Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Система управління навчальним контентом»

08-54.БКР.006.00.000 ПЗ

Виконав: студент 4 курсу, групи 1СП-21б
спеціальності 123 — «Комп'ютерна інженерія»
освітня програма — «Системне програмування»
Гресько М. Р.

Керівник: к.т.н., доц. кафедри ОТ
Снігур А. В.

Рецензент: д.т.н., професор. кафедри МБІС
Яремчук Ю.Є.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.
«23» 06 2025 р.

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень бакалавр
Галузь знань — 12 — Інформаційні технології
Спеціальність — 123 — Комп'ютерна інженерія
Освітньо-професійна програма — Системне програмування

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.

«21» березня 2025 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Греську Максиму Романовичу

1 Тема роботи: «Система управління навчальним контентом», керівник роботи Снігур А. В. к.т.н., доц., доцент кафедри ОТ затверджені наказом вищого навчального закладу від «20» березня 2025 року № 97

2 Термін подання студентом роботи 13.05.2025 р.

3 Вхідні дані до роботи: функціональні вимоги до застосунку, облік часу навчання за допомогою таймера, візуалізація результатів на діаграмі, календар подій, система облікових записів; середовище розробки — WebStorm; база даних — Room Base.

4 Зміст розрахунково-пояснювальної записки: вступ, огляд і аналіз теоретичних основ, вибір необхідних інструментів та технологій розробки, розробка система управління навчальним контентом, висновки, список використаних джерел, додатки.

5 Перелік ілюстративного матеріалу: панель управління курсами користувача, інтерфейс курсу для викладача.

6 Консультанти роботи наведені у табл. 1.

Таблиця 1 — Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийн
Вступ,	к.т.н., доц., доцент каф. ОТ Снігур А. В	21.03.2025	13.05.2025
Нормоконтроль	ас. каф. ОТ Швець С. І.	14.05.2025	30.05.2025

7. Дата видачі завдання 21.03.2025 р.

8. Календарний план наведений в табл. 2.

Таблиця 2 — Календарний план

№ етапу	Назва етапу	Термін виконання		Приміт
		Початок	закінчення	
1	Вибір, узгодження та затвердження теми БКР	21.03.2025	22.03.2025	
2	Аналіз літературних джерел. Попередня розробка основних розділів	25.03.2025	28.03.2025	
3	Розробка технічного завдання (ТЗ)	29.03.2025	31.03.2025	
4	Аналіз вирішення поставленої задачі на даному етапі	31.03.2025	02.04.2025	
5	Розробка архітектури додатку та основних компонентів	04.04.2020	08.04.2025	
6	Програмна реалізація функціоналу	10.04.2020	16.04.2025	
7	Розробка інтерфейсу користувача	21.04.2020	23.04.2025	
8	Тестування та оптимізація додатку	25.04.2020	28.04.2025	
9	Аналіз виконаної роботи, підготовка висновків	08.05.2020	11.05.2025	
10	Попередній захист БКР	12.05.2025	13.05.2025	

Студент М.Р. Гресько

Керівник роботи А.В. Снігур

АНОТАЦІЯ

УДК 004.77:004.738.5

Гресько М. Р. Система управління навчальним контентом. Бакалаврська кваліфікаційна робота зі спеціальності 123 — Комп'ютерна інженерія, освітня програма — Системне програмування. Вінниця: ВНТУ, 2025, 92 с.

На укр. мові. Бібліогр.: 32 назв; рис.: 39; табл.: 3.

У даній бакалаврській кваліфікаційній роботі розглянуто процес розробки веб-орієнтованої системи управління навчальним контентом (LCMS), яка дозволяє створювати, зберігати, структурувати та поширювати освітні матеріали.

У першому розділі проведено аналіз актуальності теми, сучасних освітніх платформ і технологій, які використовуються для керування навчальним процесом. Оцінено функціональні можливості систем Google Classroom, Moodle, Blackboard Learn, TalentLMS.

У другому розділі розглянуто обґрунтування вибору технологій для реалізації проєкту, включаючи HTML, CSS, JavaScript, React, Firebase, Material UI, WebStorm. Визначено основні вимоги до системи, її функціональні блоки, базу даних та архітектурні рішення.

У третьому розділі описано процес розробки системи, включно з реалізацією інтерфейсу користувача, організацією зберігання контенту та управління ролями користувачів. Особливу увагу приділено забезпеченню доступності, безпеки та зручності використання системи.

Під час тестування перевірено працездатність основного функціоналу, зібрано зворотний зв'язок щодо UX/UI. Результати розробки можуть бути застосовані у навчальних закладах або інтегровані з іншими освітніми сервісами.

Ключові слова: освітня платформа, LCMS, React, Firebase, управління курсами.

ABSTRACT

Hresko M.R. Learning Content Management System. Bachelor's thesis in specialty 123 — Computer Engineering, educational program — System Programming. Vinnytsia: VNTU, 2025, 92 p.

In Ukrainian. Bibliogr.: 32 titles; ill.: 39; tabl.: 3.

This bachelor's thesis presents the development process of a web-based Learning Content Management System (LCMS), enabling the creation, storage, structuring, and distribution of educational materials.

The first chapter analyzes the relevance of the topic and explores modern educational platforms and technologies used for managing the learning process. Functional capabilities of systems such as Google Classroom, Moodle, Blackboard Learn, and TalentLMS are evaluated.

The second chapter focuses on the selection and justification of technologies for implementation, including HTML, CSS, JavaScript, React, Firebase, Material UI, and WebStorm. The core requirements, system components, database, and architecture are defined.

The third chapter describes the development process, including user interface design, content storage, and user role management. Special attention is paid to accessibility, security, and usability.

During testing, the core functionality was verified, and user feedback on UX/UI was collected. The developed system can be used in educational institutions or integrated with other educational services.

Keywords: educational platform, LCMS, React, Firebase, course management.

