

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
ВЕБПЛАТФОРМИ ПІДТРИМКИ ОСВІТНЬОГО ПРОЦЕСУ
RESEARCH AND SOFTWARE IMPLEMENTATION OF A WEB
PLATFORM FOR SUPPORTING THE EDUCATIONAL PROCESS

Виконав(ла): студент(ка) 2 курсу, групи КНТ-114м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ХАЛІЗЕВ Н.С.

(ПРИЗВИЩЕ та ініціали)

Керівник ГЛАДКОВА О.М.

(ПРИЗВИЩЕ та ініціали)

Рецензент ДЬЯЧУК Т.С.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(КОД І НАЙМЕНУВАННЯ)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ХАЛІЗЕВА Нікіти Сергійовича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація вебплатформи підтримки освітнього процесу Research and Software Implementation of a Web Platform for Supporting the Educational Process

керівник проєкту (роботи) к.т.н., ГЛАДКОВА Ольга Миколаївна,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 30 ” вересня 2025 року № 447

2. Строк подання студентом проєкту (роботи) 08 грудня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	ГЛАДКОВА О.М., доцент		
Нормоконтролер	БСЛОВА А.В., асистент		

7. Дата видачі завдання “30” вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2-3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	4-5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7-8 тижні	Розділ 3
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	10-11 тижні	Додатки
8	Нормоконтроль та рецензування.	12 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка)

(підпис) Нікіта ХАЛІЗЕВ
(Г'мя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Ольга ГЛАДКОВА
(Г'мя ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
133 с., 26 табл., 61 рис., 3 дод., 33 джерела.

ВЕБПЛАТФОРМА, ОСВІТНЯ ПЛАТФОРМА, LMS, JAVASCRIPT, TYPESCRIPT, REACT, C#, ASP.NET, MINIMAL API.

Об'єкт дослідження – процес розробки програмного забезпечення для підтримки освітнього процесу.

Предмет дослідження – програмні засоби реалізації вебплатформи для підтримки освітнього процесу.

Мета роботи – зменшення часових витрат на організацію навчального процесу та підвищення оперативності взаємодії між його учасниками шляхом розробки вебплатформи зі застосуванням сучасних методів програмної інженерії.

Матеріали, методи та технічні засоби: вебпрограмування, мови програмування C# та Typescript, фреймворки ASP.NET та React, інтегровані середовища розробки Visual Studio 2026 та Visual Studio Code, персональний комп'ютер з процесором AMD Ryzen 5 3600 під управлінням операційної системи Microsoft Windows 11 Pro.

Результати. Створено вебплатформу підтримки освітнього процесу, яка забезпечує зручне управління навчальними матеріалами, взаємодію між викладачами та студентами, а також автоматизує низку операцій – від публікації завдань до збору результатів та аналітики активності.

Висновки. Створена вебплатформа скорочує час організації навчального процесу, підвищує доступність навчальних матеріалів та покращує комунікацію між учасниками освітнього середовища.

Галузь використання – заклади освіти, онлайн-курси, корпоративні навчальні системи, внутрішні освітні портали та платформи дистанційного навчання.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 133 pages, 26 tables, 61 figures, 3 appendixes, 33 sources.

WEB PLATFORM, EDUCATIONAL PLATFORM, LMS, JAVASCRIPT, TYPESCRIPT, REACT, C#, ASP.NET, MINIMAL API.

The object of study is the software development process for supporting the educational process.

The subject of study is the software tools for implementing a web platform to support the educational process.

The purpose of the work is to reduce the time costs for organizing the educational process and to increase the efficiency of interaction between its participants by developing a web platform using modern software engineering methods.

Materials, methods, and tools: web programming, C# and TypeScript programming languages, ASP.NET and React frameworks, Visual Studio 2026 and Visual Studio Code integrated development environments, a personal computer with an AMD Ryzen 5 3600 processor under the Microsoft Windows 11 Pro operating system.

Results. A web platform for supporting the educational process has been created, which provides convenient management of learning materials, interaction between teachers and students, and automates a range of operations – from publishing assignments to collecting results and activity analytics.

Conclusions. The created web platform reduces the time required to organize the learning process, increases the accessibility of educational materials, and improves communication between participants in the educational environment.

The field of use includes educational institutions, online courses, corporate training systems, internal educational portals, and distance learning platforms.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок.....	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Короткий опис досліджувальної предметної області.....	11
1.2 Огляд існуючих рішень.....	11
1.3 Постановка завдання.....	17
1.4 Висновки за розділом.....	18
2 Матеріали і методи.....	19
2.1 Вибір мови програмування.....	19
2.2 Вибір фреймворків.....	26
2.3 Вибір системи управління базами даних.....	29
2.4 Вибір середовища розробки.....	33
2.5 Висновки за розділом.....	35
3 Опис програми.....	37
3.1 Загальна архітектура системи.....	37
3.2 Проектування бази даних.....	38
3.3 Алгоритми та функціонування системи.....	54
3.4 Проектування інтерфейсу користувача.....	58
3.5 Висновки за розділом.....	78
4 Експлуатація, тестування та експериментальне дослідження програми.....	79
4.1 Призначення й умови застосування програми.....	79
4.1 Тестування програми.....	79
4.3 Висновки за розділом.....	82
Висновки.....	83
Перелік джерел посилання.....	84
Додаток А Технічне завдання.....	88
Додаток Б Текст програми.....	93
Додаток В Слайди презентації.....	125

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
ACID	– Atomicity, Consistency, Isolation, Durability;
FK	– Foreign key;
HTTP	– Hypertext Transfer Protocol;
IDE	– Integrated Development Environment,
JSON	– JavaScript Object Notation;
LINQ	– Language Integrated Query;
LMS	– Learning Management System;
PK	– Primary key;
REST	– Representational State Transfer;
SQL	– Structured Query Language;
URL	– Uniform Resource Locator;
БД	– база даних;
ПЗ	– програмне забезпечення;
СУБД	– система управління базами даних.

ВСТУП

Цифрова трансформація освіти є одним із найактуальніших викликів сучасного суспільства. Останнє десятиліття характеризується значним прискоренням процесів діджиталізації в усіх сферах діяльності, включаючи освітню галузь. Цифрова трансформація у сфері освіти – це комплексна робота над побудовою екосистеми цифрових рішень, включно зі створенням безпечного електронного освітнього середовища, забезпеченням необхідної цифрової інфраструктури закладів освіти, підвищенням рівня цифрової компетентності та автоматизацією збору й аналізу даних [1]. Пандемія COVID-19 та умови воєнного стану стали каталізаторами масового переходу на дистанційне навчання, змусивши заклади освіти терміново шукати ефективні інструменти для продовження навчального процесу [2].

У цьому контексті ключову роль відіграють системи управління навчанням (LMS – Learning Management System). Вони виступають цифровим майданчиком, де викладачі розміщують курси, а студенти мають змогу їх проходити, що забезпечує централізоване управління процесами. Статистика свідчить про стрімке зростання ринку таких систем, оскільки понад 73% компаній та більшість освітніх закладів вже використовують цифрові платформи [2].

Попри наявність широкого спектра LMS-рішень, значна частина з них має суттєві обмеження: складність адаптації під конкретні освітні потреби, складний інтерфейсу, обмежені можливості інтеграції та розширення функціоналу [3]. Багато з них є вимогливими до ресурсів і складними у підтримці. Це зумовлює необхідність розробки легковагових, гнучких та високопродуктивних вебплатформ, побудованих на сучасному стеку технологій. Такий підхід дозволяє створити швидке, масштабоване та зручне середовище для організації освітнього процесу, що і визначає актуальність теми магістерської роботи.

Мета роботи полягає у створенні вебплатформи для підтримки освітнього процесу, яка забезпечує ефективне управління навчальною інформацією, комунікацію учасників та автоматизацію типових завдань освітнього середовища, із застосуванням сучасних методів програмної інженерії.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Короткий опис досліджувальної предметної області

Сучасні тенденції освітньої сфери характеризуються стрімкою інтеграцією інформаційних технологій, що докорінно змінило підходи до організації навчального процесу. Вищі навчальні заклади та освітні організації активно впроваджують інформаційні платформи для управління освітнім процесом, зберігання та обробки великих масивів даних про студентів, викладачів та дослідження [4]. Основою для цієї трансформації стали Системи управління навчанням (LMS), які забезпечують інклюзивне навчальне середовище та підтримують академічний прогрес через інструменти для співпраці, дискусій та професійної підготовки [5].

Предметна область дипломної роботи охоплює комплекс програмних та методологічних засобів, спрямованих на організацію дистанційного та змішаного навчання. Ключовим елементом тут виступає LMS – програмне забезпечення, що функціонує як каркас для охоплення багатьох рівнів прогресивного навчання [5].

1.2 Огляд існуючих рішень

Ринок сучасних систем управління навчанням (LMS) пропонує широкий вибір рішень, які можна класифікувати на пропрієтарні системи (комерційні платформи із закритим кодом) та системи з відкритим вихідним кодом (Open Source) [5]. Вибір між ними залежить від наявних ресурсів, необхідного рівня технічної підтримки та експертизи персоналу освітнього закладу.

Пропрієтарні системи надають готові модулі, які пройшли перевірку якості, та забезпечують постійною технічною підтримкою, тоді як Open Source рішення дозволяють адаптувати вихідний код під специфічні потреби без витрат на ліцензійні збори [6].

1.2.1 Canvas LMS

Canvas – це потужна хмарна LMS, розроблена компанією Instructure, яка активно використовується у багатьох країнах світу та демонструє високу ефективність в організації дистанційного навчання [4]-[6]. Система вирізняється інтуїтивно зрозумілим інтерфейсом (рис. 1.1) та адаптивністю до різних методів викладання [6].

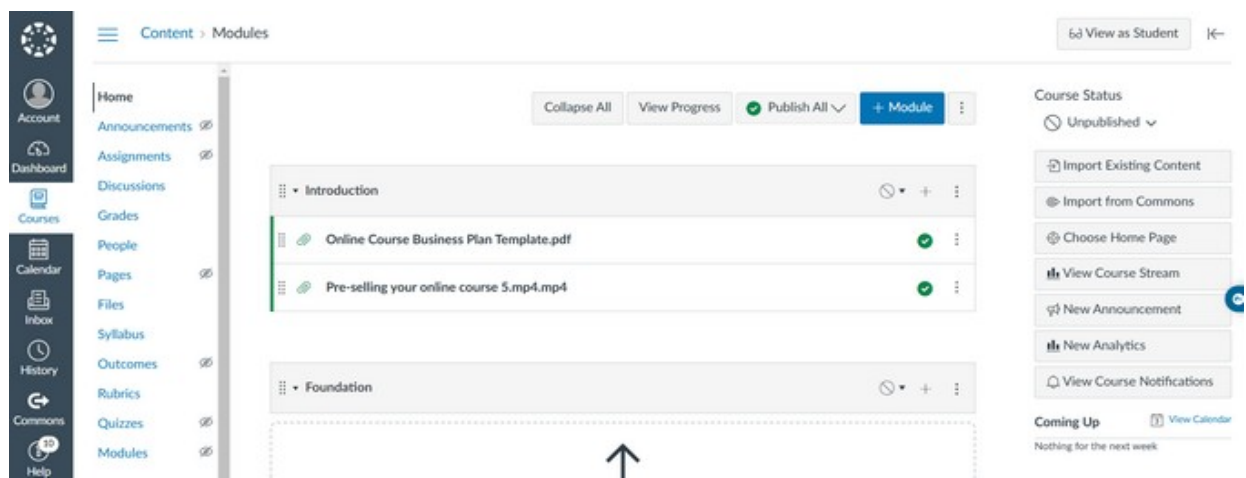


Рисунок 1.1 — Платформа Canvas LMS [7]

Платформа пропонує широкий спектр інструментів, що допомагають ефективно організувати навчання, включаючи: створення курсів, організацію завдань та взаємодію зі студентами [7]. Canvas балансує між простою використання та передовими педагогічними можливостями [8].

Переваги:

- інтуїтивний інтерфейс та дизайн;
- персоналізація навчання, функція «Mastery Paths» дозволяє автоматично персоналізувати навчання на основі успішності студента [8];
- інструменти оцінювання;
- наявність технічної підтримки.

Недоліки:

- вартість, Canvas є одним з найдорожчих рішень для організацій, де вартість ліцензії залежить від кількості користувачів та обраного пакету послуг [9];
- залежність від вендора, усі дані зберігаються у хмарі провайдера, що створює залежність від постачальника послуг щодо безпеки та доступності сервісу [5];
- наявна відкрита (Open Source) версія, функціонально обмежена відносно хмарної.

1.2.2 Moodle

Moodle є однією із найпопулярніших платформ з відкритим вихідним кодом (Open Source). Вона дозволяє навчальним закладам адаптувати програмне забезпечення під свої специфічні потреби без сплати ліцензійних зборів, проте вимагає власної технічної експертизи для налаштування та підтримки [5]-[6]. Платформа широко використовується університетами у Великій Британії та Канаді [6]. На рис. 1.2 продемонстровано інтерфейс платформи.

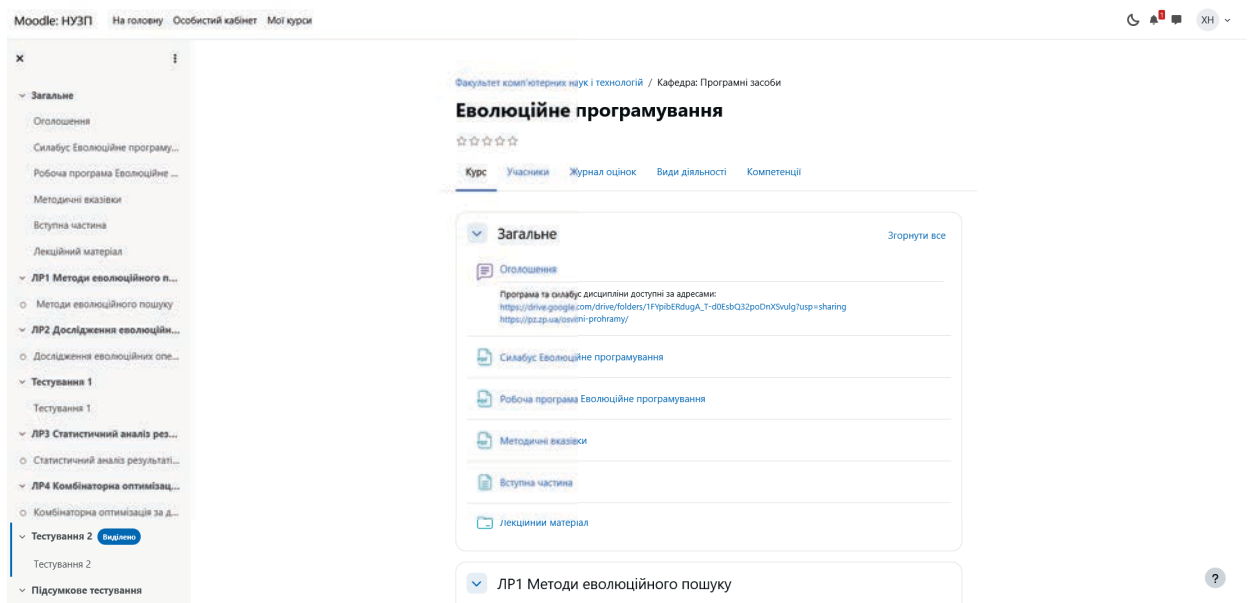


Рисунок 1.2 — Платформа Moodle

Оскільки це система з відкритим вихідним кодом, вона дозволяє повну доробку та кастомізацію, але вимагає від навчального закладу наявності власного сервера або хмарного хостингу та технічної експертизи для обслуговування [8]-[9].

Переваги:

- гнучкість та кастомізація;
- широкі педагогічні можливості;
- масштабованість;
- відсутність ліцензійних зборів.

Недоліки:

- складність розгортання;
- складний та перевантажений інтерфейс [7];
- відсутність технічної підтримки.

1.2.3 Google Classroom

Google Classroom – це інструмент, який допомагає викладачам легко створювати, організовувати та керувати онлайн-класами [7]. Він сфокусований на спрощенні управління курсами та сприянні співпраці між викладачами та студентами завдяки тісній інтеграції з сервісам Google G Suit (Google Docs, Sheets, Drive) [8]. На рис. 1.3 продемонстровано інтерфейс платформи.

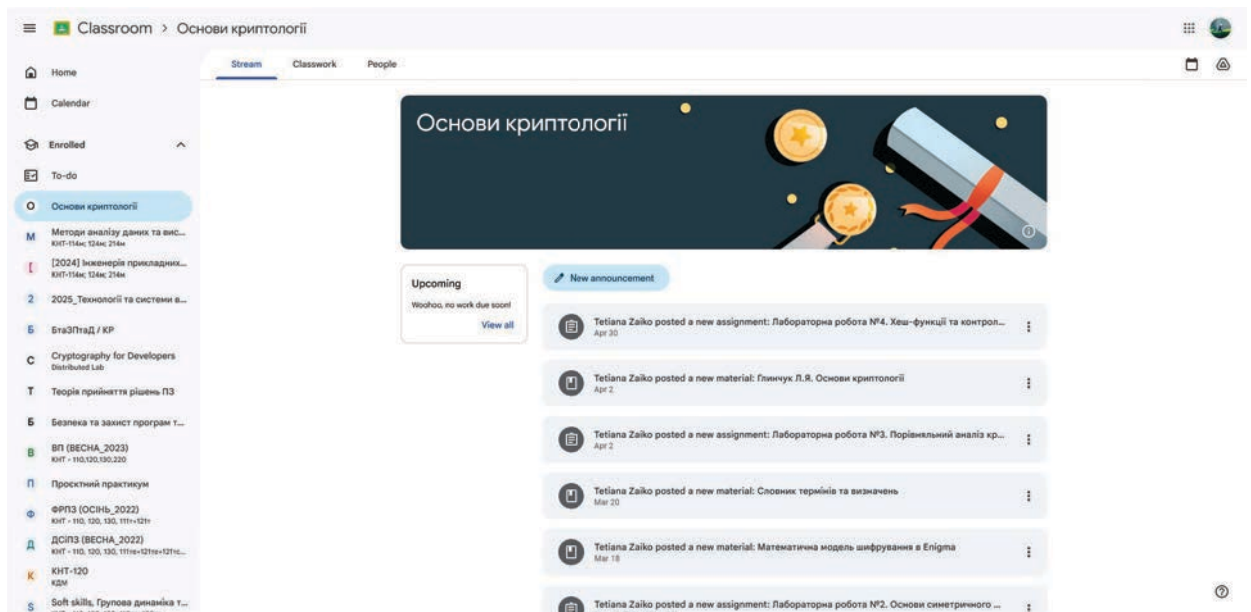


Рисунок 1.3 — Платформа Google Classroom

Це рішення ідеально підходить для тих, хто шукає простий та швидкий спосіб призначати домашні завдання, надавати швидкий зворотний зв'язок та підтримувати все в організованому стані без зайвих ускладнень [7].

Переваги:

- простота використання;
- швидке розгортання;
- глибока інтеграція з Google сервісами;
- безкоштовність;
- зручність комунікації.

Недоліки:

- обмежений педагогічний функціонал;
- відсутність кастомізації;
- залежність від екосистеми Google.

1.2.4 Порівняльний аналіз існуючих рішень

У табл. 1.1 наведено порівняльний аналіз усіх розглянутих вебплатформ для підтримки освітнього процесу.

Таблиця 1.1 — Порівняння існуючих рішень

Критерій порівняння	Canvas LMS	Moodle	Google Classroom
Тип системи	Хмарна (SaaS), або Open Source	Open Source, вимагає хостингу	Хмарна, пропрієтарна
Вартість	Висока вартість корпоративної ліцензії. Наявний безкоштовний план з обмеженим функціоналом	Безкоштовне ПЗ (витрати на хостинг та підтримку ПЗ)	Безкоштовна
Гнучкість та кастомізація	Дуже висока (плагіни, відкритий вихідний код)	Дуже висока (плагіни, відкритий вихідний код)	Низька (стандартний інтерфейс)
Простота розгортання	Дуже проста і швидка (підтримка вендора)	Складна (потребує ІТ компетенцій)	Дуже проста та швидка
Педагогічні можливості	Збалансовані	Просунуті	Базові
Аналітика та звітність	Розширена візуалізація даних	Детальні логи активності	Базова, обмежена візуалізація даних
Цільова аудиторія	Середні та великі освітні заклади, бізнес	Університети, що потребують адаптації ПЗ	Малі курси

1.3 Постановка завдання

Проведений аналіз предметної області та існуючих рішень показав, що ринок систем управління навчанням (LMS) є насиченим, проте більшість наявних платформ мають суттєві обмеження для конкретних умов використання. Комерційні рішення, такі як Canvas LMS, є високовартісними, а їх самостійне розгортання ускладнене високими технічними вимогами. Відкриті системи, зокрема Moodle, характеризуються надмірною складністю інтерфейсу та потребують значних ресурсів для адміністрування. Легковагові рішення, наприклад Google classroom, часто не забезпечують достатньої гнучкості та повноти функціоналу для організації комплексного навчального процесу.

У зв'язку з цим виникає потреба у розробці власної вебплатформи, яка поєднає зручність використання, сучасний інтерфейс та необхідний набір функцій для ефективної взаємодії викладачів і студентів.

Виходячи з аналізу переваг і недоліків існуючих систем, до розроблюваної вебплатформи висувають наступні функціональні вимоги:

- забезпечення реєстрації та авторизації користувачів (студентів, викладачів, адміністраторів);
- створення та керування навчальними курсами;
- публікація навчальних матеріалів різних форматів;
- створення завдань та можливість завантаження рішень студентами;
- оцінювання виконаних робіт та надання зворотного зв'язку;
- створення, проходження та оцінювання тестових завдань;
- перегляд статистики кожного окремого завдання та тесту;
- перегляд успішності студентів.

1.4 Висновки за розділом

У першому розділі було здійснено аналіз предметної області, пов'язаної з цифровою освітою та використанням систем управління навчанням (LMS). Встановлено, що впровадження цифрових платформ є критично важливим етапом модернізації освітнього процесу, особливо в умовах дистанційного навчання.

У ході огляду існуючих рішень було проаналізовано такі популярні системи, як Canvas LMS, Moodle та Google Classroom.

На основі виявлених, під час аналізу, недоліків аналогів обґрунтовано актуальність розробки власної вебплатформ для підтримки освітнього процесу. Нова система має бути позбавлена надмірної складності Moodle, бути економічно вигіднішою за Canvas LMS та функціональнішою за Google Classroom.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

2.1.1 Вибір мови програмування для серверної частини

У процесі розробки серверної частини вебзастосунків (бекенду) критично важливим є вибір технологічного стеку, який відповідає вимогам проєкту щодо продуктивності, масштабованості та швидкості розробки [10] – [11]. Обрана мова програмування впливає на всі аспекти життєвого циклу програмного забезпечення: від швидкості написання коду та налагодження до підтримки спільноти та простоти розгортання [10].

На сьогоднішній день для веброзробки широко використовуються такі мови як C#, PHP та Ruby. Кожна з них має унікальну філософію та технічні особливості, які необхідно враховувати при проектуванні архітектури системи [10]-[11].

2.1.1.1 C#

C# – це об'єктно-орієнтована мова зі статичною типізацією, розроблена компанією Microsoft, яка відома своєю надійністю та адаптивністю у веброзробці [11]-[12].

До основних характеристик C# належить:

- строга типізація, де компілятор забезпечує перевірку типів [11];
- розвинена екосистема, мова нерозривно пов'язана з платформою .NET, який надає стабільні інструменти та бібліотеки для створення масштабованих застосунків [11];
- висока продуктивність, оскільки C# є компільованою мовою, вона, зазвичай, демонструє вищу продуктивність порівняно з інтерпретованими мовами [11];

2.1.1.2 PHP

PHP (Hypertext Preprocessor) – це скриптова мова серверної частини, яка була спеціально створена для веброзробки і станом на 2024 рік залишається однією з найпоширеніших у світі, забезпечуючи роботу 75.2% вебсайтів [10]-[11].

Основними рисами PHP є:

- простота та швидкість розробки, мов має низький поріг входу та дозволяє швидко розроблювати вебпроекти [10]-[11];
- динамічна типізація, це надає гнучкість, але може призводити до помилок під час виконання, які важко відстежити до моменту запуску в продакшн [13];
- розвинена екосистема фреймворків, сучасно розробка на PHP базується на потужних фреймворках, таких як Laravel та Symfony, які пропонують структури та інструменти для підвищення ефективності розробки вебзастосунків [11];
- поширеність та велика кількість навчальних, та легкість розгортання на будь-якому хостингу роблять цю мову економічно ефективним вибором [10].

2.1.1.3 Ruby

Ruby – це високорівнева об'єктно-орієнтована мова, головною метою якої є підвищення продуктивності розробника та задоволення від процесу програмування [10].

Характерні особливості Ruby:

- елегантний синтаксис, код максимально наближений до природної мови, що робить його читабельним та лаконічним [14];
- фреймворк Ruby on Rails, популярність мови значною мірою зумовлена даним фреймворком, який сповідує принцип «Convention over

Configuration» (Угода понад конфігурацію), це дозволяє створювати складні вебзастосунки з мінімальною кількістю шаблонного коду [10], [14];

- швидкість прототипування робить Ruby відмінним вибором для стартапів та створення MVP (Minimum Viable Product) [14];
- головним недоліком є менша швидкість виконання коду порівняно з C# або PHP (у нових версіях), особливо у ресурсомістких завданнях [10].

2.1.1.4 Порівняння мов програмування для серверної частини

Для обґрунтованого вибору мови програмування було проведено порівняльний аналіз розглянутих технологій за ключовими критеріями. Результати аналізу наведено у табл. 2.1.

Таблиця 2.1 – Порівняння мов програмування серверної частини

Критерій порівняння	C#	PHP	Ruby
1	2	3	4
Тип типізація	Сильна, статична	Слабка, динамічна	Динамічна
Парадигма	Повністю об'єктно-орієнтована	Об'єктно-орієнтована, процедурна	Об'єктно-орієнтована
Продуктивність	Висока (компільована мова)	Середня (інтерпретована, є JIT)	Низька (поступається аналогам)
Основні фреймворки	ASP.NET Core	Laravel, Symfony	Ruby on Rails

Кінець таблиці 2.1.

1	2	3	4
Крива навчання	Помірна/Висока (складніша для новачків)	Низька (легка для вивчення)	Помірна (простий синтаксис, але складна «магія» Rails)
Швидкість розробки	Середня (вимагає більше коду)	Висока (швидкий старт)	Дуже висока (швидке прототипування)

Виходячи з проведеного аналізу, для реалізації серверної частини (бекенду) було обрано мову програмування C#. Цей вибір зумовлений необхідністю побудови надійної та масштабованої архітектури. Суворостатично типізація C# дозволяє уникнути багатьох помилок ще на етапі компіляції, а висока продуктивність платформа .NET є критичною для обробки складних завдань. Потужна підтримка з боку спільноти та наявність розвиненої документації забезпечують ефективність процесу розробки.

2.1.2 Вибір мови програмування для клієнтської частини

У сфері розробки програмного забезпечення вибір правильної мови програмування може суттєво вплинути на продуктивність застосунку, його успіх, зручність підтримки та ефективність [15]. Для розробки клієнтської частини (фронтенду) сьогодні найчастіше розглядаються такі мови програмування: JavaScript, TypeScript та Dart. Кожна з цих мов має свої унікальні особливості, переваги та недоліки, які варто зважити перед тим, як почати проєкт.

2.1.2.1 JavaScript

JavaScript – це фундаментальна та найпопулярніша мова для веброзробки, з якою знайомий майже кожен веброзробник [15]. Однією з головних причин широкого впровадження JavaScript є його універсальність: він може використовуватися як для клієнтських, так і для серверних застосунків [15].

До основних характеристик JavaScript належать:

- динамічна типізація, тобто ти змінної присвоюється під час виконання, залежно від їхнього значення, що дає розробникам гнучкість у роботі з різними типами даних без необхідності їх явного оголошення [15]-[16];
- висока швидкість розробки, завдяки відсутності етапу компіляції та динамічній природі, JavaScript дозволяє швидше створювати прототипи та прискорює процес розробки вебзастосунків [15]-[16];
- розвинена екосистема, забезпечується завдяки величезній спільноті та великій кількості готових фреймворків та бібліотек [17].

2.1.2.2 TypeScript

TypeScript – ця мова програмування є надбудовою над JavaScript, що означає, що будь-який валідний код JavaScript зазвичай є валідним кодом TypeScript [15]. Головною відмінністю є те, що TypeScript додає статичну типізацію та інші сучасні функції, яких бракує JavaScript [15].

Ключові особливості TypeScript:

- статична типізація дозволяє явно визначати типи даних у коді, що забезпечує перевірку коректності типів ще на етапі компіляції [16]. Це покращує передбачуваність коду та дозволяє виявляти помилки на ранніх стадіях [15];

- TypeScript краще підходить для великих проектів, орієнтованих на підтримку та масштабування, завдяки таким функціям, як інтерфейси, перерахування та дженерики [15]-[16];
- статична типізація забезпечує кращу підтримку рефакторингу та автодоповнення коду в IDE, що робить код більш зрозумілим і легким для обслуговування [15].

2.1.2.3 Dart

Dart – це об'єктно-орієнтована мова програмування, розроблена Google, яка використовується для створення мобільних, веб- та десктопних застосунків [17]-[18]. Популярність Dart значно зросла завдяки фреймворку Flutter, який дозволяє створювати кросплатформні застосунки [17].

Особливості Dart:

- Dart є компільованою мовою, яка може використовувати як АОТ (Ahead-of-Time), так і JIT (Just-in-Time) компіляцію, що забезпечує високу швидкість роботи та оптимізацію застосунків [17];
- підтримка суворої типізації робить його більш безпечним щодо типів, ніж JavaScript, оскільки більшість помилок виявляються під час компіляції [17];
- С-подібний синтаксис, схожий на Java або C++, що полегшує його вивчення розробникам з досвідом у об'єктно-орієнтованих мовах програмування [17]-[18].

2.1.2.4 Порівняння мов програмування для клієнтської частини

Для обґрунтованого вибору мови програмування було проведено порівняльний аналіз розглянутих технологій за ключовими критеріями. Результати аналізу наведено у табл. 2.1.

Таблиця 2.2 – Порівняння мов програмування клієнтської частини

Критерій порівняння	JavaScript	TypeScript	Dart
Тип типізації	Динамічна, слабка	Статична, сильна	Статична, сильна (опціональна)
Виявлення помилок	Під час виконання	Під час компіляції	Під час компіляції
Популярність та екосистема	Дуже висока, величезна кількість бібліотек	Висока, використовує екосистему JS	Зростаюча, але менша за JS/TS
Сфера застосування	Веб (клієнт/сервер), мобільні застосунки	Така ж, як у JavaScript	Переважно кросплатформні мобільні застосунки (Flutter)
Продуктивність	Інтерпретована	Компілюється в JS	Висока (нативна компіляція)
Підтримка ООП	На основі прототипів	Повноцінна (класи, інтерфейси)	Повноцінна (класи, інтерфейси)

Виходячи з проведеного аналізу, для реалізації клієнтської частини вебзастосунку було обрано мову програмування TypeScript. Цей вибір зумовлений необхідністю забезпечення надійності та масштабованості проекту.

TypeScript дозволяє виявляти помилки на етапі компіляції, що значно знижує ймовірність виникнення проблем під час виконання програми, на

відміну від JavaScript. Крім того, наявність статичної типізації та інтерфейсів робить код більш структурованим, зрозумілим та легшим для підтримки, що є критично важливим при роботі над складними системами або в команді [15]-[16]. Хоча Dart демонструє високу продуктивність, TypeScript має перевагу завдяки інтеграції з величезною екосистемою JavaScript та популярними веб-фреймворками, такими як React та Angular [18].

2.2 Вибір фреймворків

2.2.1 Вибір фреймворку для серверної частини

Для реалізації серверної частини (бекенду) було обрано технологію ASP.NET Core. На відміну від екосистеми JavaScript, де існує безліч фреймворків, для мови програмування C# ASP.NET є ключовим компонентом стеку технологій Microsoft та де-факто стандартом для веб-розробки [19].

ASP.NET – це потужний фреймворк для створення веб-додатків, який еволюціонував у ASP.NET Core – модульну та кросплатформну версію нового покоління [19].

