МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ
Циклова комісія (кафедра) комп’ютерної інженерії та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему

FRONTEND LMS СИСТЕМИ

Виконав: студент групи 1П-21
Спеціальності
121 Інженерія програмного забезпечення
Олександр ГОНЧАР
Керівник:
Валентин ДМИТРЮК

Черкаси 2025

АНОТАЦІЯ

Дипломна робота присвячена розробці вебзастосунку для системи управління навчанням (LMS) з використанням сучасного фронтенд-стека, зокрема Next.js, Redux Toolkit та Tailwind CSS. Метою роботи є створення адаптивної, багаторольової платформи для ефективної організації навчального процесу, доступної через браузер на будь-якому пристрої.

У ході дослідження проаналізовано функціональні вимоги до LMS, особливості ролей користувачів (студент, викладач, адміністратор) та шляхи оптимізації інтерфейсу для кожної з них. Обґрунтовано вибір архітектурних рішень і бібліотек для керування станом (Redux Toolkit для глобального стану та Zustand для локального). Реалізовано такі функції: реєстрація та авторизація, управління курсами, подача завдань, відстеження прогресу студентів, а також система повідомлень і новин.

Застосунок протестовано на різних етапах розробки, що забезпечило його стабільну роботу та відповідність технічним і користувацьким вимогам. Результати проєкту можуть бути використані як основа для подальшого розвитку систем дистанційного навчання в освітніх установах.

Ключові слова: LMS, NEXT.JS, REDUX TOOLKIT, ZUSTAND, ВЕБЗАСТОСУНОК, ДИСТАНЦІЙНЕ НАВЧАННЯ, БАГАТОРОЛЬОВА СИСТЕМА, АДАПТИВНИЙ ІНТЕРФЕЙС.

ANNOTATION

The thesis is dedicated to the development of a web application for a Learning Management System (LMS) using a modern frontend stack, including Next.js, Redux Toolkit, and Tailwind CSS. The goal of the project is to create a responsive, multi-role platform that effectively supports the educational process and is accessible via any device through a web browser.

The research includes an analysis of LMS functional requirements, user roles (student, teacher, administrator), and strategies for optimizing the user interface for each role. Architectural decisions and state management tools were justified, with Redux Toolkit used for global state and Zustand for local UI state. Key features implemented include registration and login, course management, assignment submission, student progress tracking, and a built-in messaging and announcement system.

The application was tested throughout the development cycle to ensure stable performance and compliance with technical and user requirements. The results of the project may serve as a foundation for the further development of online learning systems in academic or corporate settings.

Keywords: LMS, NEXT.JS, REDUX TOOLKIT, ZUSTAND, WEB APPLICATION, ONLINE LEARNING, MULTI-ROLE SYSTEM, RESPONSIVE INTERFACE.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ

LMS – Learning Management System (система управління навчанням)

UI – User Interface (інтерфейс користувача)

UX – User Experience (користувацький досвід)

SPA – Single Page Application (односторінковий застосунок)

API – Application Programming Interface (інтерфейс прикладного програмування)

JWT – JSON Web Token (токен для аутентифікації користувачів)

RTK – Redux Toolkit (набір інструментів для управління станом)

SSR – Server-Side Rendering (рендеринг на стороні сервера)

CSR – Client-Side Rendering (рендеринг на стороні клієнта)

DB – Database (база даних)

SQL – Structured Query Language (мова структурованих запитів)

ID – Identifier (ідентифікатор)

MVP – Minimum Viable Product (мінімально життєздатний продукт)

CMS – Content Management System (система управління контентом)

QA – Quality Assurance (забезпечення якості)

UXR – User Experience Research (дослідження користувацького досвіду)

CSV – Comma-Separated Values (формат даних для таблиць)

JSON – JavaScript Object Notation (формат обміну даними)

SVG – Scalable Vector Graphics (масштабована векторна графіка)

PDF – Portable Document Format (переносимий формат документів)

CMS – Course Management System (система керування курсами – окремий функціонал LMS)

ВСТУП

Сучасний світ диктує нові вимоги до освіти, зокрема до її гнучкості, доступності та інтерактивності. Дистанційне навчання, яке набуло значного поштовху під час пандемії COVID-19, залишається актуальним і продовжує розвиватися. Одним із ключових інструментів для організації ефективного онлайн-навчання є Learning Management System (LMS) – програмні платформи, що забезпечують управління курсами, тестуванням та навчальними матеріалами. Проблема, яка потребує вирішення, полягає у відсутності зручної, сучасної та адаптованої до потреб українських навчальних закладів LMS з інтуїтивним інтерфейсом, гнучким функціоналом та можливістю масштабування. Існуючі рішення (наприклад, Moodle, Google Classroom) часто є або надто складними для користувачів, або обмеженими у функціональності.

Актуальність теми обумовлена зростаючою потребою в інтерактивних системах дистанційного навчання, які поєднують зручність, безпеку та ефективність. Особливо це важливо для шкіл та університетів, де потрібно забезпечити індивідуальний підхід до навчання, автоматизацію перевірки знань та зручний доступ до матеріалів.

Метою даної роботи є розробка фронтенд-частини LMS-системи з використанням сучасних технологій (React, Tailwind CSS, Redux Toolkit), яка забезпечить:

- зручну реєстрацію та авторизацію користувачів (учні, вчителі, адміністратори);
- перегляд навчальних курсів (відео, документи, посилання);
- проходження тестів з автоматичною перевіркою;
- адміністрування контенту та користувачів;
- аналітику успішності.

Об’єкт дослідження – процес дистанційного навчання та сучасні LMS-системи.

Предмет дослідження – розробка фронтенд-частини інформаційної системи для дистанційного навчання.

Завдання дослідження:

- Проаналізувати існуючі LMS-системи та їхні недоліки.
- Визначити вимоги до функціоналу та інтерфейсу системи
- Розробити архітектуру фронтенд-частини з використанням React та сучасних бібліотек.
- Реалізувати основний функціонал (авторизація, курси, тестування, адмін-панель).
- Забезпечити зручний інтерфейс та адаптивність для різних пристроїв.

Методи дослідження:

- Аналіз існуючих рішень;
- Проєктування UI/UX з використанням Figma;
- Розробка компонентів на React;
- Тестування інтерфейсу (юзабіліті, продуктивність).

Наукова новизна полягає у поєднанні сучасних фронтенд-технологій (ShadCN UI, Zod для валідації) для створення більш інтуїтивної та безпечної LMS.

Практична значимість: система може бути використана в школах, університетах або корпоративних тренінгах для підвищення ефективності дистанційного навчання.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД СУЧАСНИХ LMS-СИСТЕМ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз ринку та існуючих рішень

Сучасний освітній простір стрімко трансформується під впливом цифровізації, що призводить до постійного зростання попиту на якісні системи дистанційного навчання. Проведений аналіз показав, що більшість популярних LMS були розроблені ще на початку 2010-х років і з того часу не зазнали суттєвих змін у підході до користувацького досвіду. Наприклад, інтерфейс Moodle, незважаючи на свою функціональність, залишається перевантаженим і неінтуїтивним для нових користувачів, особливо для покоління Z, яке звикло до мінімалістичних мобільних інтерфейсів.

Сьогодні на ринку навчальних платформ існує велика кількість LMS (Learning Management Systems), кожна з яких має свої переваги та недоліки. Серед найпопулярніших можна виокремити:

– Moodle – відкрита платформа з широкою функціональністю, але складним інтерфейсом для звичайних користувачів.

– Google Classroom – проста у використанні, але обмежена в гнучкості та аналітиці.

– Canvas – потужна система для університетів, але високовартісна для впровадження.

– Blackboard – корпоративне рішення зі складною архітектурою.

Цікавим аспектом є те, що сучасні освітні тенденції (такі як мікронавчання, гейміфікація) практично не реалізовані в традиційних системах. Наприклад, платформи типу Google Classroom не підтримують інтерактивні елементи на кшталт вікторин у реальному часі або системи мотивації через досягнення. Це створює значний простір для вдосконалення та інновацій у розробці нових LMS-

рішень. Провівши порівняльний аналіз, можна виділити ключові проблеми сучасних LMS:

– Складність інтерфейсу – багато систем (наприклад, Moodle) мають застарілий дизайн, що ускладнює навігацію.

– Недостатня адаптивність – не всі платформи коректно працюють на мобільних пристроях.

– Обмежена функціональність безкоштовних версій (Google Classroom не має глибокої аналітики успішності).

– Відсутність гнучкої системи тестування – багато LMS не підтримують сучасні формати завдань (drag-and-drop, інтерактивні питання).

Ці недоліки обумовлюють необхідність створення нової системи, яка поєднає зручність, сучасний дизайн та потужну функціональність.

1.2 Вимоги до сучасної LMS-системи

При формулюванні вимог до системи було враховано не лише технічні аспекти, але й психологічні особливості сприйняття інформації сучасними студентами. Згідно з дослідженням Microsoft (2015), середня тривалість концентрації уваги зменшилася з 12 секунд у 2000 році до 8 секунд у 2013, що потребує особливого підходу до дизайну навчальних інтерфейсів [Microsoft, 2015].

На основі аналізу існуючих рішень та потреб користувачів були сформульовані основні вимоги до розроблюваної системи. Функціональні вимоги:

– Реєстрація/авторизація через ролі (учень, викладач, адмін).

– Управління курсами (додавання, редагування, призначення групам).

– Робота з навчальними матеріалами (PDF, відео, посилання).

– Система тестування з автоматичною перевіркою (одинарний/множинний вибір, текстові відповіді).

– Статистика успішності для учнів та викладачів.

– Адмін-панель для керування користувачами та курсами.

Нефункціональні вимоги:

– Швидкодія – завантаження сторінок має бути ≤ 1.5 сек.

– Адаптивність – підтримка мобільних пристроїв (Mobile-First підхід).

– Безпека – JWT-аутентифікація, HTTPS, захист від XSS/SQL-ін’єкцій.

– Зручний UI/UX – інтуїтивна навігація, мінімалістичний дизайн.

Важливим викликом стала необхідність балансу між функціональністю та простотою. Наприклад, система повинна надавати викладачам потужні інструменти аналітики успішності, але при цьому не перетворюватися на «монстра» типу Blackboard, де налаштування звичайного тесту може займати до 30 хвилин. Особливу увагу приділено мобільній версії, оскільки понад 60% студентів регулярно використовують смартфони для навчання.

1.3 Вибір технологій для фронтенд-розробки

Глибокий аналіз сучасних фронтенд-технологій показав, що вибір React є оптимальним не лише через його популярність [Stack Overflow, 2023], але й через можливість подальшого масштабування [React Docs, 2024]. Наприклад, використання Server Components у Next.js могло б значно покращити продуктивність при роботі з великими навчальними матеріалами. Оскільки система має бути швидкою, масштабованою та зручною, було обрано наступний стек:

Вибір технологічного стеку зумовлений вимогами до швидкодії, масштабованості та зручності системи. Приклад такого стеку представлено в таблиці 1.1.

Таблиця 1.1 – Вибраний стек технологій для реалізації LMS

Технологія	Причина вибору
React	Віртуальний DOM для швидкого рендерингу, велика спільнота, підтримка TypeScript.
Tailwind CSS	Утилітарний підхід для швидкої стилізації без зайвих CSS-файлів.
ShadCN UI	Готові доступні (a11y) компоненти для побудови сучасного інтерфейсу.
Redux Toolkit	Керування глобальним станом (користувачі, курси, тести).
Zod	Схемна валідація форм для підвищення надійності.
React Router	Навігація між сторінками з підтримкою динамічних маршрутів.

Цікавим рішенням стало використання ShadCN UI замість традиційних бібліотек типу Material UI. Це дозволить створити унікальну дизайн-систему, яка не буде виглядати «штампованою», як більшість сучасних освітніх платформ. Додатковою перевагою є те, що Tailwind CSS дає змогу легко кастомізувати стилі під конкретні потреби навчального закладу. Ці технології дозволяють створити продуктивний, легкий у підтримці та масштабований фронтенд, що є критично важливим для навчальних систем.

1.4 Вибір технологій для фронтенд-розробки

Особливу увагу було приділено питанням доступності (accessibility), оскільки сучасні освітні платформи повинні бути пристосовані для людей з особливими потребами. Це включає підтримку скрінрідерів, керування з клавіатури та відповідність стандарту WCAG 2.1.