ЗМІСТ

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій освіта дедалі більше переходить у цифрове середовище, що зумовлює потребу в ефективних засобах організації, збереження та поширення навчального контенту. Традиційні методи подачі навчального матеріалу вже не повністю відповідають вимогам гнучкості, доступності й інтерактивності, які висуває сучасний освітній процес. Саме тому системи управління навчальним контентом (Learning Content Management Systems, LCMS) набувають особливої **актуальності**.

Такі системи дозволяють централізовано керувати освітніми матеріалами, здійснювати їх персоналізацію для окремих користувачів, надавати доступ до контенту в реальному часі та забезпечувати контроль засвоєння знань. Із розвитком веб-технологій і хмарних сервісів створення інтуїтивно зрозумілих, масштабованих і безпечних LCMS стало важливим напрямом у сфері освітніх IT-рішень.

Отже, створення сучасної, зручної та адаптивної системи управління навчальним контентом є актуальним завданням, що сприятиме вдосконаленню освітнього процесу як у закладах освіти, так і в корпоративному навчанні.

Метою роботи є розробка програмного забезпечення у вигляді системи управління навчальним контентом, яке дозволяє створювати, зберігати, структурувати та поширювати навчальні матеріали з використанням сучасних веб-технологій.

Задачі дослідження:

— провести аналіз існуючих систем управління навчальним контентом, визначити їх переваги та недоліки;

— дослідити вимоги до ефективної LCMS у контексті дистанційного й змішаного навчання;

— обґрунтувати вибір архітектурних рішень, мов програмування, бібліотек і сервісів для реалізації системи;

— розробити структуру бази даних для зберігання навчальних матеріалів, користувачів і статистики їхньої активності;

— створити веб-інтерфейс для взаємодії з контентом, управління курсами, користувачами та звітністю;

— реалізувати функціонал реєстрації, авторизації, управління ролями користувачів та обліку прогресу навчання;

— протестувати працездатність і зручність використання створеної системи в умовах, наближених до реального використання.

Об’єкт дослідження — інформаційна система для організації, управління та поширення навчального контенту.

Предметом дослідження є методи та засоби створення веб-орієнтованих систем управління навчальним контентом із використанням сучасних фреймворків, хмарних сервісів і технологій зберігання та обробки даних.

Новизна отриманих результатів полягає у тому, що запропоновано інноваційні підходи до управління навчальним контентом та інтерактивну взаємодію між учасниками освітнього процесу. Рішення може бути застосоване в навчальних закладах різного рівня, для корпоративного навчання або в рамках неформальної освіти. Гнучка архітектура системи дозволяє масштабувати її відповідно до потреб організації.

Апробація результатів бакалаврської роботи: зроблено доповідь на конференцію «Молодь в науці: дослідження, проблеми, перспективи» (МН-2025)[1].

Методи дослідження. У роботі використано методи системного аналізу, проєктування програмного забезпечення, об’єктно-орієнтованого програмування, моделювання користувацьких сценаріїв, а також експериментальні методи тестування інтерфейсів та функціональних можливостей. Для реалізації системи застосовано сучасні фреймворки (наприклад, React або Vue), бази даних (наприклад, Firestore або PostgreSQL), та хмарні інструменти (Firebase, GitHub тощо).

1 ОГЛЯД І АНАЛІЗ ТЕОРЕТИЧНИХ ОСНОВ

1.1 Визначення освітніх платформ

У сучасному цифровому суспільстві освітні платформи відіграють ключову роль у трансформації традиційного навчального процесу. У наукових джерелах та літературі, що стосується педагогкиі та інформаційним технологіям, поняття система управління навчанням (Learning Management System, LMS) або освітня платформа визначається як спеціалізоване програмне забезпечення або веборієнтоване середовище, яке забезпечує організацію, підтримку та контроль освітнього процесу за допомогою інформаційно-комунікаційних технологій.

Типова LMS є централізованим середовищем, яке дозволяє викладачам створювати, розміщувати, оновлювати та адаптувати навчальний контент, автоматизувати адміністративні функції (реєстрація, складання розкладу, облік присутності), а також відслідковувати прогрес та активність здобувачів освіти. Універсальність таких платформ забезпечує можливість їх використання як у повністю дистанційній, так і в змішаній формі навчання [2].

Серед основних функціональних характеристик LMS варто виокремити такі:

— автоматизація рутинних адміністративних завдань (керування користувачами, розподіл доступу, генерація звітів);

— організація та поширення навчальних матеріалів у різноманітних форматах (текст, відео, інтерактивні вправи);

— підтримка зворотного зв’язку, інструментів оцінювання та аналітики;

— можливість масштабування та адаптації контенту, зокрема повторне використання модулів у різних курсах;

— персоналізація освітньої траєкторії згідно з індивідуальними потребами студентів;

— забезпечення інтерактивності через форуми, чати, відеоконференції та групові завдання.

З точки зору викладача, освітня платформа дозволяє не лише публікувати навчальний контент, а й планувати заняття, задавати завдання, здійснювати контроль за виконанням робіт, забезпечувати оцінювання, а також аналізувати ефективність навчання. Для студентів LMS виступає у ролі інструмента доступу до матеріалів, засобу комунікації з викладачем і одногрупниками, джерела зворотного зв’язку, а також місця, де зберігається історія навчання.

Окремо слід підкреслити, що сучасні освітні платформи інтегрують можливості для використання мобільних пристроїв, підтримують міжнародні стандарти обміну навчальними модулями (наприклад, SCORM, xAPI), а також можуть бути інтегровані з іншими інформаційними системами — бібліотеками, базами даних, електронними щоденниками тощо.

Таким чином, LMS є потужним інструментом, що забезпечує ефективне управління освітнім процесом у цифровому середовищі та відкриває широкі можливості як для викладачів, так і для студентів.

