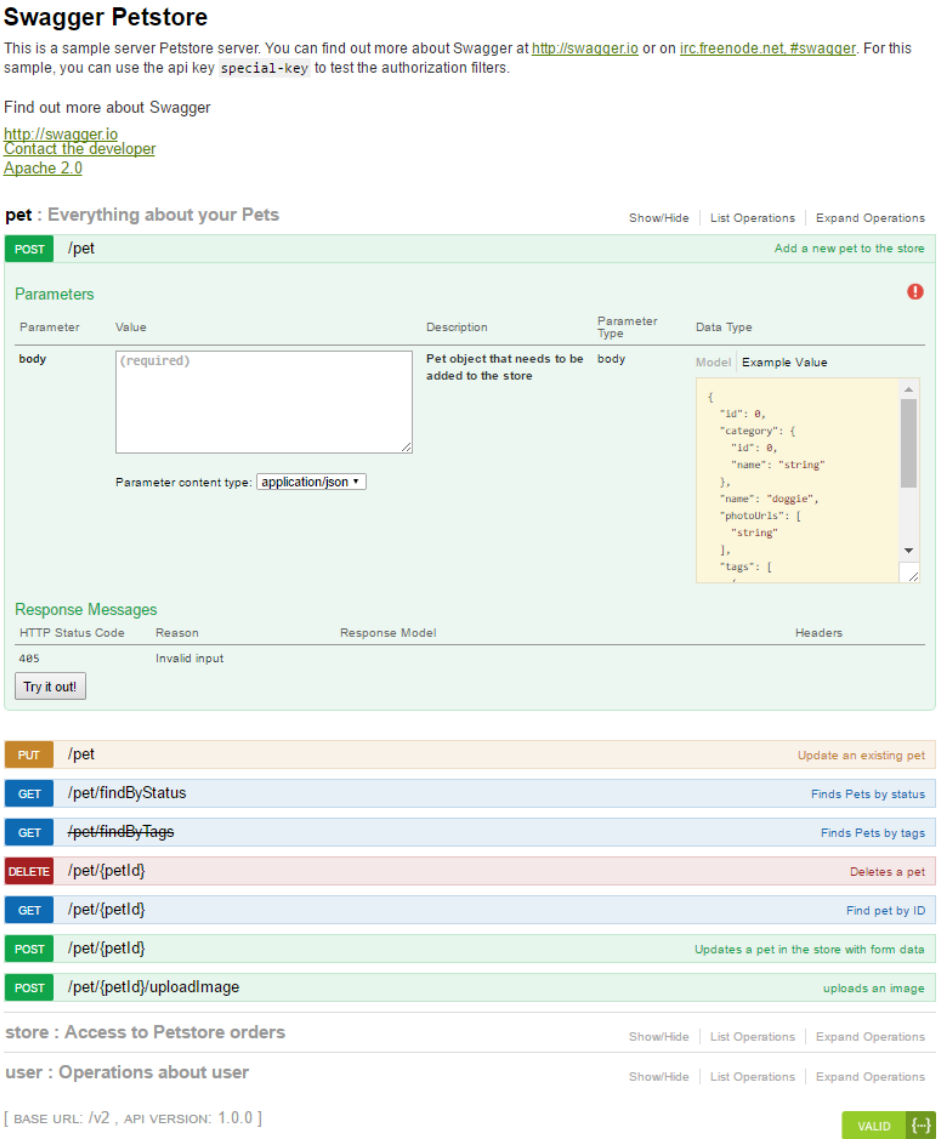

Рисунок 2.3 – Приклад роботи Swagger.

Swagger – це набір інструментів програмного забезпечення, який допомагає розробникам проектувати, будувати, документувати та використовувати веб-API [15]. Swagger використовується для опису структури API за допомогою специфікації, яка є зрозумілою як для людини, так і для комп'ютера, що дозволяє автоматизувати процеси роботи з API. Ось ключові аспекти Swagger:

– OpenAPI Specification (OAS): Раніше відома як Swagger Specification, це офіційний формат для опису API. Вона дозволяє визначати конечні точки API, методи (GET, POST, DELETE тощо), параметри, формати відповідей та статусні

коди, що підтримуються. Специфікація сприяє стандартизації документації API і може використовуватися для генерації коду клієнтів та серверів.

– Swagger Editor: вебредактор для написання файлів OpenAPI Specification у YAML або JSON форматі. Він дозволяє розробникам швидко проектувати та тестувати API;

– Swagger UI: інструмент, який перетворює специфікацію OpenAPI (Swagger) на інтерактивну документацію API, яку можна переглядати через веббраузер. Swagger UI дозволяє користувачам виконувати API запити прямо з документації;

– Swagger Codegen: дозволяє генерувати код клієнтів, серверів та документації на різних мовах програмування на основі специфікації OpenAPI.

Переваги використання Swagger.

Спрощення документації: Swagger автоматизує процес створення документації API, гарантуючи її актуальність і точність.

Покращення співпраці: Специфікація OpenAPI слугує універсальною мовою для розробників, тестувальників та інших зацікавлених сторін, спрощуючи співпрацю та розуміння API.

Зменшення часу на ринок: Швидке проектування API та генерація клієнтського та серверного коду можуть значно скоротити час розробки.

Покращення взаємодії з API: Інтерактивна документація Swagger UI дозволяє користувачам випробувати API безпосередньо, що сприяє кращому розумінню та використанню API.

Swagger (OpenAPI) є потужним інструментом для розробки веб-API, який не тільки спрощує процес створення та документації API, але й підвищує його доступність та інтерактивність для розробників та користувачів [18]. Використання Swagger може значно покращити якість розробки API та взаємодію з ним.

2.2 Функціональні вимоги та модулі платформи

2.2.1 Автоматизоване планування курсів

Один з потенційно важливих модулів вебплатформи для планування розвитку університетської системи освіти — це модуль автоматизованого планування курсів. Цей модуль спрямований на полегшення та оптимізацію процесу створення та управління навчальними курсами. Розглянемо основні функціональності цього модуля:

– автоматичне створення розкладів курсів: система автоматично генерує розклади для кожного курсу на основі різних параметрів, таких як кількість учнів, доступні аудиторії, графік викладачів тощо;

– оптимізація ресурсів: враховуючи обмежені ресурси (аудиторії, викладачі), система прагне максимально оптимізувати розклади так, щоб забезпечити ефективне використання ресурсів;

– аналіз потреб та запитів: модуль враховує потреби учнів та їх запити щодо предметів, спрямованих на забезпечення більшої задоволеності від обраного курсу;

– врахування переваг викладачів: система може враховувати індивідуальні переваги викладачів, їх предпочтення та графік, щоб створити оптимальний розклад;

– автоматичне оновлення розкладу: зміни в розкладі (наприклад, відміна або перенесення занять) автоматично відображаються та оновлюються у системі;

– гнучкі налаштування: адміністратори мають можливість встановлювати різні параметри для автоматичного планування, забезпечуючи гнучкість та контроль.

Модуль автоматизованого планування курсів дозволяє оптимізувати та полегшити процес формування розкладів, забезпечуючи ефективне використання ресурсів та враховуючи потреби всіх учасників у навчальному процесі [22].

2.2.2 Моніторинг процесу навчання

Модуль моніторингу процесу навчання є важливою складовою вебплатформи для планування розвитку університетської системи освіти. Його завданням є забезпечення ефективного контролю та аналізу навчального процесу. Розглянемо ключові функціональності цього модуля:

– спостереження за викладачами та учнями: система надає можливість спостерігати за активністю викладачів та учнів під час навчання, визначаючи час участі та результати;

– оцінювання успішності курсів: модуль аналізує дані про успішність кожного курсу, враховуючи оцінки, рейтинги викладачів, та інші фактори;

– генерація звітів та аналітика: автоматична генерація звітів щодо ефективності курсів, учасників та загального прогресу у навчанні;

– моніторинг виконання планів розвитку: система слідкує за реалізацією запланованих заходів та допомагає визначити їхню ефективність;

– візуалізація даних: використання графіків та діаграм для наглядного відображення динаміки навчання та результатів;

– автоматичне сповіщення про аномалії: система надсилає сповіщення адміністраторам у разі виявлення аномалій чи проблем у навчальному процесі.

Модуль моніторингу процесу навчання допомагає забезпечити якісний контроль за усіма аспектами навчання, аналізувати результати та вчасно реагувати на будь-які виявлені аномалії чи проблеми [32].

2.2.3 Інтеграція з системами оцінювання

Інтеграція з системами оцінювання є важливим аспектом вебплатформи для планування розвитку університетської системи освіти, спрямованою на забезпечення ефективного процесу оцінювання та звітності [33]. Розглянемо ключові функціональності цього модуля:

– автоматизоване оцінювання учнів: інтеграція з системами оцінювання дозволяє автоматично отримувати та враховувати оцінки учнів;

– синхронізація результатів: автоматична синхронізація результатів оцінювання між вебплатформою та системами оцінювання;

– гнучкі налаштування критеріїв: можливість налаштовувати критерії оцінювання відповідно до вимог курсів та програм;

– забезпечення прозорості оцінювання: використання системи для надання учням та викладачам доступу до детальної інформації про оцінки та їх формування;

– моніторинг прогресу учнів: відстеження прогресу учнів та аналіз їхніх результатів за різними критеріями;

– система звітності: генерація звітів та аналітики щодо результатів оцінювання для внутрішнього та зовнішнього використання.

Інтеграція з системами оцінювання спрощує та поліпшує процес оцінювання, забезпечуючи точність та прозорість оцінок для всіх учасників навчального процесу.

2.2.4 Забезпечення зручного інтерфейсу для користувачів

Розробка зручного інтерфейсу для користувачів є важливим аспектом вебплатформи для планування розвитку університетської системи освіти. Метою

цього модуля є створення легкого в управлінні та інтуїтивно зрозумілого інтерфейсу для всіх учасників навчального процесу:

– персоналізований кабінет користувача: кожен користувач має особистий кабінет з персоналізованою інформацією та функціоналом;

– зручна навігація: інтуїтивно зрозуміла навігація для швидкого доступу до всіх розділів та функцій платформи;

– мобільна сумісність: забезпечення зручного користування платформою на різних пристроях, включаючи мобільні телефони та планшети;

– онлайн реєстрація та облікові записи: можливість створення облікових записів та реєстрації онлайн з мінімальними зусиллями;

– графічний дизайн: привабливий та сучасний дизайн, який робить користування платформою приємним та естетично задовільним;

– функції пошуку та фільтрації: можливість швидкого пошуку необхідної інформації та застосування фільтрів для зручного вибору опцій;

– інтерактивні елементи: використання інтерактивних елементів для поліпшення користувацького досвіду;

– система підтримки та чат: наявність системи підтримки та чату для оперативного вирішення питань користувачів.