Основні причини вибору ASP.NET Core:

- ASP.NET Core демонструє одну з найвищих швидкостей обробки HTTP-запитів серед сучасних бекенд-фреймворків завдяки оптимізованій роботі з пам'яттю та асинхронному програмуванню [19];
- фреймворк підтримує багат шарову архітектуру, що дозволяє чітко розділити бізнес-логіку та логіку обробки запитів. Це спрощує створення RESTful API для взаємодії з клієнтом [19];
- платформа надає вбудовані інструменти для автентифікації, авторизації та захисту від поширених веб-атак, що є критично важливим для освітньої платформи, де зберігаються персональні дані студентів;
- тісна інтеграція з середовищем Visual Studio забезпечує зручне налагодження, тестування та розгортання застосунку [19];

2.2.2 Вибір фреймворку для клієнтської частини

При розробці сучасних веб-платформ, зокрема систем управління навчанням (LMS), вибір правильного інструментарію для фронтенду є критичним етапом, оскільки він впливає на продуктивність, масштабованість та зручність підтримки коду [20]. На ринку JavaScript-технологій існує значна конкуренція, проте експерти виділяють трьох основних лідерів: Angular, React та Vue [21]. Кожен з цих інструментів має свої архітектурні особливості та сфери застосування.

2.2.2.1 Angular

Angular – це повноцінний фреймворк, розроблений компанією Google, який пропонує комплексне рішення для створення надійних застосунків [20]-[21]. Він написаний на TypeScript та використовує реальний DOM, що забезпечує сувору типізацію та чітку структуру [20]-[21]. До його переваг належать двостороннє зв'язування даних, вбудована ін'єкція залежностей та архітектура MVC, що дозволяє ефективно організовувати код [20]-[21]. Проте, цей фреймворк має високий поріг входження через складність вивчення TypeScript та велику кількість вбудованих концепцій [20].

2.2.2.2 React

React – це бібліотека для створення інтерфейсів користувача, розроблена компанією Meta (Facebook) у 2013 році [21]. Головною особливістю React є використання віртуального DOM, який дозволяє оновлювати лише змінені частини сторінки, значно підвищуючи швидкодію застосунку [21]. React базується на компонентному підході, де інтерфейс

розбивається на незалежні модулі, які можна багаторазово використовувати [20]-[21]. Це забезпечує гнучкість архітектури та спрощує тестування [21].

2.2.2.3 Vue

Vue – це прогресивний фреймворк, створений Еваном Ю, який позиціонується як проміжний варіант між Angular та React [21]. Він відомий своєю простотою, легкістю інтеграції та гнучкістю, що робить його ідеальним для проєктів малого та середнього розміру [20]. Vue використовує віртуальний DOM та забезпечує реактивність даних [20]. Однак, у порівнянні з конкурентами, він має меншу екосистему та меншу підтримку з боку великих технологічних корпорацій [20]-[21].

2.2.2.4 Порівняння фреймворків для клієнтської частини

Для наочного зіставлення ключових технічних характеристик розглянутих технологій було сформовано порівняльну табл. 2.3.

Таблиця 2.3 – Порівняльна характеристика фреймворків та бібліотек

Критерій порівняння	Angular	React	Vue
1	2	3	4
Тип	Повноцінний фреймворк	Бібліотека для побудови інтерфейсів	Прогресивний фреймворк
Розробник	Google	Meta (Facebook)	Еван Ю та спільнота (Open Source)

Основна мова розробки	TypeScript	JavaScript (JSX) / TypeScript (TSX)	JavaScript / TypeScript
-----------------------	------------	-------------------------------------	-------------------------

Кінець таблиці 2.3.

1	2	3	4
DOM	Real DOM	Virtual DOM	Virtual DOM
Прив'язка даних	Двостороння	Одностороння	Двостороння
Поріг входження	Високий (вимагає вивчення великої кількості концепцій)	Середній (потрібно розуміння JSX та екосистеми)	Низький (простий синтаксис на базі HTML/JS)
Архітектура та структура	Жорстка модульна архітектура	Гнучка (архітектуру обирає розробник)	Гнучка (можливість поступового впровадження)
Сфера застосування	Масштабні корпоративні проєкти	SPA, мобільні додатки, соціальні мережі	Стартапи, інтерфейси середньої складності, SPA

На основі проведеного порівняльного аналізу (табл. 2.3) для реалізації клієнтської частини веб-платформи було обрано бібліотеку React. Незважаючи на те, що React позиціонується як бібліотека, а не повноцінний фреймворк, його екосистема та технічні можливості повністю покривають потреби розробки сучасних SPA-застосунків (Single Page Application).

2.3 Вибір системи управління базами даних

Вибір системи управління базами даних (СУБД) є критичним етапом проектування архітектури вебплатформи, оскільки від цього рішення залежать продуктивність, масштабованість, цілісність даних та зручність подальшої підтримки програмного продукту. СУБД виступає посередником між базою даних, додатками та користувацькими інтерфейсами, забезпечуючи ефективне управління та організацію даних [22].

Для розробки системи управління навчанням (LMS) необхідно враховувати специфіку даних: наявність чітких зв'язків між сутностями (студенти, курси, оцінки), необхідність зберігання контенту та забезпечення транзакційної цілісності. Сучасні СУБД поділяються на два основні типи: реляційні (SQL) та нереляційні (NoSQL) [22].

2.3.1 Аналіз типів баз даних

Реляційні бази даних (SQL) організовують дані у таблиці з рядками та стовпцями, мають чітко визначену схему та підтримують зв'язки між таблицями за допомогою первинних та зовнішніх ключів [22]. Вони базуються на мові структурованих запитів (SQL) та забезпечують відповідність стандарту ACID (атомарність, узгодженість, ізолюваність, довговічність), що є критично важливим для фінансових систем та додатків, де цілісність даних є пріоритетом [22].

Нереляційні бази даних (NoSQL) використовують різні моделі даних, такі як документо-орієнтована, ключ-значення або графова, та дозволяють зберігати неструктуровані дані без фіксованої схеми [22]. Вони забезпечують високу продуктивність та легке горизонтальне масштабування, проте часто мають слабші гарантії безпеки та ACID порівняно з реляційними системами [22].

Враховуючи потребу LMS у зберіганні структурованої інформації про успішність студентів та зв'язки між курсами, для порівняльного аналізу було обрано три найпопулярніші СУБД: MySQL, PostgreSQL та MongoDB.

2.3.2 Порівняльна характеристика СУБД

2.3.2.1 MySQL

MySQL – це одна з найпопулярніших реляційних баз даних з відкритим вихідним кодом, яка є основою стеку LAMP [22]. Вона характеризується простотою синтаксису, високою швидкістю роботи з операціями читання та широкою підтримкою хмарних платформ [22]. Однак, MySQL має певні обмеження щодо масштабованості та відповідності стандартам SQL порівняно з іншими рішеннями [22].

2.3.2.2 PostgreSQL

PostgreSQL – це потужна об'єктно-реляційна СУБД, відома своєю надійністю, гнучкістю та відповідністю технічним стандартам [23]. Вона підтримує як реляційні, так і нереляційні типи даних (зокрема JSON), що робить її універсальним рішенням [23]. PostgreSQL забезпечує виконання складних запитів, підтримує розширені процедури та пропонує надійні механізми забезпечення цілісності даних [23].

2.3.2.3 MongoDB

MongoDB – це документо-орієнтована NoSQL база даних, яка зберігає дані у форматі JSON-подібних документів [22]. Вона забезпечує високу

гнучкість схеми, дозволяючи швидко змінювати структуру даних без складних міграцій, та ефективно масштабується горизонтально [22]. Проте, MongoDB має вищі вимоги до пам'яті та слабшу підтримку транзакцій порівняно з SQL-системами [22].

2.3.2.4 Порівняння баз даних

У табл. 2.4 наведено порівняльний аналіз розглянутих систем відповідно до потреб проєкту.

Таблиця 2.4 – Порівняльна характеристика систем управління базами даних

Критерій порівняння	MySQL	PostgreSQL	MongoDB
1	2	3	4
Тип бази даних	Реляційна	Об'єктно-реляційна	NoSQL (Документна)
Структура даних	Таблиці, сувора схема	Таблиці, підтримка JSON та користувацьких типів	Гнучкі JSON-документи
Масштабованість	Вертикальна	Вертикальна, висока продуктивність при складних запитах	Горизонтальна

Підтримка ACID	Так (через InnoDB)	Так, повна підтримка транзакцій	Обмежена
Підтримка JSON	Обмежена	Розширена	Нативна
Складність запитів	Стандартні SQL запити	Складні запити, розширені процедури	Обмежені можливості

На основі проведеного аналізу для реалізації вебплатформи підтримки освітнього процесу було обрано систему управління базами даних PostgreSQL. Ключовим аргументом на користь цього рішення стала гібридна природа системи: будучи об'єктно-реляційною, вона вдало поєднує строгість реляційної моделі з гнучкістю NoSQL.

Це дозволяє надійно фіксувати чіткі зв'язки між студентами, групами та оцінками, водночас використовуючи тип даних JSONB для ефективного зберігання різноманітного контенту навчальних матеріалів і динамічних структур тестів.

2.4 Вибір середовища розробки

Ефективність процесу створення програмного забезпечення значною мірою залежить від обраного інструментарію. Вибір інтегрованого середовища розробки (IDE) впливає на швидкість написання коду, зручність відладки коду, можливості рефакторингу та інтеграцію з системами контролю версій.

Для реалізації вебплатформи було проаналізовано три найбільш популярні рішення на ринку: Microsoft Visual Studio, Visual Studio Code та JetBrains Rider.

2.4.1 Visual Studio 2022

Visual Studio 2022 – це комплексне інтегроване середовище розробки від Microsoft, яке є стандартом індустрії для великих корпоративних проєктів, особливо в екосистемі .NET [24]. Це повнофункціональна IDE, що надає широкий набір інструментів «з коробки», включаючи потужний відладчик, профайлери та інструменти для тестування [24].

2.4.2 Visual Studio Code

Visual Studio Code – це легковаговий редактор коду з відкритим вихідним кодом, який здобув величезну популярність завдяки своїй швидкості та гнучкості [24]. На відміну від класичної Visual Studio, VS Code є редактором коду, який може конкурувати з IDE за допомогою розширень [24]-[25].

2.4.3 JetBrains Rider

JetBrains Rider – це кросплатформна IDE, що базується на платформі IntelliJ та використовує рушій ReSharper для аналізу коду [25]-[26]. Rider позиціонується як швидка та потужна альтернатива Visual Studio, яка об'єднує інтелект ReSharper з продуктивністю платформи IntelliJ [25]-[26].

2.4.4 Порівняння середовищ розробки

Порівняння характеристик середовищ розробки наведено у табл. 2.5.

За результатами проведеного аналізу, основним інструментом для реалізації програмного продукту було обрано JetBrains Rider. Цей вибір насамперед зумовлений необхідністю балансу між продуктивністю та функціональними можливостями.

На відміну від Visual Studio 2022, яка часто демонструє високе споживання ресурсів та затримки в роботі інтерфейсу на складних проєктах, Rider забезпечує високу швидкодію та стабільність, використовуючи евристичні алгоритми для прискорення збірки [26].

Водночас, порівняно з Visual Studio Code, Rider є повноцінним інтегрованим середовищем, що не потребує складного налаштування численних плагінів для отримання базового функціоналу [27].

Таблиця 2.5 – Порівняльна характеристика середовищ розробки

Критерій порівняння	Visual Studio 2022	Visual Studio Code	JetBrains Rider
Тип ПЗ	Комплексна IDE	Редактор коду	Комплексна IDE
Кросплатформність	Відсутня	Наявна	Наявна
Аналіз коду	Високий	Базовий (залежить від плагінів)	Дуже високий (ReSharper)
Продуктивність	Низька (ресурсоємна)	Середня	Висока
Налаштування	Готова до роботи	Потребує налаштування плагінів	Готова до роботи
Підтримка перегляду БД	Відсутня	Залежить від плагінів	Вбудована
Вартість	Безкоштовна (Community) / Платна	Безкоштовна	Безкоштовна (Non-Commercial) / Платна

2.5 Висновки за розділом

На основі проведеного аналізу технологічних стеків для реалізації серверної частини вебзастосунку було обрано мову програмування C#. Дане рішення ґрунтується на необхідності побудови надійної та масштабованої архітектури, де сувора статична типізація дозволяє мінімізувати кількість помилок ще на етапі компіляції. Порівняно з такими мовами, як PHP та Ruby, C# разом із платформою .NET демонструє вищу продуктивність та стабільність, що є критично важливим для обробки складних завдань та ефективної підтримки життєвого циклу програмного забезпечення. Ключовим компонентом бекенд-розробки визначено фреймворк ASP.NET Core, який забезпечує високу швидкість обробки HTTP-запитів, надійний захист даних та зручну інтеграцію з інструментами розробки.

Для розробки клієнтської частини було обрано мову TypeScript, яка завдяки статичній типізації та інтерфейсам забезпечує вищу передбачуваність і структурування коду порівняно з JavaScript. Хоча такі альтернативи, як Dart, пропонують високу швидкість роботи, TypeScript має суттєву перевагу завдяки повній інтеграції з величезною екосистемою JavaScript-бібліотек та популярними вебфреймворками. У процесі дослідження було також детально розглянуто характеристики провідних фронтенд-інструментів, таких як Angular, React та Vue, що дозволило визначити оптимальні підходи до створення компонентної архітектури та забезпечення швидкодії інтерфейсу користувача.

3 ОПИС ПРОГРАМИ

3.1 Загальна архітектура системи

Архітектура програмного забезпечення є фундаментальним планом або структурою, яка описує, як організована система та як різні її компоненти взаємодіють між собою [28]. Для розробки даної системи було обрано клієнт-серверну архітектуру (рис.3.1).

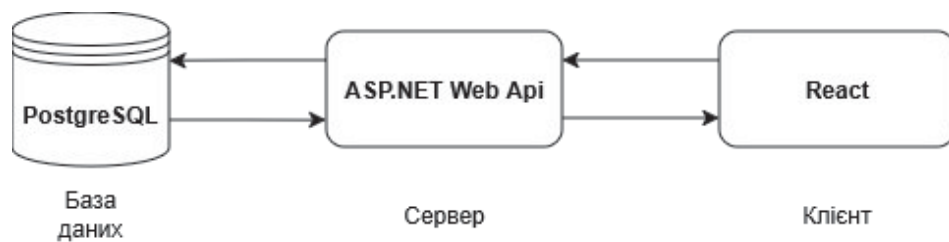


Рисунок 3.1 – Схема клієнт-серверної архітектури

Ця модель поділяє застосунок на дві основні частини: клієнтську, яка взаємодіє з користувачем, та серверну, яка відповідає за обробку даних та їх зберігання [29]. Принцип роботи полягає в тому, що клієнт надсилає запити через мережу до сервера, сервер отримує ці запити, обробляє їх згідно з бізнес-логікою та повертає відповідний результат. З точки зору рівневої організації, система реалізована як багаторівнева архітектура, яка зазвичай включає три основні рівні:

- рівень представлення, реалізований за допомогою React. Цей рівень відповідає за відображення інформації користувачу та формування запитів до сервера [30]. Він працює у веб-браузері користувача;
- рівень бізнес-логіки: реалізований за на базі ASP.NET Minimal API. Цей рівень виступає посередником, який обробляє вхідні запити від клієнта, виконує необхідні обчислення та валідацію [30]-[31]. Вибір ASP.NET Minimal API дозволяє створити легковаговий та швидкий сервер, що відповідає принципам простоти розгортання та прямої комунікації, характерним для сучасних веб-застосунків;

– рівень даних: відповідає за безпосереднє зберігання та вилучення інформації з бази даних [30].

Взаємодія між клієнтом та сервером відбувається за допомогою стандартного протоколу HTTP (Hypertext Transfer Protocol), що є загальноприйнятим стандартом для веб-систем [30]. Клієнтська частина формує HTTP-запити, а серверна частина, обробивши їх, надсилає відповідь.

3.2 Проектування бази даних

ER-діаграму бази даних наведено на рис. 3.2.

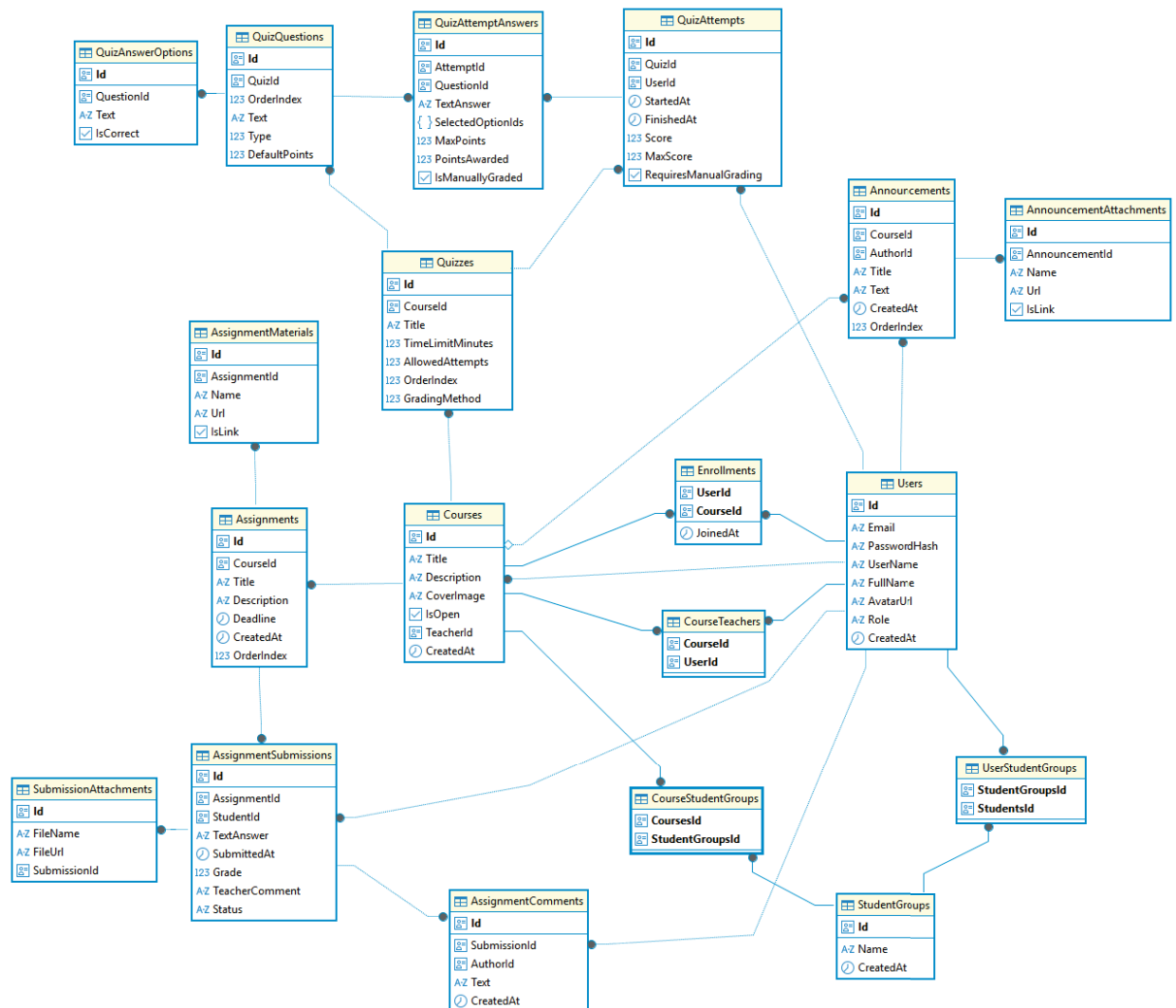


Рисунок 3.2 – ER-діаграма основних сутностей бази даних

База даних для системи управління навчанням (LMS) є центральним компонентом, що забезпечує організацію, зберігання та ефективне управління даними користувачів, навчальних курсів, студентських груп, оголошень та результатів оцінювання. Структура бази даних спроектована таким чином, щоб підтримувати складні зв'язки між сутностями та забезпечувати високу швидкість обробки запитів.

Концептуальна модель даних базується на ключових сутностях предметної області: користувачі, навчальні курси, академічні групи, завдання та системи тестування. Нижче наведено детальний опис структури таблиць бази даних (табл. 3.1 – 3.19). Таблиця користувачів наведена в табл. 3.1.

Таблиця 3.1 – Структура таблиці Users

Поле	Тип даних	Обмеження	Опис
Id	uuid	PRIMARY KEY, NOT NULL	Унікальний ідентифікатор користувача
Email	text	UNIQUE, NOT NULL	Електронна пошта користувача
PasswordHash	text	NOT NULL	Хешований пароль
UserName	text	UNIQUE, NOT NULL	Ім'я користувача (логін)
FullName	text	NOT NULL	Повне ім'я
AvatarUrl	text	NULL	Посилання на зображення профілю
Role	text	NOT NULL	Роль користувача в системі
CreatedAt	timestamp tz	NOT NULL	Дата створення облікового запису

Сутність користувачів (Users) призначена для зберігання персональної інформації, даних для автентифікації та налаштувань профілю користувачів системи.

Обов'язковими атрибутами є унікальний ідентифікатор (Id), електронна пошта (Email), хешований пароль (PasswordHash) та ім'я користувача (UserName). Додатково зберігається опис профілю та посилання на зображення обкладинки (CoverImage).

Сутність академічних груп (StudentGroups) використовується для організації студентів у групи для полегшення управління навчальним процесом. Обов'язковими атрибутами є ідентифікатор групи (Id) та її назва (Name).

Також фіксується дата створення запису. Таблиця академічних груп має наступну структуру (табл. 3.2).

Таблиця 3.2 – Структура таблиці StudentGroups

Поле	Тип даних	Обмеження	Опис
Id	uuid	PRIMARY KEY, NOT NULL	Унікальний ідентифікатор групи
Name	text	NOT NULL	Назва групи
CreatedAt	timestampz	NOT NULL	Дата створення запису

Сутність зв'язку користувачів та груп (UserStudentGroups) слугує для реалізації відношення між студентами та їхніми академічними групами. Вона містить зовнішні ключі на ідентифікатор групи (StudentGroupsId) та ідентифікатор студента (StudentsId). Структуру таблиці наведено нижче (табл. 3.3).

Таблиця 3.3 – Структура таблиці UserStudentGroups

Поле	Тип даних	Обмеження	Опис
StudentGroups Id	uuid	PK, FK (StudentGroups), NOT NULL	Ідентифікатор групи
StudentsId	uuid	PK, FK (Users), NOT NULL	Ідентифікатор студента

Сутність навчальних курсів (Courses) є центральним елементом для зберігання інформації про освітні програми в системі. Обов'язковими атрибутами є унікальний ідентифікатор (Id), назва курсу (Title), ідентифікатор основного викладача (TeacherId) та дата створення. Додатково зберігається опис змісту (Description), посилання на обкладинку (CoverImage) та статус доступності курсу (IsOpen). Таблиця навчальних курсів має наступну структуру (табл. 3.4).

Таблиця 3.4 – Структура таблиці Courses

Поле	Тип даних	Обмеження	Опис
Id	uuid	PRIMARY KEY, NOT NULL	Унікальний ідентифікатор курсу
Title	text	NOT NULL	Назва курсу
Description	text	NULL	Опис змісту курсу
CoverImage	text	NULL	Обкладинка курсу
IsOpen	bool	NOT NULL	Статус доступності курсу
TeacherId	uuid	FOREIGN KEY (Users), NOT NULL	Ідентифікатор основного викладача
CreatedAt	timestamp z	NOT NULL	Дата створення запису

Сутність реєстрацій на курси (Enrollments) призначена для фіксації приєднання конкретних користувачів до навчальних курсів. Вона містить ідентифікатори користувача (UserId) та курсу (CourseId), а також дату приєднання до навчання (JoinedAt). Таблиця реєстрацій має наступну структуру (табл. 3.5).

Таблиця 3.5 – Структура таблиці Enrollments

Поле	Тип даних	Обмеження	Опис
UserId	uuid	PK, FK (Users), NOT NULL	Ідентифікатор користувача
CourseId	uuid	PK, FK (Users), NOT NULL	Ідентифікатор курсу
JoinedAt	timestampz	NOT NULL	Дата приєднання до курсу

Сутність зв'язку курсів та груп (CourseStudentGroups) дозволяє записувати цілі академічні групи за відповідними навчальними курсами. Обов'язковими атрибутами є ідентифікатори курсу (CoursesId) та студентської групи (StudentGroupsId). Структура цієї таблиці наведена нижче (табл. 3.6).

Таблиця 3.6 – Структура таблиці CourseStudentGroups

Поле	Тип даних	Обмеження	Опис
CoursesId	uuid	PK, FK (Courses), NOT NULL	Ідентифікатор курсу
StudentGroupsId	uuid	PK, FK (StudentGroups), NOT NULL	Ідентифікатор студентської групи

Сутність викладачів курсів (CourseTeachers) призначена для управління списком викладачів, які мають доступ до ведення конкретного курсу. Вона містить ідентифікатори курсу (CoursesId) та користувача-викладача (UserId). Таблиця викладачів курсів має наступну структуру (табл.3.7).

Таблиця 3.7 – Структура таблиці CourseTeachers

Поле	Тип даних	Обмеження	Опис
CoursesId	uuid	PK, FK (Courses), NOT NULL	Ідентифікатор курсу
UserId	uuid	PK, FK (Users), NOT NULL	Ідентифікатор користувача (викладача)

Таблиця тестів має наступну структуру (табл. 3.8).

Таблиця 3.8 – Структура таблиці Quizzes

Поле	Тип даних	Обмеження	Опис
1	2	3	4
Id	uuid	PRIMARY KEY, NOT NULL	Унікальний ідентифікатор тесту
CourseId	uuid	FOREIGN KEY (Courses), NOT NULL	Курс, до якого належить тест
Title	text	NOT NULL	Назва тесту
TimeLimitMinutes	int4	NOT NULL	Ліміт часу у

			хвилинах
--	--	--	----------

Кінець таблиці 3.8.

1	2	3	4
AllowedAttempts	int4	NOT NULL	Дозволена кількість спроб
OrderIndex	int4	NOT NULL	Індекс сортування
GradingMethod	int4	NOT NULL	Метод оцінювання

Сутність тестів (Quizzes) використовується для створення тестів у межах навчальних курсів. Обов'язковими атрибутами є ідентифікатор (Id), назва тесту (Title) та зв'язок із курсом (CourseId). Також визначаються параметри проходження: ліміт часу (TimeLimitMinutes), дозволена кількість спроб (AllowedAttempts), метод оцінювання та індекс сортування.

Сутність спроб тестування (QuizAttempts) зберігає інформацію про кожне проходження тесту користувачем. Вона містить дані про час початку (StartedAt) і завершення (FinishedAt), отриманий бал (Score), максимальний бал (MaxScore) та позначку про необхідність ручної перевірки викладачем. Структура таблиці спроб наведена нижче (табл. 3.9).

Таблиця 3.9 – Структура таблиці QuizAttempts

Поле	Тип даних	Обмеження	Опис
1	2	3	4
Id	uuid	PRIMARY KEY, NOT NULL	Ідентифікатор спроби
QuizId	uuid	FK (Quizzes), NOT NULL	Ідентифікатор тесту

UserId	uuid	FK (Users), NOT NULL	Ідентифікатор користувача
--------	------	----------------------	---------------------------

Кінець таблиці 3.9.

1	2	3	4
StartedAt	timestampz	NOT NULL	Початок спроби
FinishedAt	timestampz	NULL	Завершення спроби
Score	float8	NULL	Отриманий бал
MaxScore	float8	NOT NULL	Максимально можливий бал
RequiresManualGrading	bool	NOT NULL	Чи потрібна перевірка викладачем

Сутність запитань (QuizQuestions) призначена для формування контенту конкретного тесту. Обов'язковими атрибутами є текст запитання (Text), його тип (Type), порядковий номер (OrderIndex) та кількість балів за замовчуванням (DefaultPoints). Таблиця запитань має наступну структуру (табл. 3.10).

Таблиця 3.10 – Структура таблиці QuizQuestions

Поле	Тип даних	Обмеження	Опис
1	2	3	4
Id	uuid	PRIMARY KEY, NOT NULL	Ідентифікатор запитання
QuizId	uuid	FK (Quizzes), NOT NULL	Ідентифікатор тесту

OrderIndex	int4	NOT NULL	Індекс сортування
------------	------	----------	-------------------

Кінець таблиці 3.10.

1	2	3	4
Text	text	NOT NULL	Текст запитання
Type	int4	NOT NULL	Тип запитання (напр. вибір варіанту)
DefaultPoints	float8	NOT NULL	Кількість балів за замовчуванням

Сутність варіантів відповідей (QuizAnswerOptions) містить перелік можливих відповідей для закритих типів запитань. Вона включає ідентифікатор запитання (QuestionId), текст варіанту (Text) та логічну позначку про його правильність (IsCorrect). Таблиця варіантів відповідей має наступну структуру (табл. 3.11).

Таблиця 3.11 – Структура таблиці QuizAnswerOptions

Поле	Тип даних	Обмеження	Опис
Id	uuid	PRIMARY KEY, NOT NULL	Ідентифікатор варіанту відповіді
QuestionId	uuid	FK (QuizQuestions), NOT NULL	Ідентифікатор запитання
Text	text	NOT NULL	Текст варіанту відповіді
IsCorrect	bool	NOT NULL	Позначка, чи є

			відповідь правильною
--	--	--	-------------------------

Сутність відповідей у спробі (QuizAttemptAnswers) фіксує відповіді, надані користувачем під час конкретної спроби проходження тесту. Вона зберігає текст відповіді або список обраних варіантів (SelectedOptionIds), нараховані бали (PointsAwarded) та інформацію про те, чи була відповідь оцінена вручну. Таблиця відповідей має наступну структуру (табл. 3.12).

Таблиця 3.12 – Структура таблиці QuizAttemptAnswers

Поле	Тип даних	Обмеження	Опис
Id	uuid	PRIMARY KEY, NOT NULL	Ідентифікатор запису
AttemptId	uuid	FK (QuizAttempts), NOT NULL	Ідентифікатор спроби
QuestionId	uuid	FK (QuizQuestions), NOT NULL	Ідентифікатор запитання
TextAnswer	text	NULL	Текстова відповідь користувача
SelectedOptionIds	jsonb	NULL	Список вибраних варіантів (JSON)
MaxPoints	float8	NOT NULL	Максимально можливі бали
PointsAwarded	float8	NULL	Нараховані бали

			за відповідь
IsManuallyGraded	bool	NOT NULL	Чи була відповідь оцінена вручну

Сутність оголошень (Announcements) призначена для публікації інформаційних та організаційних повідомлень у системі. Вона містить заголовок (Title), текст повідомлення (Text), ідентифікатор автора (AuthorId) та дату створення. Оголошення може бути як глобальним, так і прив'язаним до конкретного курсу (CourseId). Таблиця оголошень має наступну структуру (табл. 3.13).

Таблиця 3.13 – Структура таблиці Announcements

Поле	Тип даних	Обмеження	Опис
Id	uuid	PRIMARY KEY, NOT NULL	Ідентифікатор оголошення
CourseId	uuid	FK (Courses), NULL	Курс, до якого належить оголошення (може бути порожнім, тоді оголошення глобальне)
AuthorId	uuid	FK (Users), NOT NULL	Автор оголошення
Title	text	NOT NULL	Заголовок
Text	text	NOT NULL	Текст оголошення
CreatedAt	timestampz	NOT NULL	Дата створення запису

OrderIndex	int4	NOT NULL	Індекс сортування
------------	------	----------	-------------------

Сутність додатків до оголошень (AnnouncementAttachments) дозволяє прикріплювати до повідомлень файли або зовнішні ресурси. Вона містить унікальний ідентифікатор (Id), ідентифікатор відповідного оголошення (AnnouncementId), назву додатка (Name) та посилання на нього (Url). Також фіксується ознака того, чи є ресурс прямим посиланням (IsLink). Таблиця додатків до оголошень має наступну структуру (табл. 3.14).

Таблиця 3.14 – Структура таблиці AnnouncementAttachments

Поле	Тип даних	Обмеження	Опис
Id	uuid	PRIMARY KEY, NOT NULL	Ідентифікатор дodatка
AnnouncementId	uuid	FK (Announcements), NOT NULL	Ідентифікатор оголошення
Name	text	NOT NULL	Назва файлу або посилання
Url	text	NOT NULL	URL-адреса ресурсу
IsLink	bool	NOT NULL	Чи є ресурс зовнішнім посиланням

Сутність завдань (Assignments) призначена для організації та контролю практичної діяльності студентів (завдань) у межах навчального курсу. Обов'язковими атрибутами є ідентифікатор (Id), назва (Title),

детальний опис (Description), кінцевий термін виконання (Deadline) та дата створення.

Завдання обов'язково прив'язується до конкретного курсу (CourseId) та має індекс для сортування (OrderIndex). Таблиця завдань має наступну структуру (табл. 3.15).