На основі проведеного аналізу основна задача проекту формулюється так:

Розробити фронтенд-частину LMS-системи на React, яка забезпечить зручний інтерфейс для дистанційного навчання, тестування та аналітики успішності, з можливістю подальшого масштабування.

Ключові етапи розв’язання задачі:

– Проєктування UI/UX (Figma).

– Розробка авторизації (JWT, Protected Routes).

– Реалізація головного інтерфейсу (курси, матеріали, тести).

– Побудова адмін-панелі (CRUD для користувачів/курсів).

– Тестування (зручність використання, продуктивність, безпека).

Очікуваний результат:

– Сучасний інтерфейс (адаптивний, зручний для користувачів).

– Гнучка система тестування (різні типи завдань, автоматична перевірка).

– Можливість інтеграції з бекендом (REST API, WebSockets для сповіщень).

Висновок до розділу 1

Проведений аналіз виявив значний технологічний розрив між сучасними вимогами до навчальних платформ і можливостями існуючих рішень. Запропонований підхід до розробки, що поєднує сучасні фронтенд-технології з урахуванням психології сприйняття, дозволить створити дійсно інноваційний продукт.

Важливим результатом аналізу стало усвідомлення, що успішна LMS має бути не лише інструментом доставки контенту, а й платформою для комунікації та мотивації. Це особливо актуально в умовах, коли дистанційне навчання стає не тимчасовим заходом, а постійною складовою освітнього процесу.

РОЗДІЛ 2
АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Загальна архітектура системи

Розроблення архітектури LMS-системи почалася з глибокого аналізу вимог до продуктивності та масштабованості. Використання мікросервісного підходу дозволило розділити систему на логічно незалежні модулі, кожен з яких відповідає за певну функціональність. Це особливо важливо для навчальних платформ, де можлива значна дисперсія навантаження на різні компоненти.

Фронтенд-архітектура базується на принципах Feature-Sliced Design, що дозволяє ефективно організувати код для великих додатків. Кожен функціональний модуль (наприклад, автентифікація або робота з курсами) має власну структуру з чітко визначеними межами.

Сучасна LMS-система потребує ретельно продуманої архітектури, яка поєднує гнучкість, продуктивність та зручність розробки. Пропонована система побудована на принципах мікросервісної архітектури з чітким розділенням відповідальностей між компонентами.

Основні архітектурні рішення (рисунок 2.1):

– Фронтенд: React 19 з використанням функціональних компонентів і хуків.

– Стан менеджмент: Redux Toolkit + RTK Query для ефективної роботи з API.

– Стилізація: Tailwind CSS + ShadCN UI компоненти.

– Маршрутизація: React Router 6 з lazy loading.

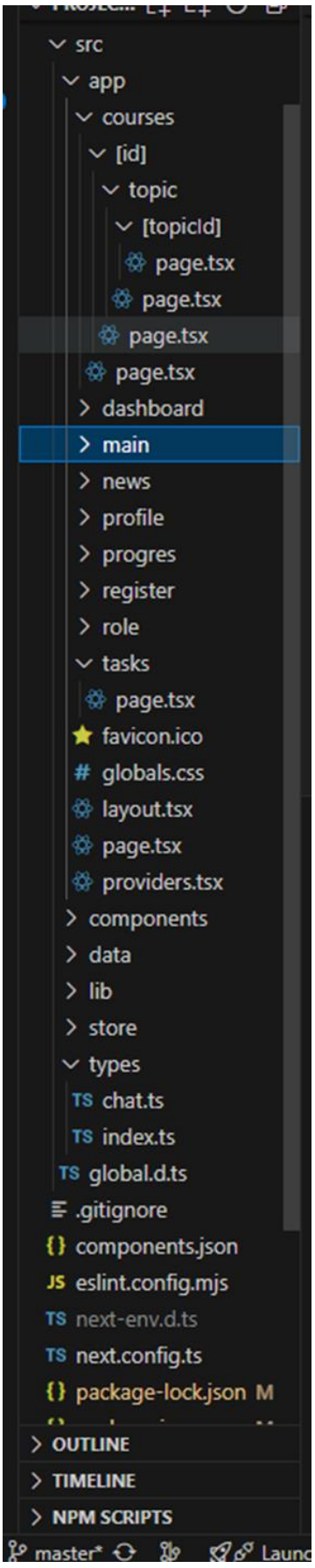

Рисунок 2.1 – Схема архітектури

Архітектура системи побудована на основі модульного підходу, що дозволяє спростити масштабування, супровід та розподіл відповідальностей між окремими частинами застосунку. У таблиці 2.1 наведено основні модулі системи, їх функціональне призначення та використані технології.

Таблиця 2.1 – Взаємодія основних модулів системи

Модуль	Відповідальність	Технології
Auth	Автентифікація та авторизація	JWT, React Hook Form
Courses	Управління курсами	RTK Query, Suspense
Testing	Система тестування	Zustand, Canvas API
Admin	Адміністрування	TanStack Table, Chart.js

2.2 Діаграма станів додатку

Управління станом у такій складній системі вимагає ретельного планування. Ми використали комбінацію Redux Toolkit для глобального стану та Zustand для локальних станів компонентів. Особливості запропонованої реалізації:

– Нормалізація даних: курси та користувачі зберігаються у вигляді словників для швидкого доступу.

– Оптимістичні оновлення: інтерфейс реагує на дії користувача без очікування відповіді сервера.

– Персистентність: критичні дані (токен, налаштування) зберігаються в localStorage.

Управління станом у системі LMS реалізовано з використанням Redux Toolkit для глобального стану (користувач, курси, тести) та Zustand для локального стану (UI-компоненти, модальні вікна тощо). На рисунку 2.2 представлено діаграму станів, що ілюструє логіку зміни станів додатку в залежності від дій користувача.

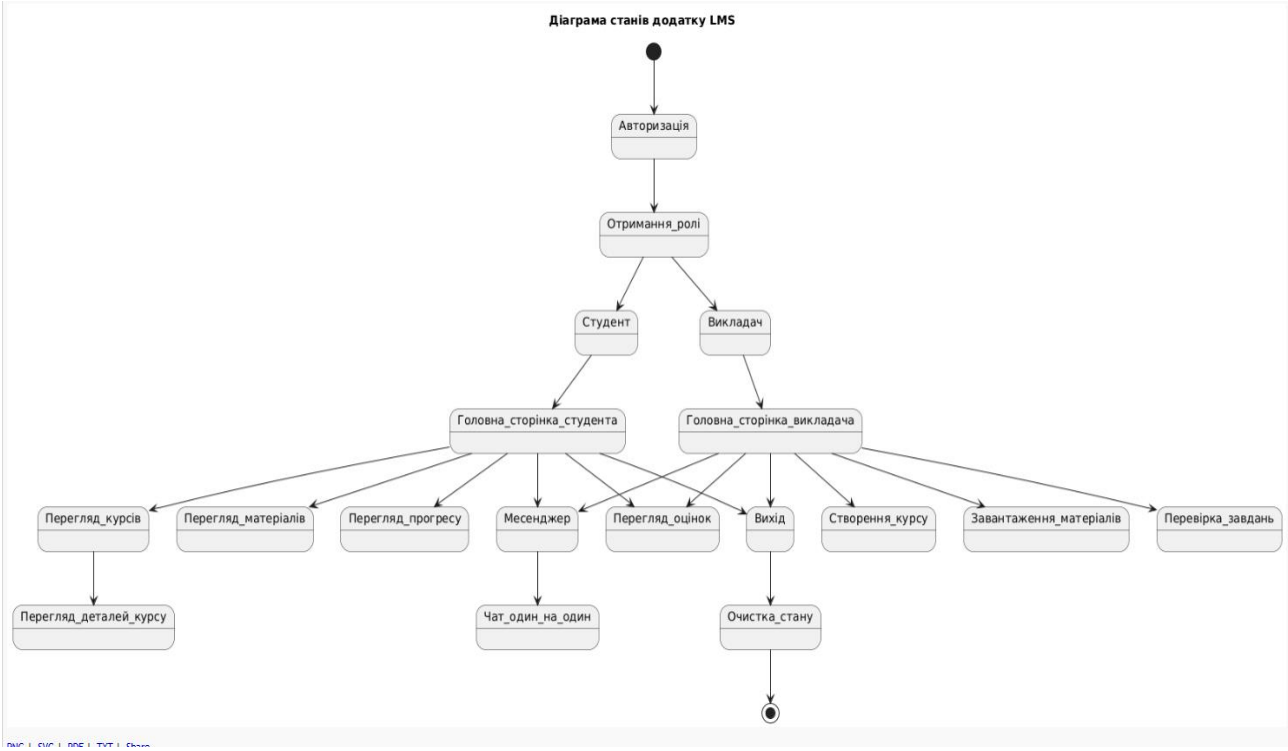

Рисунок 2.2 – Діаграма станів додатку

2.3 Схема бази даних (бекенд частина)

Незважаючи на те, що кваліфікаційна робота зосереджена на фронтенді, важливо розуміти, як має працювати повноцінна система. Орієнтовно, бекенд має бути побудований на Node.js (Express/NestJS) з PostgreSQL (ER-діаграма на рисунку 2.3). Основні таблиці:

– users - інформація про користувачів
– courses - навчальні курси
– materials - навчальні матеріали
– tests - тести та питання
– user_progress - прогрес студентів

Для реалізації функціональності системи було розроблено реляційну структуру бази даних, що охоплює основні логічні сутності та їх взаємозв’язки. У таблиці 2.2 наведено перелік основних сутностей бази даних, їхні ключові поля та зв’язки між ними.

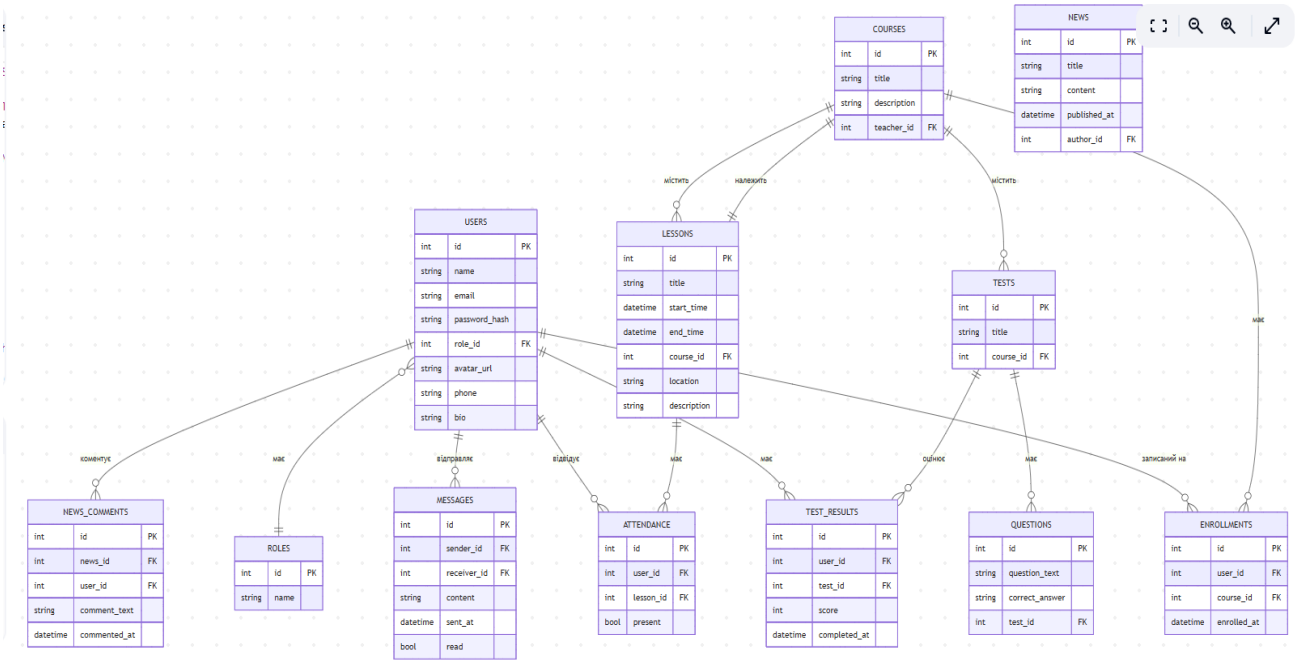

Рисунок 2.3 – ER-діаграма

Таблиця 2.2 – Основні сутності бази даних системи

Сутність	Поля	Зв'язки
users	id, email, role, password_hash	courses, tests
courses	id, title, description	materials, tests
materials	id, type, content, course_id	courses
tests	id, questions, course_id	courses, attempts
attempts	id, user_id, test_id, score	users, tests

Приклад REST API ендпоінтів:

– GET /api/courses - отримання списку курсів

– POST /api/tests/submit - відправка результату тесту

– PATCH /api/users/profile - оновлення профілю

2.4 Реалізація інтерфейсів

Студентський інтерфейс у розробленій LMS-системі побудований із фокусом на зручність, доступність і швидку навігацію. Користувачі мають змогу швидко ознайомитися з актуальною інформацією про поточні курси, терміни

здачі завдань, особистий прогрес та розклад занять. Інтерфейс реалізований таким чином, щоб студент міг максимально ефективно організувати власний навчальний процес, не витрачаючи зайвого часу на пошук необхідної інформації.