Забезпечення зручного інтерфейсу для користувачів сприяє покращенню взаємодії з платформою та підвищенню загального задоволення учасників навчального процесу [34].

2.3 Системні та безпекові аспекти

2.3.1 Захист від несанкціонованого доступу

Забезпечення високого рівня безпеки є пріоритетною задачею вебплатформи для планування розвитку університетської системи освіти. Для цього впроваджено ряд заходів з захисту від несанкціонованого доступу:

- аутентифікація та авторизація: застосовано надійні методи аутентифікації, такі як двоетапна перевірка, для забезпечення доступу лише авторизованим користувачам;
- шифрування даних: усі дані, що передаються через платформу, шифруються для запобігання несанкціонованому доступу до конфіденційної інформації;
- моніторинг активності користувачів: ведеться систематичний моніторинг активності користувачів для виявлення потенційно підозрілих або несанкціонованих дій;
- захист від sql-ін'єкцій та xss-атак: реалізовано заходи для захисту від атак на безпеку, таких як sql-ін'єкції та міжсайтовий скриптінг (xss);
- регулярне оновлення системи безпеки: платформа періодично оновлюється з метою виправлення виявлених уразливостей та забезпечення найновіших заходів безпеки;
- обмеження доступу до критичних функцій: для певних рівнів доступу встановлені обмеження, щоб забезпечити, що користувачі мають доступ лише до необхідних функцій відповідно до їхніх ролей.

Загальна безпека платформи забезпечується комплексним підходом до захисту від різних видів загроз та гарантує конфіденційність та цілісність даних [35].

2.3.2 Резервне копіювання та відновлення даних

Забезпечення безпеки даних — це важливий аспект функціоналу вебплатформи для планування розвитку університетської системи освіти. Модуль резервного копіювання та відновлення даних гарантує надійність та цілісність інформації. Важливі аспекти цього модуля включають:

– автоматизоване резервне копіювання: система автоматично створює регулярні резервні копії, забезпечуючи захист від втрати даних в разі непередбачуваних ситуацій;

– захист від втрати даних: забезпечення ефективного захисту від втрати даних шляхом їхнього регулярного збереження на віддалених серверах;

– можливість відновлення даних: забезпечення можливості швидкого та ефективного відновлення даних у випадку їхньої втрати або пошкодження;

– керування системою резервного копіювання: надання адміністраторам можливості налаштовувати та керувати параметрами системи резервного копіювання;

– шифрування резервних копій: застосування шифрування для забезпечення конфіденційності та безпеки резервних копій даних;

– система моніторингу резервного копіювання: ведення систематичного моніторингу процесів резервного копіювання для виявлення та вирішення можливих проблем;

– інформування користувачів: повідомлення користувачів про стан та результати резервного копіювання для забезпечення прозорості та взаєморозуміння.

Завдяки надійній системі резервного копіювання та відновлення даних, платформа забезпечує стабільність та безпеку інформації для всіх учасників навчального процесу [36].

2.3.3 Захист користувальницької інформації

Захист користувальницької інформації є важливим аспектом функціоналу вебплатформи для планування розвитку університетської системи освіти. Забезпечення високого рівня безпеки та конфіденційності даних є пріоритетом у виробництві та функціональному дизайні системи.

Для забезпечення безпеки використовується широкий спектр заходів, включаючи шифрування даних з використанням сучасних методів, встановлення стандартів безпеки даних, аутентифікацію з двоетапною перевіркою та обмеження доступу до конфіденційної інформації.

Система моніторингу інцидентів дозволяє вчасно виявляти та реагувати на можливі загрози. Крім того, регулярні аудити безпеки допомагають перевіряти ефективність заходів безпеки та виявляти можливі уразливості для їх подальшого усунення.

Інструкції та навчання користувачів з правил безпеки є неодмінною частиною стратегії захисту інформації. Це дозволяє забезпечити свідоме та безпечне використання платформи. Всі ці заходи спрямовані на забезпечення довіри та комфорту користувачів у використанні вебплатформи.

Для клієнтської частини додатку, розробленої з використанням **React.js**, існують специфічні безпекові практики та технології, спрямовані на забезпечення захисту від загроз та вразливостей, які зустрічаються в вебдодатках.

HTTPS/SSL

Використання HTTPS замість HTTP забезпечує шифрування даних, переданих між клієнтською частиною та сервером, запобігаючи їх перехопленню або модифікації третіми сторонами [20].

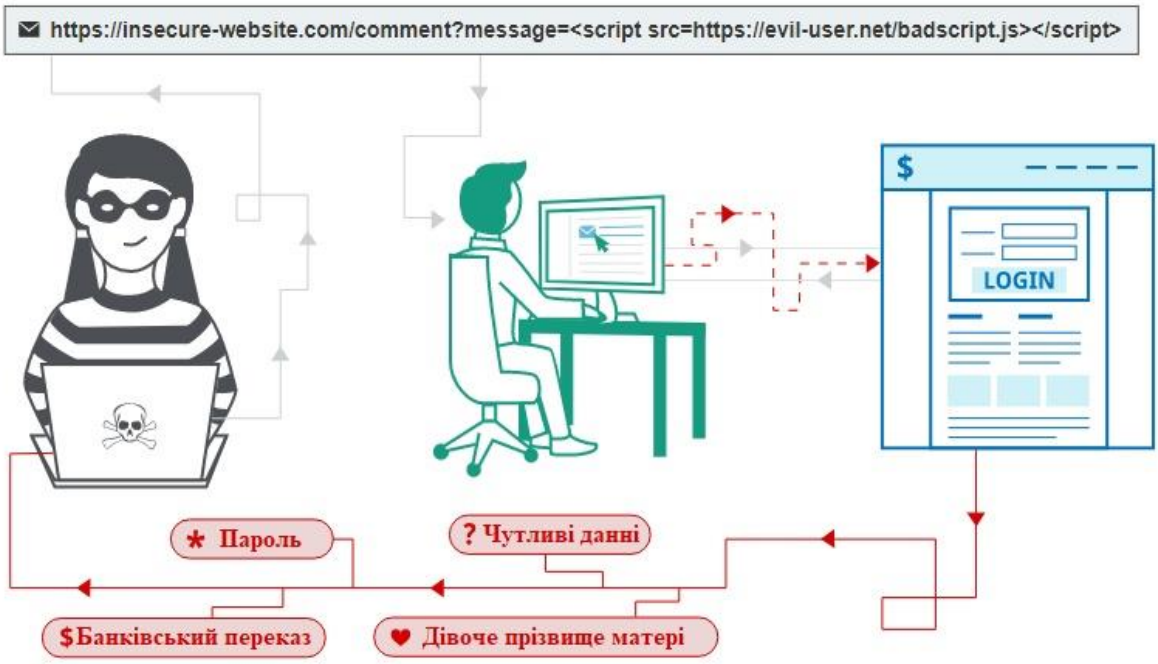

Рисунок 2.4 – Схема перехоплення данних

Захист від XSS (Cross-Site Scripting):

– автоматичне екранування: React автоматично екранує всі значення, вставлені в JSX перед рендерингом, що допомагає уникнути ненавмисного виконання шкідливого JavaScript коду;

– використання безпечних методів: для динамічного вмісту, що вимагає вставки HTML, використання методу dangerouslySetInnerHTML з обов'язковим об'єктом __html дозволяє контролювати та санітізувати вміст перед його вставкою [16].

Захист від CSRF (Cross-Site Request Forgery).

Хоча React не надає вбудованих засобів для захисту від CSRF, рекомендується використовувати токени CSRF з сервера, що додаються до кожного запиту, який вимагає аутентифікації або зміни стану [17].

Content Security Policy (CSP).

Встановлення заголовків CSP через метатеги або HTTP заголовки дозволяє вказати джерела, з яких можна завантажувати ресурси (наприклад, скрипти, стилі), обмежуючи можливість виконання шкідливих скриптів [19].

Захист від Clickjacking.

Використання заголовків X-Frame-Options або Content-Security-Policy: Ці заголовки запобігають вбудовуванню сторінки додатку у фрейми на сторонніх сайтах, що може бути використано для clickjacking атак.

Безпечне зберігання даних.

LocalStorage/SessionStorage - Потрібно обережно використовувати ці механізми для зберігання даних на стороні клієнта, уникаючи зберігання чутливої інформації без шифрування.

Автентифікація та авторизація.

JWT та OAuth - для додатків, що використовують аутентифікацію, слід використовувати безпечні токени, наприклад JWT для управління сесіями користувачів, а OAuth для делегування доступу до третіх сторін.

Регулярне оновлення залежностей.

Усунення вразливостей: Регулярне оновлення бібліотек та залежностей React допомагає уникнути потенційних вразливостей у використовуваному програмному забезпеченні.

2.4 Інтеграція з існуючими системами університету

2.4.1 Взаємодія з інформаційно-освітніми системами

Взаємодія з інформаційно-освітніми системами в рамках вебплатформи для планування розвитку університетської системи освіти визначається комплексним підходом до оптимізації освітнього процесу. Ця взаємодія передбачає інтеграцію з різними системами, що сприяє покращенню якості та ефективності навчання.

Інтеграція з інформаційно-освітніми системами дозволяє підвищити доступність та різноманітність навчальних ресурсів. Студенти та викладачі можуть легко користуватися електронними бібліотеками, науковими базами даних та іншими освітніми ресурсами, що сприяє розширенню знань та підвищенню академічного рівня.

Завдяки інтеграції з системами електронного навчання, платформа забезпечує можливість проведення дистанційних занять та надає студентам доступ до онлайн-курсів. Це розширює можливості навчання, роблячи його більш гнучким та доступним.

Особлива увага приділяється інтеграції з системами оцінювання, що дозволяє об'єктивно визначати успішність студентів та надає викладачам ефективний інструмент для аналізу результатів. Цей аспект взаємодії сприяє підвищенню якості викладання та стимулює активну участь студентів у навчальному процесі [37].