Таблиця 3.15 – Структура таблиці Assignments

Поле	Тип даних	Обмеження	Опис
Id	uuid	PRIMARY KEY, NOT NULL	Ідентифікатор завдання
CourseId	uuid	FK (Courses), NULL	Курс, до якого належить завдання
Title	text	NOT NULL	Назва завдання
Description	text	NULL	Детальний опис
Deadline	timestampz	NULL	Кінцевий термін здачі
CreatedAt	timestampz	NOT NULL	Дата створення запису
OrderIndex	int4	NOT NULL	Індекс сортування

Сутність матеріалів до завдань (AssignmentMaterials) використовується для надання студентам допоміжних файлів або інструкцій, необхідних для виконання конкретного завдання. Вона включає ідентифікатор завдання (AssignmentId), назву матеріалу (Name) та посилання на ресурс (Url) з позначкою типу (IsLink). Таблиця матеріалів до завдань має наступну структуру (табл. 3.16).

Таблиця 3.16 – Структура таблиці AssignmentSubmissions

Поле	Тип даних	Обмеження	Опис
1	2	3	4
Id	uuid	PRIMARY KEY, NOT NULL	Ідентифікатор зданої роботи

Кінець таблиці 3.16.

1	2	3	4
AssignmentId	uuid	FK (Assignments), NOT NULL	Ідентифікатор завдання
StudentId	uuid	FK (Users), NOT NULL	Студент, що здав роботу
TextAnswer	text	NULL	Текстова відповідь студента
SubmittedAt	timestampz	NOT NULL	Дата та час подання
Grade	float8	NULL	Оцінка за роботу
TeacherComment	text	NULL	Коментар викладача
Status	text	NOT NULL	Поточний статус (здано, перевірено)

Сутність зданих робіт (AssignmentSubmissions) фіксує результати виконання завдань студентами та процес їх оцінювання викладачем.

Вона містить ідентифікатори завдання (AssignmentId) та студента (StudentId), текст відповіді (TextAnswer), час здачі (SubmittedAt), оцінку (Grade) та коментар викладача (TeacherComment). Також відстежується поточний статус роботи (Status). Таблиця зданих робіт має наступну структуру (табл. 3.17).

Таблиця 3.17 – Структура таблиці SubmissionAttachments

Поле	Тип даних	Обмеження	Опис
Id	uuid	PRIMARY KEY, NOT NULL	Ідентифікатор файлу
FileName	text	NOT NULL	Назва завантаженого файлу
FileUrl	text	NOT NULL	Посилання на файл на сервері
SubmissionId	uuid	FK (AssignmentSubmissions), NOT NULL	Ідентифікатор відповіді

Сутність файлів до зданих робіт (SubmissionAttachments) призначена для зберігання документів, які студенти завантажують як частину своєї відповіді на завдання. Обов'язковими атрибутами є назва файлу (FileName), посилання на файл (FileUrl) та ідентифікатор відповідної здачі (SubmissionId). Таблиця файлів до зданих робіт має наступну структуру (табл. 3.18).

Таблиця 3.18 – Структура таблиці AssignmentMaterials

Поле	Тип даних	Обмеження	Опис
1	2	3	4
Id	uuid	PRIMARY KEY, NOT NULL	Ідентифікатор додатка

Кінець таблиці 3.18.

1	2	3	4
AnnouncementId	uuid	FK (Assignments), NOT NULL	Ідентифікатор завдання
Name	text	NOT NULL	Назва матеріалу
Url	text	NOT NULL	Шлях до файлу або посилання
IsLink	bool	NOT NULL	Чи є матеріал посиланням

Сутність коментарів до робіт (AssignmentComments) забезпечує комунікацію між викладачами та студентами щодо конкретних завдань. Вона зберігає ідентифікатор здачі (SubmissionId), дані про автора коментаря (AuthorId), текст повідомлення (Text) та дату його створення (CreatedAt). Таблиця коментарів до робіт має наступну структуру (табл. 3.19).

Таблиця 3.19 – Структура таблиці AssignmentComments

Поле	Тип даних	Обмеження	Опис
1	2	3	4

Id	uuid	PRIMARY KEY, NOT NULL	Ідентифікатор коментаря
SubmissionId	uuid	FK (AssignmentSubmissions), NOT NULL	Ідентифікатор зданої роботи
AuthorId	uuid	FK (Users), NOT NULL	Автор коментаря

Кінець таблиці 3.19.

1	2	3	4
Text	text	NOT NULL	Текст повідомлення
CreatedAt	timestampz	NOT NULL	Дата та час написання

3.3 Алгоритми та функціонування системи

Для опису динамічної поведінки розробленої системи та візуалізації послідовності дій користувачів було обрано діаграми діяльності (Activity Diagrams). Вони дозволяють детально розглянути бізнес-логіку основних процесів, розділяючи відповідальність між клієнтською частиною, серверною частиною та базою даних.

Відповідно до функціональних вимог, визначених у розділі 1, ключовими процесами системи є задача навчальних завдань, керування контентом курсів та автоматизоване оцінювання знань.

3.3.1 Алгоритм задачі завдання студентом

Алгоритм здачі завдання студентом (рис. 3.3) починається з авторизації користувача та переходу до інтерфейсу конкретного курсу, де система через API запитує список доступних завдань. Після вибору конкретного пункту клієнтська частина відображає умови завдання та перевіряє поточну дату відносно встановленого терміну здачі. Якщо дедлайн ще не настав, студент отримує можливість завантажити файл або ввести текстову відповідь. Надіслані дані проходять первинну валідацію на стороні React, після чого передаються HTTP-запитом на сервер. ASP.NET-сервер перевіряє цілісність отриманих даних, асоціює їх з ідентифікатором користувача та створює новий запис у таблиці підтверджених робіт. Завершується алгоритм оновленням статусу завдання в інтерфейсі студента на «Здано» та генерацією внутрішнього сповіщення для викладача.

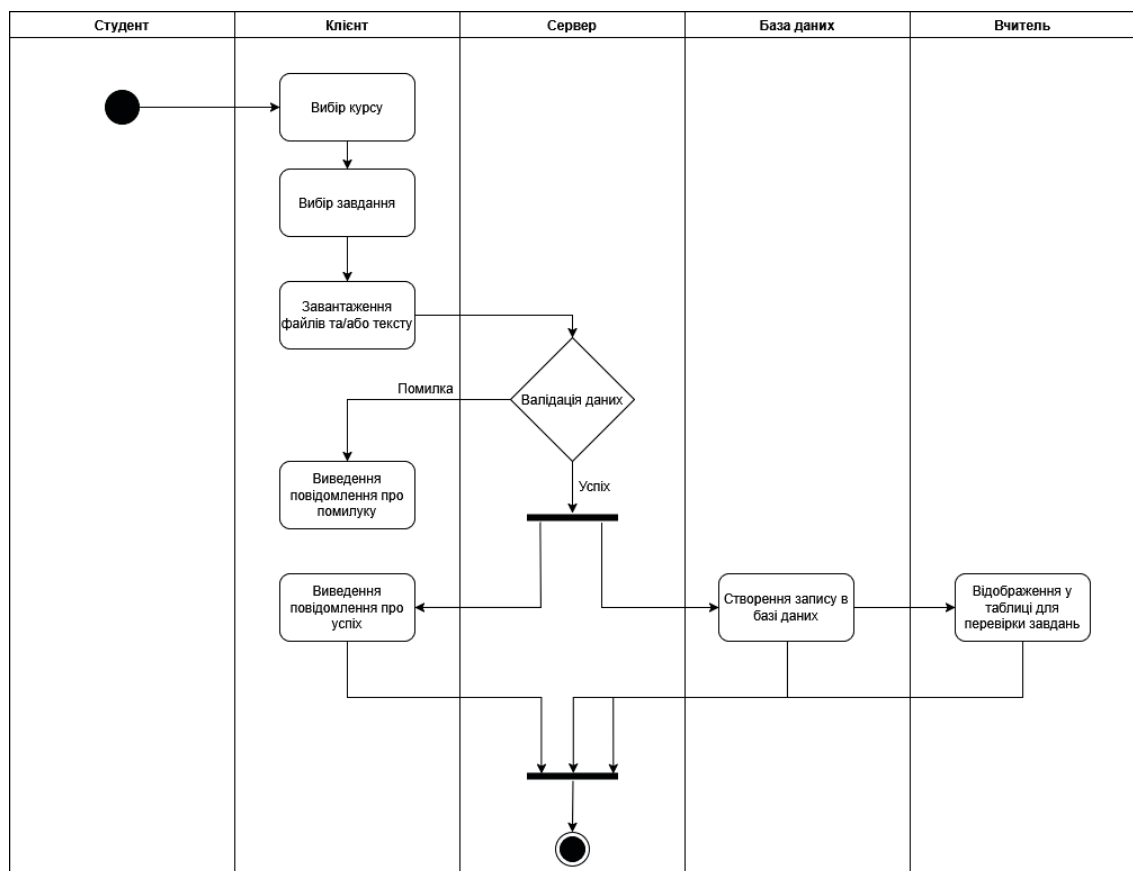


Рисунок 3.3 – Алгоритм здачі завдання студентом

3.3.2 Алгоритм створення курсу та зарахування студентів

Процес створення курсу та зарахування до нього студентів (рис. 3.4) відображає адміністративну логіку роботи викладача. На першому етапі викладач заповнює форму з метаданими курсу, такими як назва, опис та зображення обкладинки, які після збереження у базі даних формують унікальну сутність курсу з прив'язкою до власника. Після створення каркаса курсу алгоритм передбачає етап наповнення його навчальними матеріалами та оголошеннями. Ключовою частиною функціонування цього модуля є механізм управління доступом, який дозволяє викладачу додавати студентів двома способами: шляхом індивідуального пошуку або через масове зарахування цілих академічних груп. Кожна така дія створює відповідні зв'язки у таблицях реєстрацій, що автоматично робить курс доступним у персональних кабінетах обраних користувачів після його активації.

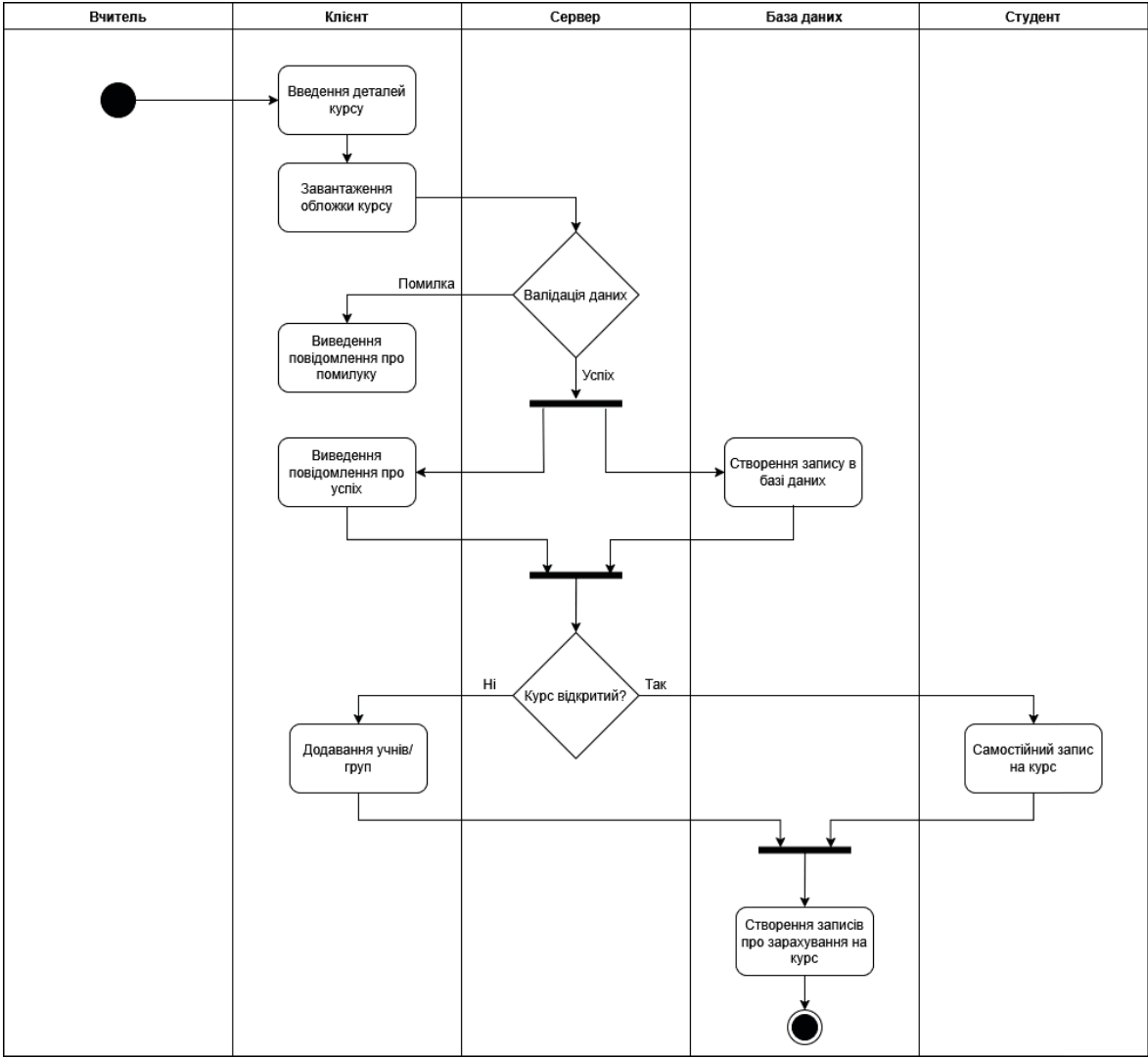


Рисунок 3.4 – Алгоритм створення курсу та зарахування студентів

3.3.3 Алгоритм перевірки виконаної роботи викладачем

Алгоритм перевірки виконаної роботи викладачем (рис. 3.5) замикає цикл взаємодії у системі завдань та відповідає за зворотний зв'язок. Процес розпочинається з перегляду викладачем списку всіх надісланих робіт для конкретного завдання. Після вибору роботи викладач отримує доступ до поданих студентом матеріалів, включаючи текстові відповіді та прикріплені файли, що витягуються з сховища на сервері. На основі аналізу виконаної роботи викладач вводить оцінку та, за потреби, додає текстовий коментар з рекомендаціями. При збереженні результату серверна частина оновлює запис у таблиці зданих робіт, фіксуючи бал та час перевірки, а також змінює статус

роботи на «Оцінено». Кінцевим етапом алгоритму є миттєве оновлення інформації на сторінці завдання та журналі оцінок студента, де він може ознайомитися з отриманим балом та зауваженнями викладача.

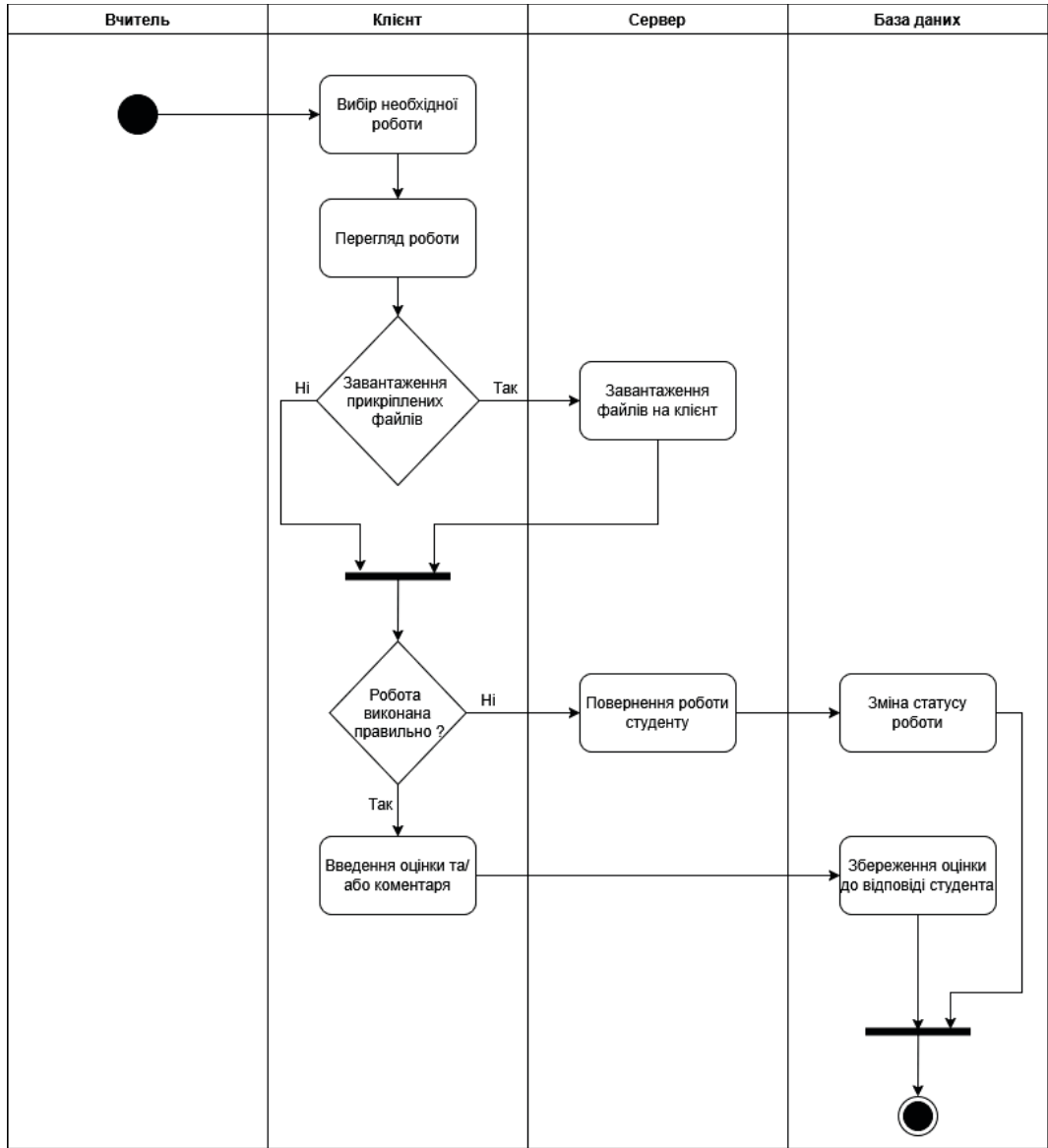


Рисунок 3.5 – Алгоритм перевірки виконаної студентом завдання

3.4 Проектування інтерфейсу користувача

На головній сторінці (рис. 3.6) відображаються важливі системні сповіщення для користувачів. Зокрема, наведено приклад оголошення про планові технічні роботи.

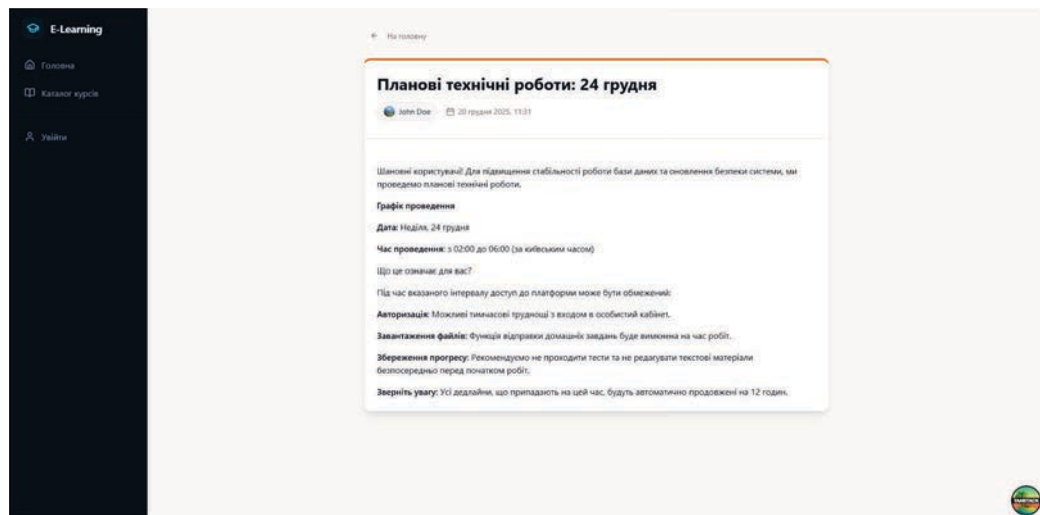


Рисунок 3.6 – Головна сторінка

Інтерфейс входу до системи (рис. 3.7) містить стандартну форму для введення імені та пароля користувача. Користувач може авторизуватися для доступу до свого особистого кабінету та навчальних матеріалів.

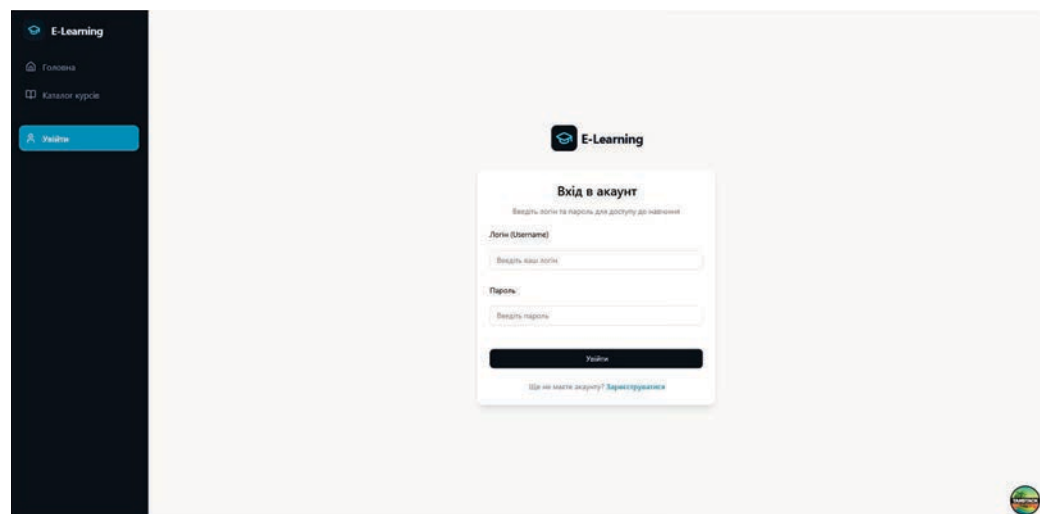
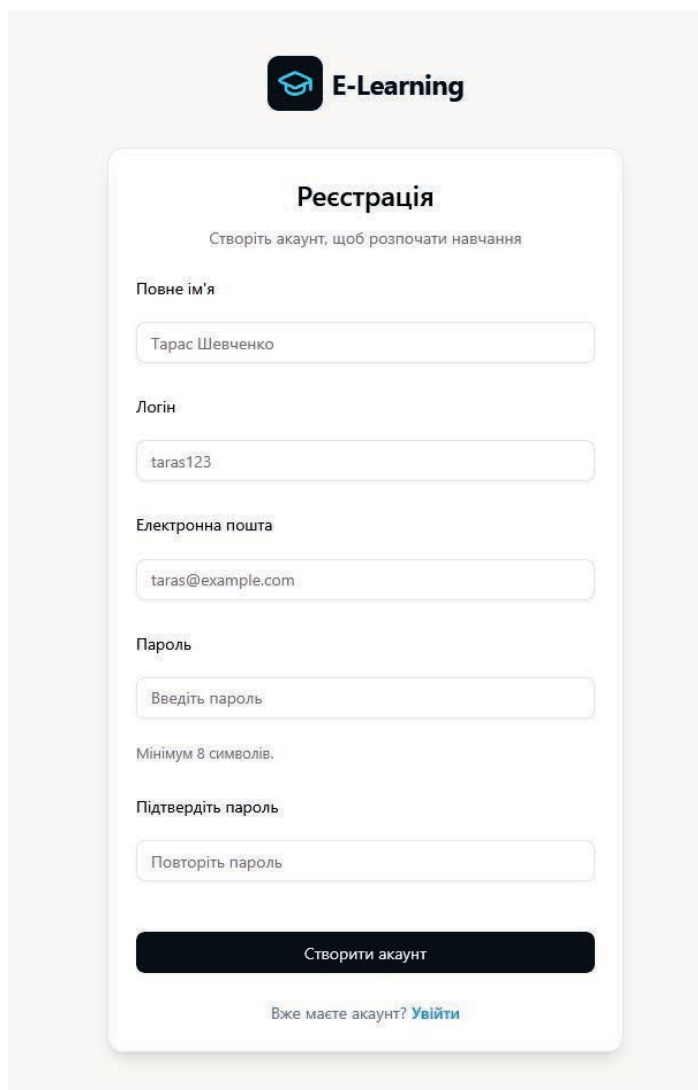


Рисунок 3.7 – Сторінка авторизації

Форма створення нового акаунту (рис. 3.8) вимагає від користувача введення повного імені, діючої адреси електронної пошти, а також встановлення та підтвердження пароля для забезпечення безпеки доступу.



The image shows a registration form for an 'E-Learning' system. At the top, there is a logo with a graduation cap icon and the text 'E-Learning'. Below the logo, the title 'Реєстрація' (Registration) is centered, followed by the subtitle 'Створіть акаунт, щоб розпочати навчання' (Create an account to start learning). The form contains several input fields: 'Повне ім'я' (Full name) with the value 'Тарас Шевченко', 'Логін' (Login) with 'taras123', 'Електронна пошта' (Email) with 'taras@example.com', 'Пароль' (Password) with the placeholder 'Введіть пароль', and 'Підтвердіть пароль' (Confirm password) with the placeholder 'Повторіть пароль'. A note 'Мінімум 8 символів.' (Minimum 8 characters.) is placed between the password and confirm password fields. At the bottom of the form is a dark button labeled 'Створити акаунт' (Create account). Below the button, there is a link: 'Вже маєте акаунт? [Увійти](#)' (Already have an account? [Login](#)).

Рисунок 3.8 – Сторінка реєстрації

На рис. 3.9 показано інтерфейс призначений для викладачів, який дозволяє налаштовувати параметри навчального курсу. Можна вказати назву, детальний опис, додати співвикладачів через систему пошуку, обрати тип курсу (відкритий або закритий) та завантажити зображення обкладинки.

← Назад **Редагування курсу**

Створити новий курс

Назва курсу

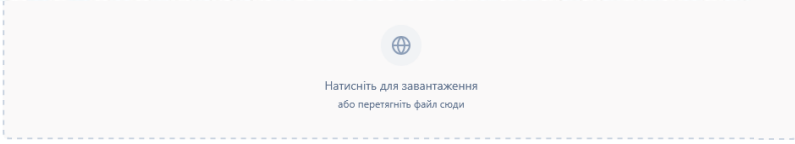
Опис

Співвикладачі Сбрано: 0

Співвикладачі матимуть доступ до редагування контенту курсу, створення завдань та оцінювання студентів.

Відкритий курс ☒ Студенти можуть записуватися самостійно. ☐

Обкладинка курсу



Натисніть для завантаження або перетягніть файл сюди

Підтримуються формати: JPG, PNG, WEBP. Макс розмір: 5MB.

Зберегти курс

Рисунок 3.9 – Форма створення курсу

На сторінці опису курсу (рис. 3.10) відображається загальна інформація про обраний курс.

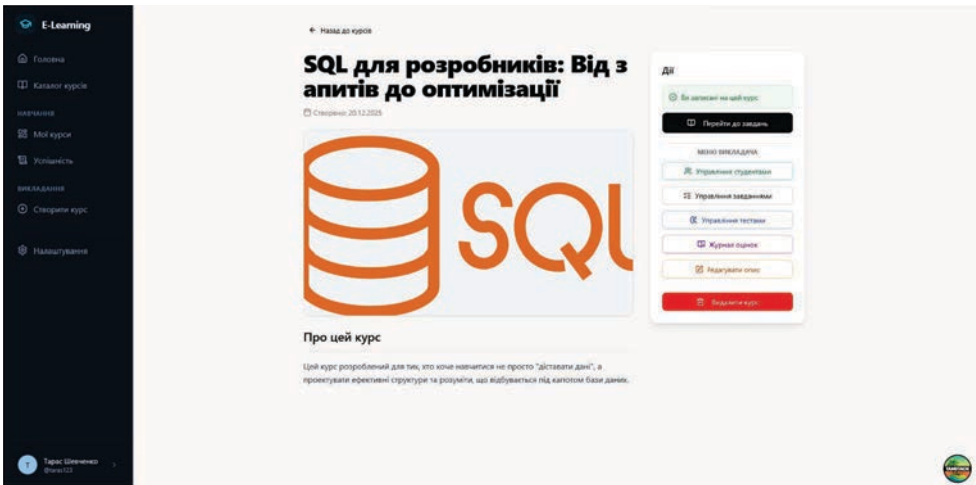


Рисунок 3.10 – Сторінка управління/опис курсу

Назва курсу, опис цілей навчання, список викладачів, а також кнопки для управління до розділів навчання, практичних завдань та тестів, якщо користувач є вчителем або адміністратором.

Сторінка управління завданнями курсу (рис. 3.11) містить перелік усіх доступних практичних робіт у межах курсу. Для кожного завдання вказується його назва та кінцевий термін здачі (дедлайн).

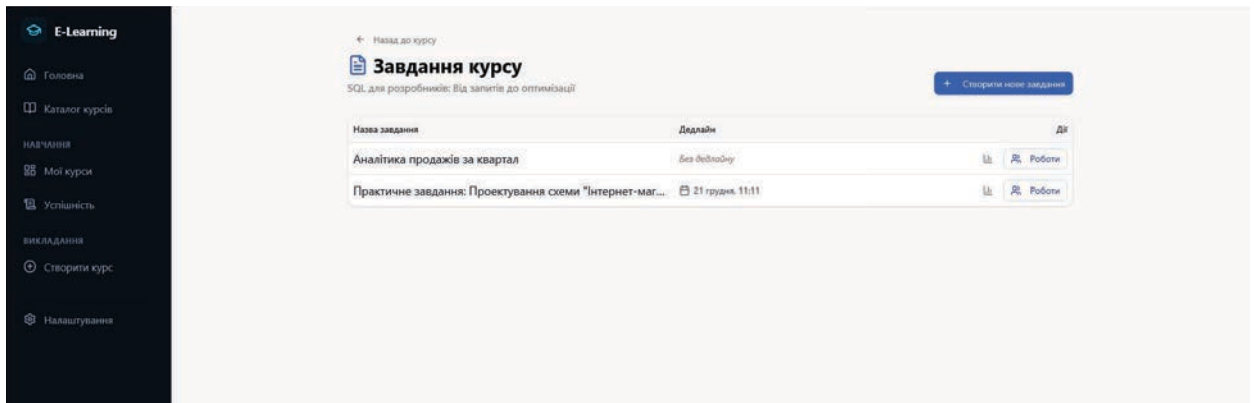


Рисунок 3.11 – Сторінку управління завданнями курсу

Форма створення завдання (рис. 3.12) дозволяє викладачу додати нове практичне завдання, вказавши її назву та детальні інструкції або умови виконання для студентів.

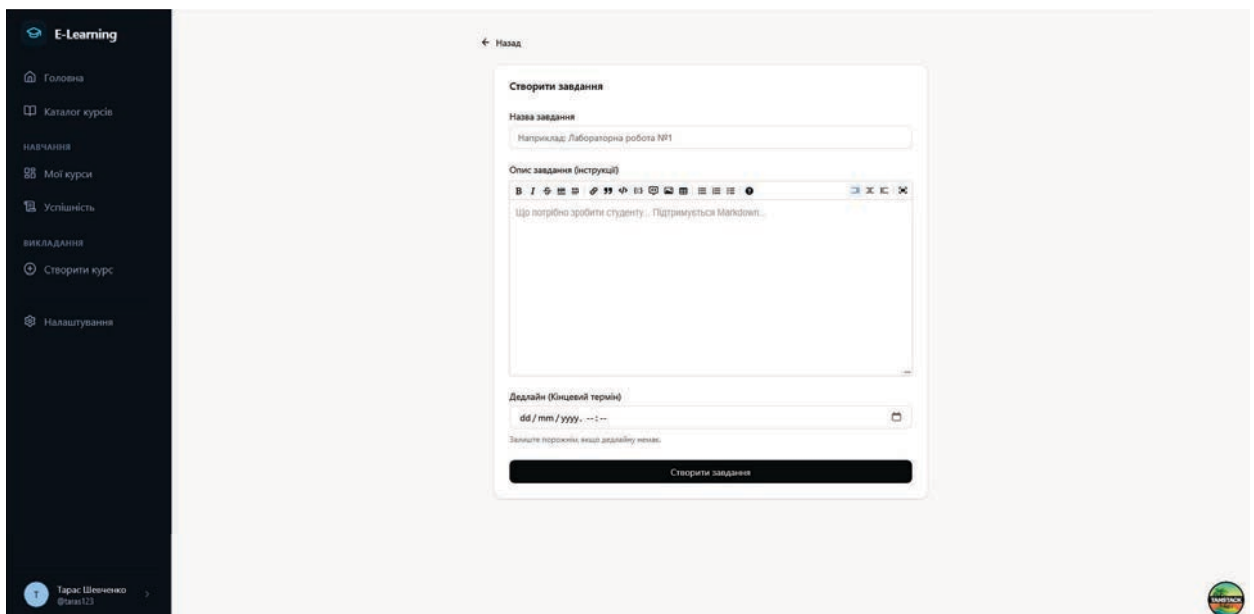


Рисунок 3.12 – Форма створення завдання

Сторінка перегляду та здачі завдання (рис. 3.13) містить опис технічного завдання, статус роботи, поле для завантаження файлу з відповіддю та блок приватних коментарів для спілкування з викладачем.