У межах кожного курсу студенти можуть переглядати навчальні матеріали, які завантажив викладач, а також самостійно завантажувати файли-відповіді на завдання. Усі дії, пов’язані з подачею робіт, зручно структуровані, що дозволяє легко відслідковувати статус кожного завдання. Для комунікації передбачено вбудований месенджер, який забезпечує обмін повідомленнями між студентом та викладачем або групою, що дозволяє оперативно отримувати зворотній зв’язок та роз’яснення. Крім того, в студентському інтерфейсі реалізовано доступ до загальних оголошень та новин курсу, що сприяє інформованості та залученості учня до навчального процесу.

Головна сторінка студента включає:

– Поточні курси – перелік курсів, на які студент записаний.

– Останні завдання – список нових або невиконаних завдань із вказанням дедлайнів.

– Успішність – таблиця або віджет із оцінками за кожен курс і завдання.

– Розклад пар – зручний перегляд дат та тем занять у вигляді таблиці або календаря.

Матеріали курсу (рисунок 2.4):

– Завантаження файлів – студент має змогу переглядати завантажені викладачем навчальні матеріали (документи, архіви, тощо).

– Скачування матеріалів – усі файли доступні для завантаження для подальшого використання офлайн.

– Коментарі до матеріалів – опціональна можливість залишати коментарі або питання до завантажених файлів.

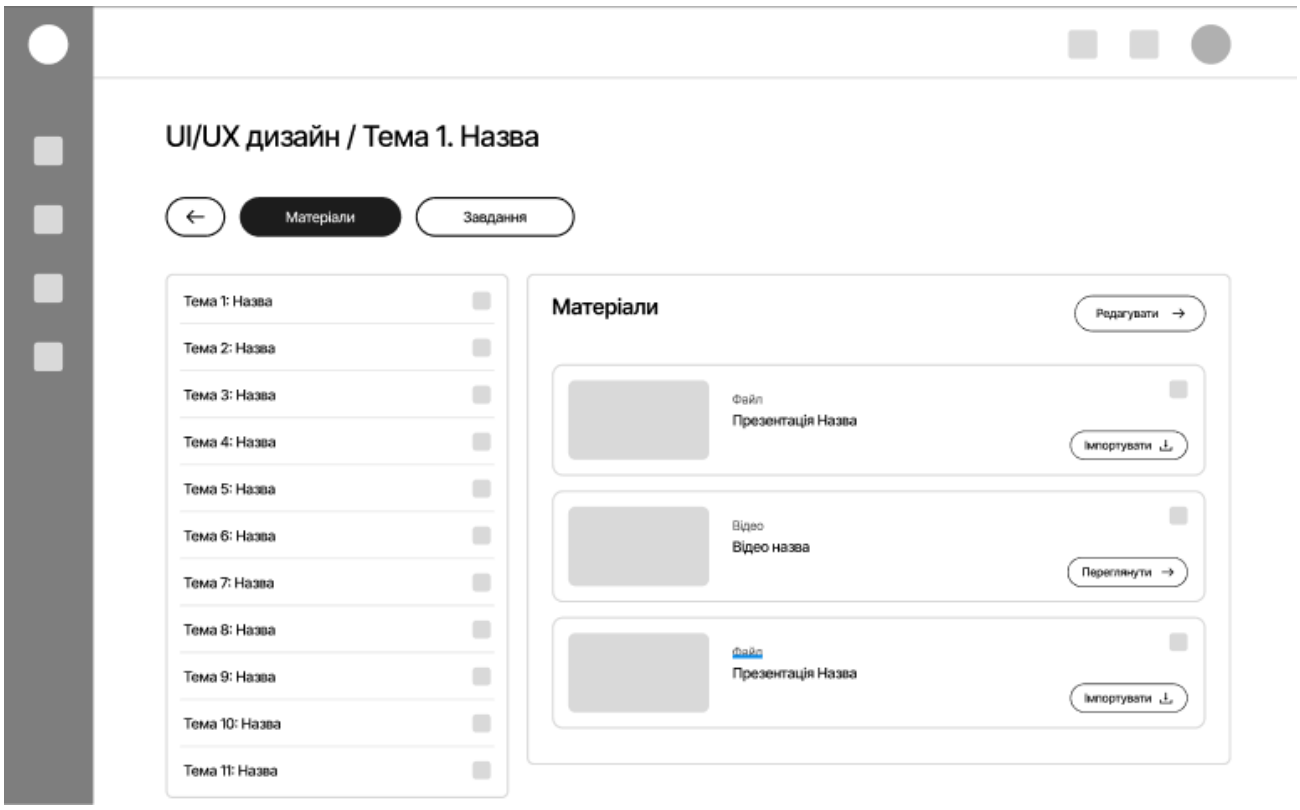

Рисунок 2.4 – Матеріали для студентського інтерфейсу

Завдання (рисунок 2.5):

– Передача робіт – студенти можуть завантажити власні файли-відповіді на певні завдання курсу.

– Список зданих завдань – перегляд усіх завантажених робіт з вказанням дати та статусу перевірки.

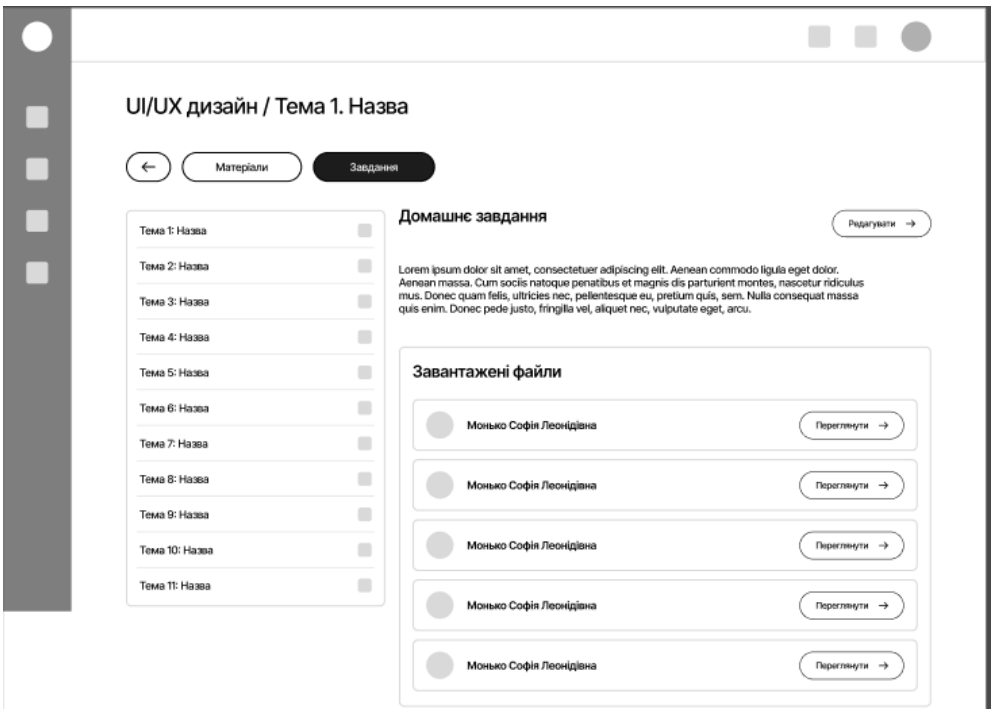

Рисунок 2.5 – Завдання для студентського інтерфейсу

Месенджер (рисунок 2.6): особисті повідомлення (чат між студентами та викладачами); історія листування (збереження попередніх діалогів у зручному вигляді).

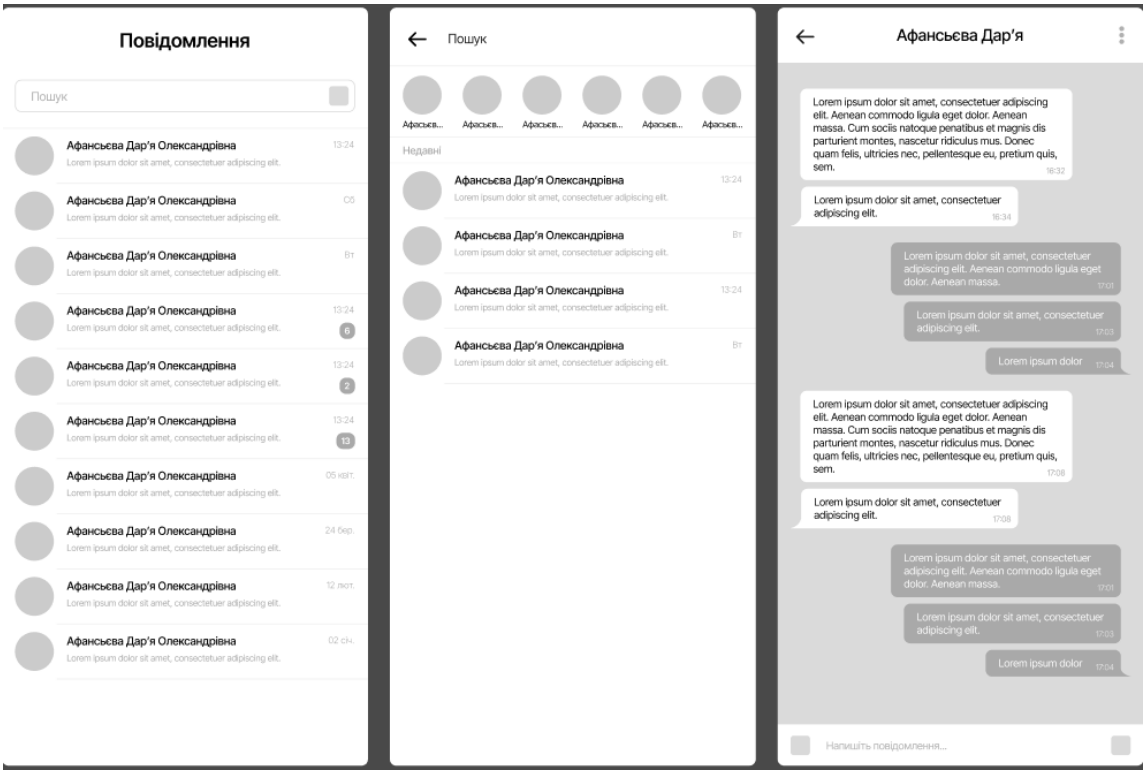

Рисунок 2.6 – Месенджер

Новини (рисунок 2.7) передбачають функціональність системних оголошень: – інформацію про важливі події, зміни в розкладі або оновлення курсу.