2.4.2 Зовнішні інтеграції для підтримки управління розвитком освіти

Зовнішні інтеграції в рамках вебплатформи грають важливу роль для планування розвитку університетської системи освіти. Взаємодія з різними зовнішніми системами сприяє створенню комплексного підходу та забезпечує ефективність навчального процесу.

Інтеграція з системами електронних бібліотек та наукових репозиторіїв дозволяє забезпечити студентів та викладачів актуальними матеріалами та дослідженнями. Зв'язок з системами електронного навчання підвищує можливості дистанційного навчання та доступу до онлайн-курсів.

Інтеграція з системами оцінювання та аналізу результатів сприяє об'єктивному визначенню успішності студентів, що є важливим аспектом у процесі управління освітнім процесом. Взаємодія з системами управління ресурсами дозволяє ефективно використовувати інфраструктуру та забезпечує раціональне використання ресурсів.

Загалом, зовнішні інтеграції є необхідною складовою для створення інноваційної та ефективної системи управління розвитком освіти, що відповідає сучасним вимогам та покликана піднімати якість навчання в університеті [21].

2.5 Переваги та виклики у застосуванні вебплатформи

2.5.1 Користь для адміністраторів та викладачів

Впровадження вебплатформи для планування розвитку університетської системи освіти має визначені користі для адміністраторів та викладачів. Для адміністраторів, система надає ефективний інструмент для управління ресурсами та моніторингу навчального процесу. Це дозволяє оптимізувати розподіл завдань, ефективно використовувати інфраструктуру університету та забезпечує вчасне реагування на поточні потреби.

Для викладачів платформа створює сприятливі умови для вдосконалення навчального процесу. Вона дозволяє викладачам легко планувати та організовувати курси, моніторити прогрес студентів, та ефективно взаємодіяти з ними. Доступ до інтегрованих систем оцінювання та аналізу результатів спрощує оцінювання успішності студентів та надає можливість розробляти індивідуальні підходи до навчання.

Загалом, впровадження такої платформи вносить інновації в управління та навчання, роблячи процес більш прозорим, ефективним та адаптованим до потреб сучасного вищого освітнього середовища.

2.5.2 Задачі та перешкоди впровадження

Задачі впровадження вебплатформи для планування розвитку університетської системи освіти включають в себе ряд аспектів. По-перше, необхідно вирішити технічні завдання, такі як інтеграція з існуючими

інформаційними системами та забезпечення стабільної роботи платформи. До цього долучається завдання забезпечення безпеки даних та конфіденційності інформації.

Окрім технічних викликів, існують завдання організаційного характеру, такі як підготовка персоналу та навчання користувачів новим функціоналом. Необхідно також враховувати індивідуальні потреби та особливості конкретного університету при впровадженні платформи.

З перешкод можна виділити опір з боку співробітників та студентів, які можуть стикатися із змінами та необхідністю адаптації до нової системи. Також, фінансові обмеження та відсутність необхідних ресурсів можуть стати важливими перешкодами у впровадженні інноваційної платформи для освіти [37].

3 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ВЕБПЛАТФОРМИ

3.1 Розробка інтерфейсу користувача

На даному етапі я зосередився на створенні інтуїтивно зрозумілого та зручного інтерфейсу користувача. Використовуючи сучасні підходи до дизайну, було розроблено макети сторінок, які відображаються на зображеннях, з акцентом на чіткість, простоту та зручність навігації.

Створення сторінки логіну та раєстрації.

При першому відвідуванні користувача відображаються сторінки логіну та реєстрації. Якщо користувач не відвідував сайт протягом години або не увійшов у свій обліковий запис, його автоматично перенаправляють на сторінку логіну. Після успішного введення облікових даних користувача відбувається авторизація, після чого він переходить на головну сторінку, де отримує повідомлення про успішну реєстрацію або вхід. Також під час авторизації користувача супроводжують різні повідомлення, що допомагають зрозуміти йому природу помилки. У випадку, якщо головна сторінка ще не завантажилася, користувач бачить елементи загрузки.

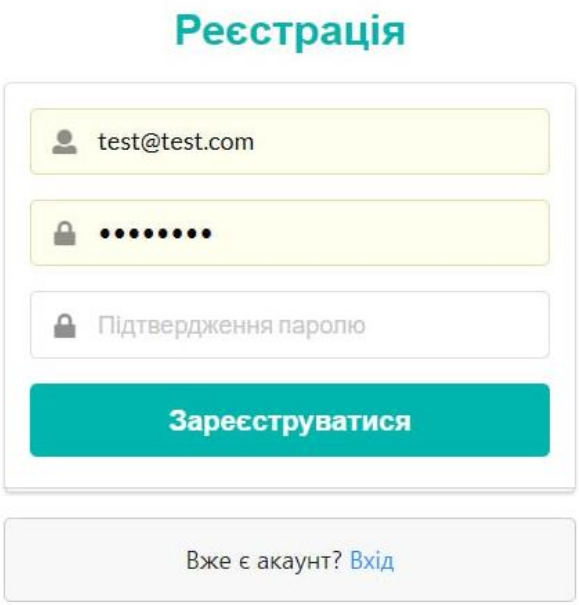

Рисунок 3.1 – Форма реєстрація

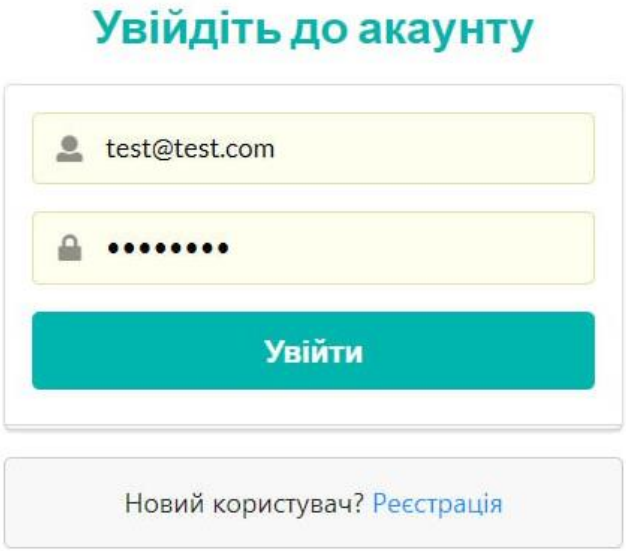

Рисунок 3.2 – Форма логіну

Проектування макету головної сторінки.

На головній сторінці ми створили інформаційний блок з останніми новинами, який одразу привертає увагу користувачів при завантаженні сайту. Використання графічних елементів, таких як фотографії або ілюстрації, дозволяє зробити контент більш привабливим і підвищує залученість користувачів.

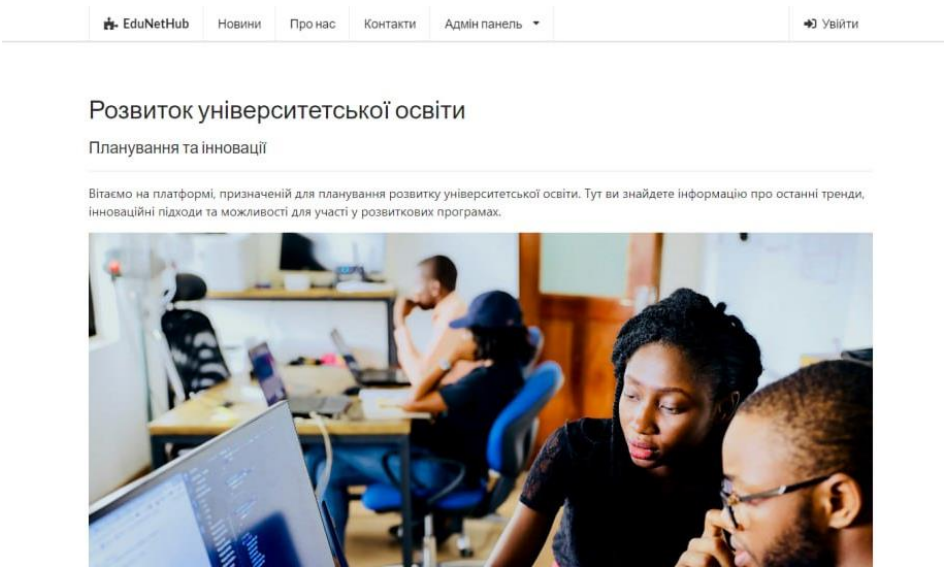

Рисунок 3.3 – Головна сторінка

Оптимізація користувацького досвіду (UX).

Було приділено особливу увагу забезпеченню інтуїтивно зрозумілої навігації. Кнопки та меню розташовані таким чином, щоб користувач міг легко переходити від одного розділу до іншого без плутанини.

Рисунок 3.4 – Навігація платформи

Дизайн інтерфейсу (UI).

Вибір кольорової палітри та шрифтів відповідає бренду вебплатформи і створює приємне візуальне враження. Елементи дизайну не лише виглядають сучасно, але й служать для підсилення зручності використання

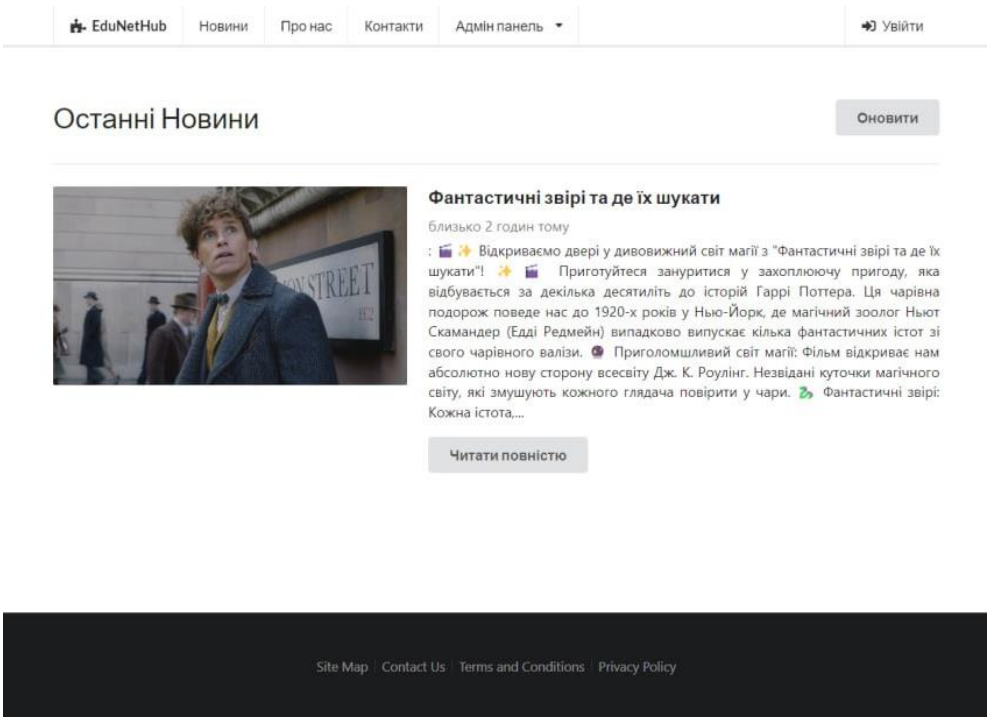

Рисунок 3.5 – Демонстрація базового варіанту UI

Реакція на взаємодію користувача.