1.2 Особливості застосування систем управління навчальним контентом

Системи управління навчанням (LMS) виконують важливу роль в організації освітнього процесу як у навчальних закладах, так і в корпоративному середовищі. Основним завданням LMS є підвищення ефективності планування, реалізації та моніторингу навчальних заходів. Завдяки централізації контенту та функціональним можливостям такі системи забезпечують низку важливих переваг, серед яких варто виокремити такі аспекти [3].

Оптимізація управління освітнім процесом. LMS дозволяє об’єднати всі навчальні матеріали, завдання, тестування та аналітику в єдиному середовищі, що значно спрощує доступ до освітнього контенту та адміністрування курсів. Централізація ресурсів зменшує витрати часу і фінансів, підвищує прозорість навчальних процесів та дає змогу швидко реагувати на зміни потреб користувачів.

Підвищення ефективності звітності та моніторингу успішності. Вбудовані інструменти аналітики та формування звітів дозволяють відстежувати прогрес

кожного учасника, фіксувати рівень виконання завдань, результати тестів, а також активність у курсах. Це дає змогу керівникам навчання ухвалювати обґрунтовані рішення на основі реальних даних.

Інтеграція формального та неформального навчання. LMS створює умови для поєднання класичних освітніх підходів із соціальними та асинхронними форматами навчання. Залучення інтерактивних елементів, форумів, чатів, відеоконференцій та групових проєктів підвищує мотивацію до навчання та сприяє розвитку навичок співпраці.

Розширення адміністративних можливостей. Адміністратори освітнього процесу можуть ефективніше контролювати розклад курсів, призначення викладачів, доступи користувачів, розподіл ресурсів, а також налаштовувати індивідуальні траєкторії навчання. Вони мають змогу бачити, які теми потребують додаткової уваги, і вчасно втручатися у навчальний процес.

Покращення якості та доступності навчального контенту. Завдяки LMS викладачі отримують зручні інструменти для розробки курсів, інтеграції мультимедійного контенту, налаштування тестів, оцінювання та гейміфікації. Водночас студенти можуть самостійно опановувати матеріал у зручному темпі, переглядати матеріали повторно, отримувати миттєвий зворотний зв’язок та рекомендації щодо покращення результатів.

Таким чином, LMS не лише автоматизує ключові аспекти організації освітнього процесу, а й сприяє підвищенню якості навчання, розвитку навичок самостійної роботи, індивідуалізації підходів та забезпеченню постійного контролю за навчальними досягненнями.

1.3 Розвиток систем керування контентом

Використання комп’ютерних засобів для підтримки освітнього процесу має довгу історію, яка передує появі сучасних систем управління навчанням (Learning Management Systems — LMS). Концепції комп’ютерної підтримки навчання виникли ще у першій половині XX століття, і хоча тодішні підходи значно

відрізнялися від сучасних систем, саме вони стали основою для подальшої еволюції у сфері освітніх технологій [4].

На початкових етапах розвитку використовувалися терміни Computer-Assisted Learning (CAL), Computer-Aided Instruction (CAI) та Computer-Based Instruction (CBI). Ці форми передбачали застосування програмованих засобів, які пропонували користувачеві навчальні завдання та елементи контролю знань. Такі системи, як правило, були ізольованими рішеннями, з обмеженими можливостями щодо масштабування, персоналізації та збору статистичних даних про навчальний прогрес.

Першу спробу автоматизованого навчання можна простежити ще у 1924 році, коли психолог Сідні Л. Прессі (Sidney L. Pressey) створив механічну навчальну машину, яку вважають прототипом майбутніх LMS. Пристрій дозволяв подавати студентам запитання з кількома варіантами відповідей та фіксувати результати, виконуючи функцію елементарного тестування. Фото пристрою наведено на рис. 1.1.

Рисунок 1.1 — Механічна навчальна машина Сідні Л. Прессі

У 1930 році М. Е. Лазерт розробив проблемний циліндр — ще одну механічну систему для навчання, що реалізовувала адаптивний підхід: залежно від правильності відповідей учня змінювалась послідовність наступних запитань. У 1956 році було створено перші моделі адаптивного навчання, які вже реагували на рівень знань студента.

Однак справжній прорив у розвитку систем управління навчанням розпочався з появою персональних комп'ютерів. У 1970-х роках компанія Hewlett-Packard представила один із перших настільних ПК, що зробило можливим масове застосування комп'ютерів у навчальному середовищі.

Поява Інтернету в 1980—1990-х роках докорінно змінила концепцію навчання, зробивши можливим віддалений доступ до курсів. У 1990 році компанія SoftArc випустила першу програму для LMS, яка дозволяла організаціям адмініструвати навчальні матеріали через локальні мережі [5].

Поступово з розвитком веб-технологій LMS перетворилися на багатофункціональні інструменти, які поєднують можливості керування навчальним контентом, оцінювання, комунікації та звітності. Починаючи з 2000-х років, з'явилися такі добре відомі системи, як Moodle, Blackboard, Canvas, Edmodo, які не лише забезпечили новий рівень якості навчання, а й впровадили адаптивність, інтерактивність, підтримку SCORM-стандартів та хмарних технологій.

У сучасних умовах системи управління навчальним контентом еволюціонували у складні веборієнтовані платформи, які підтримують мобільний доступ, штучний інтелект, аналітику навчального прогресу, інтеграцію з іншими сервісами та можливість кастомізації під конкретні освітні цілі.

Розвиток систем управління навчанням (LMS) відбувався у тісному взаємозв'язку з технологічним прогресом, проходячи через кілька ключових етапів еволюції. Перші спроби автоматизувати освітні процеси спиралися на механічні пристрої, що забезпечували базові функції управління навчанням. Згодом, із появою комп'ютерних систем та мережевих технологій, LMS

трансформувалися у складні цифрові платформи, що дозволяли інтегрувати мультимедійні матеріали, взаємодіяти з користувачами в реальному часі та адаптувати освітній контент відповідно до індивідуальних потреб.