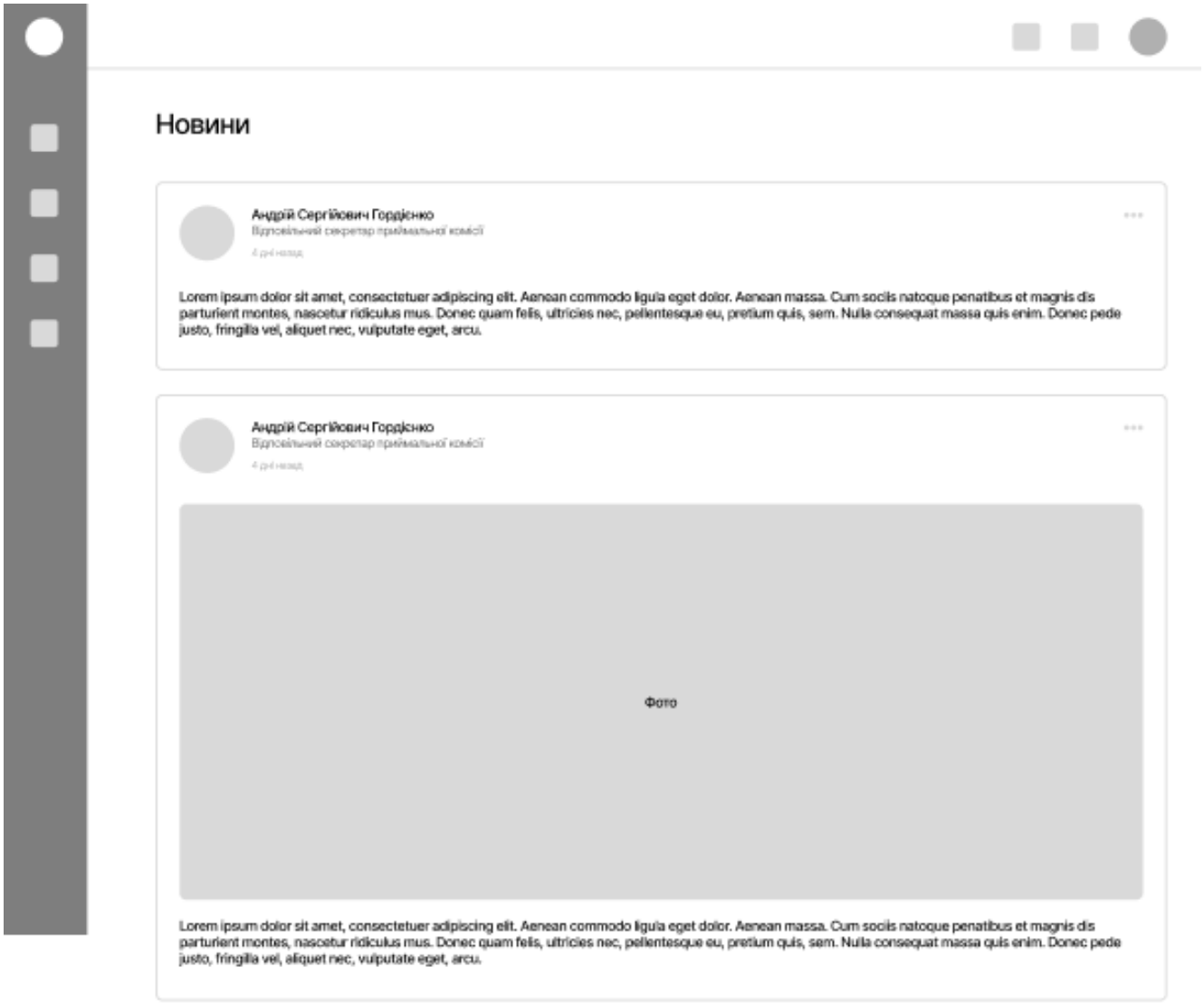

Рисунок 2.7 – Новини для студентського інтерфейсу

2.5 Система повідомлень

Система повідомлень у платформі LMS реалізована як повноцінний інструмент для ефективної комунікації між усіма учасниками навчального процесу. Основна мета – забезпечити студентів і викладачів зручним способом обговорення завдань, розкладу, прогресу та організаційних моментів.

Месенджер підтримує як особисті повідомлення, так і групові чати в межах курсів, дозволяючи швидко обмінюватися інформацією, файлами й уточненнями.

Інтерфейс чату адаптований під мобільні та десктопні пристрої. Для комфорту користувачів реалізовано систему сповіщень: наприклад, нагадування про дедлайни або нові повідомлення в чаті курсу. Передача повідомлень відбувається через WebSocket, що забезпечує реальний час, а дані зберігаються в базі, що дозволяє повертатися до історії листування в будь-який момент.

Функціональність месенджера легко розширюється, зокрема в майбутньому можлива інтеграція з мобільними push-сповіщеннями чи email-розсилками. Особисті повідомлення:

– Обмін текстовими повідомленнями
– Перегляд історії листування

Сповіщення:

– Автоматичні повідомлення про нові завдання чи оновлення
– Нагадування про дедлайни завантаження файлів
– Можливість увімкнення/вимкнення окремих типів сповіщень

Технічна реалізація:

– WebSocket-з’єднання для миттєвої доставки повідомлень
– Зберігання історії в базі даних із кешуванням на клієнті
– Пошук по чатах (реалізовано через фільтрацію локальних даних)
– Базове шифрування повідомлень при передачі (HTTPS + можливість розширення до end-to-end)

З метою забезпечення високої швидкодії, масштабованості та зручності у використанні системи, було обрано відповідний набір сучасних технологій. Детальний опис обраного технологічного стеку наведено в таблиці 2.3.

Таблиця 2.3 – Обґрунтування вибору технологічного стеку для реалізації LMS

Технологія	Використання	Особливості	Чому саме вона
React	Побудова UI, компонентна структура	Швидкість, декларативність	Зручна побудова динамічних інтерфейсів
Redux Toolkit	Управління глобальним станом	Мінімум коду, DevTools	Оптимізоване рішення для складних станів
ShadCN UI	UI-компоненти для чату, форм, кнопок	Гарний дизайн, Tailwind інтеграція	Сучасний вигляд без зайвого CSS
Tailwind CSS	Стилізація компонентів	Швидка розробка, mobile-first	Легко адаптується під різні екрани
WebSocket (через socket.io)	Реалізація реального часу	Миттєвий обмін, стабільність	Не потребує ручного оновлення сторінки
Node.js + Express	Backend API (якщо писали вручну) або mock	Обробка запитів	Простота, сумісність з фронтендом
PostgreSQL / SQLite	Зберігання повідомлень, історії чату	Реляційна БД	Стабільна робота з даними
HTTPS + JWT	Захист передачі даних, автентифікація	Безпечне з’єднання	Гарантія конфіденційності

2.6 Система повідомлень

У запропонованій LMS-платформі взаємодія між клієнтською та серверною частинами реалізована через RESTful API. Такий підхід дозволяє централізовано керувати даними, забезпечує масштабованість і спрощує налагодження окремих частин системи. Ендпоінти логічно згруповані за сутностями: курси, користувачі, повідомлення, файли, розклад, що робить API передбачуваним і зручним у використанні.

Аутентифікація побудована на основі JWT. Кожен запит до захищених маршрутів супроводжується токеном, який підтверджує роль і права користувача. У випадку закінчення сесії використовується refresh-токен для безшовного

продовження роботи. Це дозволяє зберігати високий рівень безпеки без шкоди для зручності користувача.

На стороні клієнта API-запити обробляються через власний API-клієнт з єдиною обробкою помилок. Це забезпечує єдину логіку виводу повідомлень, захоплення збоїв і можливість централізованого журналювання. Крім того, всі відповіді API мають уніфікований формат, що суттєво полегшує роботу фронтенду, зокрема з валідацією, відображенням даних і оновленням користувацького інтерфейсу.

У майбутньому архітектура API дозволяє легко перейти до підтримки GraphQL або gRPC при розширенні функціональності або підключенні мобільних застосунків.

Для забезпечення стабільної роботи фронтенду потрібна архітектура API:

RESTful API:

– Чітко структуровані ендпоінти для взаємодії з ресурсами: курсами, файлами, розкладом, повідомленнями, користувачами.

– Використання стандартних HTTP-методів: GET, POST, PUT, DELETE.

– Відповіді у форматі JSON з уніфікованою структурою (status, message, data).

Система авторизації:

– JWT (JSON Web Token) для захищеного доступу до приватних ендпоінтів.

– Refresh-токени для продовження сесії без повторного входу.

– Розмежування прав доступу через ролі (student, teacher, admin), перевірка на бекенді.

Обробка помилок:

– Уніфіковані коди помилок (401, 403, 422, 500)

– Логування на сервері всіх критичних збоїв

– Відображення користувачеві дружніх повідомлень про помилки

Приклади роботи з API представлено на лістингу 2.1.

Лістинг 2.1 – Програмний каркас для взаємодії з API

```
// Отримання розкладу студента
const fetchSchedule = async () => {
  try {
    const response = await api.get('/schedule');
    return response.data;
  } catch (error) {
    handleApiError(error);
  }
};

// Надсилання домашнього завдання
const uploadHomework = async (formData) => {
  try {
    const response = await api.post('/homework/upload', formData, {
      headers: { 'Content-Type': 'multipart/form-data' },
    });
    return response.data;
  } catch (error) {
    handleApiError(error);
  }
};
```

З метою обґрунтованого вибору протоколу взаємодії між клієнтською та серверною частинами системи було проведено порівняльний аналіз популярних підходів до побудови API. У таблиці 2.4 представлено порівняння REST API, GraphQL та gRPC за ключовими критеріями.

Таблиця 2.4 – Порівняльна характеристика підходів до реалізації API

Критерій	REST API (використовується)	GraphQL	gRPC
Формат запитів	HTTP + JSON	HTTP + JSON	HTTP/2 + Protocol Buffers
Гнучкість запитів	Статичні ендпоінти	Вибір полів у запиті	Висока, контрактна система
Простота реалізації	Простий для реалізації	Потрібен сервер GraphQL	Складна реалізація
Продуктивність	Добра	Залежить від реалізації	Висока (бінарний протокол)
Документація	Swagger, OpenAPI	Playground, GraphiQL	Proto файли (IDL)
Кешування	Добре підтримується	Потребує доп. реалізації	Складніше кешувати

Висновок до розділу 2

Розроблена архітектура клієнтської частини LMS-платформи вже сьогодні відповідає ключовим вимогам сучасних навчальних рішень, орієнтованих на масштабованість, зручність користування та гнучкість. Фронтенд-платформа побудована за принципами модульності, що забезпечує простоту підтримки та подальшого розвитку. Реалізовані інтерфейси для трьох основних типів користувачів – студентів, викладачів і адміністраторів – дозволяють ефективно взаємодіяти з освітнім контентом, розкладом, повідомленнями, а також забезпечують швидкий відгук і комфортну навігацію.

Особливу увагу приділено реалізації системи стану, яка забезпечує узгодженість даних у складних взаємозалежних модулях: від журналу оцінювання до модуля сповіщень. Продумана архітектура компонування інтерфейсів дозволяє легко додавати нові розділи або оновлювати вже існуючі без порушення загальної логіки додатку. Завдяки використанню сучасних інструментів, таких як React, Redux Toolkit і TailwindCSS, було досягнуто високого рівня продуктивності, адаптивності й естетичності.

На цьому етапі завершено фронтенд-розробку та створено концепцію для бекенд-частини системи, яка має відповідати таким характеристикам: чітка RESTful API-структура, захищена система авторизації з JWT і ролями доступу, оптимізована взаємодія з базою даних та підтримка реального часу (наприклад, у модулі повідомлень). Хоч серверна логіка ще перебуває у стадії розробки, її концептуальна модель уже дає змогу уявити повноцінне функціонування системи як завершеного продукту.

З огляду на архітектурні рішення, що вже реалізовані, платформа має всі передумови для масштабування та подальшого розвитку. У перспективі можливе впровадження інтеграції з зовнішніми освітніми сервісами, створення мобільного застосунку, додавання модулів аналітики навчального прогресу та AI-асистента. Таким чином, створена система може слугувати надійною основою

для побудови повноцінного сучасного LMS-рішення, здатного працювати у реальних умовах навчальних закладів різного рівня.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Загальні положення реалізації

У цьому розділі детально описано процес реалізації навчальної системи «mIGHT», що є результатом практичного етапу кваліфікаційної роботи. Основна мета розробки – створити інтуїтивно зрозумілу, функціональну та гнучку платформу для підтримки навчального процесу, адаптовану під різні типи користувачів. У ході розроблення акцент було зроблено на якісну реалізацію клієнтської частини з використанням сучасного фронтенд-стека, що забезпечує масштабованість і зручність розширення системи у майбутньому.