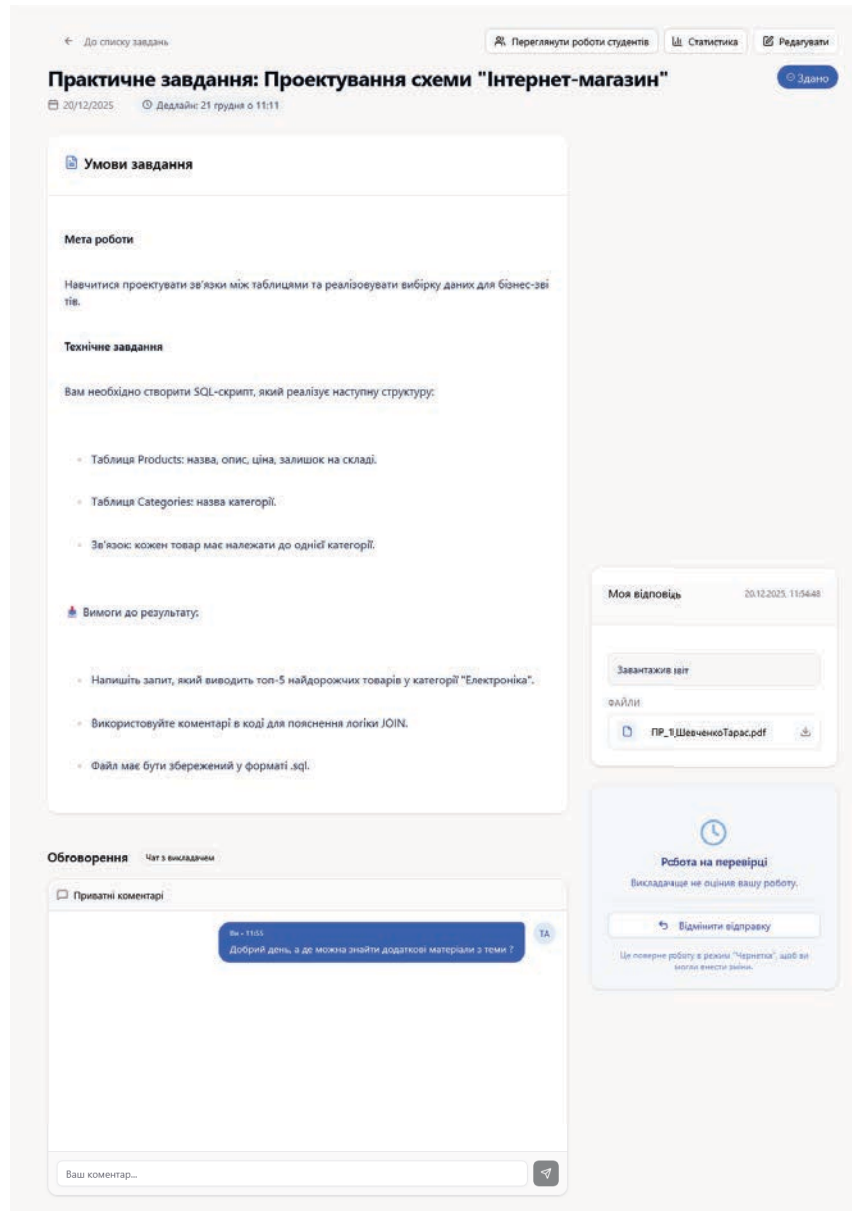


Рисунок 3.13 – Сторінка перегляду/здачі завдання

Сторінка перегляду зданих робіт студентами (рис. 3.14), де відображаються всі надіслані студентами роботи за конкретним завданням. Це дозволяє здійснювати моніторинг активності групи та переходити до перевірки окремих файлів.

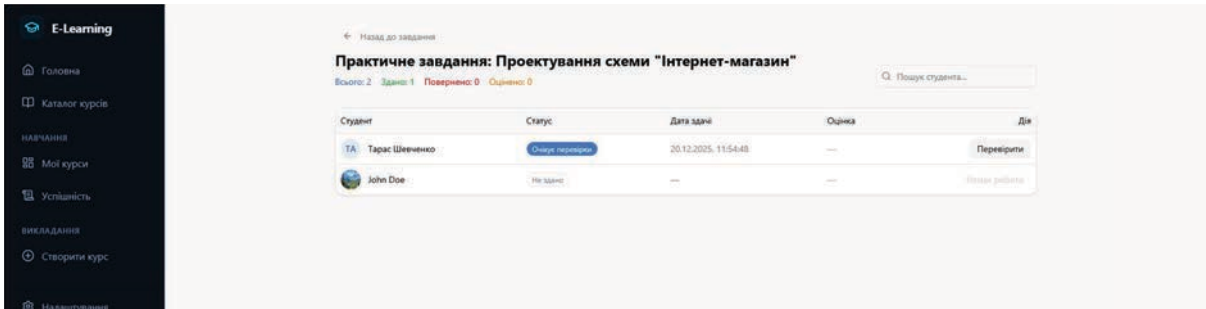


Рисунок 3.14 – Сторінка перегляду зданих робіт студентами

На сторінці оцінювання завдання (рис. 3.15), викладач має можливість переглянути завантажений студентом файл, виставити оцінку у відповідне поле та написати розгорнутий коментар із зауваженнями або порадами. На сторінці управління тестами курсу (рис. 3.16) наведено список створених тестів курсу з інформацією про ліміти часу для кожного з них.

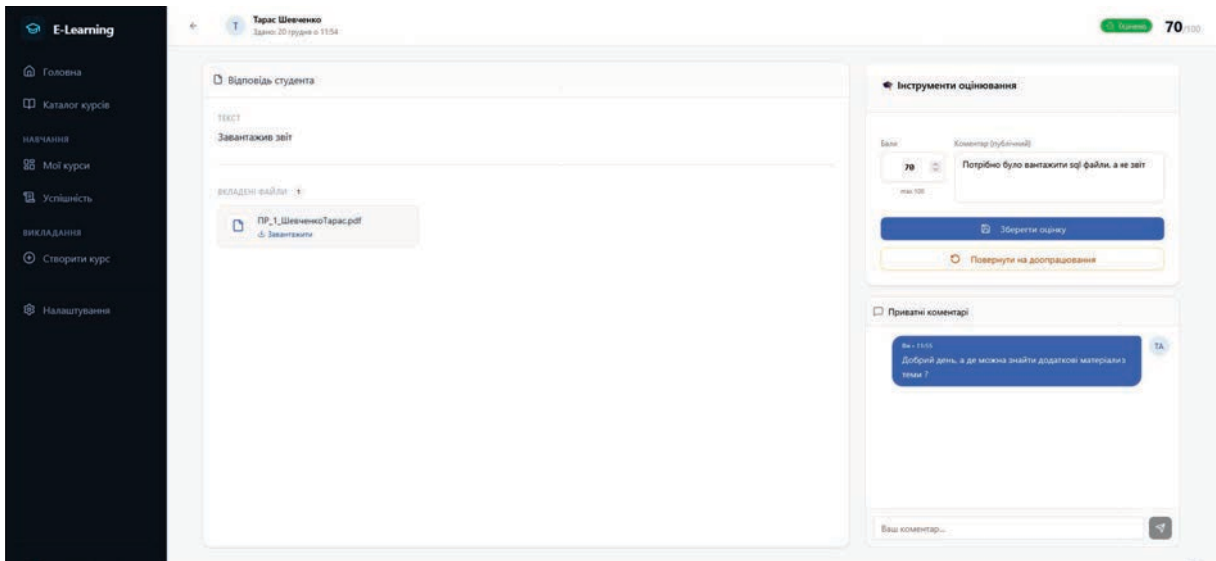


Рисунок 3.15 – Сторінка оцінювання завдання

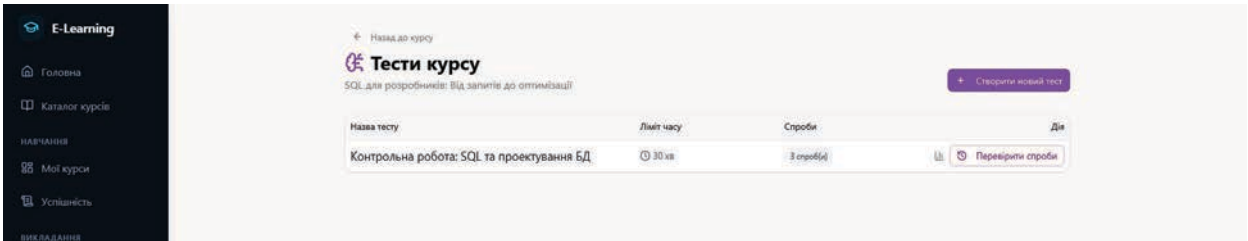


Рисунок 3.16 – Сторінку управління тестами курсу

Інтерфейс для створення тестів (рис. 3.17), що дозволяє налаштувати назву, кількість спроб, ліміт часу та метод оцінювання. Також він містить інструменти для додавання питань різних типів: одинарного вибору, множинного вибору або відкритої відповіді.

← Назад

Конструктор тесту

Назва тесту

Введіть назву...

Час (хвилини)

30

Кількість спроб

3

Метод оцінювання

Найкраща оцінка

#1

Одинарний вибір

Балів: 1

Яка команда використовується для повного видалення таблиці разом із її структурою з бази даних?

☐

DELETE TABLE

☐

TRUNCATE TABLE

☒

DROP TABLE

☐

REMOVE TABLE

+ Додати варіант відповіді

#2

Множинний вибір

Балів: 3

Які з наведених операторів використовуються для фільтрації даних у SQL запитих? (Оберіть усі правильні варіанти)

☒

WHERE

☐

ORDER BY

☒

HAVING

☒

BETWEEN

☐

GROUP BY

+ Додати варіант відповіді

#3

Відкрита відповідь

Балів: 6

Опишіть різницю між INNER JOIN та LEFT JOIN. У яких випадках варто використовувати саме LEFT JOIN?

Це питання вимагає текстової відповіді. Студент побачить поле для вводу тексту. Автоматичне оцінювання для цього типу недоступне.

+ Додати питання

Зберегти тест

Рисунок 3.17 – Сторінка створення тесту

Сторінка безпосереднього виконання тесту студентом (рис. 3.18). Вона відображає таймер зворотного відліку, текст питання, варіанти відповідей та кнопку для завершення тестування.

Контрольна робота: SQL та проектування БД

29:42

Питань: 3

1. Яка команда використовується для повного видалення таблиці разом із її структурою з бази даних?

☐ REMOVE TABLE

☐ TRUNCATE TABLE

☒ DROP TABLE

☐ DELETE TABLE

2. Які з наведених операторів використовуються для фільтрації даних у SQL запитих? (Оберіть усі правильні варіанти)

☒ WHERE

☐ ORDER BY

☐ GROUP BY

☒ BETWEEN

☒ HAVING

3. Опишіть різницю між INNER JOIN та LEFT JOIN. У яких випадках варто використовувати саме LEFT JOIN?

Незнаю

Завершити тест

Рисунок 3.18 – Сторінка проходження курсу

Після завершення тестування на екрані з'являється повідомлення про попередні результати (рис. 3.19), яке відображає відсотковий показник успішності проходження тесту.

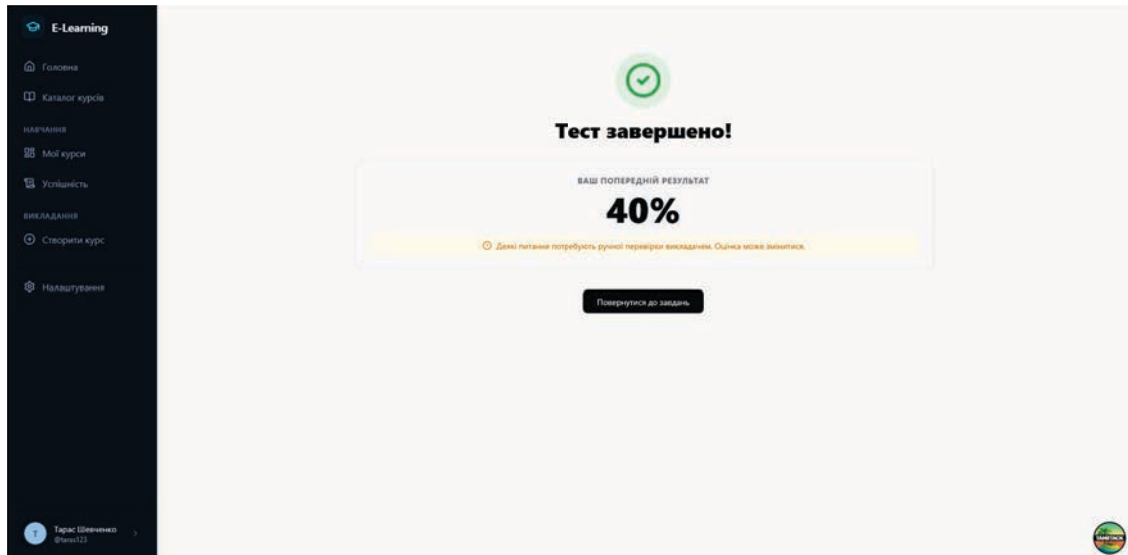


Рисунок 3.19 – Повідомлення про попередні результати тесту

На сторінці тесту (рис. 3.20) користувач може ознайомитися з інструкцією, побачити кількість доступних та використаних спроб, перейти до відповідей на спробі, а також розпочати тестування або перейти до редагування (для викладача).

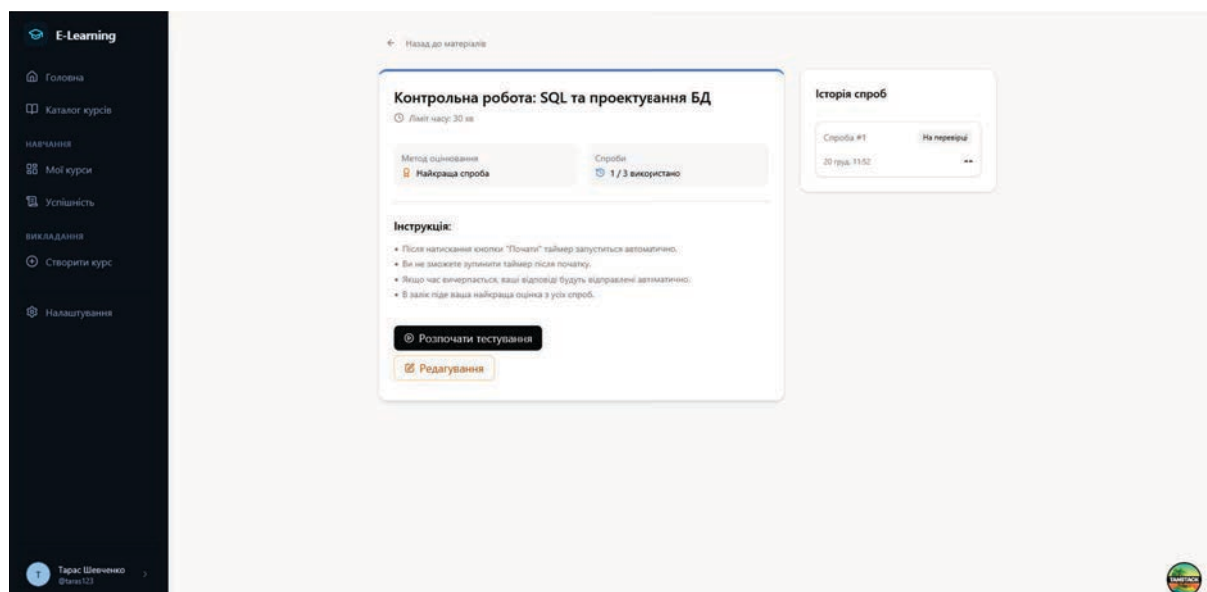


Рисунок 3.20 – Сторінка тесту

Сторінка перевірки тесту (рис. 3.21) демонструє детальний звіт за кожним питанням, де позначено правильні відповіді та вибір студента, що дозволяє виставити оцінку (для вчителя) або провести роботу над помилками (для студента).

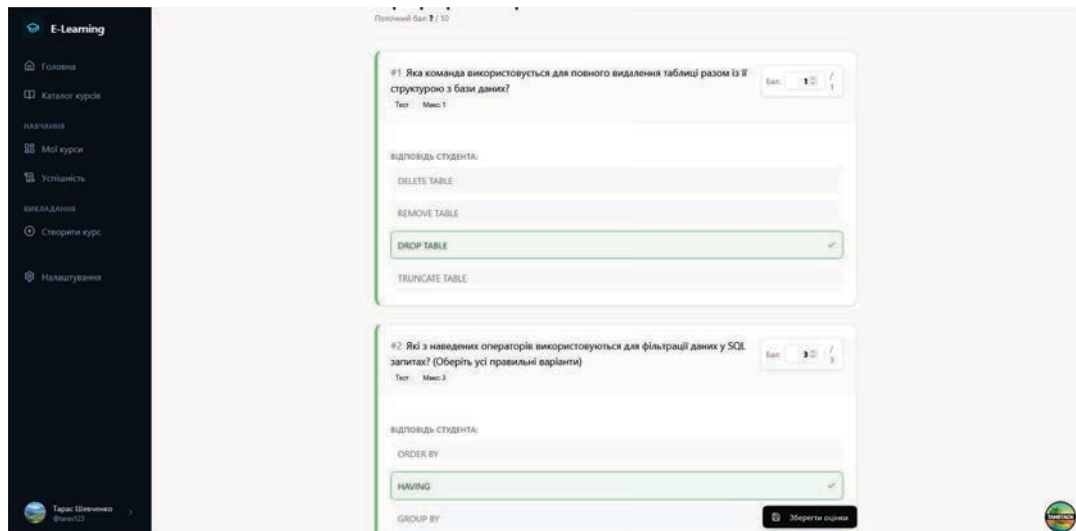


Рисунок 3.21 – Сторінка перевірки тесту

Для створення новинних повідомлень використовується сторінка створення оголошень (рис. 3.22), яка містить текстове поле для введення змісту публікації та надає можливість прикріплювати гіперпосилання та файли до оголошень.

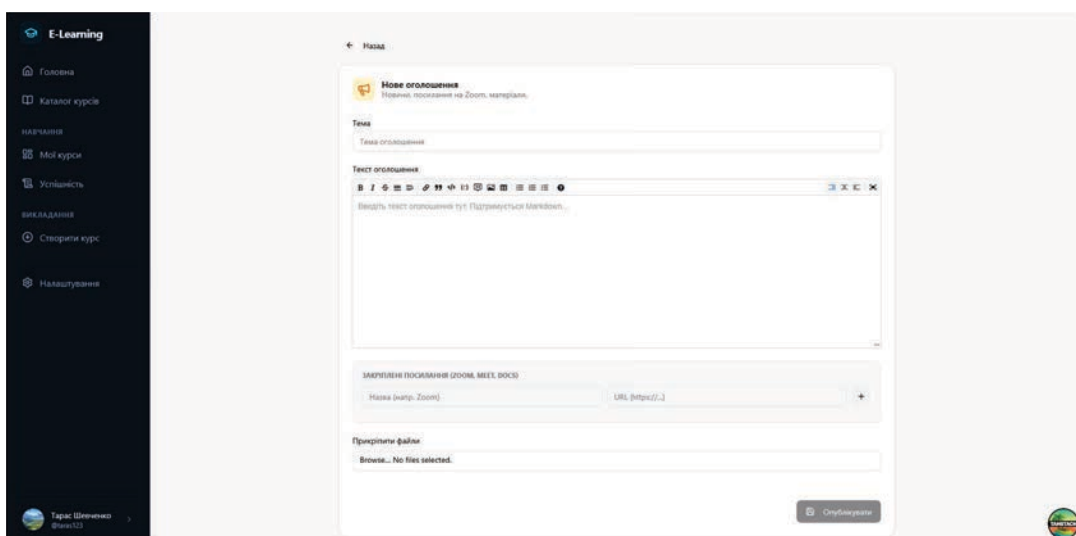


Рисунок 3.22 – Сторінка створення оголошень

На сторінці матеріалів курсу (рис. 3.23) структуровано відображаються всі навчальні ресурси, новини модуля, тести та практичні завдання курсу.

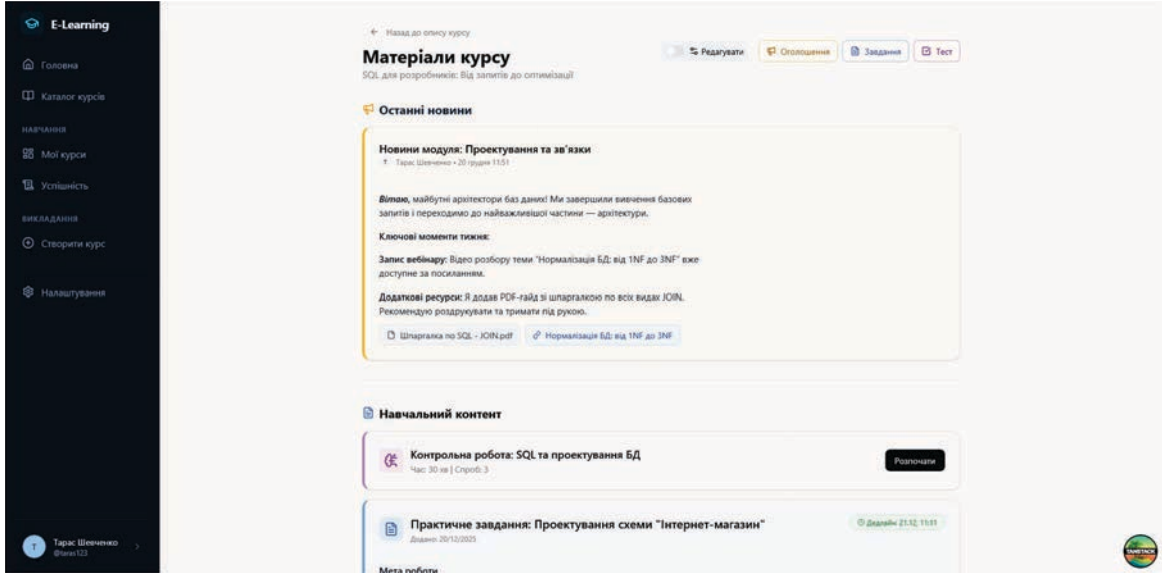


Рисунок 3.23 – Загальна стрічку курсу

Список учасників навчального процесу відображається на сторінці перегляду студентів курсу (рис. 3.24), де наведено дані про всіх зареєстрованих на курс осіб.

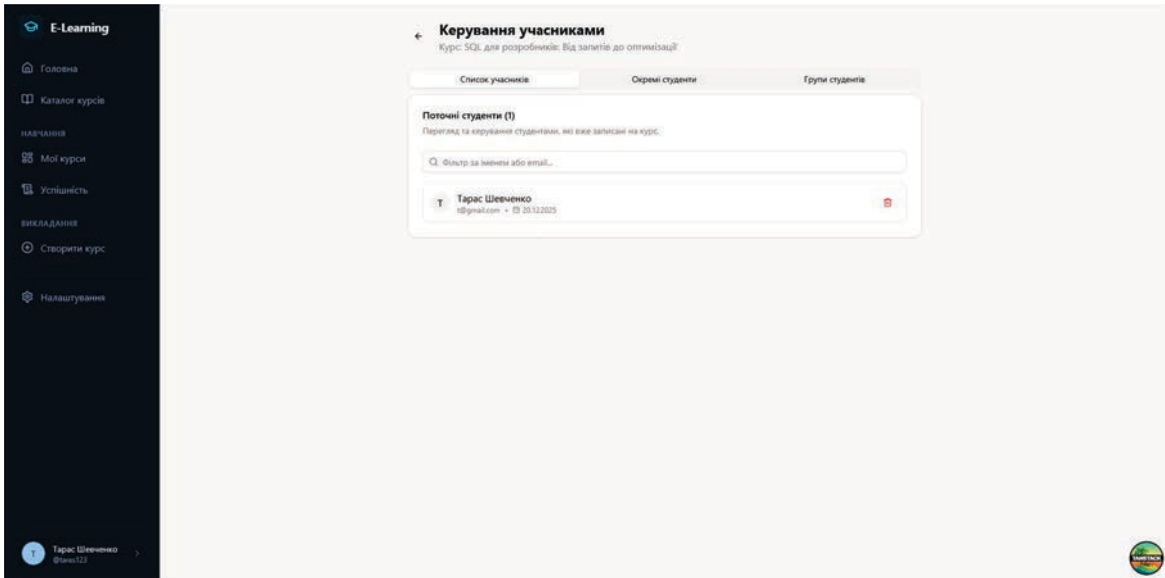


Рисунок 3.24 – Сторінка зі списком студентів зарахованих на курс

На сторінках для зарахування студентів (рис. 3.25) або академічних груп (рис. 3.26) відображаються доступні для реєстрації на курс студенти та академічні групи.

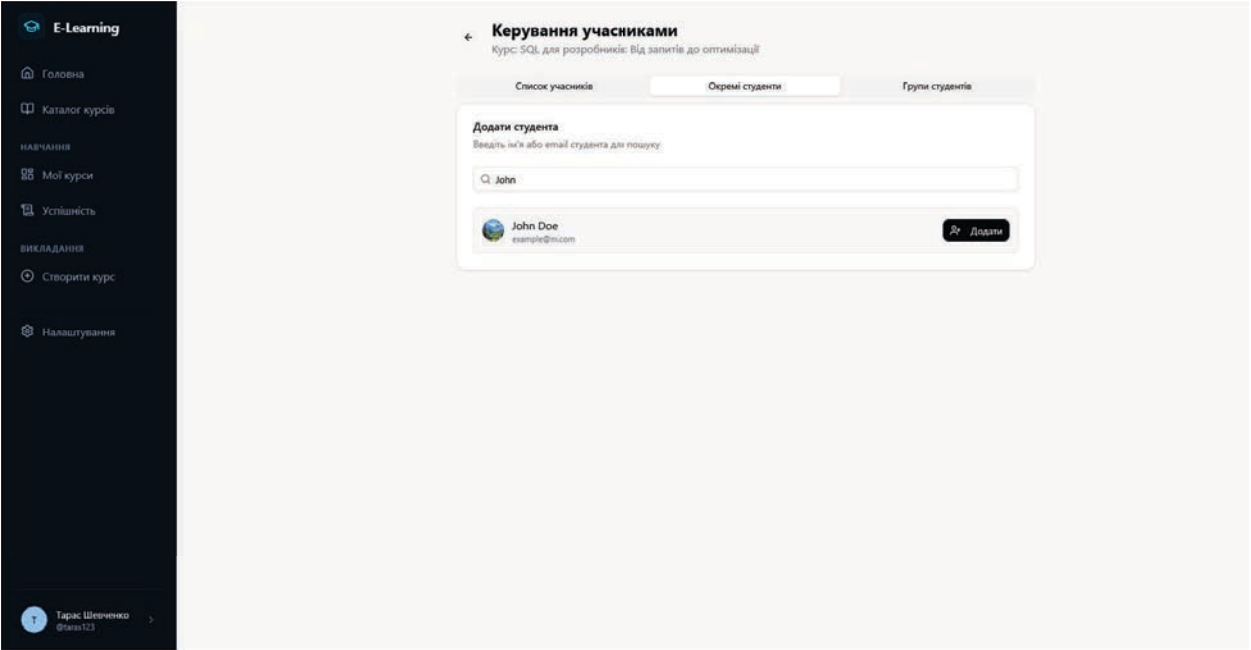


Рисунок 3.25 – Сторінка для зарахування студентів на курс

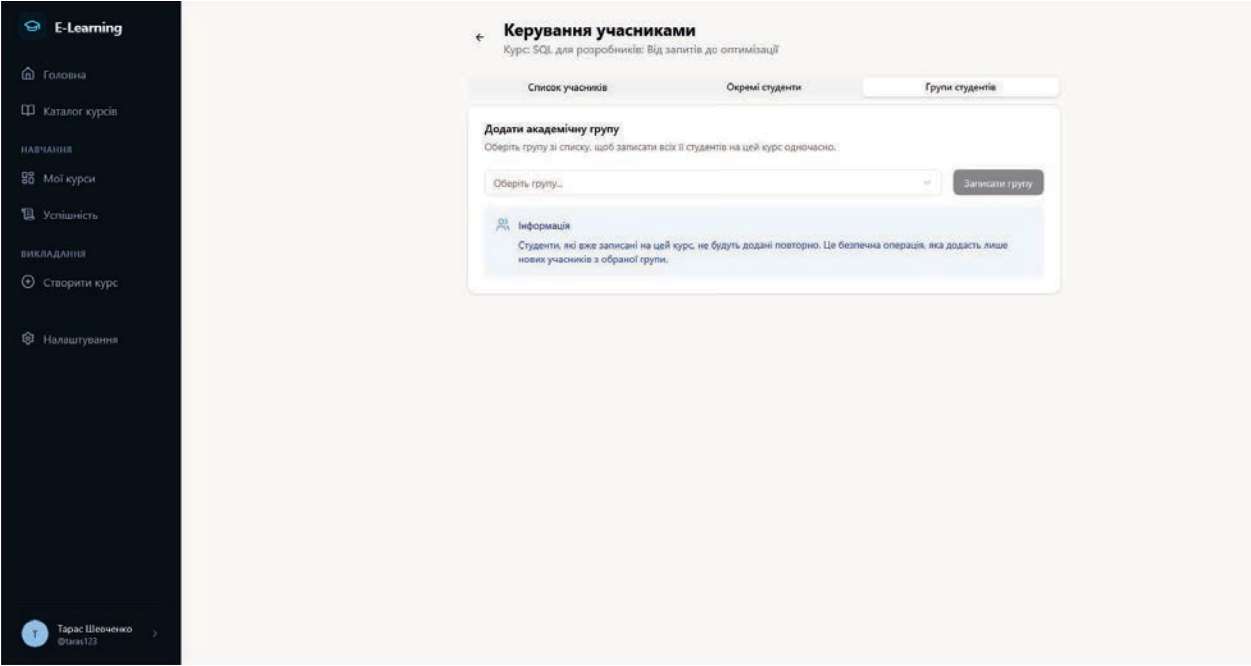


Рисунок 3.26 – Сторінка для зарахування академічних груп на курс

Кожен тест або практичне завдання має сторінку із статистичними даними (рис. 3.27 – 3.28), яка вказує на середній бал роботи по курсу, розподіл балів по гістограмі, активність студентів, середній час проходження та аналіз правильності відповіді по запитанням для тестів.