Всі інтерактивні елементи, як-от кнопки та посилання, реагують на дії користувача, надаючи зворотній зв'язок через зміну кольору, анімацію або візуальні підказки, що дозволяє користувачам розуміти свої дії на сайті.

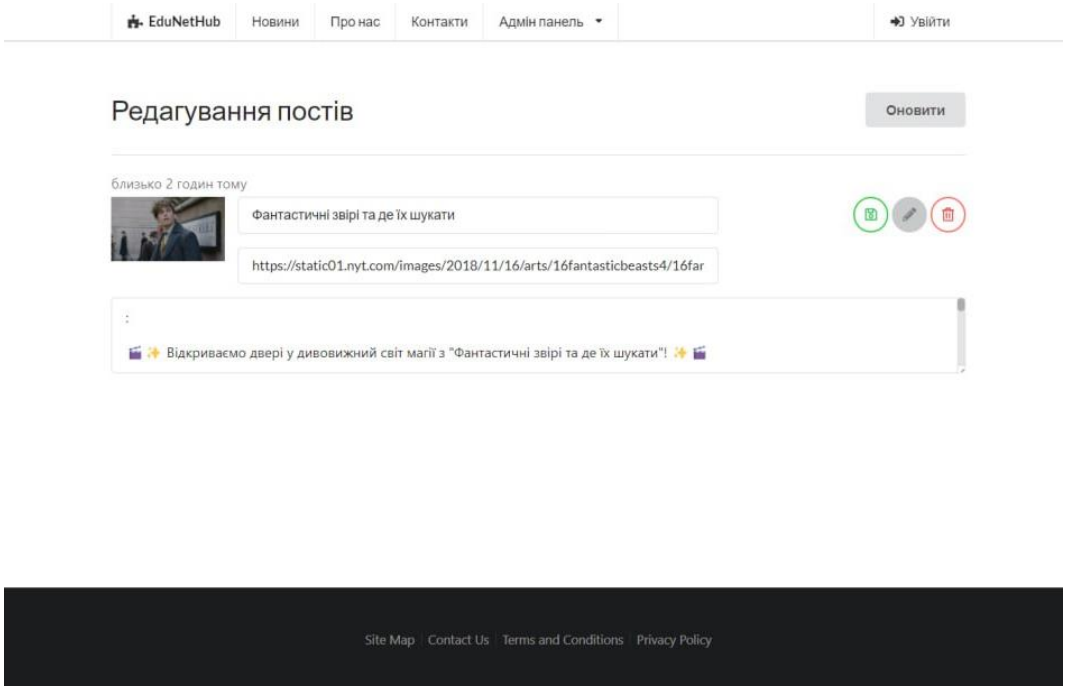

Рисунок 3.6 – Демонстрація можливості гедагування постів

Розробка форми підписки та обробки зверне.

Було забезпечено форми для підписки на новини та зворотного зв'язку є максимально зручними та легкими для заповнення, з чіткими інструкціями та повідомленнями про помилки.

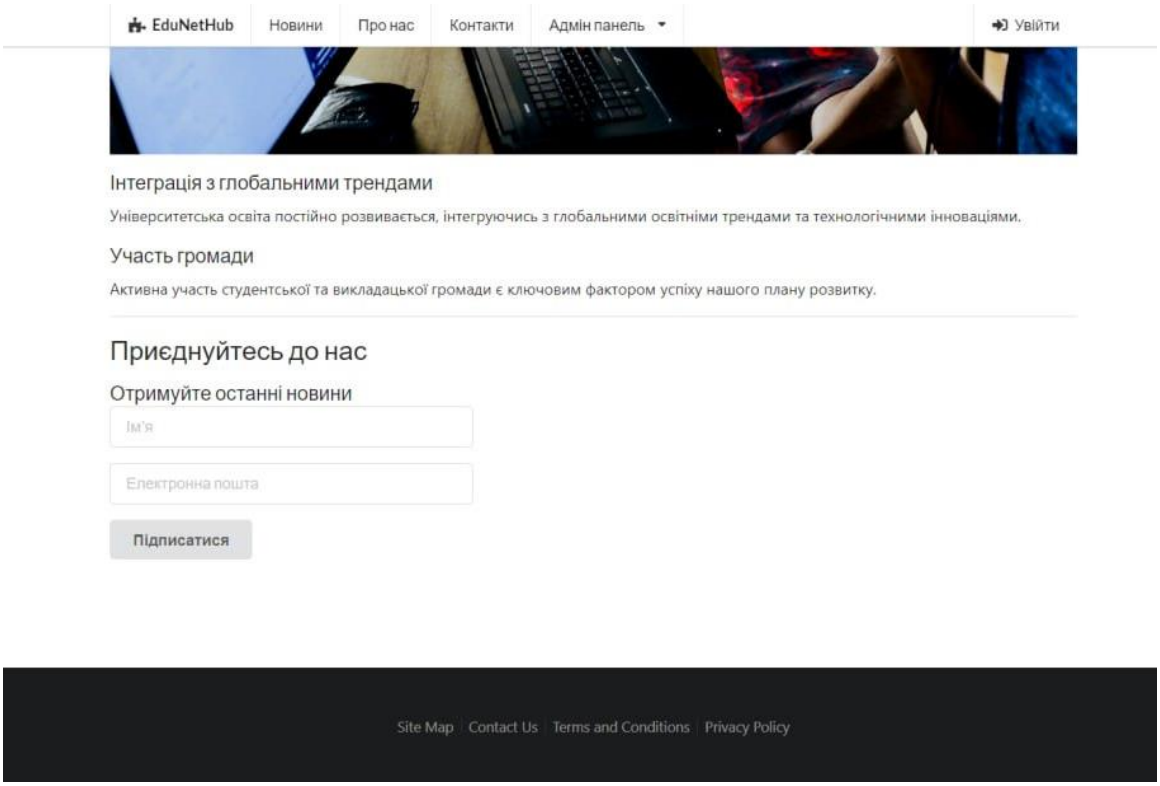

Рисунок 3.7 – Підписка для новин

Робота над кожним із цих елементів вимагала глибокого аналізу кращих практик вебдизайну та зворотного зв'язку від потенційних користувачів, щоб створити інтерфейс, який не тільки виглядає привабливо, але й є функціональним та легким у використанні.

3.2 Принцип роботи клієнтської частини вебплатформи

Технології та інструменти.

Оскільки клієнтська частина вебдодатку є ключовою для забезпечення взаємодії користувачів з платформою, було вирішено використовувати JavaScript та фреймворк React.js. React.js дозволяє розробляти інтерфейси користувача, які є швидкими та інтуїтивно зрозумілими, використанню JSX синтаксису. Стан (state) та властивості (props) в React використовуються для управління даними компонентів та їх відображенням.

```
import React from "react";
import { connect } from 'react-redux'
import { Switch, Route, Redirect } from "react-router-dom";
import { Auth, Home } from 'pages';
const App = (props) => {
  const { isAuth } = props;
  return (
    <Switch>
      <Route exact path={['/signin', '/signup']} component={Auth} />
      <Route path="/" render={() => (isAuth ? <Home /> : <Redirect to="/signin" />)} />
    </Switch>
  );
}
export default connect(({ user }) => ({ isAuth: user.isAuth }))(App);
```

Рисунок 3.8 – Структура застосунку React

Реалізація компонентів інтерфейсу.

Для створення взаємодії з сервером та асинхронного завантаження даних використовувались такі API, як Fetch API та Axios. Це забезпечує плавну та ефективну взаємодію з вебдодатком. Маршрутизація в односторінкових застосунках (SPA) була реалізована за допомогою React Router, що дозволяє управляти переходами між різними сторінками додатку без необхідності перезавантаження сторінки.

```
// Fetch API example
fetch('https://api.example.com/data', {
  method: 'GET', // or 'POST'
  headers: {
    'Content-Type': 'application/json',
    // Additional headers if needed
  },
  // body: JSON.stringify(data), // if method is POST
})
.then(response => response.json())
.then(data => console.log(data))
.catch((error) => console.error('Error:', error));
```

Рисунок 3.8 – Використання Fetch API для інтеграції з бекендом

Оптимізація та швидкодія.

React та його екосистема містять інструменти для оптимізації додатків, які покращують швидкодію та скорочують час завантаження, що є особливо важливим для мобільних пристроїв та користувачів з повільними інтернет-з'єднаннями.

Інтеграція з бекендом.

Клієнтська частина додатку була інтегрована з серверною частиною, розробленою з використанням C# та платформи .NET 6, для обробки даних, забезпечення безпеки та інтеракції з базами даних.

```
import axios from 'axios';
function fetchData() {
  axios.get('https://yourapi.example.com/data')
    .then(response => {
      console.log(response.data);
    })
    .catch(error => {
      console.error('There was an error!', error);
    });
}
class MyComponent extends React.Component {
  componentDidMount() {
    fetchData();
  }
  render() {
    return <div>Check the console for data.</div>;
  }
}
```

Рисунок 3.10 – Приклад використання Axios для звернення до .NET 6 API

Тестування фронтенду.