Фронтенд-архітектура реалізована на основі React з використанням функціональних компонентів і хуків. Управління станом здійснюється через Redux Toolkit у поєднанні з RTK Query для асинхронної взаємодії з бекендом. Для зберігання критичних даних, таких як токени, використано LocalStorage. Стилізація реалізована за допомогою Tailwind CSS і компонентів ShadCN UI, що дозволило досягти сучасного й уніфікованого вигляду інтерфейсу. Інтерфейс розроблявся адаптивним, з урахуванням можливості роботи на різних типах пристроїв – як десктопних, так і мобільних.

Хоча бекенд-частина наразі перебуває в процесі проектування, уся фронтендна логіка побудована таким чином, щоб надалі забезпечити легку інтеграцію з API за допомогою стандартизованих запитів і чітко визначених структур відповідей.

3.2 Реалізація інтерфейсів користувачів

Інтерфейс системи адаптовано під дві основні ролі: студент та викладач. Для кожного типу користувача реалізовано унікальний інтерфейс із відповідною навігацією та функціональністю.

Для студента після авторизації відкривається головна сторінка, що містить список доступних курсів, можливість переглядати матеріали, проходити тести, переглядати оцінки, слідкувати за власним прогресом та користуватися системою новин. Інтерфейс є простим і зрозумілим, адаптованим до потреб студента та зручним як на ПК, так і на мобільному пристрої.

Для викладача реалізовано інтерфейс, що дозволяє керувати навчальними матеріалами, створювати завдання та тести, переглядати та оцінювати виконані студентами роботи, а також стежити за їхнім прогресом. Викладач має доступ до розширеного функціоналу, який дозволяє більш гнучко управляти навчальним процесом.

Важливою частиною реалізації є інтерфейс користувача, що адаптується відповідно до ролі. На рисунках 3.1 та 3.2 продемонстровано інтерфейси студентів і викладачів відповідно.

Рисунок 3.1 – Інтерфейс студента

Рисунок 3.2 – Інтерфейс викладача

3.3 Реалізовані сторінки системи

У межах реалізації системи було створено такі ключові сторінки для кожного типу користувача:

– Головна сторінка – відображає загальну інформацію, список доступних курсів та швидкий доступ до основних розділів системи.

– Сторінка завдань – містить список активних завдань, опис кожного завдання, дедлайни та можливість завантажити рішення.

– Матеріали курсу – сторінка для ознайомлення з навчальними матеріалами (тексти, презентації, посилання тощо). Студент може переглядати матеріали, а викладач – додавати та редагувати їх.

– Прогрес студента – розділ для перегляду успішності користувача. Студенти бачать свою динаміку, а викладачі – успішність усіх студентів.

– Профіль користувача – персональна сторінка, де можна змінити особисті дані, перевірити загальну інформацію про користувача, переглянути активність у системі.

– Сторінка новин – показує актуальні оголошення та новини системи, що можуть бути додані адміністрацією або викладачем.

– Авторизація та реєстрація – окремі сторінки для входу в систему та створення нового облікового запису з вибором ролі.

Відповідні сторінки на рисунках 3.3-3.11. Головна сторінка студента відображає загальну інформацію про навчальний процес, включаючи список доступних курсів, останні новини та швидкий доступ до основних розділів системи, таких як завдання, матеріали та прогрес. Інтерфейс адаптований для зручного перегляду на різних пристроях (рисунок 3.3).

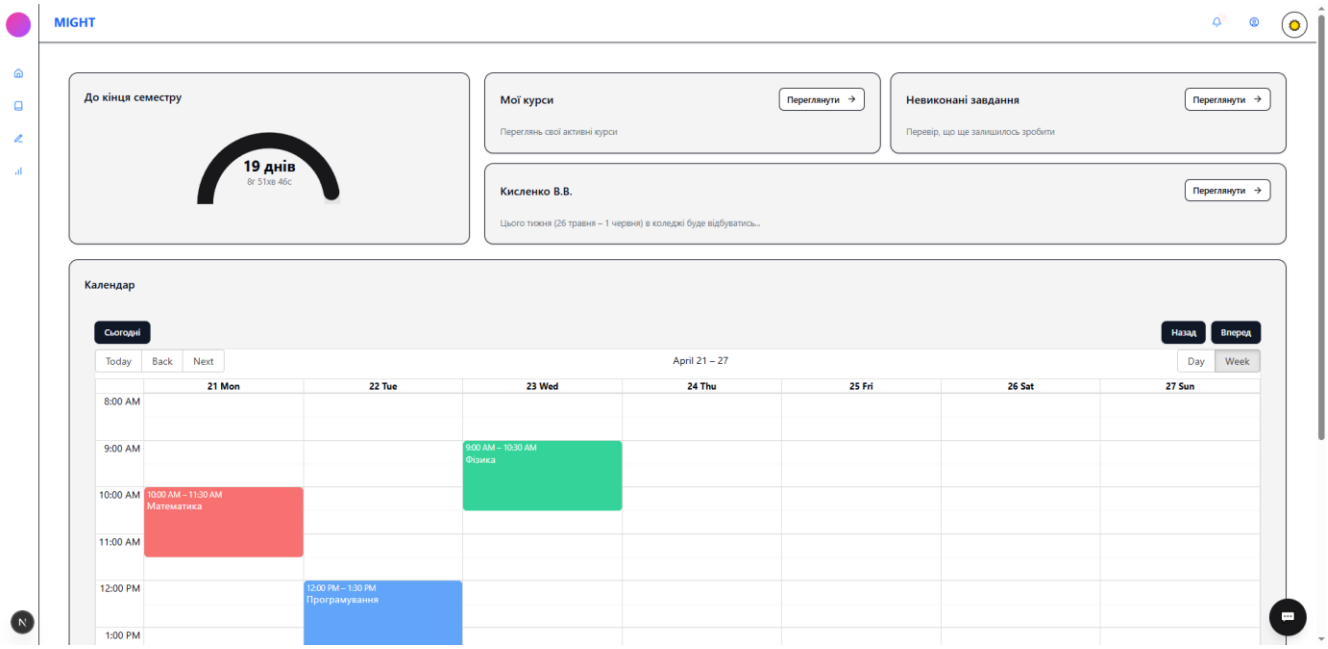

Рисунок 3.3 – Головна сторінка студента

Інтерфейс викладача на головній сторінці містить панель управління курсами, огляд поточного стану завдань студентів, а також швидкий доступ до створення матеріалів і оцінювання. Відображаються також повідомлення від адміністрації та актуальні новини (рисунок 3.4).

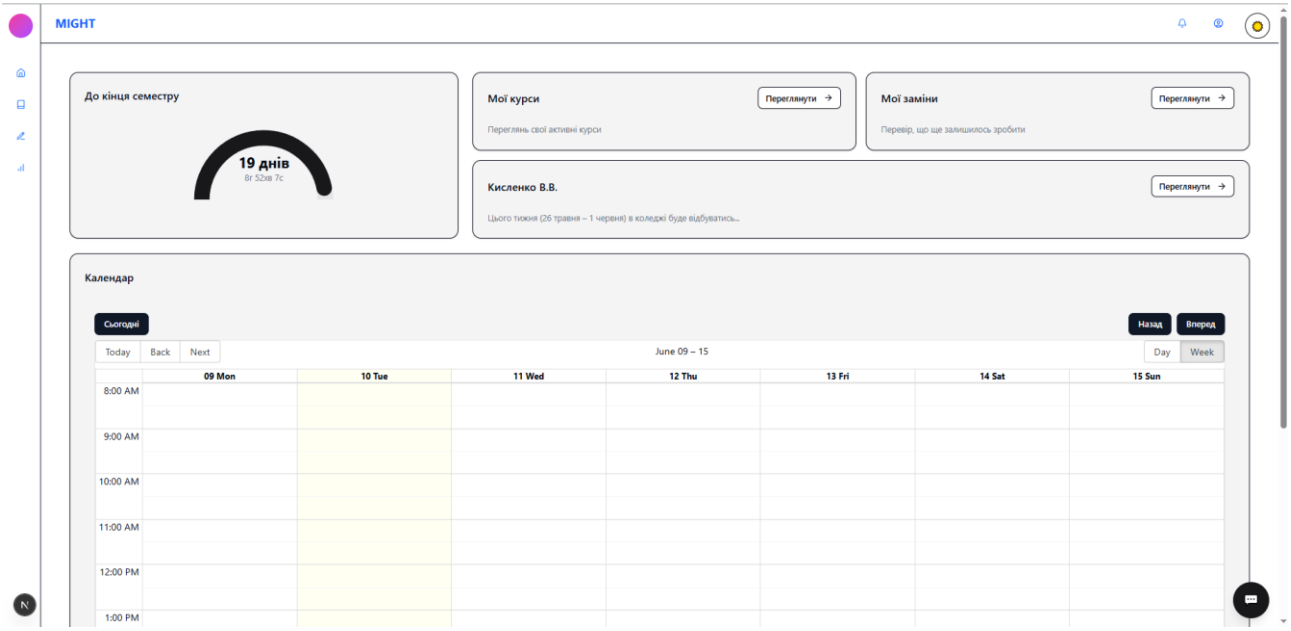

Рисунок 3.4 – Головна сторінка викладача

На сторінці завдань студент бачить список активних завдань із описом, датами дедлайнів та можливістю завантажити власні рішення. Викладачі мають можливість створювати нові завдання, редагувати наявні та переглядати надіслані роботи студентів (рисунок 3.5).

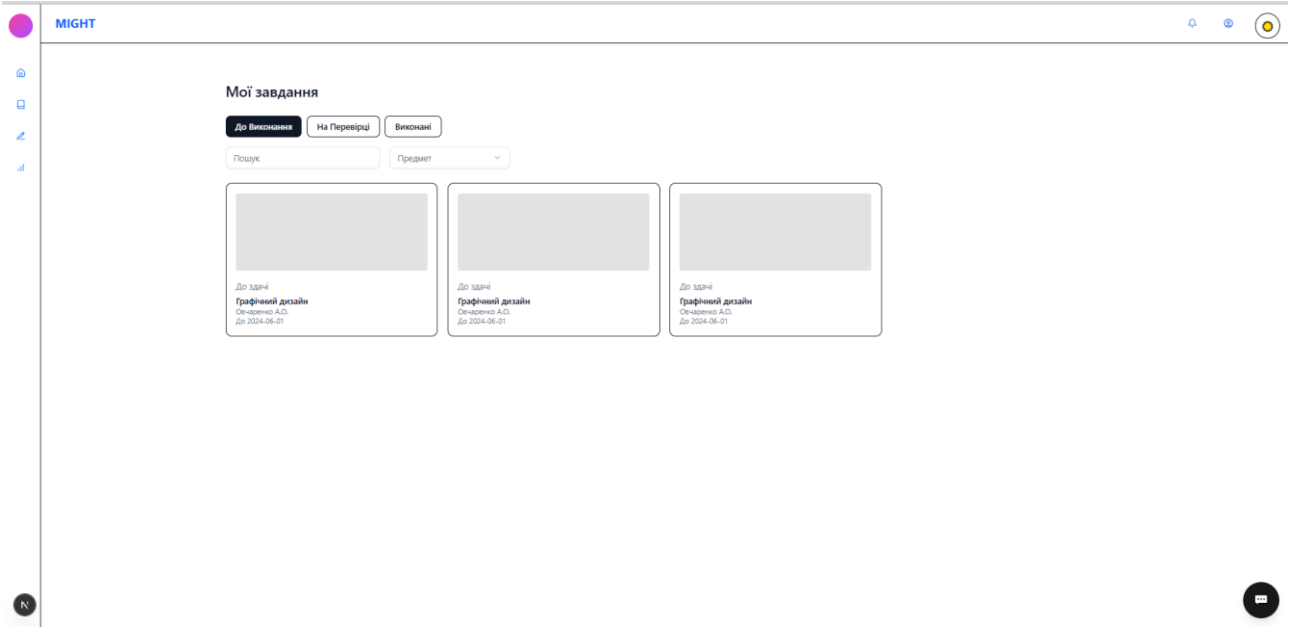

Рисунок 3.5 – Сторінка завдань

Ця сторінка містить навчальні матеріали: тексти, презентації, посилання на зовнішні ресурси. Студенти можуть переглядати та завантажувати матеріали (рисунок 3.6).