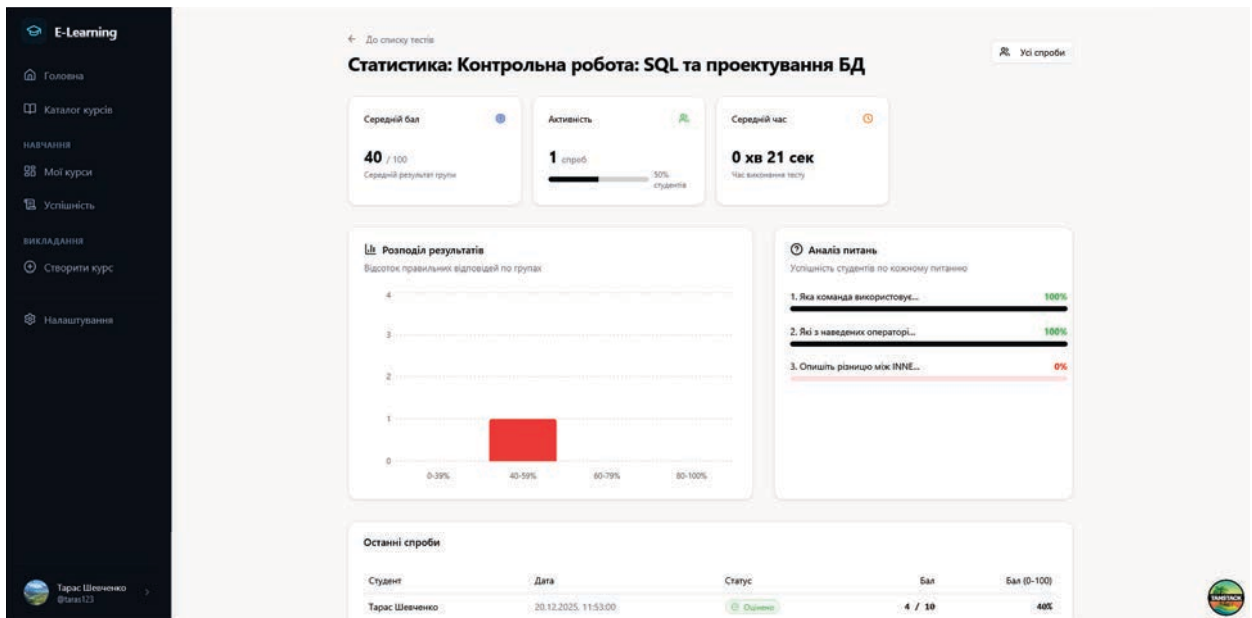


Рисунок 3.27 – Сторінка статистики тестування

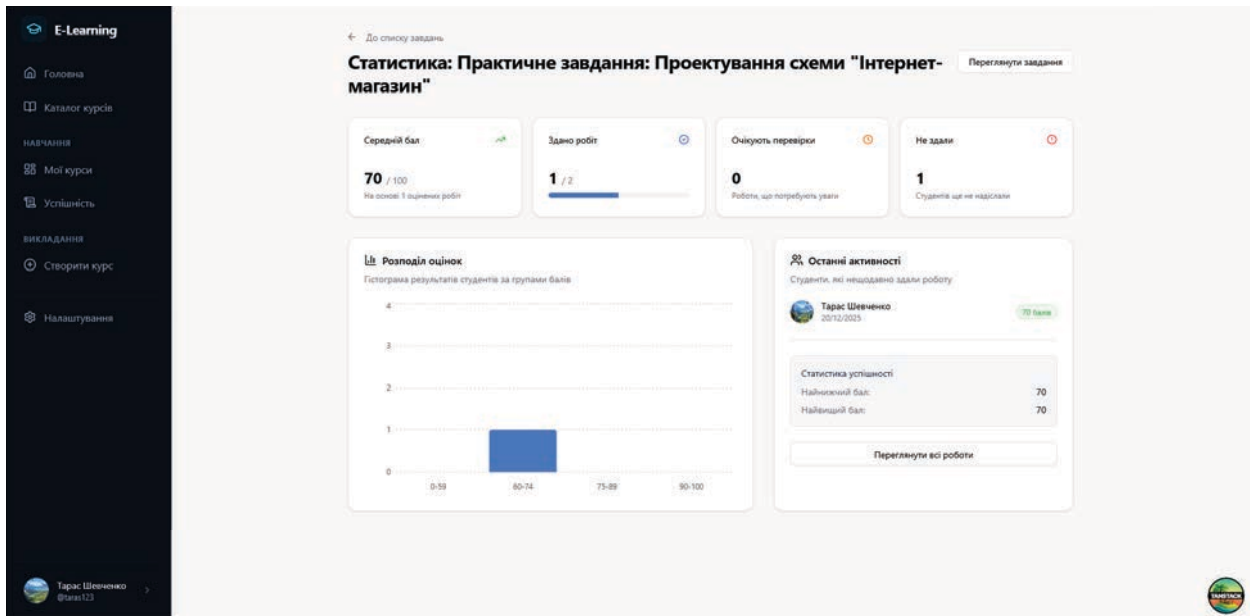


Рисунок 3.28 – Сторінка статистики завдання

Сторінку відображення каталогу курсів (рис. 3.29) на які записаний студент, відображені доступні курси для вивчення.

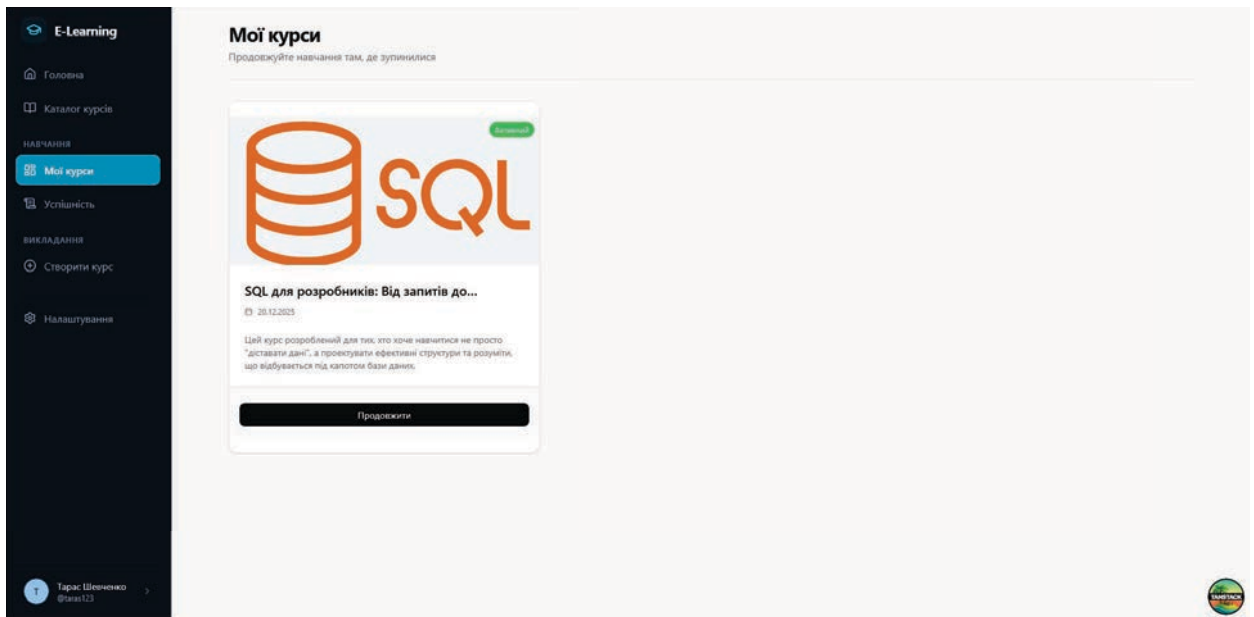


Рисунок 3.29 – Сторінка каталог курсів студента

У вікні редагування профілю (рис. 3.30) користувач має можливість змінити персональні дані, зокрема прізвище, ім'я та налаштування безпеки облікового запису.

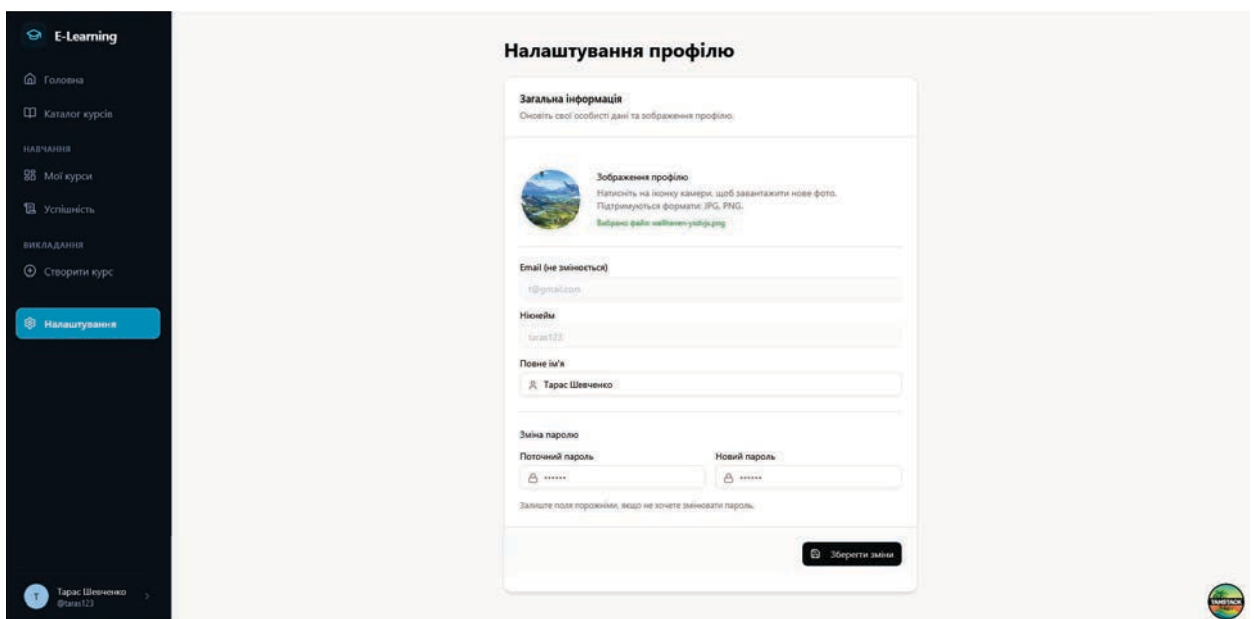


Рисунок 3.30 – Сторінка налаштування профілю користувача

На сторінці журналу оцінок по курсу (рис. 3.31) вчитель має змогу побачити всі виставлені оцінки за практичні завдання та тести, які містить курс.

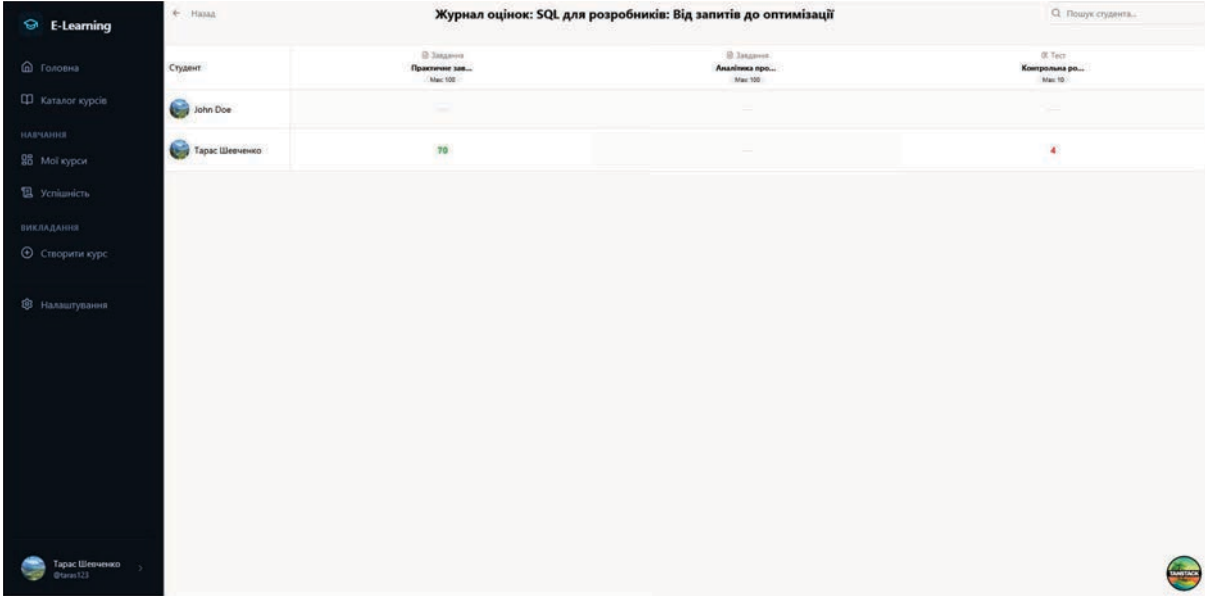


Рисунок 3.31 – Сторінка журнал оцінок по курсу

Журнал оцінок студента (рис. 3.32), надає змогу студенту переглянути його оцінки за завдання та тести, а також побачити розрахований середній бал у 100 бальній системі.

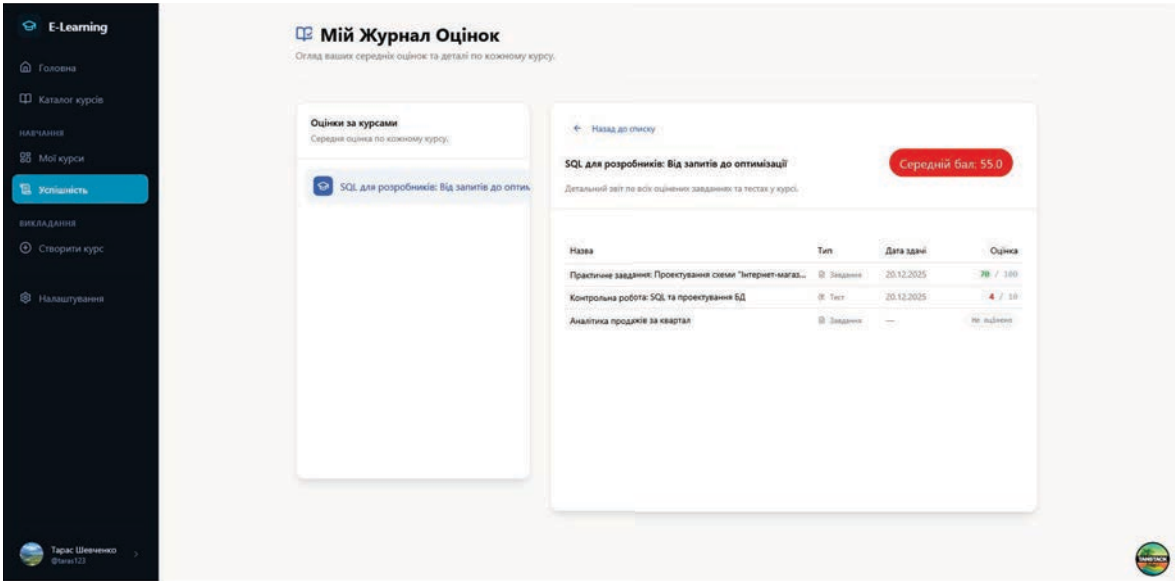


Рисунок 3.32 – Сторінка журнал оцінок студента

Бокова панель адміністратора (рис. 3.33) містить навігаційне меню для швидкого переходу між розділами управління користувачами, групами та глобальними оголошеннями.

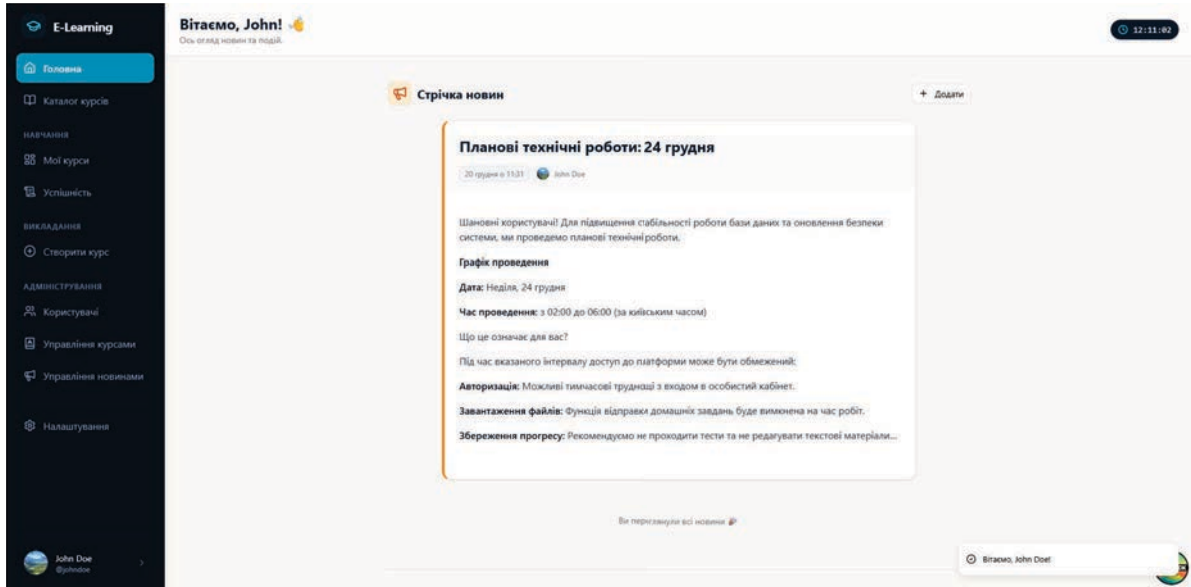


Рисунок 3.33 – Бокова панель адміністратора

Для роботи з базою користувачів призначена адміністративна сторінка управління користувачами (рис. 3.34), яка дозволяє переглядати список усіх акаунтів та виконувати їх пошук.

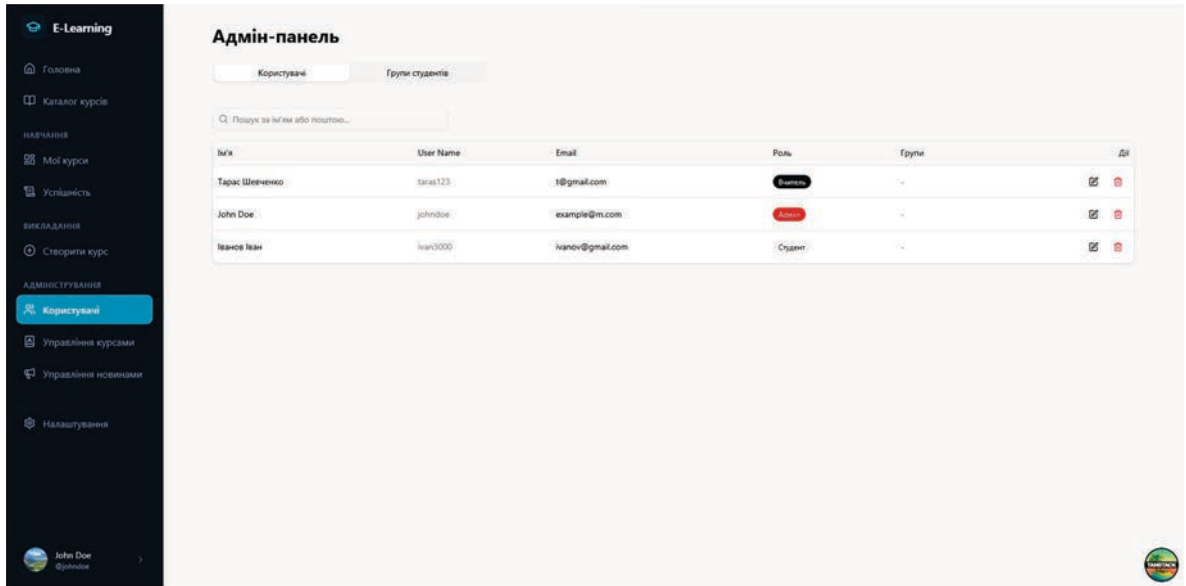
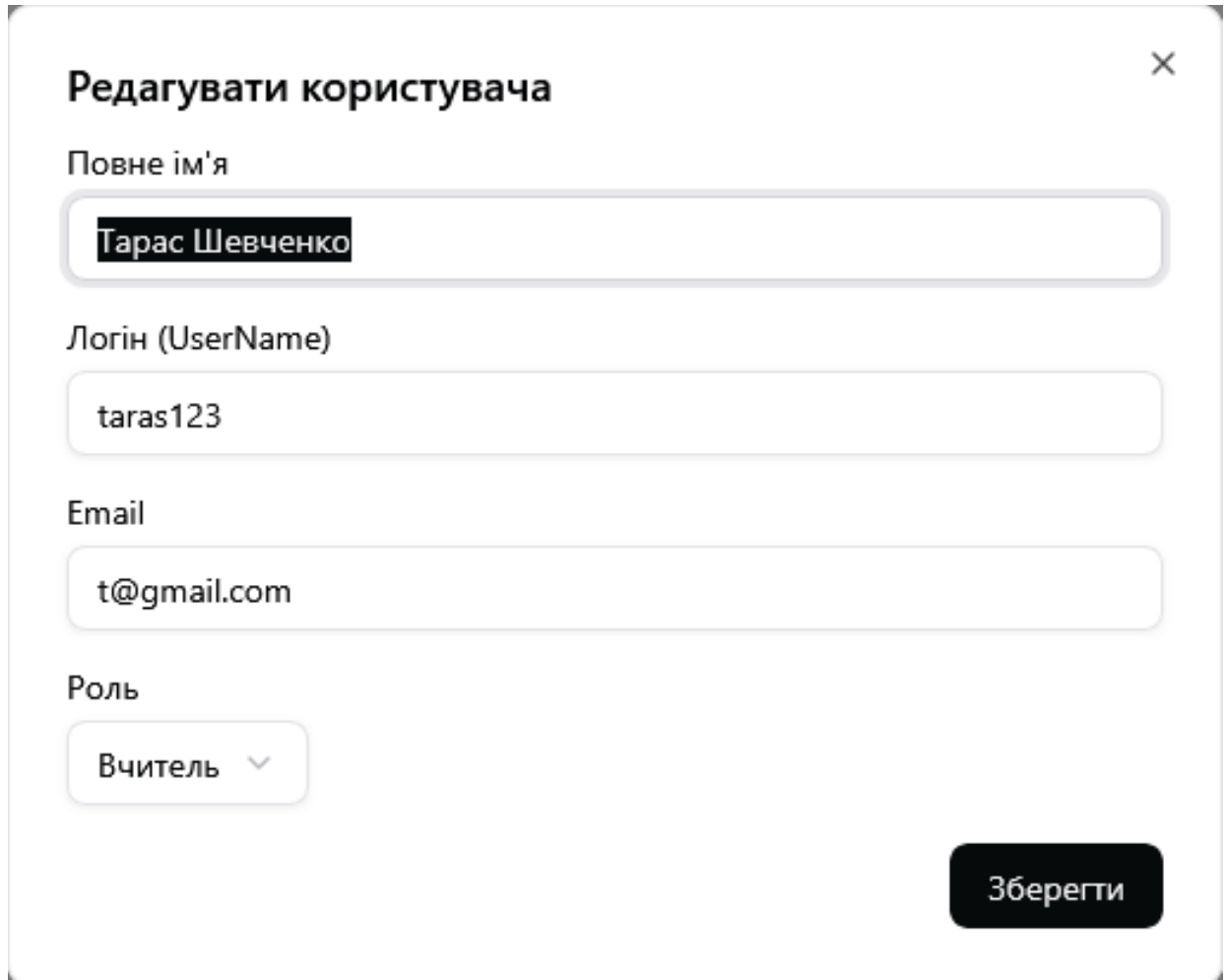


Рисунок 3.34 – Адміністративна сторінка для управління користувачами

Форма редагування користувача в адмін-панелі (рис. 3.35) дає змогу змінювати системну роль користувача (наприклад, призначити вчителем/адміністратором) та коригувати його дані.



Редагувати користувача

Повне ім'я

Тарас Шевченко

Логін (UserName)

taras123

Email

t@gmail.com

Роль

Вчитель

Зберегти

Рисунок 3.35 – Діалог редагування користувача

На сторінці управління групами (рис. 3.36) відображається перелік академічних груп із зазначенням кількості студентів та дати створення.

Для додавання нових структурних одиниць використовується форма створення групи, де задається назва та обираються учасники зі списку.

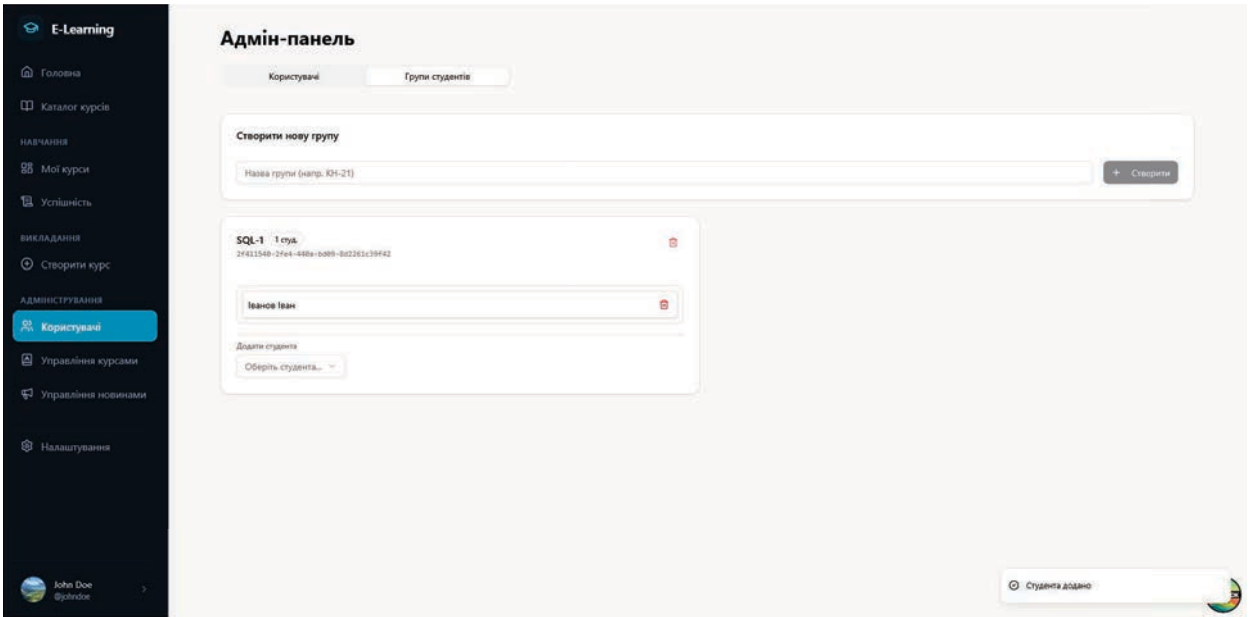


Рисунок 3.36 – Адміністративна сторінка для управління академічними групами

На сторінці управління курсами (рис. 3.37) відображається перелік курсів вебплатформи, власник (головний викладач курсу), статус курсу (відкритий/закритий), дата створення та дії редагування й видалення курсу.

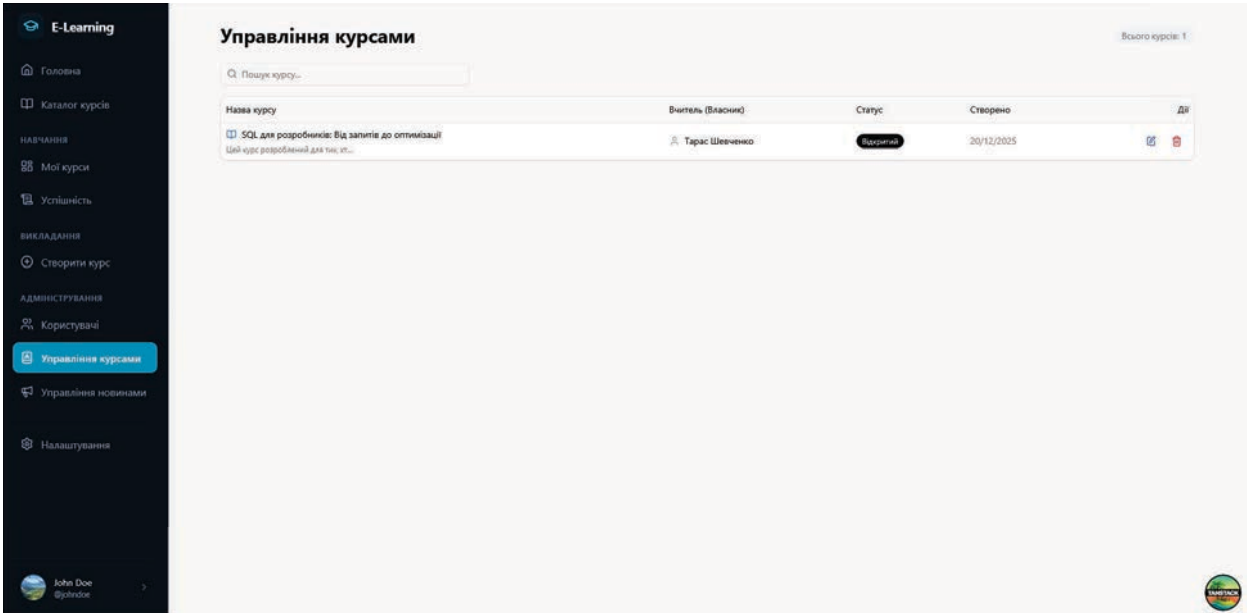


Рисунок 3.37 – Адміністративна сторінка для управління курсами

У формі редагування курсу (рис. 3.38) адміністратор має змогу змінити базові дані курсу (назву та опис) та змінити власника курсу, або перепризначити співвикладачів (вони мають змогу редагувати навчальний зміст курсу), й змінити статус курсу.

×

Редагувати курс

Змініть інформацію про курс, власника або налаштування доступу.

Назва

SQL для розробників: Від запитів до оптимізації

Опис

Цей курс розроблений для тих, хто хоче навчитися не просто "діставати дані", а проектувати ефективні структури та розуміти, що відбувається під капотом бази даних.

Власник курсу (Головний викладач)

Тарас Шевченко (t@gmail.com) ▾

Увага: Зміна власника передасть повні права на курс іншому користувачу.

Співвикладачі

Тарас Шевченко
t@gmail.com

Власник

John Doe
example@m.com

Позначені користувачі зможуть редагувати матеріали та оцінювати студентів.

Курс відкритий для самостійного запису

Скасувати

Зберегти зміни

Рисунок 3.38 – Діалог для редагування курсу

На сторінка для управління вмістом курсу (рис. 3.39) відображається список всіх елементів курсу: оголошення, тести та практичні завдання.

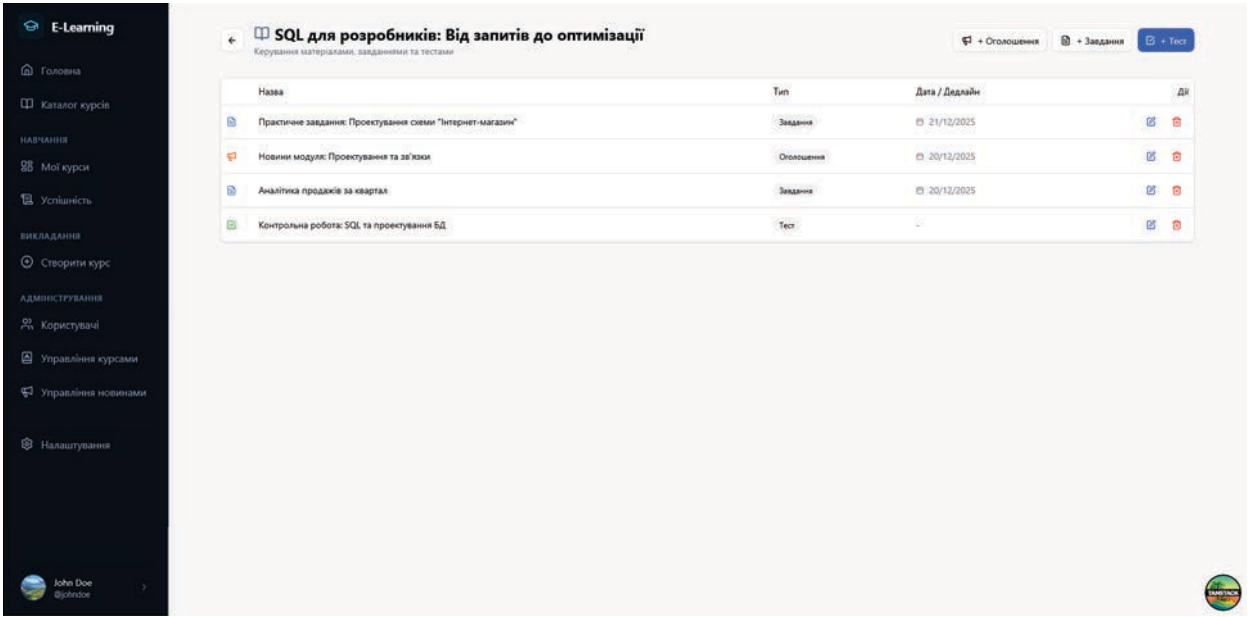


Рисунок 3.39 – Адміністративна сторінка для управління вмістом курсу

Адміністративна сторінка для управління глобальними оголошеннями (рис. 3.40) відображає список оголошень із головної сторінки вебплатформи.