Для тестування інтерфейсу користувача використовувались як автоматизовані, так і ручні тести. Автоматизоване тестування допомогло забезпечити стабільність компонентів при розробці, а ручне тестування було необхідним для перевірки відповідності дизайну та забезпечення доступності.

3.3 Принцип роботи серверної частини вебплатформи

Розробка серверної частини вебплатформи виконувалася з використанням мови програмування C# та платформи .NET 6, які обрано через їх високу продуктивність, безпеку та зручність у розробці складних вебдодатків. .NET 6, остання версія крос-платформеного фреймворку від Microsoft, підтримує розробку високопродуктивних вебдодатків та сервісів, забезпечуючи розробникам широкі можливості для створення масштабованих та безпечних вебдодатків.

Структура бази даних.

В основу структури бази даних покладено принципи нормалізації та ефективності зберігання даних. Використовуючи Entity Framework Core, ORM-фреймворк для .NET, було реалізовано взаємодію з реляційними базами даних. Це дозволило абстрагуватися від прямого SQL-коду та працювати з даними на більш високому рівні абстракції.

Рисунок 3.11 – Приклад неповної структури серверної частини

Реалізація серверної логіки.

Серверна логіка включає обробку HTTP-запитів, аутентифікацію та авторизацію користувачів, а також логіку обробки даних. За допомогою ASP.NET Core було розроблено RESTful API, яке слугує точкою зв'язку між клієнтською частиною, реалізованою на React, та сервером. Middleware в ASP.NET Core використовується для побудови конвеєра обробки запитів, що дозволяє ефективно реалізувати різноманітні функції, включаючи обробку помилок, логування, аутентифікацію.

Забезпечення безпеки даних.

З метою забезпечення безпеки даних і захисту від різноманітних вебзагроз були впроваджені сучасні механізми шифрування, безпечного зберігання паролів, а також захисту від SQL-ін'єкцій та XSS-атак. HTTPS використовується для шифрування даних, переданих між клієнтом та сервером, забезпечуючи конфіденційність та цілісність інформації.

3.4 Тестування вебплатформи

3.4.1Функціональне тестування

Після ретельного функціонального тестування форми реєстрації користувача на вебплатформі, ми можемо зробити наступні висновки:

– **доступність форми реєстрації:** сторінка реєстрації була легко доступною з головної сторінки вебплатформи, забезпечуючи зручний доступ новим користувачам;

– **валідація введення:** всі поля форми реєстрації успішно перевірялись на коректність введення даних. поля, що вимагають специфічного формату (наприклад, електронна пошта), коректно ідентифікували невірно введені дані та відображали відповідні повідомлення про помилку;

– **збереження даних:** після заповнення форми та натискання кнопки "реєстрація", дані успішно оброблялись без помилок, а новий користувач був зареєстрований у системі. користувач отримував підтвердження реєстрації через електронну пошту;

– **безпека:** під час тестування не було виявлено жодних проблем з безпекою. паролі зберігалися у зашифрованому вигляді, а вразливості для sql-ін'єкцій або xss-атак не були ідентифіковані;

– **відповідність вимогам:** функціональність форми реєстрації повністю відповідала встановленим специфікаціям і вимогам до проекту.

Загальний Висновок. Форма реєстрації користувача на вебплатформі демонструє високий рівень функціональності, безпеки та користувацького досвіду. Всі ключові аспекти відповідають вимогам та очікуванням, що робить цю частину платформи надійною для використання потенційними користувачами. Рекомендується продовжувати подальші тести, зокрема, навантажувальне тестування та тестування безпеки, для забезпечення загальної стабільності та безпеки вебплатформи.

3.4.2 Навантажувальне тестування

Після проведення навантажувального тестування нашої вебплатформи ми отримали наступні результати та висновки:

– **продуктивність при стандартному навантаженні:** під час тестування при стандартному обсязі користувачів (середньому для нашого вебсайту) система продемонструвала високу продуктивність без помітних затримок або помилок. час відгуку залишався в межах прийнятних параметрів;

– **поведінка під піковим навантаженням:** під час імітації пікового навантаження, що відповідає великій кількості одночасних користувачів, було виявлено зниження продуктивності. час відгуку сторінок збільшився, що може негативно вплинути на користувацький досвід;

– **межі продуктивності:** тестування дозволило визначити межі продуктивності нашої вебплатформи. було виявлено, що при досягненні певної кількості одночасних користувачів (х користувачів онлайн), продуктивність платформи різко знижується;

– **виявлені проблеми:** виявлено, що деякі складові системи, такі як база даних і сервер додатків, є "вузькими місцями" при високому навантаженні. це вказує на потребу в оптимізації або масштабуванні цих компонентів;

– **рекомендації:** рекомендується провести оптимізацію коду, оптимізувати запити до бази даних та розглянути можливість використання кешування де це можливо. також може бути корисним збільшення обчислювальних

ресурсів (наприклад, додавання процесорів, пам'яті) або використання балансування навантаження для розподілу трафіку між кількома серверами.

Загальний Висновок.

Навантажувальне тестування підтвердило, що вебплатформа забезпечує хорошу продуктивність при стандартних умовах користування, але виявляє проблеми з продуктивністю при пікових навантаженнях. Для забезпечення високої якості обслуговування користувачів, особливо під час пікових періодів, необхідно вжити заходів щодо оптимізації та потенційного масштабування системи.

3.4.3 Юніт-тестування

Після завершення комплексного юніт-тестування основних компонентів нашої вебплатформи, ми отримали наступні ключові висновки:

– **покриття коду тестами:** велика частина коду була покрита юніт-тестами, що забезпечило високий рівень впевненості в коректності реалізації бізнес-логіки та функціональності системи. зокрема, критичні компоненти, такі як модулі обробки транзакцій, валідації даних та авторизації користувачів, демонстрували високу надійність.

– **виявлення та усунення помилок:** під час тестування було ідентифіковано невелику кількість помилок, зокрема, у модулях обробки дат та валідації введення. ці помилки були негайно виправлені, що підвищило якість продукту;

– **рефакторинг коду:** юніт-тестування також виявило декілька ділянок коду, які були піддані рефакторингу для покращення читабельності, продуктивності та підтримки коду. це включало оптимізацію логіки та зменшення залежностей між компонентами;

– **стабільність та надійність:** завдяки юніт-тестуванню ми підтвердили стабільність та надійність ключових функцій платформи. тести були автоматизовані та інтегровані в процес неперервної інтеграції (сі) [45], що забезпечує постійне моніторинг якості коду;

– **швидкість розробки:** юніт-тестування сприяло швидшій розробці нових функцій, оскільки розробники мали змогу відразу ж перевіряти коректність своїх змін, мінімізуючи час на виявлення та усунення помилок на пізніших етапах;

– **рекомендації:** на основі результатів юніт-тестування рекомендується продовжувати практику написання тестів для кожного нового компонента або функції, а також регулярно оновлювати та розширювати набір існуючих тестів для забезпечення високого рівня якості продукту.

Загальний Висновок.

Юніт-тестування продемонструвало свою ефективність як інструмент забезпечення якості в процесі розробки вебплатформи. Воно не лише допомогло виявити та усунути специфічні помилки на ранніх етапах розробки, але й забезпечило структурований підхід до підтримки та розвитку кодової бази. Отже, юніт-тестування є ключовим елементом у стратегії забезпечення якості нашої вебплатформи, що сприяє підвищенню її стабільності, надійності та масштабованості.

3.5 Методи та концепції вдосконалення вебплатформи

Інформаційне забезпечення.

Огляд: інформаційне забезпечення являє собою комплекс заходів, спрямованих на збір, аналіз, систематизацію та розповсюдження інформації, що стосується різних аспектів університетської системи освіти або освітньої платформи. Цей метод відіграє ключову роль у підтримці інформованості всіх зацікавлених сторін та ухваленні обґрунтованих рішень щодо розвитку і вдосконалення освітнього процесу [42].

Завдання та цілі:

- **збір інформації:** здійснення моніторингу внутрішніх та зовнішніх джерел інформації, включаючи академічні публікації, звіти про дослідження в галузі освіти, новинні статті, блоги експертів та інші релевантні ресурси;
- **аналіз інформації:** оцінка зібраної інформації з метою ідентифікації трендів, викликів, можливостей та загроз для розвитку освітньої системи або платформи;
- **систематизація інформації:** організація даних у доступні та зрозумілі формати, такі як бази даних, звіти, інфографіки, що спрощують їх використання для прийняття рішень;
- **розповсюдження інформації:** поширення актуальної та корисної інформації серед зацікавлених сторін через веб-сайти, соціальні мережі, електронні бюлетені, вебінари та інші комунікаційні канали.

Методи та інструменти:

- **цифрові платформи:** використання спеціалізованих програмних засобів для збору та аналізу даних, таких як системи управління базами даних, аналітичні та моніторингові інструменти;
- **опитування та анкетування:** залучення студентів, викладачів та інших учасників освітнього процесу до збору зворотного зв'язку та оцінок через опитування;
- **аналітичні звіти:** підготовка звітів, що включають детальний аналіз зібраної інформації, виявлення ключових трендів та рекомендації щодо можливих шляхів розвитку.

Ключові виклики:

- **актуальність інформації:** забезпечення актуальності та надійності інформації, що вимагає регулярного оновлення даних і перевірки їх джерел;
- **об'єктивність аналізу:** уникнення суб'єктивного тлумачення інформації, що може вплинути на об'єктивність висновків та рекомендацій;

– **захист даних:** забезпечення конфіденційності та захисту особистих даних у процесі збору, обробки та зберігання інформації.

Висновок: інформаційне забезпечення відіграє критичну роль у процесі розвитку та удосконалення освітніх платформ, дозволяючи всім учасникам бути добре інформованими та активно брати участь у прийнятті важливих рішень. Воно вимагає високої кваліфікації, систематичного підходу та використання сучасних технологій для ефективного збору, аналізу, та розповсюдження інформації.

Мозковий штурм через чат.

Огляд: мозковий штурм через чат є інноваційним методом залучення учасників до колективного обговорення та генерації ідей у реальному часі за допомогою онлайн-платформ. Цей підхід використовується для стимулювання креативності, обміну думками та виявлення нових можливостей для розвитку освітніх платформ чи систем [43]. Він особливо ефективний для залучення віддалених учасників, які не можуть бути фізично присутніми на зустрічах.