Рисунок 3.6 – Сторінка матеріалів курсу для студента

Ця сторінка містить навчальні матеріали — тексти, презентації, посилання на зовнішні ресурси. Викладачі можуть додавати нові, редагувати або видаляти вже існуючі (рисунок 3.7).

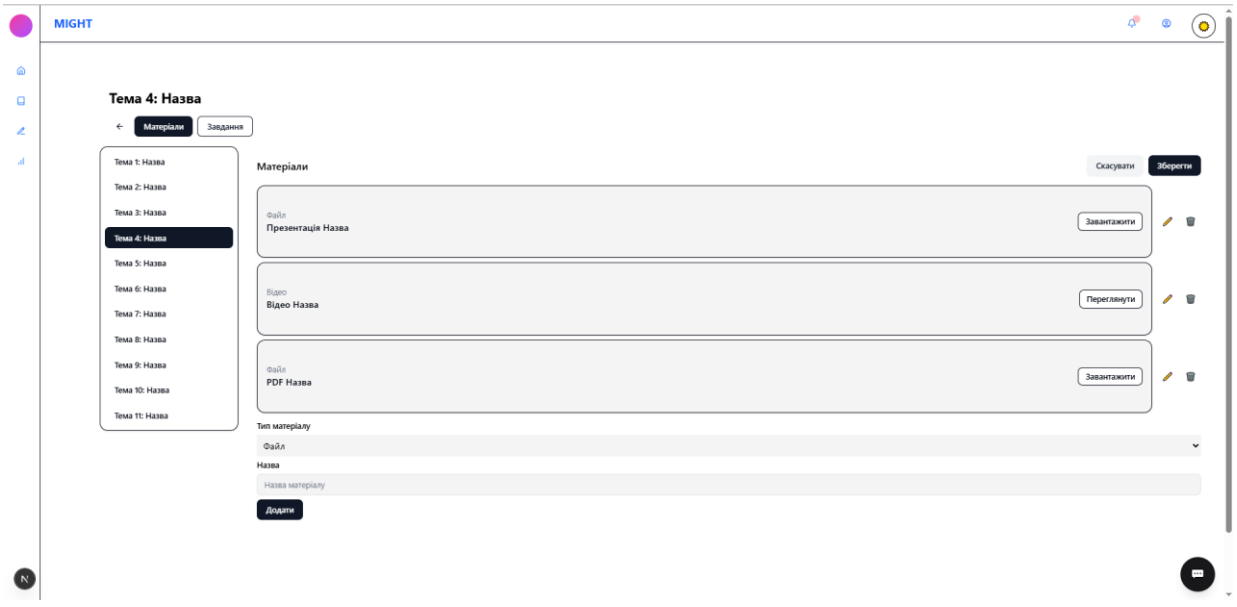

Рисунок 3.7 – Сторінка матеріалів курсу для викладача

Розділ прогресу відображає успішність користувача. Студенти бачать свої оцінки, статистику виконання завдань та динаміку навчання. Викладачі мають доступ до успішності усіх студентів курсу з можливістю аналізу та експорту даних (рисунок 3.8).

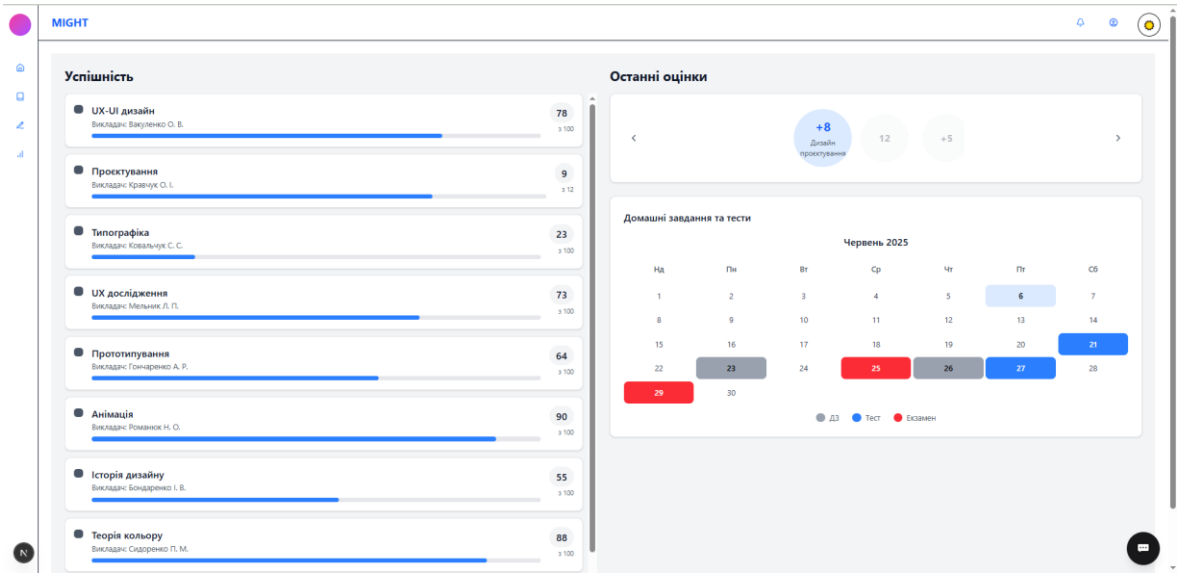

Рисунок 3.8 – Сторінка прогресу студента

Особистий профіль дозволяє користувачу переглядати та змінювати особисті дані, налаштовувати параметри облікового запису та переглядати історію активності в системі (рисунок 3.9).

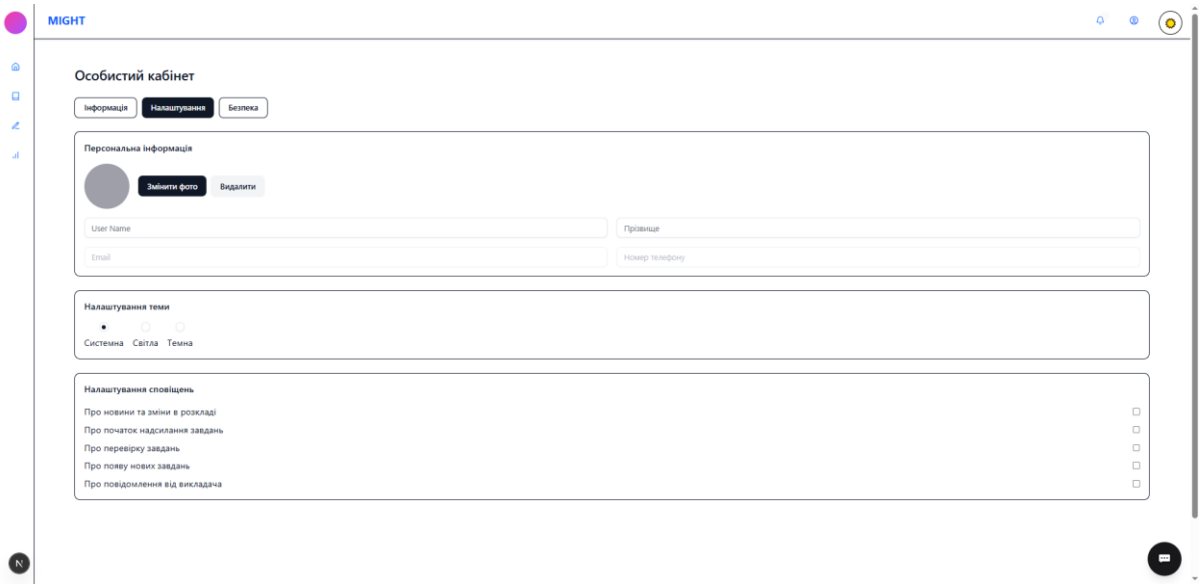

Рисунок 3.9 – Профіль користувача

Ця сторінка відображає актуальні оголошення та новини системи, які можуть бути додані адміністрацією або викладачами. Користувачі отримують важливу інформацію про зміни в розкладі, нові можливості чи події (Рисунок 3.10).

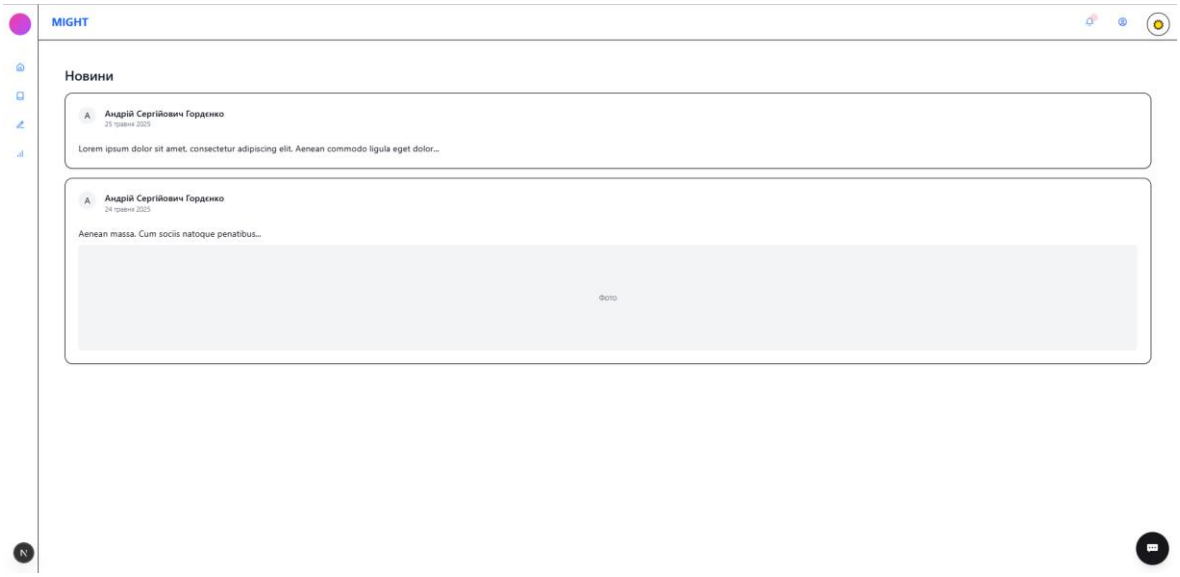

Рисунок 3.10 – Сторінка новин

Окрема сторінка входу в систему, де користувач вводить свої облікові дані для доступу до LMS. Включає опції відновлення паролю та вибору ролі під час реєстрації (рисунок 3.11).

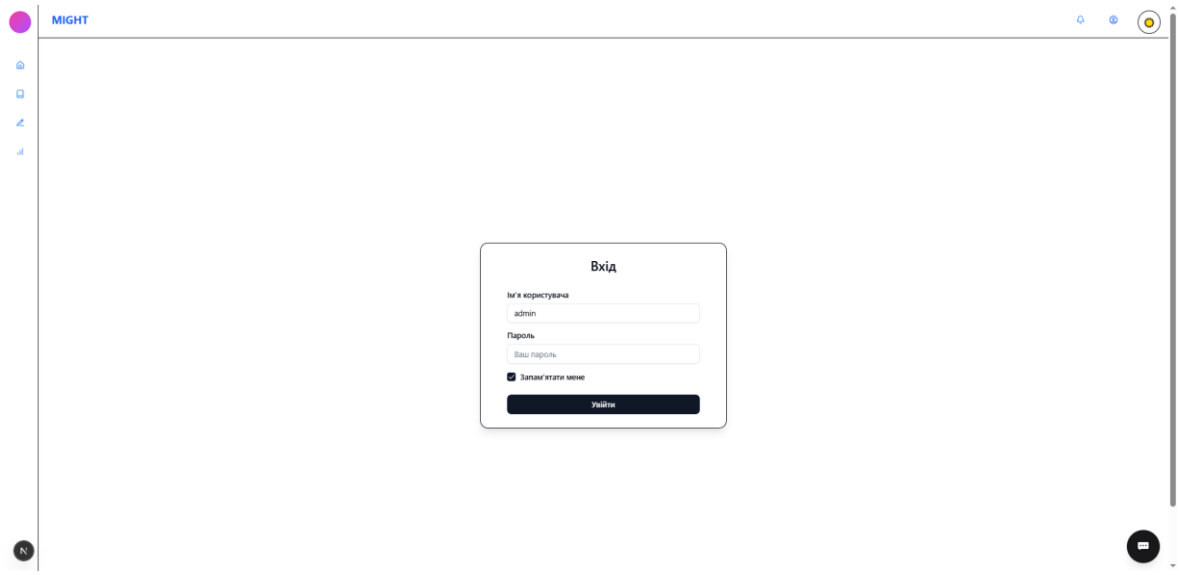

Рисунок 3.11 – Сторінка авторизації

Кожна сторінка була протестована з точки зору UX-дизайну шляхом проведення юзабіліті-тестування, яке включало:

– Спостереження за поведінкою користувачів під час взаємодії з інтерфейсом (moderated testing).