Кожен технологічний прорив — від локальних комп'ютерних програм до хмарних платформ із підтримкою штучного інтелекту — призводив до розширення функціоналу LMS, їх гнучкості та інтерактивності. Це сприяло підвищенню доступності освіти, дозволяючи користувачам взаємодіяти із системою з будь-якої точки світу. Вдосконалення алгоритмів обробки даних та персоналізації навчального процесу також значно покращило якість освіти, роблячи її більш ефективною та адаптованою до потреб студентів і викладачів.

Таким чином, LMS продовжують розвиватися, інтегруючи новітні технології для створення динамічного, персоналізованого та максимально ефективного освітнього середовища, що наведено на рис. 1.2.

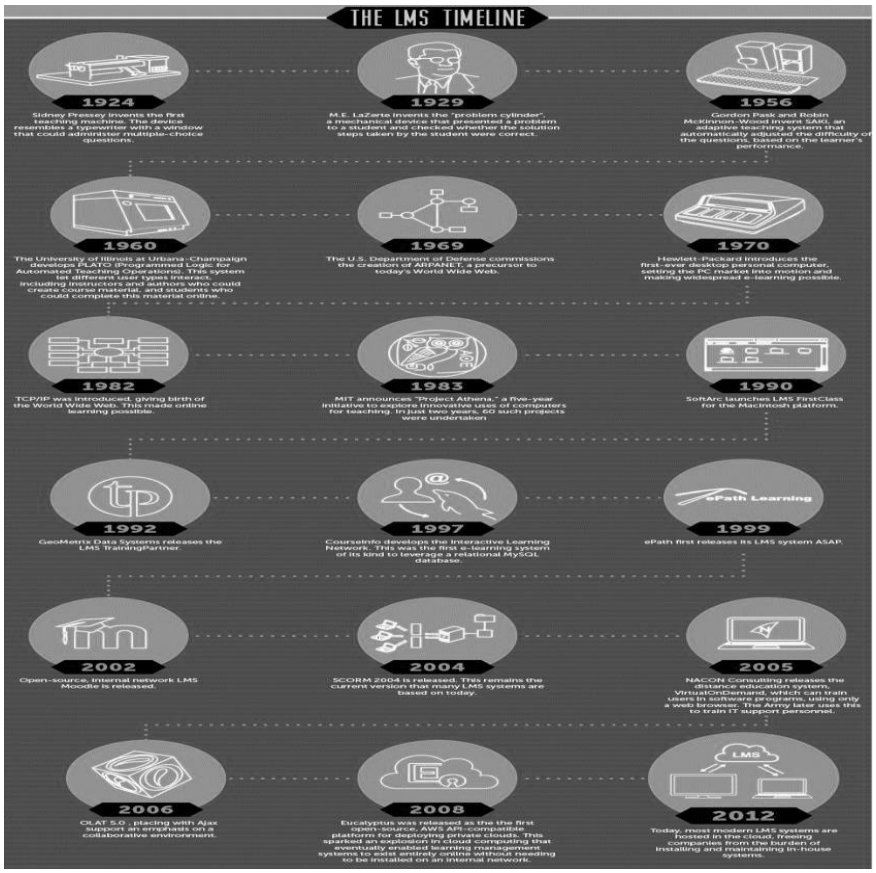

Рисунок 1.2 — Історія розвитку LMS

1.4 Огляд та аналіз сучасних освітніх платформ

Google Classroom — це інтегрована освітня платформа, розроблена компанією Google, яка надає комплекс інструментів для організації цифрового навчального процесу. Вона дозволяє викладачам створювати та поширювати завдання, здійснювати перевірку робіт, оцінювання та зворотний зв’язок у режимі реального часу. Первісною метою розробки Google Classroom було зменшення залежності від паперових матеріалів у навчанні та забезпечення зручної взаємодії між викладачами й учнями, зокрема в освітніх установах, що використовують Chromebook.

З початком масового впровадження дистанційного навчання Google Classroom набув ширшого поширення, ставши однією з провідних платформ для управління навчальним контентом. Платформа інтегрується з іншими сервісами Google, такими як Google Документи, Таблиці, Презентації, Форми, Сайти, Gmail, Календар, а також підтримує засоби відеозв’язку — Google Meet і Google Hangouts. Фото головного вікна Google Classroom наведено на рисунку 1.3.

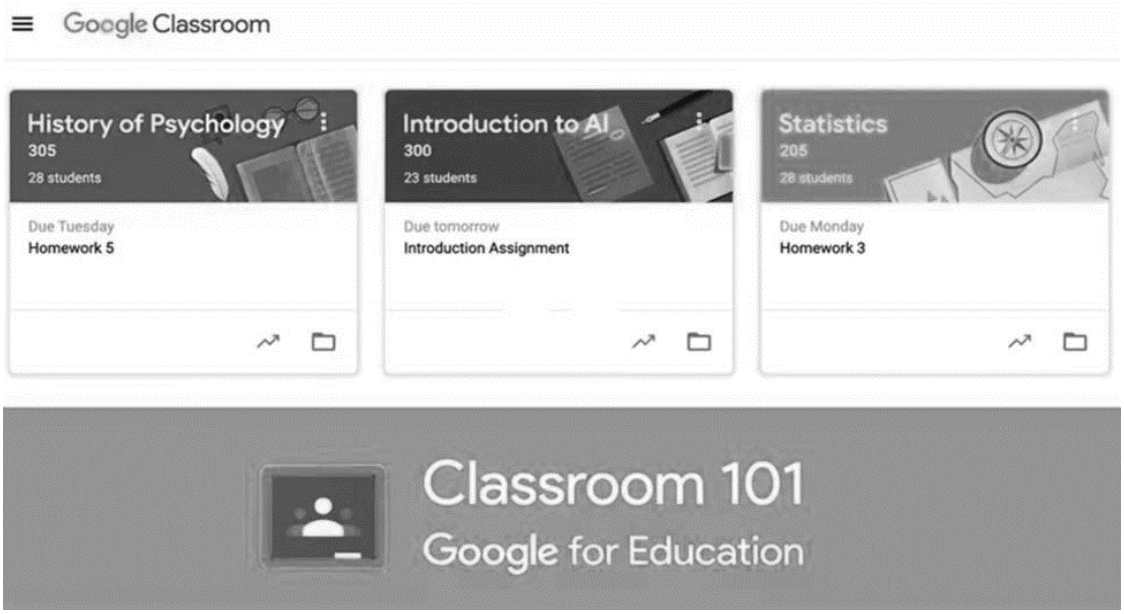

Рисунок 1.3 — Головне вікно Google Classroom

Функціональні можливості для викладачів:

— організація відеозустрічей через Google Meet;

— створення курсів, завдань, тестів, опитувань, управління матеріалами та оцінками;

— додавання навчального контенту з youtube, google диска, інтерактивних форм та інших ресурсів;

— надання коментарів, рекомендацій і персоналізованого зворотного зв’язку учням;

— публікація оголошень і запитань у стрічці курсу;

— можливість залучення батьків і опікунів через email-розсилки зі зведенням про стан виконання завдань учнями.

Функціональні можливості для учнів:

— перегляд, виконання та здача завдань у зручному форматі;

— отримання оцінок, коментарів і рецензій від викладача;

— комунікація в межах курсу за допомогою коментарів або електронної пошти.

Функціональні можливості для батьків:

— автоматичне надсилання звітів із коротким оглядом діяльності та успішності учнів;

— отримання сповіщень про майбутні або прострочені завдання.

Функціональні можливості для адміністраторів:

— централізоване керування обліковими записами користувачів та дозволами;

— налаштування навчальних курсів, списків учнів, додавання або видалення учасників;

— захист даних і конфіденційності відповідно до політик Google Workspace for Education;

— доступ до технічної підтримки 24/7.

Таким чином, Google Classroom є потужним інструментом для організації гнучкого, доступного та ефективного освітнього середовища, що відповідає сучасним вимогам до цифрового навчання.

Moodle (Modular Object-Oriented Dynamic Learning Environment) — це потужна система управління навчанням (LMS), яка активно використовується в освітньому процесі для організації дистанційного та змішаного навчання. Вона є програмним забезпеченням з відкритим вихідним кодом і розповсюджується на безоплатній основі, що дозволяє установам повністю адаптувати платформу до власних потреб. Moodle підтримує гнучке налаштування, модульну архітектуру з широким набором плагінів, а також має активну глобальну спільноту викладачів, розробників і користувачів [6].

На відміну від комерційних LMS, таких як Instructure Canvas або Blackboard, Moodle вимагає розгортання і налаштування на власних серверах, якщо не використовується платна хмарна версія. Це дозволяє адміністраторам мати повний контроль над функціональністю та конфігурацією системи, включаючи налаштування єдиного входу (SSO), мовної локалізації, безпеки та зовнішнього вигляду.

Основні функціональні можливості Moodle наведено на рис. 1.4.

Управління сайтом:

— повне адміністрування платформи здійснюється через спеціальний інтерфейс адміністратора;

— налаштування системи може здійснюватися як на етапі інсталяції, так і після запуску;

— можливе гнучке редагування зовнішнього вигляду сайту: кольорова гама, шрифти, розміщення елементів;

— функціонал системи може бути розширений за рахунок додаткових модулів і плагінів;

— платформа підтримує багатомовність завдяки мовним пакетам, що забезпечує адаптацію до будь-якої країни;

— відкритий код дозволяє модифікувати будь-які компоненти відповідно до потреб організації.

Керування користувачами:

— підтримується кілька способів реєстрації: самореєстрація, ручна реєстрація адміністратором, за допомогою LDAP тощо;

— система автоматично надсилає нагадування користувачам про відновлення паролів через електронну пошту;

— реалізовані сучасні механізми захисту від несанкціонованого доступу.

— особисті профілі користувачів можуть містити інформацію за вибором самого користувача;

— призначення користувачів на курси можливе через ключ доступу, ручне додавання або інші методи;

— рольова система управління правами доступу дозволяє детально налаштовувати дозволи на рівні курсів, модулів тощо.