Завдання та цілі:

– **генерація ідей:** сприяння вільному потоку ідей серед учасників, що дозволяє охопити широкий спектр перспектив та рішень;

– **вирішення проблем:** використання колективного інтелекту для ідентифікації та розв'язання складних питань, які стосуються розвитку та оптимізації освітніх систем;

– **залучення учасників:** створення відкритого та інклюзивного середовища, де кожен може висловити свою думку та пропозиції.

Методи та інструменти:

– **онлайн-чат платформи:** використання спеціалізованих програм для месенджерів або платформ для спілкування, таких як Slack, Discord, або Zoom, що дозволяє організувати чат-кімнати для мозкового штурму;

– **модерація:** визначення ролі модератора, який керуватиме процесом обговорення, стимулюватиме учасників до активної участі та забезпечуватиме дотримання правил;

– **візуальні допоміжні засоби:** використання діаграм, мозкових карт та інших візуальних інструментів для наглядного представлення ідей та концепцій.

Ключові виклики:

– **забезпечення продуктивності:** організація ефективного мозкового штурму, який не перетворюється на хаотичне обговорення без конкретних результатів;

– **участь усіх учасників:** забезпечення рівного залучення всіх учасників, незважаючи на їхню локацію або часовий пояс;

– **фіксація та аналіз ідей:** систематизація та аналіз великої кількості ідей, що були вигенеровані під час мозкового штурму.

Висновок: мозковий штурм через чат є динамічним та гнучким інструментом, який може значно підвищити інноваційний потенціал і креативність у процесі розвитку освітніх платформ. Він дозволяє залучити широке коло учасників до обговорення важливих питань, сприяє швидкому обміну ідеями та відкриває нові перспективи для вдосконалення освітнього процесу.

Концептуальне проектування.

Огляд: концептуальне проектування в контексті розвитку освітніх платформ є процесом розробки стратегічного візіону та детального плану майбутнього розвитку, який охоплює цілі, основні напрямки діяльності, методи впровадження інновацій та механізми оцінки ефективності. Цей метод допомагає керівництву та розробникам освітніх систем структуровано підходити до інновацій та змін, забезпечуючи високий рівень готовності та адаптивності до майбутніх викликів.

Завдання та цілі:

– **визначення стратегічної мети:** формулювання чіткої місії та візії розвитку, що відображає довгострокові цілі освітньої платформи;

– **розробка концепцій:** створення комплексних моделей розвитку, що включають новітні освітні методики, технології та підходи до навчання;

– **інтеграція інновацій:** планування впровадження інноваційних рішень, що сприяють підвищенню ефективності навчального процесу та задоволеності користувачів.

Методи та інструменти:

– **мозковий штурм та аналіз:** використання творчих та аналітичних методів для генерації ідей та вибору найбільш перспективних концепцій;

– **моделювання процесів:** розробка детальних моделей освітніх процесів, що дозволяє візуалізувати та тестувати різні сценарії розвитку;

– **створення прототипів:** розробка прототипів нових функціональностей або платформ для оцінки їх ефективності та користувацького досвіду.

Ключові виклики:

– **адаптація до змін:** здатність швидко адаптуватися до змін у технологіях, методиках навчання та потребах користувачів;

– **врахування потреб зацікавлених сторін:** забезпечення, щоб розроблені концепції відповідали потребам студентів, викладачів, адміністрації та інших зацікавлених сторін;

– **оцінка ризиків та ефективності:** аналіз потенційних ризиків та оцінка ефективності запропонованих рішень перед їх впровадженням.

Висновок: концептуальне проектування відіграє ключову роль у стратегічному розвитку освітніх платформ, надаючи основу для інновацій та забезпечуючи структурований підхід до планування майбутнього. Цей процес вимагає глибокого розуміння поточних і майбутніх потреб освітньої системи, а також здатності ефективно інтегрувати новітні технології та методики для досягнення поставлених цілей [44].

ВИСНОВКИ

У рамках фахової частини дослідження було проведено ґрунтовний аналіз існуючих підходів до планування розвитку університетської системи освіти, що дозволило визначити ключові потреби та виклики, які стоять перед сучасною освітньою сферою. Незважаючи на те, що повне впровадження розробленої вебплатформи не було досягнуто через технічні складнощі та обмеження ресурсів, отримані теоретичні знання та часткова реалізація проекту поклали міцний фундамент для подальшого розвитку та вдосконалення системи планування в університетському середовищі.

Проведене дослідження виявило значний потенціал використання цифрових технологій для оптимізації процесів планування та адміністрування в освіті, підкресливши важливість адаптації до сучасних вимог цифрової епохи. Отже, подальші дослідження мають зосередитись на розширенні функціональності вебплатформи, зокрема на інтеграції з іншими системами управління університетом, розробці алгоритмів для автоматизації процесів планування, та впровадженні інструментів для збору та аналізу даних, які допоможуть приймати обґрунтовані рішення щодо розвитку освітньої системи.

Рекомендується залучення додаткових ресурсів для долання існуючих технічних та ресурсних бар'єрів, а також активізація співпраці з IT-спеціалістами, освітніми експертами, та керівництвом університетів для забезпечення всебічної підтримки проекту. Подальше дослідження та розробка вебплатформи для планування розвитку університетської системи освіти не тільки сприятимуть покращенню освітніх процесів, але й стануть важливим кроком на шляху до створення адаптивної, ефективної та інноваційної освітньої екосистеми.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Грекова И. Кафедра : проза. Астрель : АСТ, 2011. 352 с.

2. Волконский С. Мои воспоминания : підручник. Москва : Искусство, 1992. 148 с.

3. Арас Е. Биологический закон Арндта-Шульца, феномен гормезиса : стаття. URL: https://elena-aras.ru/biologicheskij-zakon-arndta-shulca-ili-fenomengormezisa/?fbclid=IwAR2dI0m4D0UXRqw7JFQGwW0gVrGnwpgokFr47vSwIM9Z-gj4iKTCXDKGn-Y (дата звернення: 15.12.2023)

4. Синергія у єдностідистанційних форм навчання і науковихдосліджень для сталогорозвиткууніверситетськоїсистемиосвіти / Мещанінов О.П., Боровльова С.Ю., Дворецька С.В. / Науковіпраці: Науково-методичний журнал. – Т. 313. Вип. 301. Педагогічні науки. Миколаїв: Вид-во ЧНУ ім. Петра Могили, 2018. 10-14 с.

5. Зязюн І.А. Педагогіка добра: ідеали і реалії: науково-методичнийпосібник. – Київ: МАУП, 2000. 312 с.

6. Hegaret.P. 100 Specifications for the Open Web Platform and Counting: стаття. URL: https://www.w3.org/blog/2011/100-specifications-for-the-ope/ (дата звернення 17.12.2023)

7. Цапліна А. Що таке LMS та як підібрати собі LMS-систему: стаття. URL: https://sendpulse.ua/blog/learning-management-system (дата звернення: 17.12.2023).

8. Google Colaboratory. *Google Colab*: вебсайт. URL: https://colab.google/ (дата звернення: 22.12.2023).

9. TensorFlow API Versions. *Tensorflow*: вебсайт. URL: https://www.tensorflow.org/versions (дата звернення: 22.12.2023).

10. API Reference. *Matplotlib*: вебсайт. URL: https://matplotlib.org/stable/api/index.html (дата звернення: 22.12.2023).

11. React JavaScript-бібліотека для створення користувацьких інтерфейсів. *React*: вебсайт. URL: https://uk.legacy.reactjs.org/(дата звернення: 22.12.2023).

12. Сверчков В. Що має знати C# .Net розробник у 2023 році: стаття. URL: https://itvdn.com/ua/blog/article/cs-dnet-23 (дата звернення: 04.02.2024).

13. ASP.NET documentation. *Microsoft*: вебсайт. URL: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-2.1 (дата звернення: 04.02.2024).

14. Entity Framework Core. *Entity framework tutorial*: вебсайт. URL: https://www.entityframeworktutorial.net/efcore/entity-framework-core.aspx (дата звернення: 04.02.2024).

15. Swagger UI for visualizing APIs. *Swagger*: вебсайт. URL: https://swagger.io/tools/swagger-ui/ (дата звернення: 04.02.2024).

16. Загроза XSS (міжсайтовий скриптинг). *IT Блог*: вебсайт. URL: https://blog-it.com.ua/zahyst-vid-xss-csrf-ta-inshyh-typovyh-atak/ (дата звернення: 04.02.2024).

17. Загроза CSRF (міжсайтова подальша аутентифікація). *IT Блог*: вебсайт. URL: https://blog-it.com.ua/zahyst-vid-xss-csrf-ta-inshyh-typovyh-atak/ (дата звернення: 04.02.2024).

18. OpenAPI Specification. *Swagger*: вебсайт. URL: https://swagger.io/specification/ (дата звернення: 04.02.2024).

19. Content Security Policy (CSP). *Resources for Developers, by Developers*: вебсайт. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP (дата звернення: 04.02.2024).

20. What is SSL, TLS & HTTPS? *digicer*t: вебсайт. URL: https://www.digicert.com/what-is-ssl-tls-and-https (дата звернення: 04.02.2024).

21. Ващука Ф.Г. Інтеграція в європейський освітній простір: здобутки, проблеми, перспективи: монографія. Ужгород: ІВА, 2011. 375 с.

22. Косіюк М.М., Більовський К.Е., Лисак В.М. Автоматизована інформаційна система управління закладом вищої освіти «Електронний університет»: стаття. URL:

https://journal.iitta.gov.ua/index.php/itlt/article/view/5107/2104 (дата звернення: 04.02.2024).

23. About MOOCs. *MOOC*: вебсайт. URL: https://www.mooc.org/ (дата звернення: 23.12.2023).

24. Архітектура клієнт-сервер. *Електронна бібліотека*: вебсайт. URL: http://inter.ptngu.com/kompyuterni-merezhi/arhitektura-kliyent-server (дата звернення: 23.12.2023).

25. Вишнівський В.В., Гніденко М.П., Гайдур Г.І., Ільїн О.О. Організація дистанційного навчання. Створення електронних навчальних курсів та електронних тестів: навчальний посібник. Київ: ДУТ, 2014. – 140 с.