– Аналіз кліків і навігації у Figma або під час live-тестування.

– Оцінку інтуїтивності розміщення елементів, відповідності очікуванням та зручності доступу до функціоналу.

– Збір зворотного зв’язку від цільових користувачів (студентів/викладачів) у формі коротких опитувань.

– Тестування на адаптивність та доступність у різних браузерах і на мобільних пристроях.

3.4 Порівняння функціоналу студентів і викладачів

Для забезпечення зручного розмежування прав доступу та функціональних можливостей користувачів системи, було визначено ролі з відповідними діями. У таблиці 3.1 наведено розподіл функціоналу між ролями студента та викладача.

Таблиця 3.1 – Розподіл функціональних можливостей між користувачами системи

Функціонал	Студент	Викладач
Перегляд курсів	Так	Так
Створення завдань	Ні	Так
Завантаження матеріалів	Ні	Так
Проходження тестів	Так	Ні
Перевірка завдань	Ні	Так
Перегляд оцінок	Так	Так
Додавання новин	Ні	Так

3.5 Тестування системи

Для забезпечення високої якості продукту та комфортного користувацького досвіду було проведено комплексне тестування системи, що

охоплювало як функціональну перевірку, так і оцінку зручності використання (юзабіліті), адаптивності інтерфейсу, а також кросбраузерну сумісність.

Особлива увага була приділена юзабіліті – зручності та інтуїтивності роботи користувачів із системою. Проводилися сесії тестування за участі реальних користувачів різних ролей (студентів, викладачів, адміністраторів), які виконували типові сценарії: реєстрація, вхід, перегляд курсів, додавання завдань, перевірка оцінок. На основі їхніх відгуків були внесені коригування до структури меню, розташування кнопок і підказок, що суттєво покращило користувацький досвід.

Для оцінки ефективності взаємодії користувачів з інтерфейсом було складено таблицю результатів UX-тестування (наведено у таблиці 3.2):

Таблиця 3.2 – Результати UX-тестування типових сценаріїв

Сценарій	Роль	Час виконання	Коментар користувача
Реєстрація	Студент	00:25 с	«Все зрозуміло, кнопка видно чітко»
Додавання завдання	Викладач	00:38 с	«Можна трохи підняти кнопку вгору»
Перевірка оцінок	Студент	00:17 с	«Зручно, добре видно оцінки»
Зміна тема	Будь-яка	миттєво	«Темна тема дуже приємна для очей»
Доставлення повідомлень	Будь-яка	миттєво	«Спілкування в реальному часі це зручно»

UI-тестування з Playwright

Для автоматизації тестування інтерфейсу було використано Playwright – сучасний фреймворк для наскрізного (end-to-end) тестування, що підтримує

емуляцію користувацьких сценаріїв у різних браузерах. Playwright дозволив створити набір автоматичних тестів, які перевіряють:

- коректність відображення основних UI-компонентів,
- працездатність форм і навігації,
- реакцію на зміни теми (світла/темна),
- відсутність помилок при взаємодії користувача з додатком.

Код прикладу автоматизованого тесту входу за допомогою Playwright наведено в лістингу 3.1 і результат відображено на рисунку 3.12

Лістинінг 3.1 - Приклад автоматизованого тесту входу користувача

projectg > tests > TS login.spec.ts > test("Логін під адміністратором працює") callback

```
import { test, expect } from "@playwright/test";

test("Логін під адміністратором працює", async ({ page }) => {
  await page.addInitScript(() => {
    localStorage.setItem("persist:root", JSON.stringify({
      auth: JSON.stringify({
        username: "admin",
        role: "admin",
        isAuthenticated: true
      }),
    }));
  });

  await page.goto("http://localhost:3000/dashboard");

  await page.waitForURL("**/role/admin");

  await page.screenshot({ path: "tests/screenshots/admin-login-success.png", fullPage: true });

  expect(page.url()).toContain("/role/admin");
});
```

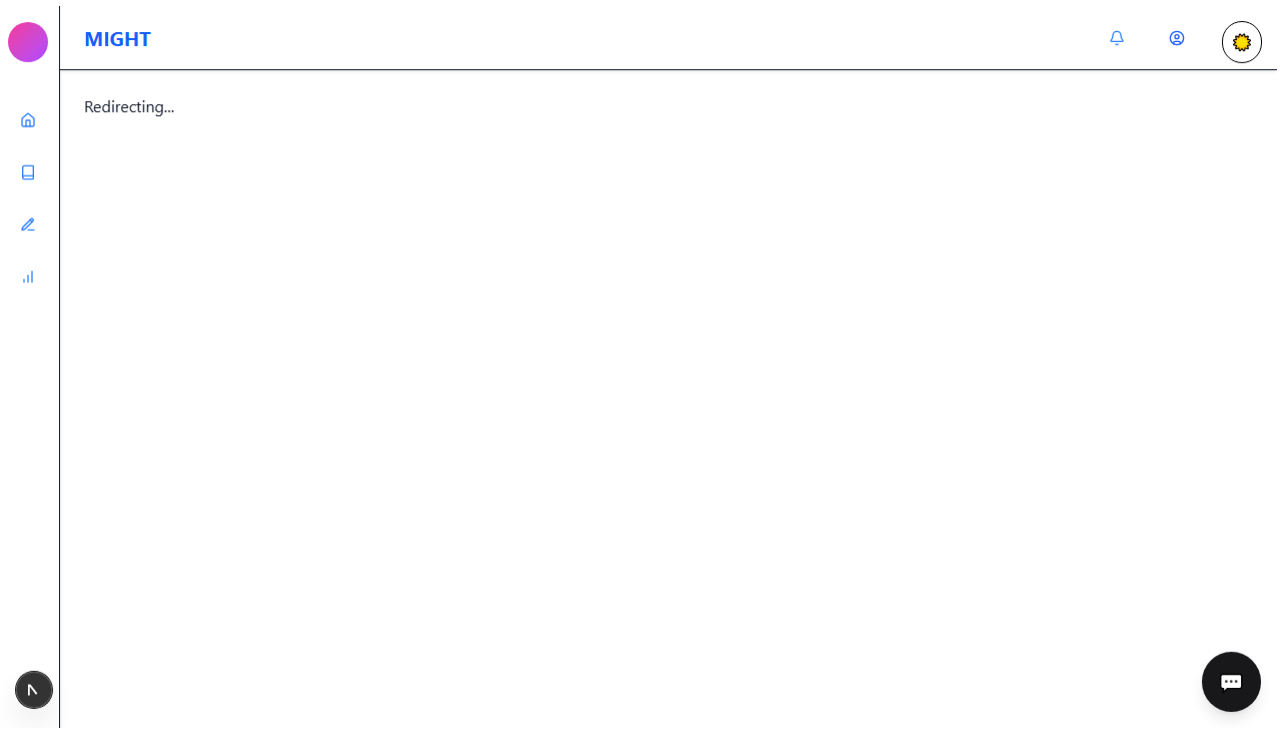

Рисунок 3.12 - Автоматизоване UI-тестування у Playwright

Перевірка кросбраузерності та адаптивності системи здійснювалась у найпоширеніших браузерах (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari), що гарантує стабільну роботу та однаковий вигляд інтерфейсу незалежно від платформи користувача. Особливу увагу було приділено підтримці адаптивного дизайну – інтерфейс коректно відображається на різних розмірах екранів, включно зі смартфонами, планшетами та десктопами. (На рисунку 3.13 зображено мобільну весрію сайту)

Під час тестування було підтверджено, що всі елементи UI автоматично підлаштовуються під розмір екрану, а зміна орієнтації (портретна/ландшафтна) не призводить до порушень верстки або функціональності.

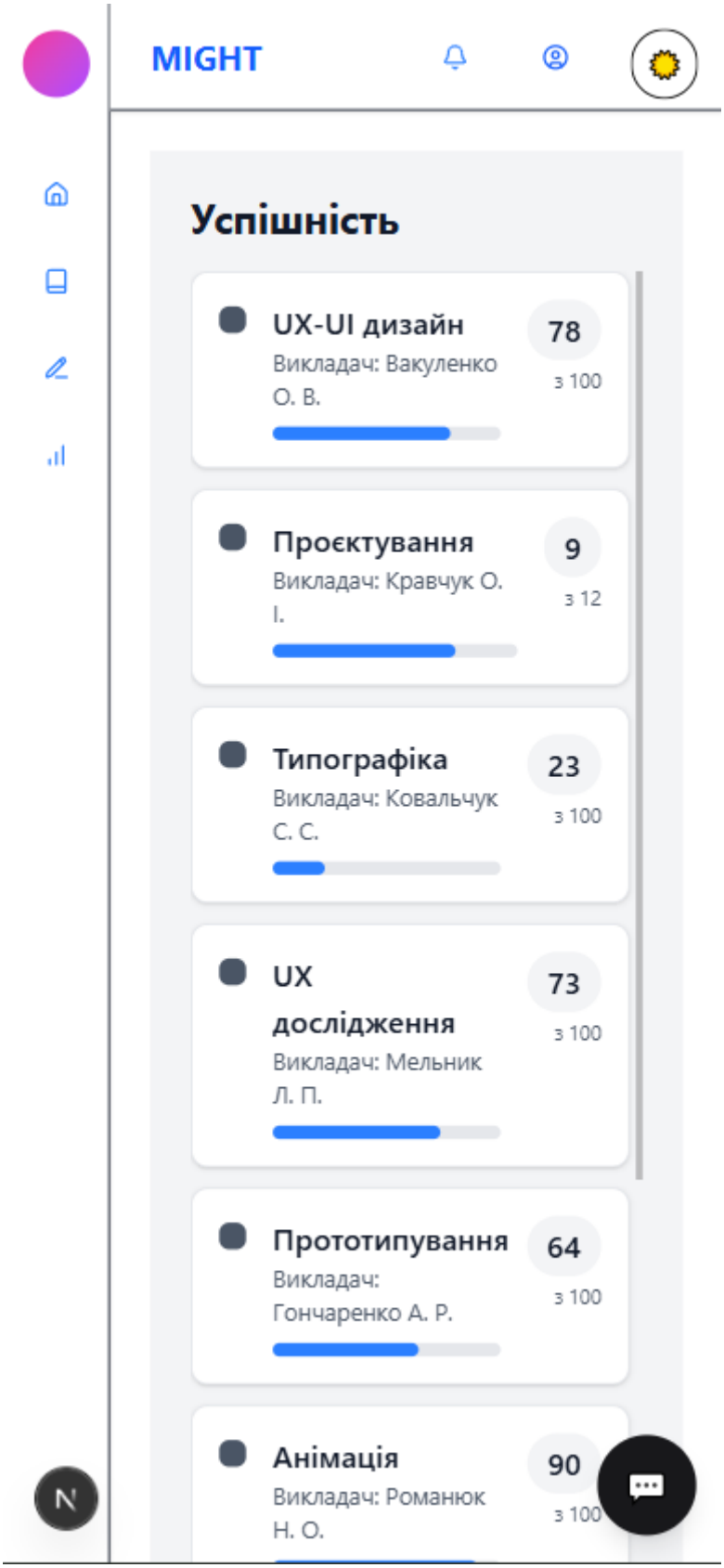

Рисунок 3.13 – Адаптивне відображення LMS на мобільному пристрої

Інтерфейс також підтримує перемикання між світлою та темною темами, що дозволяє користувачам обирати комфортний режим перегляду відповідно до

умов освітлення та власних уподобань. Тестування підтвердило, що при зміні теми всі компоненти інтерфейсу коректно оновлюють кольорову схему, зберігаючи читабельність, контрастність та цілісність дизайну.(На рисунках 3.14 та 3.15 продемонстровано відмінності у відображенні сторінки в світлій та темній темах оформлення.)