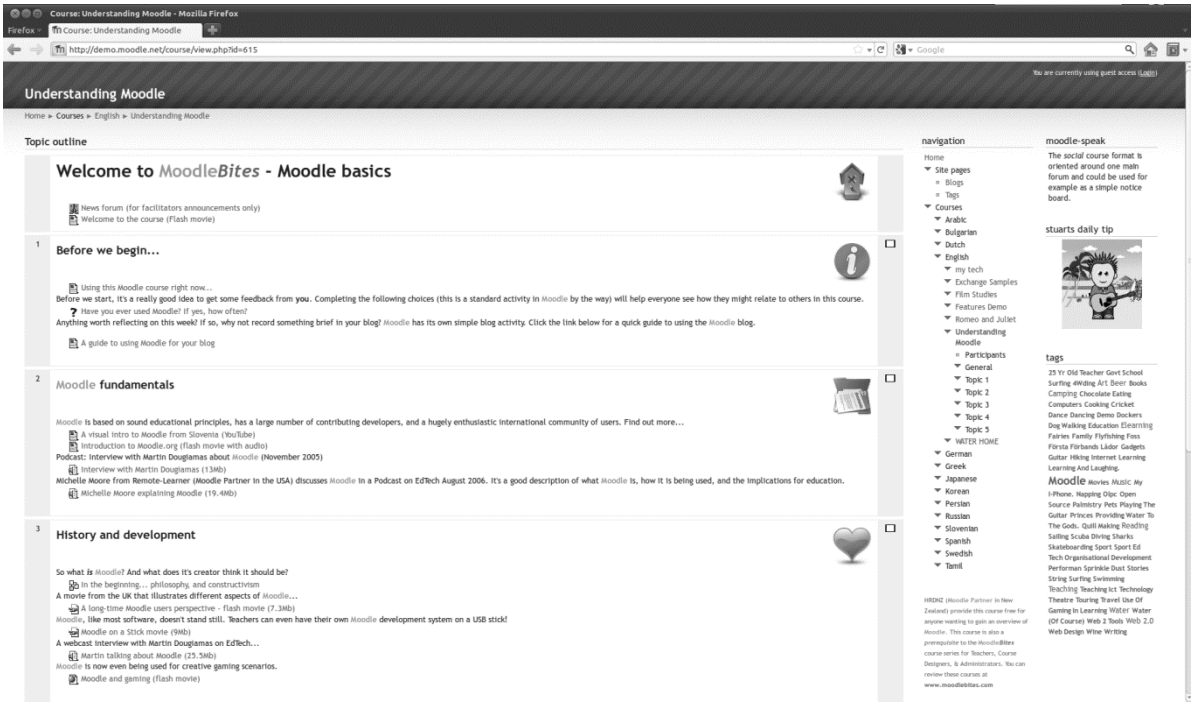

Рисунок 1.4 — Головне вікно Moodle

Управління курсами:

— викладачі мають повний контроль над курсом, якщо інше не задано адміністратором;

— підтримуються різні формати організації навчання: SCORM, топік-орієнтований, календарний тощо;

— індивідуальні налаштування курсу дозволяють гнучко адаптувати зміст до навчальних цілей;

— Moodle пропонує широкий набір інтерактивних компонентів: форуми, тести, глосарії, ресурси, чати, опитування тощо;

— система автоматично зберігає останні зміни, внесені після останньої авторизації користувача;

— забезпечується повне відстеження прогресу кожного слухача курсу;

— інтеграція з поштовими сервісами дозволяє обмін повідомленнями між викладачами та учнями;

— підтримується резервне копіювання курсів (Backup) у форматі ZIP;

— елементи курсів можна імпортувати з інших курсів для повторного використання.

Взаємодія користувачів:

— інструменти взаємодії включають: чат, блог, форум, вікі, які сприяють формуванню інтерактивного освітнього середовища.

Blackboard Learn — це потужна інтерактивна система управління навчанням (LMS), призначена для широкого спектра користувачів: закладів вищої освіти, загальноосвітніх шкіл, державних і військових установ, а також великих корпоративних структур [7]. Платформа підтримує реалізацію освітніх програм різного рівня, забезпечуючи при цьому масштабованість, надійність та багатофункціональність. Фото головного вікна Blackboard Learn наведено на рис. 1.5.

Серед основних можливостей Blackboard — створення навчального контенту, організація віртуальних класів, керування сертифікацією, а також функції для корпоративного та соціального навчання. Платформа доступна в декількох варіантах розгортання: як керований хостинг, локальна інсталяція або у форматі програмного забезпечення як послуги (SaaS). Крім того, Blackboard Learn підтримує мобільний доступ, що дозволяє здобувачам освіти переглядати курси з будь-якого місця та пристрою.

Однією з ключових особливостей платформи є інтегрований календар, що дозволяє користувачам відстежувати дедлайни. Також реалізовано потік активності, який виводить на перший план найважливіші повідомлення, завдання та курси, що потребують уваги. Blackboard Learn підтримує елементи гейміфікації, зокрема цифрові значки за досягнення, а також функціонал електронної комерції, що дозволяє продавати навчальні курси поза межами організації.