26. Blackboard App: Mobile Learning Solutions. *Blackboard* : вебсайт. URL: https://www.blackboard.com/teaching-learning/learning-management/mobile-learning-solutions (дата звернення: 04.01.2024).

27. Developer resource centre. *Moodle*: вебсайт. URL: https://moodledev.io/ (дата звернення: 04.01.2024).

28. The future of web design: how coding and webflow are revolutionizing digital agencies. *HTML5 Canvas Tutorials*: вебсайт. URL: https://www.html5canvastutorials.com/ (дата звернення: 04.01.2024).

29. Про Клас. *Support Google*: вебсайт. URL: https://support.google.com/edu/classroom/answer/6020279?hl=uk (дата звернення: 04.01.2024).

30. A flexible and secure foundstion for teaching and learning. *Google for Education:* вебсайт. URL: https://edu.google.com/workspace-for-education/editions/education-fundamentals/ (дата звернення: 04.01.2024).

31. Про затвердження Положення про вебплатформу дистанційного навчання "Всеукраїнська школа онлайн": наказ М-ва освіти і науки України від 16 червня 2023 р. № 746 *Вища школа* 2023 № 1094/40150.

32. Організація та здійснення моніторингу освітнього процесу та професійного рівня педагогічних працівників. *Всеосвіта*: вебсайт. URL:

https://vseosvita.ua/library/embed/01002xmr-092f.doc.html (дата звернення: 15.01.2024).

33. Організація дистанційного навчання в Moodle.

Організація дистанційного навчання в Moodle. *Освіта.UA:* вебсайт. URL: https://osvita.ua/vnz/high_school/72285/ (дата звернення: 16.01.2024).

34. UI/UX дизайн: принципи та методи створення зручного інтерфейсу користувача. *While Web Production:* вебсайт. URL: https://whileweb.com/uk/blog/uiux-dizajn-principi-ta-metodi-stvorennya-zruchnogo-interfejsu-koristuvacha/ (дата звернення: 15.01.2024).

35. Захист інформації від несанкціонованого доступу. *Підручник для студентів онлайн*: вебсайт. URL: https://stud.com.ua/43315/informatika/zahist_informatsiyi_nesanktsionovanogo_dostupu (дата звернення: 18.01.2024).

36. Резервне копіювання та відновлення даних. *СУЧАСНА ІНФОРМАТИКА*: вебсайт. URL: https://alextexnok.blogspot.com/p/v-behaviorurldefaultvmlo_1.html (дата звернення: 18.01.2024).

37. Захар О.Г., Тихонова Т.В. Інформаційно-освітнєсередовищєпідвищення кваліфікації як засіб забезпечення якісної післядипломної педагогічної освітивчителів інформатики: стаття. URL: https://openedu.kubg.edu.ua/journal/index.php/openedu/article/view/86/119 (дата звернення: 19.01.2024).

38. Корчажкина О.М. Стратегии повышения эффективности профессиональной деятельности учителя-предметника в условиях информатизации образования : наукова література. Мінськ: БГУ, 2010 р. 169-175 с.

39. Розробка односторінкової веб-платформи для популяризації кампанії “Небайдужі”. *UNFPA Україна*: вебсайт. URL: https://ukraine.unfpa.org/uk/62456486 (дата звернення: 20.01.2024).

40. Борис О.Типи веб-сайтів - 12 популярних веб-сайтів, створених у 2022 році: стаття. URL: https://mas-agency.com.ua/blog/article/?id=2 (дата звернення: 20.01.2024).

41. *Kats Y.* Learning Management Systems and Instructional Design: Best Practices in Online Education: наукова література. Пенсільванія: IGI Global, 2013р. 48 с.

42. Інформаційні системи і технології в статистиці. *Бібліотека економіста*: вебсайт. URL: https://library.if.ua/book/80/5658.html (дата звернення: 25.01.2024).

43. Що таке мозковий штурм та які техніки бувають. *GENESIS*: вебсайт. URL: https://www.gen.tech/post/sho-take-mozkovii-shturm-pravyla-ta-porady (дата звернення: 25.01.2024).

44. Пометун О.І., Романова Г.М. Інноваційні технології навчання: навчальний посібник. Київ: НТУ, 2017р. 172 с.

45. 10 причин, чому CI/CD важливі для DevOps. *Cloudfresh*: вебсайт. URL: https://cloudfresh.com/ua/cloud-blog/10-prichin-chomu-ci-cd-vazhlivi-dlya-devops/ (дата звернення: 25.01.2024).

ДОДАТОК А

Лістинг коду вебплатформи

Файл /Client/src/App.js

```
import React from "react";
import { connect } from 'react-redux'
import { Switch, Route, Redirect } from "react-router-dom";

import { Auth, Home } from 'pages';

const App = (props) => {
  const { isAuth } = props;
  return (
    <Switch>
      <Route exact path={['/signin', '/signup']} component={Auth} />
      <Route path="/" render={() => (isAuth ? <Home /> : <Redirect to="/signin" />)} />
    </Switch>
  );
}

export default connect(({ user }) => ({ isAuth: user.isAuth }))(App);
```

Файл /Client/src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter as Router } from 'react-router-dom'
import { Provider } from "react-redux";

import App from './App';
import { userActions } from "redux/actions";

import store from "redux/store";

import 'semantic-ui-css/semantic.min.css'
import './index.css';

store.dispatch(userActions.fetchUserData());

ReactDOM.render(
  <Provider store={store}>
    <Router>
      <App />
    </Router>
  </Provider>,
  document.getElementById('root'));
```

Файл /Client/src/redux/store.js

```
import { createStore, applyMiddleware, compose } from "redux";
import thunk from "redux-thunk";

import rootReducer from "./reducers";

const composeEnhancers = window.__REDUX_DEVTOOLS_EXTENSION_COMPOSE__ || compose;

const middleware = [thunk];

const store = createStore(
  rootReducer,
  composeEnhancers(applyMiddleware(...middleware))
```

```
);

export default store;
```

Файл \Client\src\components\EditPostItem\ index.jsx

```
import React, { useState } from 'react';
import { Button, Item, Form, TextArea, Header, Icon } from 'semantic-ui-react';
import { formatDistance } from 'date-fns';
import { uk } from 'date-fns/locale';
import { connect } from 'react-redux';

import { postActions } from 'redux/actions';
import openNotification from 'utils/helpers/openNotification';

const EditPostItem = ({ image, header, description, date, id, fetchEditPost, fetchDeletePost }) => {
  const [isEditing, setIsEditing] = useState(false);
  const [isSaving, setIsSaving] = useState(false);
  const [isDeleting, setIsDeleting] = useState(false);
  const [editedText, setEditedText] = useState({
    header: '',
    description: '',
    image: '',
  });

  const handleEdit = (e) => {
    if (isEditing) {
      setIsEditing(false);
    } else {
      setIsEditing(true);
      setEditedText({
        header,
        description,
        image,
      });
    }
  };

  const handleSave = (e) => {
    if (
      editedText.header !== header ||
      editedText.description !== description ||
      editedText.image !== image
    ) {
      setIsSaving(true);
      fetchEditPost(
        {
          name: editedText.header,
          content: editedText.description,
          imageLink: editedText.image,
        },
        id,
      ).then(() => {
        setIsEditing(false);
        setIsSaving(false);
        openNotification({
          title: 'Отлично!',
          text: 'Пост обновлен',
          type: 'success',
        });
      });
    } else {
```

```
    openNotification({
      text: 'Пост остался без изменений',
      type: 'info',
    });
    setIsEditing(false);
  }
};

const handleDelete = (e) => {
  setIsDeleting(true);
  fetchDeletePost(id).then(() => {
    setIsDeleting(false);
    openNotification({
      title: 'Отлично!',
      text: 'Пост удален',
      type: 'success',
    });
  });
};

const handleChange = (e) => {
  setEditedText({ ...editedText, [e.target.name]: e.target.value });
};

const dateToWord = (date) => formatDistance(date, new Date(), { addSuffix: true, locale: uk });

return (
  <Item>
    <Item.Content>
      <Form>
        <Item.Meta style={{ marginTop: 0 }}>{dateToWord(date)}</Item.Meta>
        <Button
          loading={isDeleting}
          onClick={handleDelete}
          floated="right"
          circular
          color="red"
          inverted
          icon={<Icon name="trash alternate outline" color="red" />}
        />
        <Button
          onClick={handleEdit}
          floated="right"
          circular
          color="grey"
          inverted
          icon={<Icon name="pencil" color="grey" />}
        />
        {isEditing ? (
          <>
            <Item.Image
              src={editedText.image || 'https://le-go.net/images/noimage.gif'}
              size="small"
              floated="left"
            />
            <Button
              loading={isSaving}
              onClick={handleSave}
              floated="right"
              circular
```

```
            color="green"
            inverted
            icon={<Icon name="save outline" color="green" />}
           />
           <Form.Field style={{ display: 'inline-block' }} width={9}>
            <input
             placeholder="Заголовок"
             type="text"
             value={editedText.header}
             name="header"
             onChange={handleChange}
            />
           </Form.Field>
           <Form.Field style={{ display: 'inline-block' }} width={9}>
            <input
             placeholder="Посилання на зображення"
             type="text"
             value={editedText.image}
             name="image"
             onChange={handleChange}
            />
           </Form.Field>
           <TextArea
            placeholder="Основний контент"
            style={{ minHeight: 100 }}
            value={editedText.description}
            name="description"
            onChange={handleChange}
           />
          </>
         ) : (
          <>
           <Item.Image
            src={image || 'https://le-go.net/images/noimage.gif'}
            size="small"
            floated="left"
           />
           <Header style={{ marginTop: 0 }} content={header} />
           <Item.Description style={{ textAlign: 'justify' }}>{description}</Item.Description>
          </>
         )}
        </Form>
      </Item.Content>
    </Item>
  );
};

export default connect(null, postActions)(EditPostItem);
```

Файл \EduNetHub\EduNetHubApi\Program.cs

```
using EduNetHubApi.Data;
using EduNetHubApi.Options;
using EduNetHubApi.ServiceInterfaces;
using EduNetHubApi.Services;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using Microsoft.OpenApi.Models;
using System.Text;
```