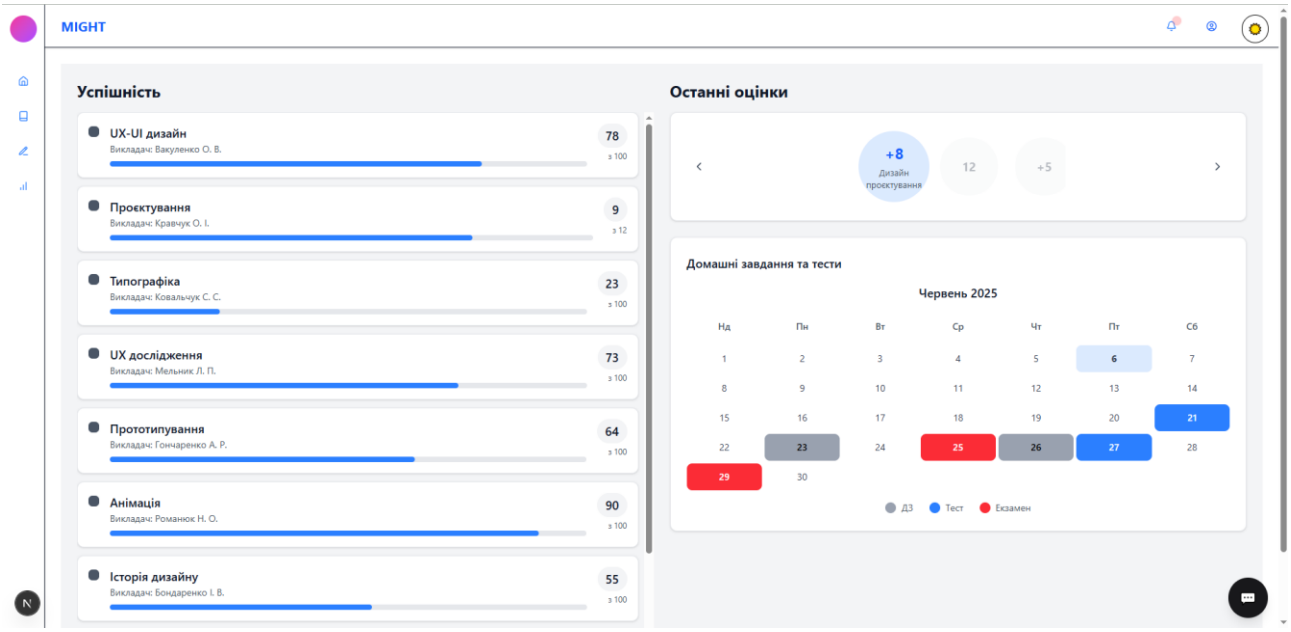

Рисунок 3.14 – Перемикання між темами (light)

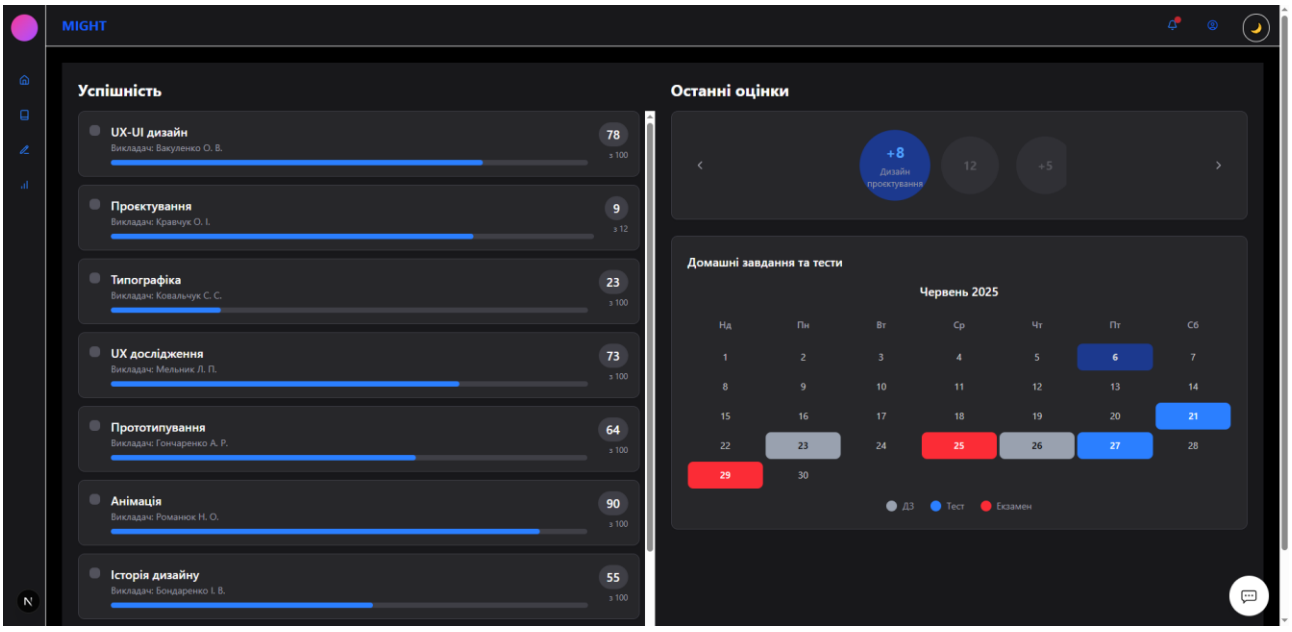

Рисунок 3.15 – Перемикання між темами (dark)

Висновок до розділу 3

У цьому розділі було висвітлено практичну реалізацію програмної системи «mIGHT». Розроблений інтерфейс враховує потреби як студентів, так і викладачів, забезпечуючи зручний доступ до навчальних матеріалів, завдань, результатів навчання та загальної комунікації в рамках навчального курсу.

Окрема увага була приділена технічному стеку та архітектурі клієнтської частини, що забезпечує легку підтримку та можливість подальшого розвитку проєкту. Було представлено ключові сторінки, що охоплюють основну логіку взаємодії користувачів із системою, а також наведено таблицю порівняння функціональних можливостей для різних типів користувачів.

Додатково, розділ охоплює тестування, що забезпечує впевненість у правильності реалізації функцій. Проведені перевірки підтвердили стабільність інтерфейсу та його відповідність заданим вимогам. У подальшому передбачається впровадження автоматизованого тестування, що дозволить забезпечити надійність проєкту на етапі експлуатації та при розширенні функціональності.

ВИСНОВКИ

За результатами виконання кваліфікаційної роботи було реалізовано повний цикл розробки мобільного застосунку системи управління навчанням (LMS) для студентів, викладачів та адміністрації навчального закладу. Робота охоплювала всі етапи – від збору вимог і аналізу аналогів до безпосередньої розробки інтерфейсу, реалізації функціоналу та тестування продукту.

На початковому етапі було проведено дослідження сучасних інструментів та технологій, які найкраще підходять для створення кросплатформного застосунку. В результаті аналізу було обрано стек технологій, що базується на React Native, Redux Toolkit для керування станом, та сучасних бібліотеках UI-компонентів, що забезпечили швидкість розробки та зручність у підтримці.

У ході реалізації проєкту було створено набір ключових екранів, які покривають основні сценарії взаємодії користувача із системою – перегляд курсів, завдань, оцінок, таймер семестру, а також базову навігацію між ролями користувачів. Увага приділялась логіці розподілу доступу між трьома основними ролями: студент, викладач і адміністратор. Було враховано аспекти UI/UX, щоб забезпечити інтуїтивну та комфортну взаємодію з додатком.

Через відсутність готового бекенду на момент реалізації розробка була зосереджена на клієнтській частині – побудові інтерфейсів та взаємодії між компонентами. Незважаючи на це, архітектура додатку передбачає легке масштабування та інтеграцію з API в майбутньому.

У завершальній фазі проєкту було проведено ручне тестування додатку на реальних пристроях з різними характеристиками. Основні компоненти функціонували стабільно, виявлені недоліки були оперативно виправлені.

Отже, поставлена мета кваліфікаційної роботи була досягнута. Створений мобільний додаток LMS демонструє функціональну реалізацію ключових елементів електронної системи навчання, має зручний та сучасний інтерфейс і

може бути використаний як основа для подальшої розробки повноцінної багатофункціональної платформи дистанційного навчання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. React Documentation. URL: https://react.dev (дата звернення: 09 червня 2025 р.)

2. Redux Toolkit Documentation. URL: https://redux-toolkit.js.org (дата звернення: 09 червня 2025 р.)

3. RTK Query API Reference. URL: https://redux-toolkit.js.org/rtk-query/overview (дата звернення: 09 червня 2025 р.)

4. Tailwind CSS Documentation. URL: https://tailwindcss.com/docs (дата звернення: 09 червня 2025 р.)

5. ShadCN UI. URL: https://ui.shadcn.com (дата звернення: 09 червня 2025 р.)

6. MDN Web Docs (Mozilla Developer Network). URL: https://developer.mozilla.org (дата звернення: 09 червня 2025 р.)

7. JavaScript.info - Довідник з сучасного JavaScript. URL: https://javascript.info (дата звернення: 09 червня 2025 р.)

8. Глушаков С. О. Web-програмування. HTML, CSS, JavaScript. - Київ: «Міра», 2020. – 312 с.

9. Литвиненко В. В. Системи управління навчанням: архітектура, функціональність, безпека. - Харків: ХНУРЕ, 2021. – 140 с.

10. Радченко С. П. Побудова веб-додатків на основі React. - Харків: ФОП Сидоренко, 2022. – 98 с.

11. State of JavaScript 2023 - Аналітичний звіт щодо використання JS-технологій. URL: https://stateofjs.com (дата звернення: 09 червня 2025 р.)

12. Stack Overflow Developer Survey 2023 URL: https://survey.stackoverflow.co/2023 (дата звернення: 09 червня 2025 р.)

13. Visual Studio Code - офіційна документація редактора коду. URL: https://code.visualstudio.com (дата звернення: 09 червня 2025 р.)

14. Основи програмної інженерії / під ред. В. С. Прохоренка. -Київ: Наукова думка, 2019. - 344 с.

15. Next.js Documentation.. URL: https://nextjs.org/docs (дата звернення: 09 червня 2025 р.)

16. ISO/IEC 9126:2001 - Міжнародний стандарт якості програмного забезпечення. URL: https://iso.org/standard/22749.html (дата звернення: 09 червня 2025 р.)

17. Базовий курс з програмної інженерії на сайті Coursera. URL: https://www.coursera.org (дата звернення: 09 червня 2025 р.)

18. Prettier - автоматичне форматування коду. URL: https://prettier.io (дата звернення: 09 червня 2025 р.)

19. ESLint - інструмент для перевірки якості JavaScript-коду. URL: https://eslint.org (дата звернення: 09 червня 2025 р.)

20. ISO/IEC/IEEE 12207 - Системи та програмна інженерія. Процеси життєвого циклу програмного забезпечення. URL: https://www.iso.org/standard/43447.html (дата звернення: 09 червня 2025 р.)

21. ISO/IEC/IEEE 12207 - Системи та програмна інженерія. Процеси життєвого циклу програмного забезпечення. URL: https://www.iso.org/standard/43447.html (дата звернення: 09 червня 2025 р.)

22. Stackoverflow - Веб-фреймворки та технології. URL: https://survey.stackoverflow.co/2023/#section-most-popular-technologies-web-frameworks-and-technologies (дата звернення: 09 червня 2025 р.)

23. Nature - Соціальне навчання. URL: https://www.nature.com/articles/s41467-020-16856-8 (дата звернення: 09 червня 2025 р.)

24. Mermaid Live Editor - онлайн-редактор діаграм. URL: https://mermaid.live/ (дата звернення: 09 червня 2025 р.)

ДОДАТОК А – ПОСИЛАННЯ НА ПЕРВИННИЙ КОД ДОДАТКА «mIGHT»

Первинний код розробленого програмного засобу для обчислення метрик програмного коду, а також проведення статистичного аналізу результатів розміщено в git-репозиторії за адресою:
https://github.com/B1toks/projectG