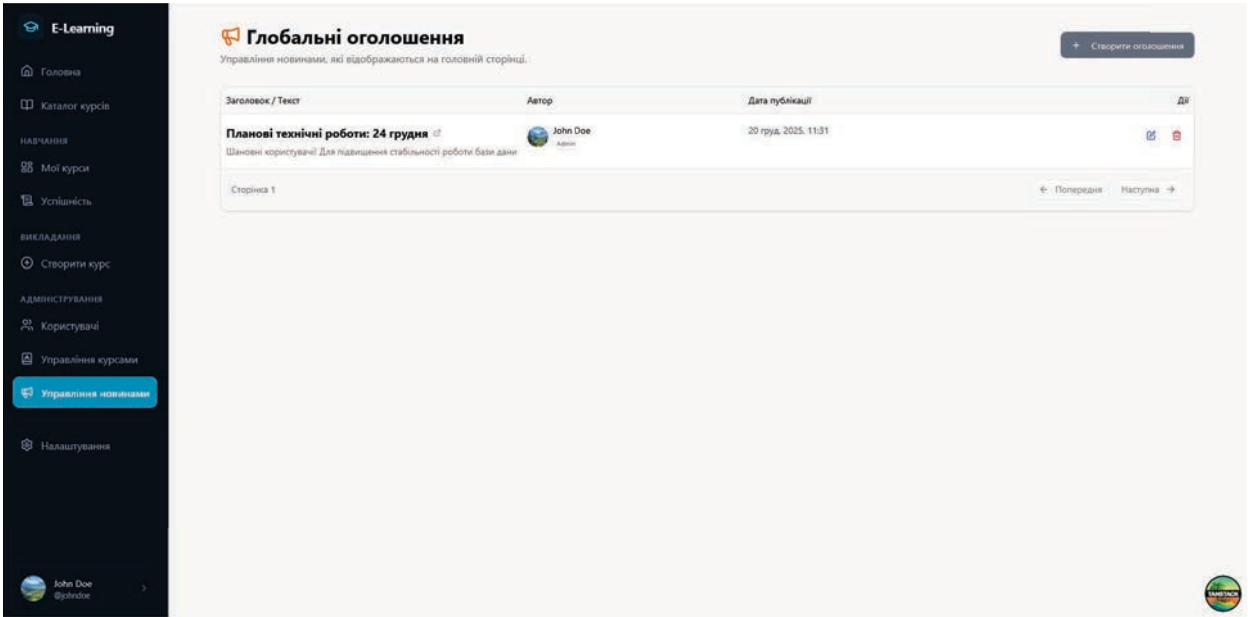


Рисунок 3.40 – Адміністративна сторінка для управління глобальними оголошеннями

3.5 Висновки за розділом

У третьому розділі було проведено детальне проектування архітектури та бази даних системи управління навчанням, що дозволило закласти надійний фундамент для подальшої реалізації програмного продукту. Застосована клієнт-серверна модель із чітким розподілом на рівень представлення, рівень бізнес-логіки та рівень даних забезпечує високу гнучкість розробки та масштабованість системи. Вибір стеку технологій на базі ASP.NET Minimal API для серверної частини та React для клієнтської дозволив створити високопродуктивну архітектуру, яка мінімізує витрати ресурсів на обробку HTTP-запитів та гарантує швидкий відгук інтерфейсу користувача.

Важливим результатом роботи стало проектування реляційної моделі бази даних у середовищі DBeaver, яка виступає центральним компонентом для зберігання та управління освітніми даними. Побудована ER-діаграма та деталізована структура таблиць дозволили врахувати складні взаємозв'язки між ключовими сутностями, такими як користувачі, навчальні курси, академічні групи та системи тестування.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ.

4.1 Призначення й умови застосування програми

Розроблена вебплатформа призначена для підтримки освітнього процесу через ефективне управління навчальною інформацією, комунікацію учасників та автоматизацію типових завдань. Основна мета системи полягає у створенні швидкого та масштабованого середовища для організації дистанційного та змішаного навчання.

Мінімальні системні вимоги та програмне забезпечення, що необхідні для стабільного функціонування серверної частини платформи:

- процесор: Intel Xeon E3-1220 або аналогічний з частотою від 3 ГГц;
- оперативна пам'ять: від 4 Gb RAM;
- дисковий простір: SSD об'ємом від 20 Gb (залежно від обсягу навчальних матеріалів);
- вебсервер: Nginx;
- середовище виконання: .NET 10 (Backend), NodeJS (Frontend);
- база даних: PostgreSQL.

Система побудована за архітектурою клієнт-сервер. Клієнтська частина забезпечує взаємодію користувача з інтерфейсом, а серверна – обробку запитів, роботу з базою даних та безпеку збереження інформації.

4.1 Тестування програми

Тестування проводилось з метою перевірки відповідності розробленого продукту функціональним вимогам, а також для оцінки надійності та якості роботи системи в реальних умовах. Процес включав перевірку всіх ключових модулів: від авторизації до системи оцінювання та аналітики.

Тестування розробленої вебплатформи включає наступні кроки. Першочергово проводиться функціональне тестування, під час якого система перевіряється на відповідність вимогам і специфікаціям [32]. Базою для цього етапу є перевірка того, що кожна функція програми працює належним чином шляхом надання вхідних даних та порівняння фактичного результату з очікуваним [32]. Оскільки цей вид тестування зосереджений на зовнішній поведінці програмного забезпечення і не вимагає знань внутрішнього вихідного коду, він здебільшого вважається тестуванням «чорного ящика» [32].

Важливою складовою є тестування зручності, яке допомагає визначити рівень навченості та зрозумілості інтерфейсу для кінцевого користувача [33]. Мета такого дослідження – переконатися, що користувачі зможуть самостійно та швидко розібратися в продукті, що безпосередньо впливає на конверсію та задоволеність ресурсом [33].

Під час тестування зручності оцінюються такі критерії, як інтуїтивність інтерфейсу, ефективність використання системи та рівень толерантності до помилок, що дозволяє користувачеві швидко виправити ситуацію у разі некоректних дій [33]. Окрему увагу приділяють зручності заповнення форм, наявності підказок та логічному розташуванню елементів навігації, щоб процес взаємодії з платформою був максимально простим і приємним [33].

4.2.1 Функціональне тестування

Метою функціонального тестування була перевірка коректності виконання програмою всіх завдань, визначених у постановці завдання. Нижче представлено чек-лист (табл. 4.1) з результатами перевірки основних можливостей платформи.

Таблиця 4.1 – Чек-лист функціонального тестування

№	Тест	Результат
1	2	3
1	Аутентифікація та профілі	
a	Перевірити реєстрацію нових користувачів	Passed
b	Перевірити авторизацію за логіном та паролем	Passed
c	Перевірити можливість зміни аватарки, імені користувача та паролю на сторінці налаштування профілю	Passed
2	Управління курсами	
a	Перевірити можливість створення нового курсу викладачем	Passed
b	Перевірити публікацію навчальних матеріалів різних форматів (оголошення, практичні завдання, тести)	Passed
3	Завдання та оцінювання	
a	Перевірити завантаження студентом виконаного завдання	Passed
b	Перевірити можливість викладача виставити оцінку та надати відгук	Passed
4	Тестування та статистика	
a	Перевірити створення та проходження онлайн-тестів	Passed
b	Перевірити відображення статистики успішності студентів	Passed

4.2.2 Тестування зручності сайту

Під час тестування інтерфейсу було встановлено, що платформа має сучасний, лаконічний дизайн, який не перевантажений зайвими елементами, що вирішує проблему складності аналогів, як-от Moodle.

Навігаційна панель забезпечує швидкий доступ до списку курсів, журналу оцінок та особистого кабінету. Дизайн адаптований під різні типи пристроїв, що дозволяє комфортно працювати з платформою як з комп'ютерів, так і з планшетів. Швидкість завантаження сторінок є високою завдяки використанню сучасного стеку технологій.

Було виявлено, що система забезпечує інтуїтивно зрозумілий шлях для студента від проходження тестів до здачі звіту. Проте, як і в прикладі, було помічено потребу в подальшому вдосконаленні доступності інтерфейсу для користувачів з особливими потребами (наприклад, підтримка екранних читців).

4.3 Висновки за розділом

У четвертому розділі було визначено призначення розробленої системи та встановлено умови її експлуатації. Проведено комплексне тестування функціональних можливостей вебплатформи, результати якого підтвердили повну відповідність продукту вимогам, висунутим на етапі проектування. Також було проаналізовано зручність інтерфейсу, що дозволило підтвердити перевагу розробленого рішення над існуючими аналогами в аспекті простоти та швидкодії.

ВИСНОВКИ

У результаті виконання магістерської роботи було успішно розроблено вебплатформу для підтримки освітнього процесу, що дозволило повністю досягти поставленої мети щодо зменшення часових витрат на організацію навчання та підвищення оперативності взаємодії між усіма учасниками освітнього середовища. Використання сучасних методів програмної інженерії та актуального стеку технологій, зокрема мов програмування C# і TypeScript у поєднанні з фреймворками ASP.NET та React, забезпечило створення високоефективного та масштабованого програмного продукту. Розроблена система успішно автоматизує низку критично важливих операцій від публікації навчальних завдань до збору результатів та аналітики активності, що безпосередньо сприяє підвищенню доступності матеріалів та покращенню комунікації. Проведений аналіз існуючих рішень, таких як Canvas та Moodle, підтвердив актуальність розробки більш гнучкого та легковагового інструменту, який відповідає викликам сучасної цифрової трансформації освіти. Таким чином, створена вебплатформа є дієвим програмним засобом для впровадження дистанційного та змішаного навчання, що підтверджує успішне виконання всіх поставлених завдань дослідження.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Цифрова трансформація освіти і науки | Міністерство освіти і науки України [Електронний ресурс]. – Режим доступу: <https://mon.gov.ua/tag/tsifrova-transformatsiya-osviti-i-nauki?&tag=tsifrova-transformatsiya-osviti-i-nauki>
2. Що таке LMS та як підібрати LMS-систему для навчання у 2025 році [Електронний ресурс]. – Режим доступу: <https://academyocean.com/ua/blog/post/shho-take-lms-sistema-povnii-gid-po-viboru-platforni-dlya-navcannya-u-2025-roci-1>
3. Samson, M. Challenges and barriers to learning management systems adoption in higher education: a thematic literature review. 2025. [Electronic resource] / M. Samson. – Access mode: https://www.researchgate.net/publication/394254129_CHALLENGES_AND_BARRIERS_TO_LEARNING_MANAGEMENT_SYSTEMS_ADOPTION_IN_HIGHER_EDUCATION_A_THEMATIC_LITERATURE_REVIEW
4. Akhmetov, B. Analysis of modern LMS platforms in Kazakhstan: Structure, functionality, cybersecurity [Electronic resource]. / B. Akhmetov, S. Gnatyk, B. Akhmetov, B. Tynymbayev // – Cyber Security and Data Protection 2024 (CSDP-2024) – 2024. – Vol. 3800. – P. 106-113. – Access mode: <https://ceur-ws.org/Vol-3800/short4.pdf>
5. Bradley, V. M. Learning Management System (LMS) use with online instruction. / V. M. Bradley // International Journal of Technology in Education (IJTE). – 2021. – Vol. 4, No. 1. – P. 68-92. – DOI: 10.46328/ijte.36
6. Turnbull, D. An Overview of the Common Elements of Learning Management System Policies in Higher Education Institutions / D. Turnbull, R. Chugh, J. Luck // TechTrends. – 2022. – Vol. 66, No. 5. – P. 855-867. – DOI: 10.1007/s11528-022-00752-7
7. Canvas LMS vs G-Classroom: Full Comparison in 2025 [Electronic resource]. – Access mode: <https://uteach.io/compare/canvas-lms-vs-google->

classroom

8. Bousboula, T. Comparative Study between Moodle, Canvas and Google Classroom / T. Bousboula, F. El Fadli, M. Khaldi // The International Journal of Research and Innovation in Social Science (IJRISS). – 2025. Vol. 9. No. 3 – P. 2819-2824. – DOI: 10.47772/IJRISS.2025.90300219

9. Moodle проти Canvas проти Blackboard: найкраща LMS 2023 року [Електронний ресурс]. – Режим доступу: <https://www.webhostingzone.org/uk/%D0%B1%D0%BB%D0%BE%D0%B7%D1%96/moodle-vs-canvas-vs-blackboard-%D0%BD%D0%B0%D0%B9%D0%BA%D1%80%D0%B0%D1%89%D1%96-lms-2023-%D1%80%D0%BE%D0%BA%D1%83>

10. Ruby vs PHP: What to Consider Before You Decide [Electronic resource]. – Access mode: <https://www.bairesdev.com/blog/ruby-vs-php/>

11. C# Vs PHP For Web Development in 2024 [Electronic resource]. – Access mode: <https://technologicalblog.hashnode.dev/c-vs-php-for-web-development-in-2024>

12. The Good and the Bad of C# Programming [Electronic resource]. – Access mode: <https://www.altexsoft.com/blog/c-sharp-pros-and-cons/>

13. Saglam, V. Why I Chose C# over PHP [Electronic resource] / V. Saglam. – Access mode: <https://medium.com/@vahitsaglam/why-i-chose-c-over-php-e011ee9ff082>

14. Comparing PHP and Ruby Languages: Which suits your startup best? [Electronic resource]. – Access mode: <https://proxify.io/articles/comparing-php-and-ruby-languages>

15. TypeScript vs. JavaScript — Choosing the Right Language for Your Project [Electronic resource]. – Access mode: <https://www.sencha.com/blog/typescript-vs-javascript-choosing-the-right-language-for-your-project/>

16. Difference between TypeScript and JavaScript [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/typescript/difference-between->

typescript-and-javascript/

17. Dart vs JavaScript: detailed comparison [Electronic resource]. – Access mode: <https://blog.codemagic.io/dart-vs-javascript/>

18. Difference between TypeScript and Dart Languages [Electronic resource]. – Access mode: <https://www.aalpha.net/articles/different-between-typescript-and-dart/>

19. Що розробляють на ASP.NET? [Електронний ресурс]. – Режим доступу: <https://foxminded.ua/asp-net-tse/>

20. Angular vs React vs Vue: Core Differences [Electronic resource]. – Access mode: <https://www.browserstack.com/guide/angular-vs-react-vs-vue>

21. Angular Vs React Vs Vue: Which One To Choose [Electronic resource]. – Access mode: <https://www.tatvasoft.com/blog/angular-vs-react-vs-vue/>

22. Comparing Database Management Systems: MySQL, PostgreSQL, MSSQL Server, MongoDB, Elasticsearch, and others [Electronic resource]. – Access mode: <https://www.altexsoft.com/blog/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/>

23. PostgreSQL vs. MySQL: What's the Difference? [Electronic resource]. – Access mode: <https://www.ibm.com/think/topics/postgresql-vs-mysql>

24. Visual Studio vs Visual Studio Code: The Ultimate Guide (2025) [Electronic resource]. – Access mode: <https://blog.ndepend.com/visual-studio-vs-visual-studio-code-the-ultimate-guide/>

25. Free (Non-Commercial) JetBrains Rider vs. Visual Studio 2022 Community Edition [Electronic resource]. – Access mode: <https://visualstudiomagazine.com/articles/2024/10/28/compare-new-non-commercial-jetbrains-rider-with-visual-studio-community-edition.aspx>

26. Moving away from Visual Studio to JetBrains Rider [Electronic resource]. – Access mode: <https://ankitvijay.net/2020/09/22/visual-studio-to-rider/>

27. Visual Studio vs Visual Studio Code: What's the Difference? [Electronic resource]. – Access mode: <https://codeop.tech/visual-studio-vs-visual->

studio-code-whats-the-difference/

28. Основні типи архітектури програмного забезпечення [Електронний ресурс]. – Режим доступу: <https://www.artofba.com/uk/post/main-types-of-software-architecture>

29. Singh, A. Evolutionary Architectures in Web Applications: A Comprehensive Study of Client-Server, Multi-Tier, and Service-Oriented Approaches / A. Singh // International Journal for Multidisciplinary Research (IJFMR). – 2024. – Vol. 6, No. 2. – DOI: 10.36948/ijfmr.2024.v06i02.18266

30. Nyabuto, G. Client-server Architecture, a Review / G. Nyabuto // International Journal of Advanced Science and Computer Applications. – 2024. – Vol. 3, No. 2. – DOI: 10.47679/ijasca.v3i1.48

31. Zhang, H. Architecture of Network and Client-Server model [Electronic resource] / H. Zhang // Advances in Electrical Engineering Systems. – 2013. – arXiv:1307.6665. – DOI: 10.48550/arXiv.1307.6665 – Access mode: <https://arxiv.org/abs/1307.6665>

32. Різниця між функціональним і нефункціональним тестуванням [Електронний ресурс]. – Режим доступу: <https://training.qatestlab.com/blog/technical-articles/difference-between-functional-and-non-functional-testing/>

33. Як і навіщо тестувати юзабіліті сайту [Електронний ресурс]. – Режим доступу: <https://training.qatestlab.com/blog/technical-articles/how-and-why-usability-testing/>

ДОДАТОК А
Технічне завдання

Вступ

Організація сучасного навчального процесу вимагає впровадження ефективних цифрових інструментів для дистанційної та змішаної освіти. Традиційні методи управління навчанням часто призводять до значних часових витрат на рутинні операції, такі як публікація матеріалів, збір робіт та комунікація між учасниками. Використання систем управління навчанням (LMS) дозволяє автоматизувати ці процеси, забезпечуючи централізований доступ до ресурсів та підвищуючи оперативність взаємодії викладачів і студентів.

Розробка спеціалізованої вебплатформи спрямована на створення гнучкого, легковагового та продуктивного рішення, що відповідає сучасним методам програмної інженерії.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу магістра на тему: «Розробка вебплатформи для підтримки освітнього процесу», затверджене Національним університетом «Запорізька політехніка» № 447 від 30 вересня 2025 р.

A.2 Призначення розробки

Програмний продукт (LMS-платформа) призначений для використання в освітніх закладах, центрах підвищення кваліфікації та для організації внутрішнього корпоративного навчання. Система забезпечує єдине середовище для публікації навчального контенту та контролю успішності.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення має реалізовувати наступні функції:

- авторизація та реєстрація користувачів;
- створення структури курсів, додавання текстових та мультимедійних матеріалів, посилань на ресурси;
- можливість створення викладачем важливих оголошень у межах курсу;
- автоматизована публікація навчальних завдань, встановлення термінів (дедлайнів), прикріплення файлів з відповідями;
- створення тестів із різними типами запитань, миттєва перевірка відповідей системою або ручна перевірка (у разі відкритих запитань або помилок з варіантами відповідей) та виставлення балів, встановлення часових лімітів на проходження тесту та обмеження кількості спроб;
- виставлення балів, ведення журналу успішності та надання зворотного зв'язку викладачем;
- відображення статистики активності студентів та прогресу виконання навчальної програми;
- вбудовані засоби для обміну коментарями до робіт.

A.3.2 Вимоги до інтерфейсу

Інтерфейс має бути мінімалістичним, щоб не відволікати від навчального процесу та забезпечувати зручну навігацію між навчальними матеріалами курсу.

А.3.3 Вимоги до надійності

Забезпечення цілісності даних при одночасній роботі великої кількості користувачів. Використання транзакційного підходу (ACID) для роботи з базою даних. Коректна обробка помилок (наприклад, при завантаженні файлів великого розміру або розриві з'єднання).

А.3.4 Умови експлуатації

Для експлуатації вебплатформи для підтримки освітнього процесу потрібен сучасний веббраузер, такий як Mozilla Firefox, Google Chrome, Microsoft Edge або Opera.

Для доступу до освітнього контенту та особистого кабінету користувача потрібне активне та безперебійне підключення до мережі Інтернет. Без підключення до мережі Інтернет користувач не зможе взаємодіяти з сервером.

Експлуатація серверної частини платформи має забезпечуватися середовищем з підтримкою .NET Core та доступом до бази даних, що підтримує SQL-запити.

А.3.5 Вимоги до складу та параметрів технічних засобів

ПЗ підтримки освітнього процесу має забезпечуватися серверною частиною з такими мінімальними параметрами технічних засобів:

- процесор: Intel Xeon E3-1220 або аналогічний з частотою від 3 ГГц;
- оперативна пам'ять: від 4 Gb RAM;
- дисковий простір: SSD об'ємом від 20 Gb (залежно від обсягу навчальних матеріалів);
- вебсервер: Nginx;

- середовище виконання: .NET 10 (Backend), NodeJS (Frontend);
- база даних: PostgreSQL.

A.4 Вхідні дані

Вхідними даними є профілі користувачів, контент курсів, база тестових питань, файли відповідей студентів та тексти оголошень.

A.5 Порядок контролю та приймання

Робота має виконуватися у відповідності з затвердженим завданням і контролюється керівником. Отримані результати розробки вебплатформи для підтримки освітнього процесу представляються у формі працюючого програмного продукту та пояснювальної записки.

Процедура контролю та приймання дипломної кваліфікаційної роботи завершується представленням отриманих результатів і захистом роботи перед членами екзаменаційної комісії

ДОДАТОК Б
Текст програми

Б.1 Клієнт

routes/_root.tsx

```
import { Outlet, createRootRouteWithContext } from '@tanstack/react-router'
import { TanStackRouterDevtoolsPanel } from '@tanstack/react-router-devtools'
import { TanStackDevtools } from '@tanstack/react-devtools'

import TanStackQueryDevtools from
'../integrations/tanstack-query/devtools'

import type { QueryClient } from '@tanstack/react-query'
import { Toaster } from "@components/ui/sonner"
import { Sidebar } from '@components/Sidebar'
import { NotFoundPage } from '@components/errors/NotFoundPage'

interface MyRouterContext {
  queryClient: QueryClient
}

export const Route = createRootRouteWithContext<MyRouterContext>() ({
  component: () => (
    <div className="min-h-screen bg-stone-50">
      <Sidebar />
      <main className="lg:pl-64 min-h-screen flex flex-col">
        <Outlet />
      </main>
      <TanStackDevtools
        config={{
          position: 'bottom-right',
        }}
        plugins={[
          {
            name: 'Tanstack Router',
            render: <TanStackRouterDevtoolsPanel />,
          },
          TanStackQueryDevtools,
        ]}
      />
      <Toaster />
    </div>
  ),
  notFoundComponent: NotFoundPage,
})
```

routes/index.tsx

```
/* eslint-disable @typescript-eslint/no-unnecessary-condition */
import { Link, createFileRoute } from '@tanstack/react-router'
import { useInfiniteQuery, useMutation, useQueryClient } from
'@tanstack/react-query'
import { useEffect, useRef } from 'react'
import { ArrowRight, Loader2, Megaphone, Plus, Trash2 } from 'lucide-react'

import MDEditor from '@uiw/react-md-editor'
import { toast } from 'sonner'
import { ServerClock } from '@components/ServerClock'
import { getGlobalAnnouncements } from '@api/general'
import { UserRole, useAuthUser } from '@api/queries'
import { Card, CardContent, CardHeader, CardTitle } from
'@components/ui/card'
```

```

import { Skeleton } from '@components/ui/skeleton'
import { Avatar, AvatarFallback, AvatarImage } from
'@components/ui/avatar'
import { Button } from '@components/ui/button'
import { deleteAnnouncement } from '@api/announcements'

export const Route = createFileRoute('/')({
  component: DashboardPage,
})

function DashboardPage() {
  const { user, isAuthenticated } = useAuthUser()
  const observerTarget = useRef<HTMLDivElement>(null)
  const queryClient = useQueryClient()

  const {
    data,
    fetchNextPage,
    hasNextPage,
    isFetchingNextPage,
    status,
  } = useInfiniteQuery({
    queryKey: ['global-announcements'],
    queryFn: ({ pageParam = 1 }) => getGlobalAnnouncements({ page:
pageParam, limit: 5 }),
    initialPageParam: 1,
    getNextPageParam: (lastPage, allPages) => {
      return lastPage.length === 0 ? undefined : allPages.length + 1
    },
  })

  const deleteAnnouncementMutation = useMutation({
    mutationFn: deleteAnnouncement,
    onSuccess: () => {
      toast.success("Оголошення видалено")
      queryClient.invalidateQueries({ queryKey: ['global-
announcements'] })
    },
    onError: () => toast.error("Помилка видалення")
  })

  useEffect(() => {
    const observer = new IntersectionObserver(
      (entries) => {
        if (entries[0].isIntersecting && hasNextPage && !
isFetchingNextPage) {
          fetchNextPage()
        }
      },
      { threshold: 1.0 }
    )

    if (observerTarget.current) {
      observer.observe(observerTarget.current)
    }

    return () => observer.disconnect()
  }, [hasNextPage, isFetchingNextPage, fetchNextPage])

  const announcements = data?.pages.flatMap((page) => page) || []

  return (
    <div>
      <header className="sticky top-0 z-40 bg-white/80 backdrop-blur-

```

```

md border-b px-6 py-4 flex justify-between items-center">
  <div>
    <h1 className="text-2xl font-bold text-slate-800">
      {isAuthenticated ? `Вітаємо, ${user?.fullName.split(' ')}
[0]}! 🙌` : 'Ласкаво просимо!'}
    </h1>
    <p className="text-sm text-muted-foreground hidden
sm:block">
      {isAuthenticated ? 'Ось огляд новин та подій.' :
'Увійдіть, щоб розпочати навчання.'}
    </p>
  </div>
  <ServerClock />
</header>

<div className="flex-1 p-6 md:p-8 max-w-5xl mx-auto w-full
space-y-8">
  <section className="space-y-4">
    <div className="flex items-center justify-between mb-4
sticky top-20 z-30 bg-gray-50/90 p-2 rounded-lg backdrop-blur-sm">
      <div className="flex items-center gap-2 text-slate-800">
        <div className="p-2 bg-orange-100 text-orange-600
rounded-lg">
          <Megaphone className="h-6 w-6" />
        </div>
        <h2 className="text-xl font-bold">Стрічка новин</h2>
      </div>
      {user?.role === UserRole.Admin && (
        <Button size="sm" variant="outline" asChild>
          <Link to="/announcements/create">
            <Plus className="h-4 w-4 mr-1" /> Додати
          </Link>
        </Button>
      )}
    </div>

    {status === 'pending' ? (
      <div className="flex flex-col gap-6 max-w-3xl mx-auto">
        <Skeleton className="h-64 rounded-xl w-full" />
        <Skeleton className="h-64 rounded-xl w-full" />
      </div>
    ) : status === 'error' ? (
      <div className="text-center text-red-500 py-10">Не вдалося
завантажити оголошення.</div>
    ) : announcements.length === 0 ? (
      <Card className="bg-white border-dashed shadow-sm max-
w-3xl mx-auto">
        <CardContent className="flex flex-col items-center
justify-center py-10 text-muted-foreground">
          <Megaphone className="h-10 w-10 mb-2 opacity-20" />
          <p>Наразі немає важливих оголошень.</p>
        </CardContent>
      </Card>
    ) : (
      <div className="flex flex-col gap-8 max-w-3xl mx-auto">
        {announcements.map((item) => (
          <Card key={item.id} className="hover:shadow-md
transition-shadow border-l-4 border-l-orange-400 overflow-hidden
relative group">
            {user?.role === UserRole.Admin && (
              <Button
                variant="ghost"
                size="icon"
                className="absolute top-2 right-2 opacity-0

```

```

group-hover:opacity-100 transition-opacity text-red-500 hover:text-
red-700 hover:bg-red-100 z-10"
      onClick={() => confirm("Видалити?")} &&
deleteAnnouncementMutation.mutate(item.id) }
    >
    <Trash2 className="h-4 w-4" />
  </Button>
)}
<CardHeader className="pb-3 bg-slate-50/50">
  <div className="flex justify-between items-start
gap-2">
    <Link to="/announcements/${announcementId}"
params={{ announcementId: item.id }} className="block">
      <CardTitle className="text-xl md:text-2xl
font-bold text-slate-800 group-hover:text-orange-600 transition-colors
break-all">
        {item.title}
      </CardTitle>
    </Link>
  </div>
  <div className="flex items-center gap-3 text-xs
text-muted-foreground mt-2">
    <span className="bg-slate-100 px-2 py-1 rounded-
md border">
      {new
Date(item.createdAt).toLocaleDateString('uk-UA', {
        day: 'numeric', month: 'long', hour: '2-
digit', minute: '2-digit'
      })}
    </span>
    <div className="flex items-center gap-2">
      <Avatar className="h-5 w-5">
        <AvatarImage src={item.authorAvatar} />
        <AvatarFallback className="text-
[9px]">{item.authorName?.[0]}</AvatarFallback>
      </Avatar>
      <span>{item.authorName}</span>
    </div>
  </div>
</CardHeader>
<CardContent className="pt-4">
  <div className="prose prose-slate max-w-none text-
slate-700 leading-relaxed line-clamp-[10] relative">
    <div className="absolute bottom-0 left-0 w-full
h-8 bg-gradient-to-t from-white to-transparent md:hidden"></div>
    <div data-color-mode="light">
      <MDEditor.Markdown
        source={item.text}
        style={{ backgroundColor: 'transparent',
color: 'inherit', fontSize: '1rem' }}
      />
    </div>
  </div>
  <Link to="/announcements/${announcementId}"
params={{ announcementId: item.id }} className="block">
    <div className="mt-4 text-orange-600 text-sm
font-medium flex items-center gap-1 opacity-0 group-hover:opacity-100
transition-opacity">
      Читати повністю <ArrowRight className="h-4
w-4" />
    </div>
  </Link>

```

```

        </CardContent>
      </Card>
    )))

    <div ref={observerTarget} className="flex justify-center
py-6 h-20">
      {isFetchingNextPage && <Loader2 className="h-8 w-8
animate-spin text-orange-500" />}
      {!hasNextPage && announcements.length > 0 && (
        <span className="text-muted-foreground text-sm">Ви
переглянули всі новини 🌈</span>
      )}
    </div>
  </div>
) }
</section>

{isAuthenticated && (
  <section className="pt-8 border-t">
    <h2 className="text-xl font-bold mb-4 text-
slate-800">Швидкий доступ</h2>
    <div className="grid gap-4 sm:grid-cols-2 lg:grid-cols-4">
      <Card className="hover:bg-slate-50 transition-colors
cursor-pointer group">
        <Link to="/courses">
          <CardHeader>
            <CardTitle className="text-lg group-hover:text-
cyan-600 transition-colors flex items-center gap-2">
              Мої курси <ArrowRight className="h-4 w-4
opacity-0 group-hover:opacity-100 transition-opacity" />
            </CardTitle>
            <p className="text-sm text-muted-foreground font-
normal">
              Продовжити навчання
            </p>
          </CardHeader>
        </Link>
      </Card>
    </div>
  </section>
) }
</div>
</div>
)
}

```

routes/admin/announcements.tsx

```

import { Link, createFileRoute, useNavigate } from '@tanstack/react-
router'
import { useMutation, useQuery, useQueryClient } from
'@tanstack/react-query'
import { useEffect, useState } from 'react'
import { toast } from 'sonner'
import { ArrowLeft, ArrowRight, Edit, ExternalLink, Loader2,
Megaphone, Plus, Trash2 } from 'lucide-react'
import { format } from 'date-fns'
import { uk } from 'date-fns/locale'

import { getGlobalAnnouncements } from '@api/general'
import { deleteAnnouncement } from '@api/announcements'
import { Button } from '@components/ui/button'
import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow
} from '@components/ui/table'

```

```

import { Avatar, AvatarFallback, AvatarImage } from
'@/components/ui/avatar'
import { Badge } from '@components/ui/badge'
import { Skeleton } from '@components/ui/skeleton'
import { UserRole, useAuthUser } from '@api/queries'

export const Route = createFileRoute('/admin/announcements') ({
  component: AdminAnnouncementsPage,
})

function AdminAnnouncementsPage() {
  const queryClient = useQueryClient()
  const [page, setPage] = useState(1)
  const pageSize = 10

  const { user, isAuthenticated, isLoadingAuth } = useAuthUser()
  const navigate = useNavigate()

  useEffect(() => {
    if (!isLoadingAuth && (!isAuthenticated || user?.role !==
UserRole.Admin)) {
      toast.error("Доступ заборонено")
      navigate({ to: '/' })
    }
  }, [isLoadingAuth, isAuthenticated, user, navigate])

  const { data: announcements, isLoading, isPlaceholderData } =
useQuery({
    queryKey: ['global-announcements', page],
    queryFn: () => getGlobalAnnouncements({ page, limit: pageSize }),
    placeholderData: (previousData) => previousData,
  })

  const deleteMutation = useMutation({
    mutationFn: deleteAnnouncement,
    onSuccess: () => {
      toast.success("Оголошення видалено")
      queryClient.invalidateQueries({ queryKey: ['global-
announcements'] })
    },
    onError: () => toast.error("Не вдалося видалити оголошення")
  })

  const handleDelete = (id: string) => {
    if (confirm("Ви дійсно бажаєте видалити це глобальне оголошення?
Це незворотна дія.")) {
      deleteMutation.mutate(id)
    }
  }

  return (
    <div className="container mx-auto py-8 px-4">
      <div className="flex flex-col sm:flex-row items-center justify-
between mb-8 gap-4">
        <div>
          <h1 className="text-3xl font-bold flex items-center gap-2">
            <Megaphone className="h-8 w-8 text-orange-500" />
            Глобальні оголошення
          </h1>
          <p className="text-muted-foreground mt-1">
            Управління новинами, які відображаються на головній
сторінці.
          </p>
        </div>
      </div>
    </div>
  )