```
namespace EduNetHubApi
{
public class Program
{
public static void Main(string[] args)
{
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddCors();
builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

builder.Services.AddDbContext<DataContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

builder.Services.AddIdentity<IdentityUser, IdentityRole>(config =>
{
config.Password.RequireNonAlphanumeric = false;
})
.AddRoles<IdentityRole>()
.AddRoleManager<RoleManager<IdentityRole>>()
.AddEntityFrameworkStores<DataContext>();

builder.Services.AddScoped<IPostService, PostService>();
builder.Services.AddScoped<IImageService, ImageService>();
builder.Services.AddScoped<IIdentityService, IdentityService>();

var jwtSettings = new JwtSettings();
builder.Configuration.Bind(nameof(jwtSettings), jwtSettings);
builder.Services.AddSingleton(jwtSettings);

var tokenValidationParameters = new TokenValidationParameters
{
ValidateIssuerSigningKey = true,
IssuerSigningKey = new SymmetricSecurityKey(Encoding.ASCII.GetBytes(jwtSettings.Secret)),
ValidateIssuer = false,
ValidateAudience = false,
RequireExpirationTime = false,
ValidateLifetime = true
};
builder.Services.AddSingleton(tokenValidationParameters);

builder.Services.AddAuthentication(x =>
{
x.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
x.DefaultScheme = JwtBearerDefaults.AuthenticationScheme;
x.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(x =>
{
x.SaveToken = true;
x.TokenValidationParameters = tokenValidationParameters;
});

builder.Services.AddSwaggerGen(x =>
{
x.SwaggerDoc("v1", new OpenApiInfo { Title = "EduNetHub", Version = "v1" });
```

```
x.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme
{
Description = "JWT Authorization header using the bearer scheme",
Name = "Authorization",
In = ParameterLocation.Header,
Type = SecuritySchemeType.ApiKey,
Scheme = "Bearer"
});

x.AddSecurityRequirement(new OpenApiSecurityRequirement
{
{
new OpenApiSecurityScheme
{
Reference = new OpenApiReference
{
Type = ReferenceType.SecurityScheme,
Id = "Bearer"
},
Scheme = "oauth2",
Name = "Bearer",
In = ParameterLocation.Header,

},
new List<string>()
}
});
});
var app = builder.Build();

if (app.Environment.IsDevelopment())
{
app.UseSwagger();
app.UseSwaggerUI();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseAuthorization();
app.UseAuthentication();

var swaggerOptions = new SwaggerOptions();
builder.Configuration.GetSection(nameof(SwaggerOptions)).Bind(swaggerOptions);

app.UseSwagger(option => { option.RouteTemplate = swaggerOptions.JsonRoute; });

app.UseSwaggerUI(option =>
{
option.SwaggerEndpoint(swaggerOptions.UiEndpoint, swaggerOptions.Description);
});

app.UseCors(options => {
options.AllowAnyOrigin();
options.AllowAnyHeader();
options.AllowAnyMethod();
});

app.MapControllers();
```

```
app.Run();
}
}
}
```

Файл /EduNetHub/EduNetHubApi/appsettings.json

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=(local);Database=EduNetHub;Integrated Security=true;"
  },
  "JwtSettings": {
    "Secret": "B3E1AAC8A5D4472986AD022AE766ED1D",
    "TokenLifetime": "01:00:00"
  },
  "SwaggerOptions": {
    "JsonRoute": "swagger/{documentName}/swagger.json",
    "Description": "EduNetHub API",
    "UiEndpoint": "v1/swagger.json"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

Файл /EduNetHub/EduNetHubApi/Controllers/IdentityController.cs

```
using EduNetHubApi.Contracts;
using EduNetHubApi.Contracts.Requests;
using EduNetHubApi.Contracts.Responses;
using EduNetHubApi.Extentions;
using EduNetHubApi.ServiceInterfaces;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace EduNetHubApi.Controllers
{
public class IdentityController : Controller
    {
        private readonly IIdentityService _identityService;

        public IdentityController(IIdentityService identityService)
        {
            _identityService = identityService;
        }

        [HttpPost(ApiRoutes.Identity.Register)]
        public async Task<IActionResult> Register([FromBody] UserRegistrationRequest request)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(new AuthFailedResponse
                {
                    Errors = ModelState.Values.SelectMany(x => x.Errors.Select(xx => xx.ErrorMessage))
                });
            }

            var authResponse = await _identityService.RegisterAsync(request.Email, request.Password);

            if (authResponse is null)
```

```
    {
        return BadRequest(new AuthFailedResponse
        {
            Errors = new[] { "Something went wrong" }
        });
    }

    if (!authResponse.Success)
    {
        return BadRequest(new AuthFailedResponse
        {
            Errors = authResponse.Errors
        });
    }

    return Ok(new AuthSuccessResponse
    {
        Token = authResponse.Token,
        RefreshToken = authResponse.RefreshToken
    });
}

[HttpPost(ApiRoutes.Identity.Login)]
public async Task<IActionResult> Login([FromBody] UserLoginRequest request)
{
    var authResponse = await _identityService.LoginAsync(request.Email, request.Password);

    if (!authResponse.Success)
    {
        return BadRequest(new AuthFailedResponse
        {
            Errors = authResponse.Errors
        });
    }

    return Ok(new AuthSuccessResponse
    {
        Token = authResponse.Token,
        RefreshToken = authResponse.RefreshToken
    });
}

[HttpPost(ApiRoutes.Identity.Refresh)]
public async Task<IActionResult> Refresh([FromBody] RefreshTokenRequest request)
{
    var authResponse = await _identityService.RefreshTokenAsync(request.Token, request.RefreshToken);

    if (!authResponse.Success)
    {
        return BadRequest(new AuthFailedResponse
        {
            Errors = authResponse.Errors
        });
    }

    return Ok(new AuthSuccessResponse
    {
        Token = authResponse.Token,
        RefreshToken = authResponse.RefreshToken
    });
```

```
        }

        [Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
        [HttpGet(ApiRoutes.Identity.GetMe)]
        public async Task<IActionResult> GetMe()
        {
            var userData = await _identityService.GetUserDataAsync(HttpContext.GetUserId());

            if (userData is null) return NotFound();

            return Ok(userData);
        }
    }
}
```

Файл /EduNetHub/EduNetHubApi/Controllers/PostsController.cs

```
using EduNetHubApi.Contracts;
using EduNetHubApi.Contracts.Requests;
using EduNetHubApi.Contracts.Responses;
using EduNetHubApi.Extentions;
using EduNetHubApi.Models;
using EduNetHubApi.ServiceInterfaces;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace EduNetHubApi.Controllers
{
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
public class PostsController : Controller
{
private readonly IPostService _postService;
private readonly IImageService _imageService;

public PostsController(IPostService postService, IImageService imageService)
{
_postService = postService;
_imageService = imageService;
}

[HttpGet(ApiRoutes.Posts.GetAll)]
public async Task<IActionResult> GetAll()
{
var posts = await _postService.GetPostsAsync();

var response = posts.Select(async post =>
{
if (post.ImageId.HasValue)
post.Image = await _imageService.GetImageByIdAsync(post.ImageId.Value);
return MapPostToPostResponse(post);
})
.Select(t => t.Result)
.ToList();

return Ok(response);
}

[HttpPut(ApiRoutes.Posts.Update)]
public async Task<IActionResult> Update([FromRoute] Guid postId, [FromBody] UpdatePostRequest request)
{
var userOwnsPost = await _postService.UserOwnsPostAsync(postId, HttpContext.GetUserId());
```

```
var IsAdmin = HttpContext.HasUserAdminRole();

if (!userOwnsPost && !IsAdmin)
return BadRequest(new { error = "You do not own this post" });

Image image;
if (!string.IsNullOrEmpty(request.ImageLink))
{
image = new Image
{
Id = new Guid(),
Path = request.ImageLink,
Name = new Guid().ToString()
};

await _imageService.CreateImageAsync(image);
}
else
{
image = new Image();
}

var post = await _postService.GetPostByIdAsync(postId);

        if (post is null)
return BadRequest(new { error = "Post is missing" });

post.Name = request.Name;
post.Content = request.Content;
post.Image = image;

var updated = await _postService.UpdatePostAsync(post);

if (!updated)
return NotFound();

return Ok(MapPostToPostResponse(post));
}

[HttpDelete(ApiRoutes.Posts.Delete)]
public async Task<IActionResult> Delete([FromRoute] Guid postId)
{
var userOwnsPost = await _postService.UserOwnsPostAsync(postId, HttpContext.GetUserId());
var IsAdmin = HttpContext.HasUserAdminRole();

if (!userOwnsPost && !IsAdmin)
{
return BadRequest(new { error = "You do not own this post" });
}

var deleted = await _postService.DeletePostByIdAsync(postId);

if (deleted)
return NoContent();

return NotFound();
}

[HttpGet(ApiRoutes.Posts.GetById)]
public async Task<IActionResult> Get([FromRoute] Guid postId)
```

```
{
var post = await _postService.GetPostByIdAsync(postId);

if (post is null)
return NotFound();

if (post.ImageId.HasValue)
post.Image = await _imageService.GetImageByIdAsync(post.ImageId.Value);

return Ok(MapPostToPostResponse(post));
}

[HttpPost(ApiRoutes.Posts.Create)]
public async Task<IActionResult> Create([FromBody] CreatePostRequest postRequest)
{
Image? image = null;
if (!string.IsNullOrEmpty(postRequest.ImageLink))
{
image = new Image
{
Id = new Guid(),
Path = postRequest.ImageLink,
Name = new Guid().ToString()
};

await _imageService.CreateImageAsync(image);
}

var post = new Post
{
Name = postRequest.Name,
UserId = HttpContext.GetUserId(),
Content = postRequest.Content,
Date = DateTime.Now,
ImageId = image?.Id
};

await _postService.CreatePostAsync(post);

var baseUrl = $"{HttpContext.Request.Scheme}://{HttpContext.Request.Host.ToUriComponent()}";
var locationUri = baseUrl + "/" + ApiRoutes.Posts.GetById.Replace("{postId}", post.Id.ToString());

var response = MapPostToPostResponse(post);

return Created(locationUri, response);
}

private static PostResponse MapPostToPostResponse(Post post)
{
return new PostResponse
{
Id = post.Id,
Date = post.Date,
Content = post.Content,
Name = post.Name,
ImageLink = post.Image?.Path
};
}
}
```