```

```

        <Button asChild size="lg" className="bg-gray-500 hover:bg-
gray-700">
        <Link to="/announcements/create">
        <Plus className="mr-2 h-5 w-5" /> Створити оголошення
        </Link>
        </Button>
    </div>

    <div className="border rounded-lg bg-white shadow-sm overflow-
hidden">
        <Table>
            <TableHeader>
                <TableRow className="bg-slate-50">
                    <TableHead className="w-[400px]">Заголовок /
Текст</TableHead>
                    <TableHead>Автор</TableHead>
                    <TableHead>Дата публікації</TableHead>
                    <TableHead className="text-right">Дії</TableHead>
                </TableRow>
            </TableHeader>
            <TableBody>
                {isLoading ? (
                    Array.from({ length: 5 }).map((_, i) => (
                        <TableRow key={i}>
                            <TableCell><Skeleton className="h-12 w-3/4"
/></TableCell>
                            <TableCell><Skeleton className="h-8 w-24"
/></TableCell>
                            <TableCell><Skeleton className="h-6 w-32"
/></TableCell>
                            <TableCell><Skeleton className="h-8 w-8 ml-auto"
/></TableCell>
                        </TableRow>
                    ))
                ) : announcements?.length === 0 ? (
                    <TableRow>
                        <TableCell colspan={4} className="text-center py-12
text-muted-foreground">
                            <div className="flex flex-col items-center gap-2">
                                <Megaphone className="h-10 w-10 opacity-20" />
                                <p>Оголошень поки немає.</p>
                            </div>
                        </TableCell>
                    </TableRow>
                ) : (
                    announcements?.map((item) => (
                        <TableRow key={item.id} className="hover:bg-
slate-50/50 transition-colors">
                            <TableCell className="align-top py-4 max-w-175">
                                <div className="flex flex-col gap-1">
                                    <Link
                                        to="/announcements/$announcementId"
                                        params={{ announcementId: item.id }}
                                        className="font-bold text-lg hover:text-
orange-600 transition-colors line-clamp-1 flex items-center gap-2
truncate"
                                    >
                                        {item.title}
                                        <ExternalLink className="h-3 w-3
opacity-50" />
                                    </Link>
                                <div className="text-sm text-muted-foreground
line-clamp-2 text-ellipsis break-words max-w-md">
                                    {item.text.replace(/[#*_`]/g, ' ')}
                                </div>
                            </TableCell>
                        </TableRow>
                    ))
                )
            </TableBody>
        </Table>
    </div>

```

```

        </div>
      </div>
    </TableCell>

    <TableCell className="align-top py-4">
      <div className="flex items-center gap-2">
        <Avatar className="h-8 w-8">
          <AvatarImage src={item.authorAvatar} />
        <AvatarFallback>{item.authorName[0]}</AvatarFallback>
      </div>
      <div className="flex flex-col">
        <span className="text-sm font-medium">{item.authorName}</span>
        <Badge variant="outline" className="w-fit text-[10px] h-5 px-1 font-normal text-muted-foreground border-slate-200">
          Admin
        </Badge>
      </div>
    </div>
  </TableCell>

  <TableCell className="align-top py-4 text-sm text-slate-600 whitespace-nowrap">
    {format(new Date(item.createdAt), "d MMM yyyy, HH:mm", { locale: uk })}
  </TableCell>

  <TableCell className="align-top py-4 text-right">
    <div className="flex justify-end gap-1">
      <Button variant="ghost" size="icon" asChild
        title="Редагувати">
        <Link to="/announcements/$announcementId/edit"
          params={{ announcementId: item.id }}>
          <Edit className="h-4 w-4 text-blue-600" />
        </Link>
      </Button>
      <Button
        variant="ghost"
        size="icon"
        onClick={() => handleDelete(item.id)}
        disabled={deleteMutation.isPending}
        title="Видалити"
        className="hover:bg-red-50"
      >
        {deleteMutation.isPending ? (
          <Loader2 className="h-4 w-4 animate-spin" />
        ) : (
          <Trash2 className="h-4 w-4 text-red-500" />
        )}
      </Button>
    </div>
  </TableCell>
</TableRow>
))
})
</TableBody>
</Table>

<div className="flex items-center justify-between px-4 py-4 border-t bg-slate-50">
  <div className="text-sm text-muted-foreground">
    Сторінка {page}
  </div>
</div>

```

```

</div>
<div className="flex gap-2">
  <Button
    variant="outline"
    size="sm"
    onClick={() => setPage((old) => Math.max(old - 1, 1))}
    disabled={page === 1 || isLoading}
  >
    <ArrowLeft className="h-4 w-4 mr-1" /> Попередня
  </Button>
  <Button
    variant="outline"
    size="sm"
    onClick={() => {
      if (!isPlaceholderData && announcements &&
announcements.length === pageSize) {
        setPage((old) => old + 1)
      }
    }}
    disabled={isPlaceholderData || (announcements &&
announcements.length < pageSize)}
  >
    Наступна <ArrowRight className="h-4 w-4 ml-1" />
  </Button>
</div>
</div>
</div>
</div>
)
}

```

/routes/admin/courses.tsx

```

/* eslint-disable @typescript-eslint/no-unnecessary-condition */
import { Link, createFileRoute, useNavigate } from '@tanstack/react-router'
import { useMutation, useQuery, useQueryClient } from '@tanstack/react-query'
import { useEffect, useState } from 'react'
import { toast } from 'sonner'
import { BookOpen, Check, Edit, Loader2, Search, Trash2, User } from 'lucide-react'

import type { Course } from '@api/courses'
import { deleteCourse, getCourses, updateCourse } from '@api/courses'
import { getAdminUsers } from '@api/admin'

import { Button } from '@components/ui/button'
import { Input } from '@components/ui/input'
import { Dialog, DialogContent, DialogDescription, DialogFooter,
DialogHeader, DialogTitle } from '@components/ui/dialog'
import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow }
from '@components/ui/table'
import { Label } from '@components/ui/label'
import { Badge } from '@components/ui/badge'
import { Switch } from '@components/ui/switch'
import { Select, SelectContent, SelectGroup, SelectItem,
SelectTrigger, SelectValue } from '@components/ui/select'
import { Textarea } from '@components/ui/textarea'
import { UserRole, useAuthUser } from '@api/queries'

export const Route = createFileRoute('/admin/courses')({
  component: AdminCoursesPage,
})

```

```

function AdminCoursesPage() {
  const queryClient = useQueryClient()

  const { user, isAuthenticated, isLoadingAuth } = useAuthUser()
  const navigate = useNavigate()

  useEffect(() => {
    if (!isLoadingAuth && (!isAuthenticated || user?.role !==
UserRole.Admin)) {
      toast.error("Доступ заборонено")
      navigate({ to: '/' })
    }
  }, [isLoadingAuth, isAuthenticated, user, navigate])

  const [search, setSearch] = useState('')
  const [editingCourse, setEditingCourse] = useState<Course |
null>(null)

  const [selectedOwnerId, setSelectedOwnerId] = useState<string>("")
  const [selectedCoTeachers, setSelectedCoTeachers] =
useState<Array<string>>([])

  const { data: courses, isLoading: coursesLoading } = useQuery({
    queryKey: ['courses'],
    queryFn: getCourses
  })

  const { data: users, isLoading: usersLoading } = useQuery({
    queryKey: ['admin', 'users'],
    queryFn: getAdminUsers
  })

  const potentialTeachers = users?.filter(u => u.role ===
UserRole.Admin || u.role === UserRole.Teacher) || []

  const updateMutation = useMutation({
    mutationFn: (vars: { id: string; data: any }) =>
updateCourse(vars),
    onSuccess: () => {
      toast.success("Курс оновлено")
      queryClient.invalidateQueries({ queryKey: ['courses'] })
      setEditingCourse(null)
    },
    onError: () => toast.error("Не вдалося оновити курс")
  })

  const deleteMutation = useMutation({
    mutationFn: (id: string) => deleteCourse(id),
    onSuccess: () => {
      toast.success("Курс видалено")
      queryClient.invalidateQueries({ queryKey: ['courses'] })
    },
    onError: () => toast.error("Не вдалося видалити курс")
  })

  useEffect(() => {
    if (editingCourse) {
      setSelectedOwnerId(editingCourse.teacherId)
      setSelectedCoTeachers(editingCourse.coTeacherIds || [])
    } else {
      setSelectedOwnerId("")
      setSelectedCoTeachers([])
    }
  })
}

```

```

    }, [editingCourse])

    const handleDelete = (course: Course) => {
      if (confirm(`Ви впевнені, що хочете видалити курс "$
{course.title}"? Цю дію неможливо скасувати.`)) {
        deleteMutation.mutate(course.id)
      }
    }

    const handleOwnerChange = (newOwnerId: string) => {
      setSelectedOwnerId(newOwnerId);
      if (selectedCoTeachers.includes(newOwnerId)) {
        setSelectedCoTeachers(prev => prev.filter(id => id !==
newOwnerId));
      }
    }

    const toggleCoTeacher = (userId: string) => {
      if (userId === selectedOwnerId) return;

      setSelectedCoTeachers(prev =>
        prev.includes(userId)
          ? prev.filter(id => id !== userId)
          : [...prev, userId]
      )
    }

    const filteredCourses = courses?.filter(c =>
      c.title.toLowerCase().includes(search.toLowerCase())
    )

    const getTeacherName = (teacherId: string) => {
      const user = users?.find(u => u.id === teacherId)
      return user ? user.fullName : <span className="text-
red-400">Невідомий (ID: {teacherId})</span>
    }

    if (coursesLoading || usersLoading) return <div className="flex
justify-center p-10"><Loader2 className="animate-spin" /></div>

    return (
      <div className="container mx-auto py-8 px-4">
        <div className="flex items-center justify-between mb-6">
          <h1 className="text-3xl font-bold">Управління
курсами</h1>
          <div className="text-sm text-muted-foreground bg-
slate-100 px-3 py-1 rounded">
            Всього курсів: {courses?.length}
          </div>
        </div>

        <div className="flex items-center gap-4 mb-4">
          <div className="relative flex-1 max-w-sm">
            <Search className="absolute left-2.5 top-2.5 h-4
w-4 text-muted-foreground" />
            <Input
              placeholder="Пошук курсу..."
              className="pl-8"
              value={search}
              onChange={e => setSearch(e.target.value)}
            />
          </div>
        </div>
      </div>
    )
  }
}

```

```

        <div className="border rounded-lg bg-white shadow-sm
overflow-hidden">
        <Table>
            <TableHeader>
                <TableRow className="bg-slate-50">
                    <TableHead>Назва курсу</TableHead>
                    <TableHead>Вчитель (Власник)</TableHead>
                    <TableHead>Статус</TableHead>
                    <TableHead>Створено</TableHead>
                    <TableHead
className="text-right">Дії</TableHead>
                </TableRow>
            </TableHeader>
            <TableBody>
                {filteredCourses?.length === 0 && (
                    <TableRow>
                        <TableCell colspan={5}
className="text-center py-8 text-muted-foreground">
                            Курсів не знайдено
                        </TableCell>
                    </TableRow>
                )}
                {filteredCourses?.map(course => (
                    <TableRow key={course.id}>
                        <TableCell className="font-medium">
                            <Button variant="ghost" asChild
className="p-0 h-auto font-medium hover:bg-transparent hover:underline
justify-start">
                                <Link
to="/courses/${courseId}/content" params={{ courseId: course.id }}>
                                    <div className="flex flex-
col items-start gap-1">
                                        <div className="flex
items-center gap-2">
                                            <BookOpen
className="h-4 w-4 text-blue-500" />
                                            {course.title}
                                        </div>
                                        <span className="text-
xs text-muted-foreground font-normal truncate max-w-[200px]">
                                            {course.description}
                                        </span>
                                    </div>
                                </Link>
                            </Button>
                        </TableCell>
                        <TableCell>
                            <div className="flex items-center
gap-2">
                                <User className="h-4 w-4 text-
slate-400" />
                                {getTeacherName(course.teacherId)}
                            </div>
                        </TableCell>
                        <TableCell>
                            <Badge variant={course.isOpen ?
"default" : "secondary"}>
                                {course.isOpen ? "Відкритий" :
"Закритий"}
                            </Badge>
                        </TableCell>
                        <TableCell className="text-sm text-

```

```
muted-foreground">
                                {new
Date(course.createdAt).toLocaleDateString() }
                                </TableCell>
                                <TableCell className="text-right">
                                    <div className="flex justify-end
gap-1">
                                        <Button variant="ghost"
size="icon" onClick={() => setEditingCourse(course)}>
                                            <Edit className="h-4 w-4
text-blue-600" />
                                                </Button>
                                            <Button variant="ghost"
size="icon" onClick={() => handleDelete(course)}>
                                                <Trash2 className="h-4 w-4
text-red-500" />
                                                    </Button>
                                                        </div>
                                                            </TableCell>
                                                                </TableRow>
                                                                    )})}
</TableBody>
</Table>
</div>


    <Dialog open={!editingCourse} onOpenChange={(open) => !
open && setEditingCourse(null)}>
        <DialogContent className="sm:max-w-[600px] max-h-
[90vh] overflow-y-auto">
            <DialogHeader>
                <DialogTitle>Редагувати курс</DialogTitle>
                <DialogDescription>
                    Змініть інформацію про курс, власника або
налаштування доступу.
                </DialogDescription>
            </DialogHeader>
            {editingCourse && (
                <form onSubmit={e => {
                    e.preventDefault()
                    const formData = new
FormData(e.currentTarget)

                    updateMutation.mutate({
                        id: editingCourse.id,
                        data: {
                            title: formData.get('title') as
string,
                            description:
formData.get('description') as string,
                            isOpen: formData.get('isOpen') ===
'on',
                            teacherId: selectedOwnerId,
                            coTeacherIds: selectedCoTeachers
                        }
                    })
                }} className="space-y-4 py-2">

                    <div className="grid gap-2">
                        <Label htmlFor="title">Назва</Label>
                        <Input id="title" name="title"
defaultValue={editingCourse.title} required />
                    </div>

                    <div className="grid gap-2">
```

```

<Label
htmlFor="description">Опис</Label>
  <Textarea id="description"
name="description" defaultValue={editingCourse.description || ""} />
</div>

  <div className="grid gap-2 p-4 border
rounded-md bg-amber-50/50 border-amber-200">
    <Label htmlFor="teacherId"
className="font-bold text-amber-900">Власник курсу (Головний
викладач)</Label>
    <Select
      name="teacherId"
      value={selectedOwnerId}
      onChange={handleOwnerChange}
    >
      <SelectTrigger className="bg-
white">
        <SelectValue
placeholder="Оберіть власника" />
      </SelectTrigger>
      <SelectContent>
        <SelectGroup>
          {potentialTeachers.map(u
=> (
            <SelectItem key={u.id}
value={u.id}>
              {u.fullName}
            </SelectItem>
          ))}
        </SelectGroup>
      </SelectContent>
    </Select>
    <p className="text-xs text-muted-
foreground">
      Увага: Зміна власника передасть
повні права на курс іншому користувачу.
    </p>
  </div>

  <div className="grid gap-2 p-4 border
rounded-md">
    <Label className="font-bold
mb-2">Співвикладачі</Label>
    <div className="grid grid-cols-1
sm:grid-cols-2 gap-2 max-h-[200px] overflow-y-auto pr-2">
      {potentialTeachers.map(u => {
        const isOwner = u.id ===
selectedOwnerId;
        const isSelected =
selectedCoTeachers.includes(u.id);

        return (
          <div
            key={u.id}
            className={`
              flex items-center
              justify-between p-2 rounded border text-sm cursor-pointer transition-
colors
              ${isSelected ?
'bg-blue-50 border-blue-200' : 'bg-white hover:bg-slate-50'}
              ${isOwner ?
'opacity-50 cursor-not-allowed bg-slate-100' : ''}

```

```

        `}
        onClick={() => !
isOwner && toggleCoTeacher(u.id)}
        >
        <div className="flex
flex-col overflow-hidden">
            <span
className="font-medium truncate">{u.fullName}</span>
            <span
className="text-xs text-muted-foreground truncate">{u.email}</span>
        </div>
        {isOwner ? (
            <Badge
variant="outline" className="text-[10px] ml-2">Власник</Badge>
            ) : isSelected && (
                <Check
className="h-4 w-4 text-blue-600 ml-2" />
            )}
        </div>
    )}
</div>
<p className="text-xs text-muted-
foreground mt-1">
        Позначені користувачі зможуть
        редагувати матеріали та оцінювати студентів.
    </p>
</div>

    <div className="flex items-center space-
x-2 pt-2">
        <Switch id="isOpen" name="isOpen"
defaultChecked={editingCourse.isOpen} />
        <Label htmlFor="isOpen">Курс відкритий
для самостійного запису</Label>
    </div>

    <DialogFooter className="mt-4 gap-2">
        <Button type="button"
variant="outline" onClick={() => setEditingCourse(null)}>
            Скасувати
        </Button>
        <Button type="submit"
disabled={updateMutation.isPending}>
            {updateMutation.isPending &&
<Loader2 className="mr-2 h-4 w-4 animate-spin" />}
            Зберегти зміни
        </Button>
    </DialogFooter>
</form>
    )}
</DialogContent>
</Dialog>
</div>
)
}

```

/routes/admin/users.tsx

```

import { createFileRoute } from '@tanstack/react-router'
import { useMutation, useQuery, useQueryClient } from
'@tanstack/react-query'
import { useState } from 'react'
import { toast } from 'sonner'

```

```

import { Edit, Loader2, Plus, Search, Trash2 } from 'lucide-react'

import type { AdminUser } from '@api/admin';
import {
  addUserToGroup,
  createGroup, deleteAdminUser, deleteGroup, getAdminGroups,
  getAdminUsers,
  removeUserFromGroup,
  updateAdminUser
} from '@api/admin'

import { Button } from '@components/ui/button'
import { Input } from '@components/ui/input'
import { Tabs, TabsContent, TabsList, TabsTrigger } from
'@components/ui/tabs'
import { Dialog, DialogContent, DialogFooter, DialogHeader,
DialogTitle } from '@components/ui/dialog'
import { Table, TableBody, TableCell, TableHead, TableHeader, TableRow
} from '@components/ui/table'
import { Label } from '@components/ui/label'
import { Badge } from '@components/ui/badge'
import { Card, CardContent, CardDescription, CardHeader, CardTitle }
from '@components/ui/card'
import { Select, SelectContent, SelectGroup, SelectItem, SelectLabel,
SelectTrigger, SelectValue } from '@components/ui/select'
import { UserRole } from '@api/queries'

export const Route = createFileRoute('/admin/users')({
  component: AdminDashboard,
})

function AdminDashboard() {
  return (
    <div className="container mx-auto py-8 px-4">
      <h1 className="text-3xl font-bold mb-6">Адмін-панель</h1>
      <Tabs defaultValue="users" className="w-full">
        <TabsList className="grid w-full max-w-md grid-cols-2
mb-8">
          <TabsTrigger
value="users">Користувачі</TabsTrigger>
          <TabsTrigger value="groups">Групи
студентів</TabsTrigger>
        </TabsList>

        <TabsContent value="users">
          <UsersManager />
        </TabsContent>

        <TabsContent value="groups">
          <GroupsManager />
        </TabsContent>
      </Tabs>
    </div>
  )
}

function UsersManager() {
  const queryClient = useQueryClient()
  const [search, setSearch] = useState('')
  const [editingUser, setEditingUser] = useState<AdminUser |
null>(null)

  const { data: users, isLoading } = useQuery({
    queryKey: ['admin', 'users'],

```

```

        queryFn: getAdminUsers
      })

      const updateMutation = useMutation({
        mutationFn: (vars: { id: string; fullName: string; email:
string; userName: string; role: string }) =>
          updateAdminUser(vars.id, {
            fullName: vars.fullName,
            email: vars.email,
            userName: vars.userName,
            role: vars.role
          }),
        onSuccess: () => {
          toast.success("Користувача оновлено")
          queryClient.invalidateQueries({ queryKey: ['admin',
'users'] })
          setEditingUser(null)
        },
        onError: (err: any) => {
          toast.error(err.message || "Помилка оновлення")
        }
      })

      const deleteMutation = useMutation({
        mutationFn: (userId: string) => deleteAdminUser(userId),
        onSuccess: () => {
          toast.success("Користувача видалено")
          queryClient.invalidateQueries({ queryKey: ['admin',
'users'] })
        },
        onError: (err: any) => {
          const msg = err.message || "Не вдалося видалити
користувача"
          toast.error(msg)
        }
      })

      const handleDelete = (user: AdminUser) => {
        if (confirm(`Ви впевнені, що хочете видалити користувача $
{user.fullName}? Це незворотня дія.`)) {
          deleteMutation.mutate(user.id)
        }
      }

      const filteredUsers = users?.filter(u =>
        u.fullName.toLowerCase().includes(search.toLowerCase()) ||
        u.email.toLowerCase().includes(search.toLowerCase())
      )

      if (isLoading) return <Loader2 className="animate-spin" />

      return (
        <div className="space-y-4">
          <div className="flex items-center gap-4">
            <div className="relative flex-1 max-w-sm">
              <Search className="absolute left-2.5 top-2.5 h-4
w-4 text-muted-foreground" />
              <Input
                placeholder="Пошук за ім'ям або поштою..."
                className="pl-8"
                value={search}
                onChange={e => setSearch(e.target.value)}
              />
            </div>
          </div>
        </div>
      )
    )
  }
}

```

```

</div>

<div className="border rounded-lg bg-white shadow-sm
overflow-hidden">
  <Table>
    <TableHeader>
      <TableRow className="bg-slate-50">
        <TableHead>Имя</TableHead>
        <TableHead>User Name</TableHead>
        <TableHead>Email</TableHead>
        <TableHead>Роль</TableHead>
        <TableHead>Группы</TableHead>
        <TableHead
className="text-right">Дії</TableHead>
      </TableRow>
    </TableHeader>
    <TableBody>
      {filteredUsers?.map(user => (
        <TableRow key={user.id}>
          <TableCell className="font-
medium">{user.fullName}</TableCell>
          <TableCell className="text-muted-
foreground">{user.userName}</TableCell>
          <TableCell>{user.email}</TableCell>
          <TableCell>
            <Badge variant={user.role ==
UserRole.Admin ? 'destructive' : user.role == UserRole.Teacher ?
'default' : 'secondary'}>
              {user.role == UserRole.Admin ?
"Admin" : user.role == UserRole.Teacher ? "Вчитель" : "Студент"}
            </Badge>
          </TableCell>
          <TableCell className="text-sm text-
muted-foreground">
            {user.groups.length > 0 ?
user.groups.join(", ") : "-"}
          </TableCell>
          <TableCell className="text-right">
            <div className="flex justify-end
gap-1">
              <Button variant="ghost"
size="icon" onClick={() => setEditingUser(user)}>
                <Edit className="h-4
w-4" />
              </Button>
              <Button
                variant="ghost"
                size="icon"
                onClick={() =>
handleDelete(user)}
                className="text-red-500
hover:text-red-700 hover:bg-red-50"
              >
                <Trash2 className="h-4
w-4" />
              </Button>
            </div>
          </TableCell>
        </TableRow>
      )})
    </TableBody>
  </Table>
</div>

```

```

        <Dialog open={!editingUser} onOpenChange={ (open) => !open
        && setEditingUser(null) }>
            <DialogContent>
                <DialogHeader>
                    <DialogTitle>Редагувати
користувача</DialogTitle>
                </DialogHeader>
                {editingUser && (
                    <form onSubmit={ (e) => {
                        e.preventDefault()
                        const formData = new
FormData(e.currentTarget)
                        updateMutation.mutate({
                            id: editingUser.id,
                            fullName: formData.get('fullName') as
string,
                            email: formData.get('email') as
string,
                            userName: formData.get('userName') as
string,
                            role: formData.get('role') as string
                        })
                    }} className="space-y-4">
                        <div className="grid gap-2">
                            <Label htmlFor="fullName">Повне
ім'я</Label>
                            <Input id="fullName" name="fullName"
defaultValue={editingUser.fullName} required />
                        </div>
                        <div className="grid gap-2">
                            <Label htmlFor="userName">Логін
(UserName)</Label>
                            <Input id="userName" name="userName"
defaultValue={editingUser.userName} required />
                        </div>
                        <div className="grid gap-2">
                            <Label htmlFor="email">Email</Label>
                            <Input id="email" name="email"
type="email" defaultValue={editingUser.email} required />
                        </div>
                        <div className="grid gap-2">
                            <Label htmlFor="role">Роль</Label>
                            <Select key={editingUser.id}
name="role"
defaultValue={String(editingUser.role)}
>
                                <SelectTrigger>
                                    <SelectValue placeholder="Роль
користувача" />
                                </SelectTrigger>
                                <SelectContent>
                                    <SelectGroup>
                                        <SelectLabel>Роль</SelectLabel>
                                        <SelectItem
value="2">Студент</SelectItem>
                                        <SelectItem
value="1">Вчитель</SelectItem>
                                        <SelectItem
value="0">Адміністратор</SelectItem>
                                    </SelectGroup>
                                </SelectContent>
                            </Select>

```

```

        </div>
        <DialogFooter>
            <Button type="submit"
disabled={updateMutation.isPending}>
                {updateMutation.isPending ?
<Loader2 className="mr-2 h-4 w-4 animate-spin" /> : null}
                Зберегти
            </Button>
        </DialogFooter>
    </form>
    )}
    </DialogContent>
</Dialog>
</div >
)
}

function GroupsManager() {
    const queryClient = useQueryClient()
    const [newGroupName, setNewGroupName] = useState('')

    const { data: groups, isLoading: isGroupsLoading } = useQuery({
        queryKey: ['admin', 'groups'],
        queryFn: getAdminGroups
    })

    const { data: users, isLoading: isUsersLoading } = useQuery({
        queryKey: ['admin', 'users'],
        queryFn: getAdminUsers
    })

    const allStudents = users?.filter(u => u.role == UserRole.Student)
    || []

    const createGroupMutation = useMutation({
        mutationFn: createGroup,
        onSuccess: () => {
            toast.success("Групу створено")
            setNewGroupName('')
            queryClient.invalidateQueries({ queryKey: ['admin',
'groups'] })
        }
    })

    const deleteGroupMutation = useMutation({
        mutationFn: (groupId: string) => deleteGroup(groupId),
        onSuccess: () => {
            toast.success("Групу видалено")
            queryClient.invalidateQueries({ queryKey: ['admin',
'groups'] })
        },
        onError: () => toast.error("Не вдалося видалити групу")
    })

    const addStudentMutation = useMutation({
        mutationFn: (vars: { groupId: string, userId: string }) =>
addUserToGroup(vars.groupId, vars.userId),
        onSuccess: () => {
            toast.success("Студента додано")
            queryClient.invalidateQueries({ queryKey: ['admin',
'groups'] })
            queryClient.invalidateQueries({ queryKey: ['admin',
'users'] })
        },
    })

```

```

      onError: () => toast.error("Не вдалося додати (можливо, вже в
группі)")
    })

    const removeStudentMutation = useMutation({
      mutationFn: (vars: { groupId: string, userId: string }) =>
removeUserFromGroup(vars.groupId, vars.userId),
      onSuccess: () => {
        toast.success("Студента видалено з групи")
        queryClient.invalidateQueries({ queryKey: ['admin',
'groups'] })
      }
    })

    const handleGroupDelete = (id: string, name: string) => {
      if (confirm(`Видалити групу "${name}"? Студенти не будуть
видалені з системи.`)) {
        deleteGroupMutation.mutate(id)
      }
    }

    if (isGroupsLoading || isUsersLoading) return <Loader2
className="animate-spin" />

    return (
      <div className="space-y-6">
        <Card>
          <CardHeader>
            <CardTitle>Створити нову групу</CardTitle>
          </CardHeader>
          <CardContent>
            <div className="flex gap-4">
              <Input
                placeholder="Назва групи (напр. КН-21)"
                value={newGroupName}
                onChange={e =>
setNewGroupName(e.target.value)}
              />
              <Button
                onClick={() =>
createGroupMutation.mutate(newGroupName)}
                disabled={!newGroupName.trim() ||
createGroupMutation.isPending}
              >
                <Plus className="mr-2 h-4 w-4" /> Створити
              </Button>
            </div>
          </CardContent>
        </Card>

        <div className="grid grid-cols-1 md:grid-cols-2 gap-6">
          {groups?.map(group => {
            const availableStudents = allStudents.filter(
              student => !group.students.some(groupStudent
=> groupStudent.id === student.id)
            );

            return (
              <Card key={group.id} className="flex flex-
col">
                <CardHeader className="pb-2">
                  <div className="flex justify-between
items-start">
                    <div>

```

```

center gap-2">
<CardTitle>{group.name}</CardTitle>
                                <Badge
variant="outline">{group.studentCount} студ.</Badge>
                                </div>
                                <CardDescription
className="text-xs font-mono mt-1">{group.id}</CardDescription>
                                </div>
                                <Button
                                variant="ghost"
                                size="icon"
                                className="h-8 w-8 text-
red-400 hover:text-red-600 hover:bg-red-50"
                                onClick={() =>
handleGroupDelete(group.id, group.name)}
                                >
                                <Trash2 className="h-4 w-4" />
                                </Button>
                                </div>
</CardHeader>
<CardContent className="flex-1 flex flex-
col gap-4">
                                <div className="flex-1 border rounded-
md p-2 max-h-48 overflow-y-auto space-y-1 bg-slate-50/50">
                                    {group.students.length === 0 ? (
                                        <p className="text-sm text-
muted-foreground text-center py-2">Група порожня</p>
                                    ) : (
                                        group.students.map(s => (
                                            <div key={s.id}
className="flex justify-between items-center text-sm p-2 bg-white
border rounded shadow-sm">
<span>{s.fullName}</span>
                                            <Button
                                            variant="ghost"
                                            size="icon"
                                            className="h-6 w-6
text-red-500 hover:text-red-700 hover:bg-red-50"
                                            onClick={() =>
removeStudentMutation.mutate({ groupId: group.id, userId: s.id })}
                                            >
                                            <Trash2
className="h-3 w-3" />
                                            </Button>
                                        </div>
                                    )
                                )
                                </div>
                                <div className="pt-2 border-t">
                                    <Label className="text-xs mb-1.5
block font-semibold text-muted-foreground">Додати студента</Label>
                                    <div className="flex gap-2">
                                        <Select
                                            key={`select-${group.id}-${
{group.students.length}}`
                                            onChange={ (userId) =>
addStudentMutation.mutate({ groupId: group.id, userId }) }
                                        >
                                            <SelectTrigger
className="h-9 text-sm">

```

```

placeholder="Оберіть студента..." />
                                <SelectValue>
                                </SelectTrigger>
                                <SelectContent>

{availableStudents.length === 0 ? (
                                <div
className="p-2 text-sm text-muted-foreground text-center">
                                Bci студенти
вже в групі
                                </div>
                                ) : (

availableStudents.map(s => (
                                <SelectItem
key={s.id} value={s.id}>

{s.fullName} <span className="text-xs text-muted-foreground">({s.email})</span>
                                </SelectItem>
                                ))
                                ))
                                </SelectContent>
                                </Select>
                                </div>
                                </div>
                                </CardContent>
                                </Card>
                                )
                                )))
                                </div>
                                </div>
                                )
}

```

/routes/announcements/create.tsx

```

import { createFileRoute, useNavigate } from '@tanstack/react-router'
import { useMutation, useQueryClient } from '@tanstack/react-query'
import { useEffect, useState } from 'react'
import { ArrowLeft, File as FileIcon, Link as LinkIcon, Loader2,
Megaphone, Plus, Save, X } from 'lucide-react'
import { toast } from 'sonner'
import MDEditor from '@uiw/react-md-editor'

import { createGlobalAnnouncement } from '@api/announcements'
import { UserRole, useAuthUser } from '@api/queries'
import { Button } from '@components/ui/button'
import { Input } from '@components/ui/input'
import { Label } from '@components/ui/label'
import { Card, CardContent, CardDescription, CardHeader, CardTitle }
from '@components/ui/card'

export const Route = createFileRoute('/announcements/create')({
  component: CreateGlobalAnnouncementPage,
})

function CreateGlobalAnnouncementPage() {
  const navigate = useNavigate()
  const queryClient = useQueryClient()
  const { user, isAuthenticated, isLoading } = useAuthUser()

  useEffect(() => {
    if (!isLoading && (!isAuthenticated || user?.role !==

```

```

UserRole.Admin)) {
    toast.error("Доступ заборонено")
    navigate({ to: '/' })
  }
}, [isLoading, isAuthenticated, user, navigate])

const [title, setTitle] = useState('')
const [text, setText] = useState('')
const [links, setLinks] = useState<Array<{ name: string, url: string
}>>([])
const [newLinkName, setNewLinkName] = useState('')
const [newLinkUrl, setNewLinkUrl] = useState('')
const [files, setFiles] = useState<Array<File>>([])

const addLink = () => {
  if (!newLinkName || !newLinkUrl) return;
  setLinks([...links, { name: newLinkName, url: newLinkUrl }]);
  setNewLinkName('');
  setNewLinkUrl('');
}

const removeLink = (index: number) => setLinks(links.filter((_, i)
=> i !== index));

const handleFileChange = (e: React.ChangeEvent<HTMLInputElement>) =>
{
  if (e.target.files)
setFiles([...files, ...Array.from(e.target.files)]);
}

const removeFile = (index: number) => setFiles(files.filter((_, i)
=> i !== index));

const mutation = useMutation({
  mutationFn: createGlobalAnnouncement,
  onSuccess: () => {
    toast.success("Глобальне оголошення опубліковано!")
    queryClient.invalidateQueries({ queryKey: ['global-
announcements'] })
    navigate({ to: '/' })
  },
  onError: () => toast.error("Не вдалося створити оголошення")
})

if (isLoading) return <div className="p-10 flex justify-
center"><Loader2 className="animate-spin" /></div>

return (
  <div className="container max-w-4xl mx-auto py-10 px-4">
    <Button variant="ghost" onClick={() => navigate({ to: '/' })}
className="mb-4">
      <ArrowLeft className="mr-2 h-4 w-4" /> На головну
    </Button>

    <Card className="border-orange-200 shadow-md">
      <CardHeader>
        <div className="flex items-center gap-3">
          <div className="p-2 bg-orange-100 text-orange-600 rounded-
lg">
            <Megaphone className="h-6 w-6" />
          </div>
          <div>
            <CardTitle>Нові глобальні оголошення</CardTitle>
            <CardDescription>Це повідомлення побачать усі

```

```

користувачі на головній сторінці.</CardDescription>
</div>
</div>
</CardHeader>
<CardContent className="space-y-6">
  <div className="space-y-2">
    <Label htmlFor="title">Тема</Label>
    <Input id="title" value={title} onChange={(e) =>
setTitle(e.target.value)} placeholder="Важлива новина..." />
  </div>

  <div className="space-y-2">
    <Label>Текст</Label>
    <div data-color-mode="light" className="pt-1">
      <MDEditor
        value={text}
        onChange={(val) => setText(val || '')}
        height={300}
        preview="edit"
      />
    </div>
  </div>

  <div className="space-y-3 p-4 bg-slate-50 rounded-lg
border">
    <Label className="text-xs uppercase text-muted-foreground
font-bold">Посилання</Label>
    <div className="flex flex-col sm:flex-row gap-2">
      <Input placeholder="Назва" value={newLinkName}
onChange={e => setNewLinkName(e.target.value)} className="flex-1" />
      <Input placeholder="URL" value={newLinkUrl} onChange={e
=> setNewLinkUrl(e.target.value)} className="flex-1" />
      <Button type="button" size="icon" variant="secondary"
onClick={addLink}><Plus className="h-4 w-4" /></Button>
    </div>
    <div className="space-y-2">
      {links.map((l, i) => (
        <div key={i} className="flex items-center justify-
between text-sm bg-white p-2 rounded border">
          <div className="flex items-center gap-2 overflow-
hidden min-w-0">
            <LinkIcon className="h-3 w-3 text-blue-500
shrink-0" />
            <a href={l.url} target="_blank"
className="truncate text-blue-600 hover:underline">{l.name}</a>
          </div>
          <button onClick={() => removeLink(i)}
className="text-slate-400 hover:text-red-500 shrink-0 ml-2"><X
className="h-4 w-4" /></button>
        </div>
      ))}
    </div>
  </div>

  <div className="space-y-3">
    <Label>Файли</Label>
    <Input type="file" multiple onChange={handleFileChange} />
    <div className="space-y-1">
      {files.map((f, i) => (
        <div key={i} className="flex items-center justify-
between text-sm bg-slate-50 p-2 rounded">
          <div className="flex items-center gap-2 min-w-0">
            <FileIcon className="h-4 w-4 text-slate-500
shrink-0" />

```

```

        <span className="truncate">{f.name}</span>
      </div>
      <button onClick={() => removeFile(i)}
className="text-slate-400 hover:text-red-500 ml-2 shrink-0"><X
className="h-4 w-4" /></button>
    </div>
  )}
</div>
</div>

<div className="flex justify-end pt-4">
  <Button size="lg" className="bg-orange-600 hover:bg-
orange-700" onClick={() => {
    mutation.mutate({
      title, text, jsonLinks: JSON.stringify(links), files
    })
  }} disabled={mutation.isPending || !title}>
    {mutation.isPending ? <Loader2 className="mr-2 h-4 w-4
animate-spin" /> : <Save className="mr-2 h-4 w-4" />}
    Опублікувати
  </Button>
</div>
</CardContent>
</Card>
</div>
)
}

```

/routes/announcements/\$announcementId/index.tsx

```

/* eslint-disable @typescript-eslint/no-unnecessary-condition */
import { Link, createFileRoute, useNavigate } from '@tanstack/react-
router'
import { useMutation, useQuery } from '@tanstack/react-query'
import { ArrowLeft, Calendar, Edit, FileIcon, LinkIcon, Loader2,
Trash2 } from 'lucide-react'
import { format } from 'date-fns'
import { uk } from 'date-fns/locale'
import { toast } from 'sonner'
import MDEditor from '@uiw/react-md-editor'

import { deleteAnnouncement, getAnnouncement } from
'@/api/announcements'
import { Button } from '@/components/ui/button'
import { Card, CardContent, CardHeader, CardTitle } from
'@/components/ui/card'
import { Avatar, AvatarFallback, AvatarImage } from
'@/components/ui/avatar'
import { Separator } from '@/components/ui/separator'
import { Skeleton } from '@/components/ui/skeleton'
import { UserRole, useAuthUser } from '@/api/queries'

export const Route =
createFileRoute('/announcements/$announcementId/') ({
  component: GlobalAnnouncementDetailPage,
})

function GlobalAnnouncementDetailPage() {
  const { announcementId } = Route.useParams()
  const navigate = useNavigate()
  const { user, isAuthenticated } = useAuthUser()

  const { data: announcement, isLoading, isError } = useQuery({
    queryKey: ['announcement', announcementId],

```

```

    queryFn: () => getAnnouncement(announcementId),
  })

  const deleteMutation = useMutation({
    mutationFn: deleteAnnouncement,
    onSuccess: () => {
      toast.success("Оголошення видалено")
      navigate({ to: '/' })
    },
    onError: () => toast.error("Помилка видалення")
  })

  if (isLoading) return <div className="container py-10"><Skeleton
className="h-96 w-full max-w-4xl mx-auto" /></div>
  if (isError || !announcement) return <div className="p-10 text-
center text-muted-foreground">Оголошення не знайдено</div>

  const isAdmin = isAuthenticated && user?.role === UserRole.Admin;
  const canEdit = isAdmin;

  const handleDelete = () => {
    if (confirm("Видалити це оголошення назавжди?")) {
      deleteMutation.mutate(announcementId)
    }
  }

  return (
    <div className="container max-w-4xl mx-auto py-8 px-4">
      <div className="mb-6 flex justify-between items-center">
        <Button variant="ghost" asChild className="px-0 text-muted-
foreground">
          <Link to="/">
            <ArrowLeft className="h-4 w-4 mr-2" /> На головну
          </Link>
        </Button>

        {canEdit && (
          <div className="flex gap-2">
            <Button variant="outline" size="sm" asChild>
              <Link to="/announcements/$announcementId/edit" params={{
announcementId }}>
                <Edit className="h-4 w-4 mr-2" /> Редагувати
              </Link>
            </Button>
            <Button variant="destructive" size="sm"
onClick={handleDelete} disabled={deleteMutation.isPending}>
              {deleteMutation.isPending ? <Loader2 className="h-4 w-4
animate-spin" /> : <Trash2 className="h-4 w-4" />}
            </Button>
          </div>
        )}
      </div>

      <Card className="shadow-lg border-t-4 border-t-orange-500">
        <CardHeader className="space-y-4">
          <div className="flex flex-col gap-2">
            <CardTitle className="text-3xl font-bold text-slate-900
leading-tight break-all">
              {announcement.title}
            </CardTitle>

            <div className="flex items-center gap-4 text-sm text-
muted-foreground mt-2">
              <div className="flex items-center gap-2 bg-slate-50 px-3

```

```

py-1.5 rounded-full border">
    <Avatar className="h-5 w-5">
      <AvatarImage src={announcement.authorAvatar ||
undefined} />
    <AvatarFallback className="text-[10px] bg-
slate-200">
      {announcement.authorName[0]}
    </AvatarFallback>
  </Avatar>
  <span className="font-medium text-
slate-700">{announcement.authorName}</span>
</div>

  <div className="flex items-center gap-1.5">
    <Calendar className="h-4 w-4" />
    <span>{format(new Date(announcement.createdAt), "d
MMM yyyy, HH:mm", { locale: uk })}</span>
  </div>
</div>
</div>
</CardHeader>

<Separator />

<CardContent className="pt-6">
  <div className="prose prose-slate max-w-none text-slate-800
leading-relaxed">
    <div data-color-mode="light">
      <MDEditor.Markdown
        source={announcement.text}
        style={{ backgroundColor: 'transparent', color:
'inherit', fontSize: '1rem' }}
      />
    </div>
  </div>

  {announcement.attachments && announcement.attachments.length
> 0 && (
    <div className="mt-8 pt-6 border-t border-dashed">
      <h4 className="text-sm font-semibold text-muted-
foreground mb-3 uppercase tracking-wider">
        Прикріплені матеріали
      </h4>
      <div className="grid grid-cols-1 sm:grid-cols-2 gap-3">
        {announcement.attachments.map((att, idx) => (
          <a
            key={idx}
            href={att.url}
            target="_blank"
            rel="noopener noreferrer"
            className={`
              flex items-center gap-3 p-3 rounded-lg border
transition-all hover:shadow-sm
              ${att.isLink
                ? "bg-blue-50 border-blue-100 hover:border-
blue-300 text-blue-700"
                : "bg-slate-50 border-slate-100 hover:border-
slate-300 text-slate-700"
              }
            `}
          >
            <div className={`p-2 rounded-md ${att.isLink ?
"bg-blue-100" : "bg-white border"}`}>
              {att.isLink ? <LinkIcon className="h-4 w-4" /> :

```

```

<FileIcon className="h-4 w-4" />
      </div>
      <span className="text-sm font-medium truncate
flex-1">{att.name}</span>
    </a>
  )))
</div>
</div>
  )
</CardContent>
</Card>
</div>
)
}

```

/routes/auth/login.tsx

```

import { Link, createFileRoute, useNavigate } from '@tanstack/react-router'
import { useMutation, useQueryClient } from '@tanstack/react-query'
import { useState } from "react"
import { toast } from "sonner"
import { GraduationCap, Loader2 } from 'lucide-react'
import type { User } from '@api/queries'
import { Button } from "@components/ui/button"
import {
  Card,
  CardContent,
  CardDescription,
  CardHeader,
  CardTitle,
} from "@components/ui/card"
import {
  Field,
  FieldGroup,
  FieldLabel,
} from "@components/ui/field"
import { Input } from "@components/ui/input"

export type AuthResponse = {
  user: User
}

export const Route = createFileRoute('/auth/login')({
  component: RouteComponent,
})

function RouteComponent() {
  const navigate = useNavigate()
  const queryClient = useQueryClient()

  const [username, setUsername] = useState("")
  const [password, setPassword] = useState("")

  const { mutate: loginMutation, isPending } = useMutation({
    mutationFn: async ({ username, password }: { username: string;
password: string }): Promise<AuthResponse> => {
      const res = await fetch("https://localhost:7214/login", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ UserName: username, Password:
password }),
        credentials: "include",
      })
    }
  })

```

```

const data = await res.json().catch(() => ({}))

if (!res.ok) {
  if (res.status === 401) {
    throw new Error("Неправильний логін або пароль")
  }
  throw new Error(data.error || data.title || "Помилка входу")
}

return data
},

onSuccess: (data) => {
  queryClient.invalidateQueries({ queryKey: ['user'] });
  toast.success(`Вітаємо, ${data.user.fullName}!`)
  navigate({ to: "/" })
},

onError: (e: any) => {
  toast.error(e.message || "Щось пішло не так")
},
))

const handleSubmit = (e: React.FormEvent) => {
  e.preventDefault()
  loginMutation({ username, password })
}

return (
  <div className='min-h-screen flex flex-col items-center justify-center p-4 bg-stone-50'>
    <Link to="/" className="flex items-center gap-2 mb-8 group">
      <div className="bg-slate-900 p-2 rounded-lg group-hover:bg-slate-800 transition-colors">
        <GraduationCap className="h-8 w-8 text-cyan-400" />
      </div>
      <span className="text-2xl font-bold tracking-tight text-slate-900">E-Learning</span>
    </Link>

    <Card className='w-full max-w-md shadow-lg border-slate-200'>
      <CardHeader className="space-y-1">
        <CardTitle className="text-2xl text-center">Вхід в акаунт</CardTitle>
        <CardDescription className="text-center">
          Введіть логін та пароль для доступу до навчання
        </CardDescription>
      </CardHeader>
      <CardContent>
        <form onSubmit={handleSubmit} className="space-y-4">
          <FieldGroup>
            <Field className="space-y-2">
              <FieldLabel htmlFor="username">Логін (Username)</FieldLabel>
              <Input
                id="username"
                type="text"
                placeholder="Введіть ваш логін"
                required
                value={username}
                onChange={(e) => setUsername(e.target.value)}
                className="bg-white"
              />
            </Field>
          </FieldGroup>
        </form>
      </CardContent>
    </Card>
  </div>
)

```

```

    </Field>

    <Field className="space-y-2">
      <div className="flex items-center justify-between">
        <FieldLabel htmlFor="password">Пароль</FieldLabel>
      </div>
      <Input
        id="password"
        type="password"
        placeholder="Введіть пароль"
        required
        value={password}
        onChange={ (e) => setPassword(e.target.value) }
        className="bg-white"
      />
    </Field>

    <Button
      type="submit"
      className="w-full mt-4 bg-slate-900 hover:bg-slate-800
text-white"
      disabled={isPending}
    >
      {isPending ? (
        <>
          <Loader2 className="mr-2 h-4 w-4 animate-spin" />
          Вхід...
        </>
      ) : "Увійти"}
    </Button>
  </FieldGroup>
</form>

  <div className="mt-6 text-center text-sm text-slate-500">
    Ще не маєте акаунту?{" "}
    <Link to="/auth/register" className="font-semibold text-
    cyan-600 hover:text-cyan-500 hover:underline">
      Зареєструватися
    </Link>
  </div>
</CardContent>
</Card>
</div>
)

```

ДОДАТОК В
Слайди презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

дипломна робота на тему:
**Дослідження та програмна реалізація вебплатформи підтримки
освітнього процесу**
Research and Software Implementation of a Web Platform for Supporting the Educational Process

- Виконав: студент гр. КНТ-114м Нікіта ХАЛІЗЕВ
- Керівник: к.т.н., доцент Ольга ГЛАДКОВА

Рисунок В.1 – Слайд 1

Об'єкт, предмет, мета роботи

- **Об'єкт дослідження – процес розробки програмного забезпечення для підтримки освітнього процесу.**
- **Предмет дослідження – програмні засоби реалізації вебплатформи для підтримки освітнього процесу.**
- **Мета роботи – зменшення часових витрат на організацію навчального процесу та підвищення оперативності взаємодії між його учасниками шляхом розробки вебплатформи зі застосуванням сучасних методів програмної інженерії.**

2

Рисунок В.2 – Слайд 2

Аналіз предметної області

Критерій порівняння	Canvas LMS	Moodle	Google Classroom
Тип системи	Хмарна (SaaS), або Open Source	Open Source, вимагає хостингу	Хмарна, пропріетарна
Вартість	Висока вартість корпоративної ліцензії. Наявний безкоштовний план з обмеженим функціоналом	Безкоштовне ПЗ (витрати на хостинг та підтримку ПЗ)	Безкоштовна
Гнучкість та кастомізація	Дуже висока (плагіни, відкритий вихідний код)	Дуже висока (плагіни, відкритий вихідний код)	Низька (стандартний інтерфейс)
Простота розгортання	Дуже проста і швидка (підтримка вендора)	Складна (потребує ІТ компетенцій)	Дуже проста та швидка
Педагогічні можливості	Збалансовані	Просунуті	Базові
Аналітика та звітність	Розширена візуалізація даних	Детальні логи активності	Базова, обмежена візуалізація даних
Цільова аудиторія	Середні та великі освітні заклади, бізнес	Університети, що потребують адаптації ПЗ	Малі курси

3

Рисунок В.3 – Слайд 3

Вибір мови програмування для бекенду

Критерій порівняння	C#	PHP	Ruby
Тип типізація	Сильна, статична	Слабка, динамічна	Динамічна
Парадигма	Повністю об'єктно-орієнтована	Об'єктно-орієнтована, процедурна	Об'єктно-орієнтована
Продуктивність	Висока (компільована мова)	Середня (інтерпретована, є JIT)	Низька (поступається аналогам)
Основні фреймворки	ASP.NET Core	Laravel, Symfony	Ruby on Rails
Крива навчання	Помірна/Висока (складніша для новачків)	Низька (легка для вивчення)	Помірна (простий синтаксис, але складна «магія» Rails)
Швидкість розробки	Середня (вимагає більше коду)	Висока (швидкий старт)	Дуже висока (швидке прототипування)

4

Рисунок В.4 – Слайд 4

Вибір мови програмування для фронтенду

Критерій порівняння	JavaScript	TypeScript	Dart
Тип типізації	Динамічна, слабка	Статична, сильна	Статична, сильна (опціональна)
Виявлення помилок	Під час виконання	Під час компіляції	Під час компіляції
Популярність та екосистема	Дуже висока, величезна кількість бібліотек	Висока, використовує екосистему JS	Зростаюча, але менша за JS/TS
Сфера застосування	Веб (клієнт/сервер), мобільні застосунки	Така ж, як у JavaScript	Переважно кросплатформні мобільні застосунки (Flutter)
Продуктивність	Інтерпретована	Компілюється в JS	Висока (нативна компіляція)
Підтримка ООП	На основі прототипів	Повноцінна (класи, інтерфейси)	Повноцінна (класи, інтерфейси)

5

Рисунок В.5 – Слайд 5

Вибір фреймворків

Критерій порівняння	Angular	React	Vue
Тип	Повноцінний фреймворк	Бібліотека для побудови інтерфейсів	Прогресивний фреймворк
Розробник	Google	Meta (Facebook)	Еван Ю та спільнота (Open Source)
Основна мова розробки	TypeScript	JavaScript (JSX) / TypeScript (TSX)	JavaScript / TypeScript
Поріг входження	Високий (вимагає вивчення великої кількості концепцій)	Середній (потрібно розуміння JSX та екосистеми)	Низький (простий синтаксис на базі HTML/JS)
Архітектура та структура	Жорстка модульна архітектура	Гнучка (архітектуру обирає розробник)	Гнучка (можливість поступового впровадження)
Сфера застосування	Масштабні корпоративні проекти	SPA, мобільні додатки, соціальні мережі	Стартапи, інтерфейси середньої складності, SPA

6

Рисунок В.6 – Слайд 6

Обрання бази даних

Критерій порівняння	MySQL	PostgreSQL	MongoDB
Тип бази даних	Реляційна	Об'єктно-реляційна	NoSQL (Документна)
Структура даних	Таблиці, сувора схема	Таблиці, підтримка JSON та користувацьких типів	Гнучкі JSON-документи
Масштабованість	Вертикальна	Вертикальна, висока продуктивність при складних запитах	Горизонтальна
Підтримка ACID	Так (через InnoDB)	Так, повна підтримка транзакцій	Обмежена
Підтримка JSON	Обмежена	Розширена	Нативна
Складність запитів	Стандартні SQL запити	Складні запити, розширені процедури	Обмежені можливості

7

Рисунок В.7 – Слайд 7

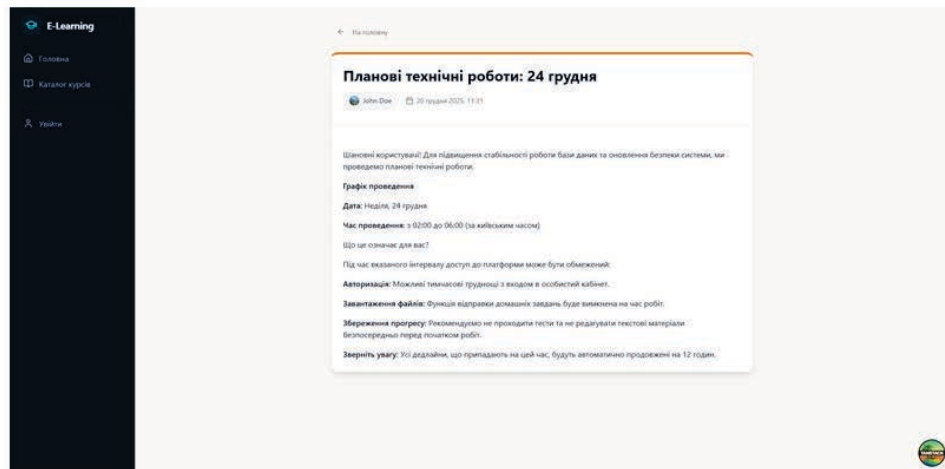
Обрання середовища розробки

Критерій порівняння	Visual Studio 2022	Visual Studio Code	JetBrains Rider
Тип ПЗ	Комплексна IDE	Редактор коду	Комплексна IDE
Кросплатформність	Відсутня	Наявна	Наявна
Аналіз коду	Високий	Базовий (залежить від плагінів)	Дуже високий (ReSharper)
Продуктивність	Низька (ресурсоемна)	Середня	Висока
Налаштування	Готова до роботи	Потребує налаштування плагінів	Готова до роботи
Підтримка перегляду БД	Відсутня	Залежить від плагінів	Вбудована
Вартість	Безкоштовна (Community) / Платна	Безкоштовна	Безкоштовна (Non-Commercial) / Платна

8

Рисунок В.8 – Слайд 8

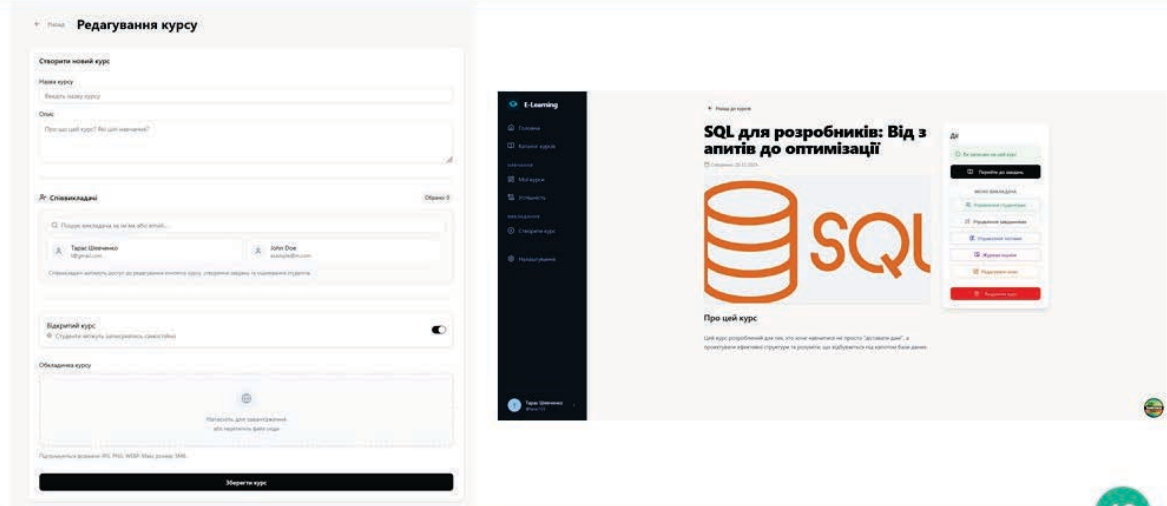
Екранні форми



11

Рисунок В.11 – Слайд 11

Екранні форми



12

Рисунок В.12 – Слайд 12

Екранні форми

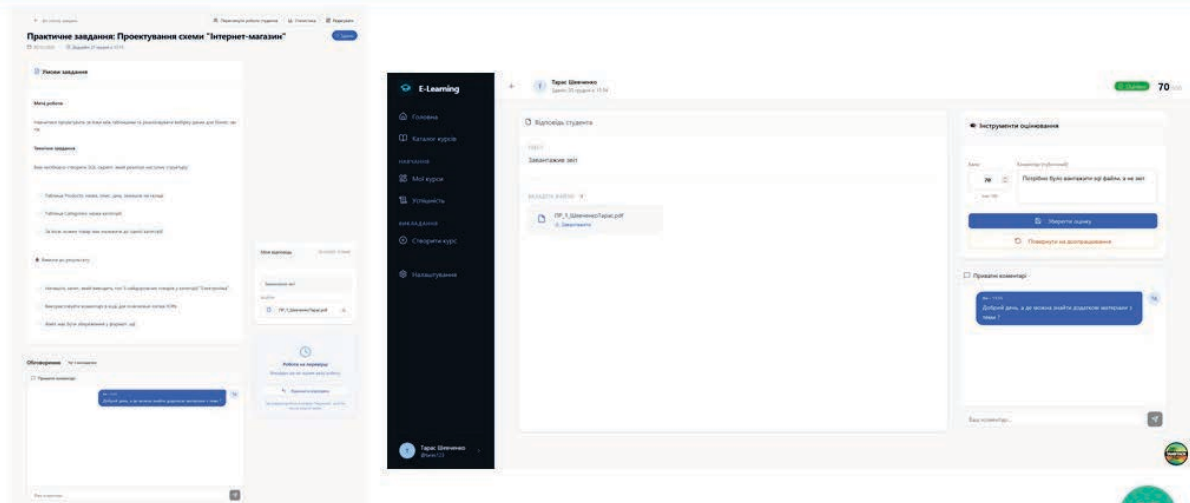


Рисунок В.13 – Слайд 13

Екранні форми

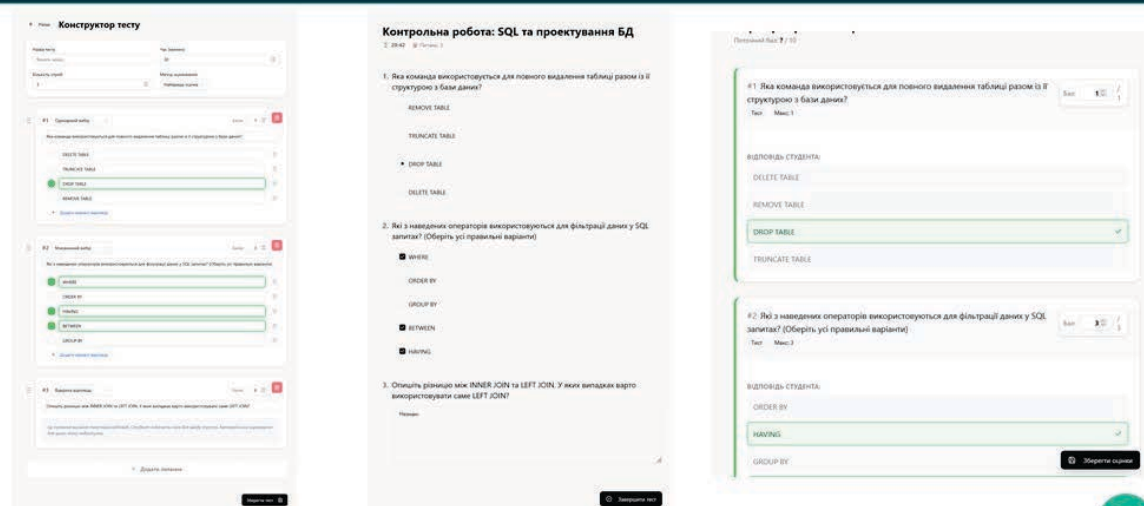
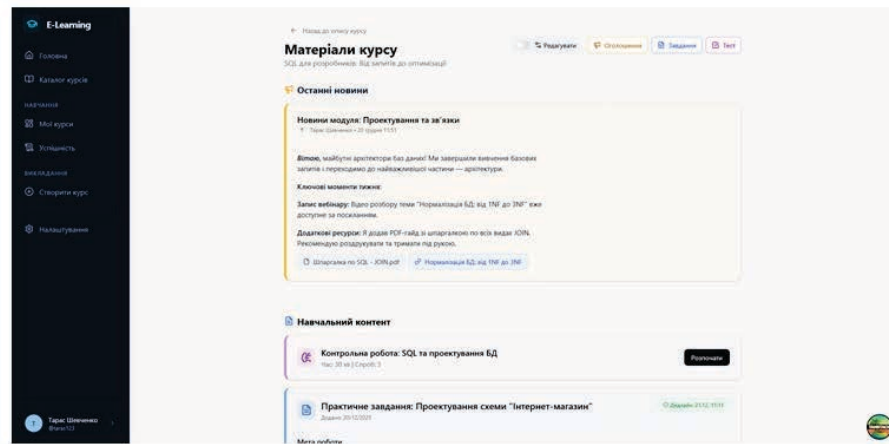


Рисунок В.14 – Слайд 14

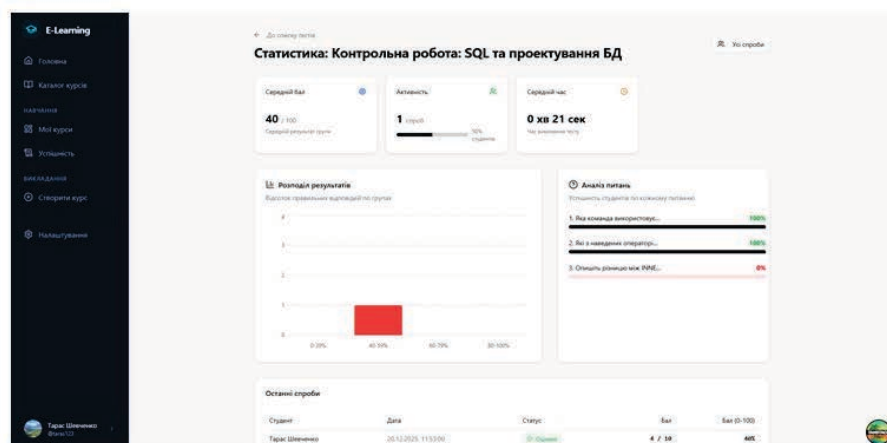
Екранні форми



15

Рисунок В.15 – Слайд 15

Екранні форми



16

Рисунок В.16 – Слайд 16

Екранні форми

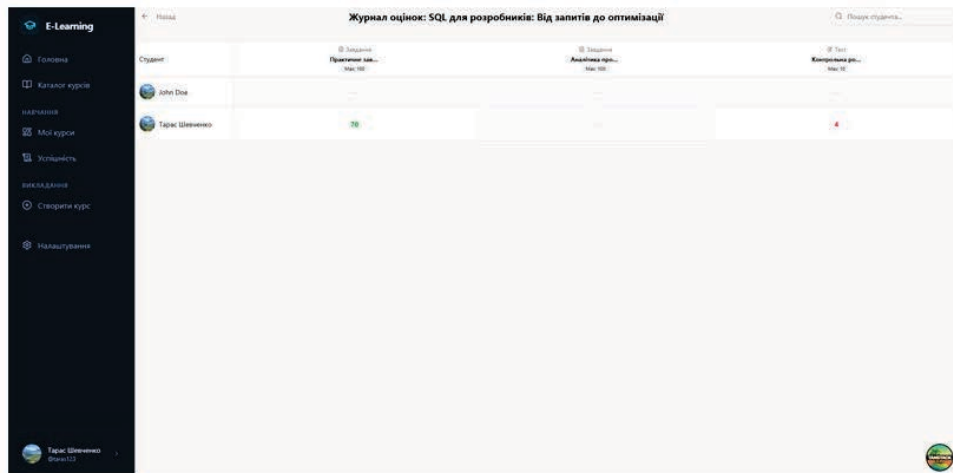


Рисунок В.17 – Слайд 17

Висновки

У результаті виконання магістерської роботи було успішно розроблено вебплатформу для підтримки освітнього процесу, що дозволило повністю досягти поставленої мети щодо зменшення часових витрат на організацію навчання та підвищення оперативності взаємодії між усіма учасниками освітнього середовища.

Рисунок В.18 – Слайд 18