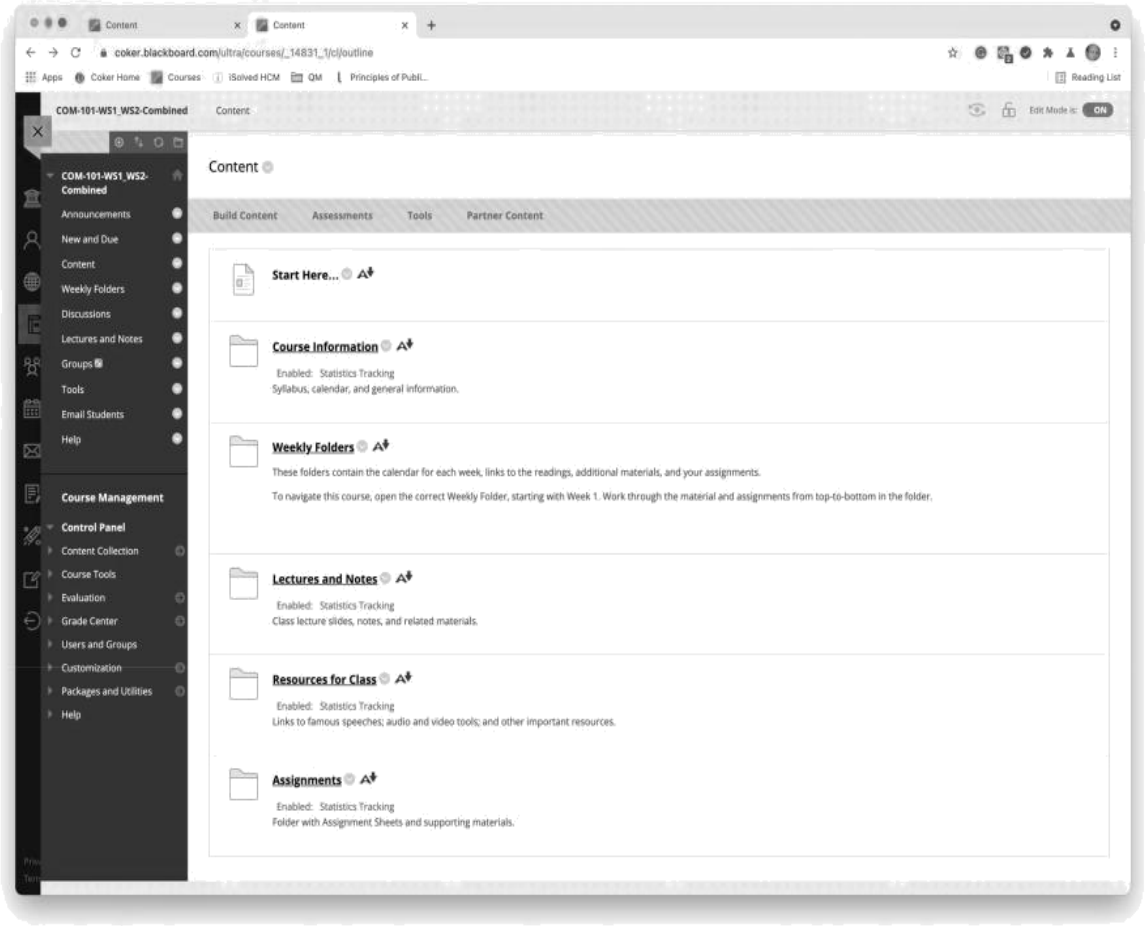

Рисунок 1.5 — Головне вікно Blackboard Learn

Однією з ключових особливостей платформи є інтегрований календар, що дозволяє користувачам відстежувати дедлайни. Також реалізовано потік активності, який виводить на перший план найважливіші повідомлення, завдання та курси, що потребують уваги. Blackboard Learn підтримує елементи

гейміфікації, зокрема цифрові значки за досягнення, а також функціонал електронної комерції, що дозволяє продавати навчальні курси поза межами організації.

Основні функціональні можливості Blackboard Learn:

— мобільний доступ: підтримка мобільних додатків для роботи на ходу;

— система сповіщень: інформування про важливі події, дедлайни, оцінки тощо;

— контент-менеджмент: створення, редагування, збереження та публікація навчальних матеріалів;

— спільна робота: інтерактивні інструменти для групової діяльності;

— контент-редактор: потужний редактор для форматування тексту, додавання медіа та інших елементів;

— аналітика: інструменти аналізу даних, формування звітів, моніторинг успішності;

— зворотній зв’язок: можливість залишати коментарі та оцінки, у тому числі автоматизовані;

— інтеграція з Blackboard Collaborate: організація онлайн-занять у режимі реального часу;

— управління курсами: формування груп, контроль доступу, сертифікація;

— панель управління (дешборд): централізоване місце для перегляду ключової інформації;

— дошка xpLor: інтеграція спільного сховища навчальних ресурсів;

— гейміфікація: цифрові значки за навчальні досягнення;

— комерційна функціональність: можливість монетизації курсів;

— обговорення календаря: координація подій та завдань;

— файловий менеджер: централізоване зберігання та управління файлами;

— автоматизація оцінювання: полегшення процесу виставлення оцінок;

— цілі та стандарти: налаштування цілей навчання та звітність за їх досягненням.

TalentLMS — це хмарна система управління навчанням (LMS), розроблена для швидкого та гнучкого створення, проведення та моніторингу онлайн-курсів і навчальних програм. Платформа орієнтована на підприємства будь-якого розміру, які прагнуть забезпечити безперервне навчання співробітників та сприяти їх професійному розвитку [8].

Завдяки простоті використання та налаштуванню, TalentLMS дозволяє створювати курси електронного навчання всього за кілька хвилин. Користувачі можуть персоналізувати навчальне середовище, додаючи власний логотип, налаштовуючи теми оформлення або використовуючи зовнішній домен, що дозволяє брендувати платформу під потреби конкретної організації. Платформа підтримує функції продажу та поширення курсів онлайн.

TalentLMS оптимізовано для роботи на різних пристроях, зокрема настільних комп’ютерах, планшетах і смартфонах під керуванням iOS та Android. Вона також пропонує рідні мобільні додатки для iOS і Android, що забезпечує зручний доступ до навчальних матеріалів у будь-який час. Інтерфейс TalentLMS наведено на рис. 1.6

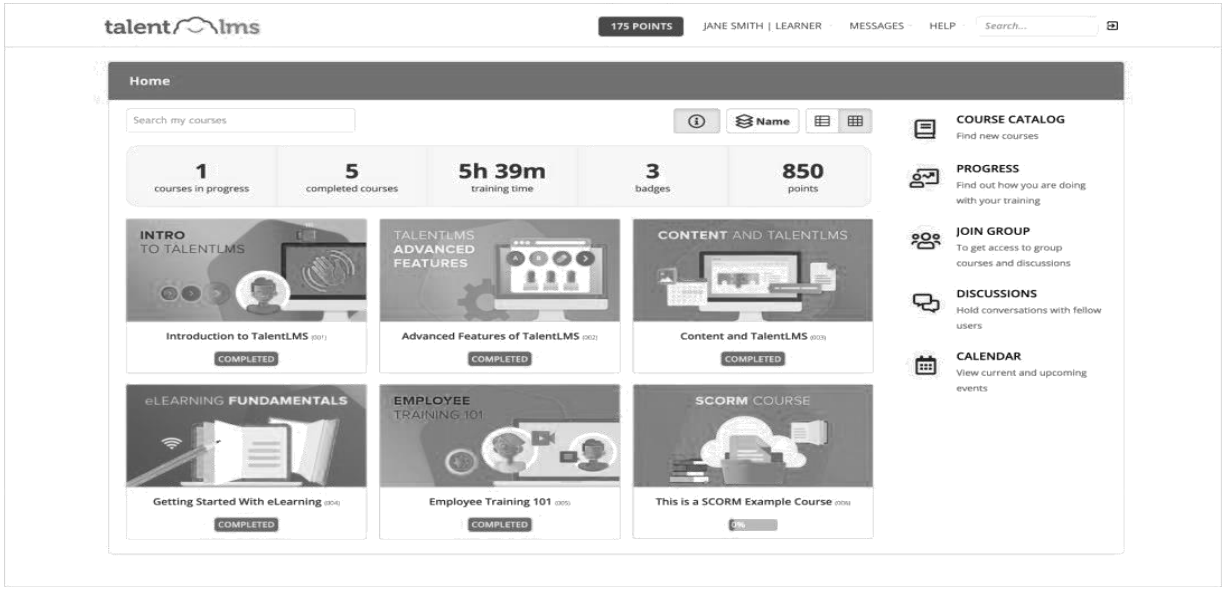

Рисунок 1.6 — Інтерфейс TalentLMS

Ключові функціональні можливості TalentLMS:

— інтуїтивно зрозумілий інтерфейс, який мінімізує час на опанування платформи і дозволяє користувачам зосередитися на навчальному контенті;

— підтримка інтеграції з існуючими навчальними матеріалами, можливість повторного використання контенту та прямого додавання ресурсів з інтернету;

Основні переваги платформи:

— швидке розгортання порталу електронного навчання завдяки хмарній архітектурі;

— наявність безкоштовного тарифного плану з можливістю гнучкого оновлення або пониження пакету;

— мінімалістичний дизайн, що сприяє зручній роботі користувачів.

Для наочності функціональні можливості LMS узагальнено в таблиці 1.1, яка порівнює ключові функції популярних систем управління навчанням.

Таблиця 1.1 — Порівняння функціональних можливостей LMS

Функція	Moodle	Google Classroom	Blackboard
Управління контентом	Підтримка SCORM, модульна структура курсів	Хмарна архітектура, адаптивний контент	Комплексне управління курсами
Оцінювання знань	Тести, завдання, автоматизоване оцінювання	Гнучкі критерії оцінювання	Аналіз відповідей, звіти
Комунікація	Форуми, чати, повідомлення	Інтеграція з Zoom, Teams	Віртуальні класи, чати
Аналітика та звітність	Звіти про прогрес, активність	Детальна аналітика прогресу	Розширені аналітичні звіти
Інтеграція з системами	API, плагіни для інтеграції	API, Google Workspace	Інтеграція з академічними системами
Безпека даних	Локальне розгортання, шифрування	Хмарна безпека AWS	Двофакторна автентифікація

На сучасному етапі існує велика кількість веб-додатків, які відрізняються за функціональністю та відповідністю різноманітним потребам користувачів. Кожен користувач обирає найбільш оптимальний варіант серед наявних систем, керуючись їхніми перевагами та недоліками. Саме на основі цих критеріїв формується загальний рейтинг популярності та рівень використання відповідних платформ.

У сфері інформаційних технологій освітні веб-платформи набирають дедалі більшої популярності та попиту, що обумовлює їх важливість у сучасній освітній діяльності. Тому розробка веб-додатків для навчання є актуальним завданням не лише для освітньої галузі, але й для суміжних сфер.

Аналізуючи вимоги користувачів до освітніх веб-платформ, можна виділити низку ключових характеристик, якими повинне володіти ефективне програмне середовище: воно має бути інтуїтивно зрозумілим, доступним, адаптованим до користувачів із різним рівнем технічного забезпечення, а також мати можливість безкоштовного використання і сумісність із різними операційними системами.

2 ВИБІР НЕОБХІДНИХ ІНСТРУМЕНТІВ ТА ТЕХНОЛОГІЙ РОЗРОБКИ

2.1 Основи HTML

HTML (HyperText Markup Language — мова гіпертекстової розмітки) є основною мовою для створення веб-сторінок та веб-документів. Вона задає структуру та форму веб-сторінки, визначаючи, як саме веб-браузери мають відображати текстову, графічну та мультимедійну інформацію. HTML не є мовою програмування, а саме мовою розмітки, яка дозволяє організувати контент у певній логічній структурі за допомогою спеціальних елементів, званих тегами.

Основне призначення HTML полягає в тому, щоб створити основу сторінки, яка може бути доповнена стилями (CSS) та динамічними функціями (JavaScript). Саме за допомогою HTML браузер отримує інформацію про те, які частини сторінки є заголовками, абзацами, списками, таблицями, зображеннями тощо [9].

HTML є стандартизованою мовою розмітки, що регулюється міжнародними організаціями W3C та WHATWG. Це забезпечує уніфіковані правила написання та інтерпретації коду в різних браузерах, що гарантує стабільне відображення веб-сторінок на різних пристроях і платформах. HTML-документ зберігається у вигляді текстового файлу з розширенням .html або .htm. Правильне іменування таких файлів є важливим, адже веб-сервер визначає спосіб обробки вмісту за типом розширення. На початку кожного HTML-документа обов’язково вказується декларація <!DOCTYPE html>. Вона повідомляє браузеру, що документ написано відповідно до стандарту HTML5 — найновішої версії мови. У разі відсутності цієї декларації або її неправильного запису браузер може активувати режим сумісності, що може погіршити відображення або функціональність сторінки. Основу HTML становлять теги — спеціальні елементи, які задають структуру вмісту. Кожен тег, як правило, має відкриваючу та закриваючу частину, між якими розміщується контент. Наприклад, тег <p> використовується для створення абзаців, а тег <img> — для вставлення зображень. Дотримання правильної

структури HTML-документа забезпечує легке сприйняття як для браузера, так і для розробника.

Коли користувач вводить URL-адресу або натискає на посилання, веб-браузер надсилає запит до веб-сервера. Сервер у відповідь шукає HTML-файл за вказаною адресою. Знайдений файл з розширенням .html передається браузеру (див. рис. 2.1), який починає обробку його вмісту. Аналіз розпочинається з декларації <!DOCTYPE html>, далі браузер розпізнає HTML-теги та формує візуальне представлення сторінки для користувача. Якщо в HTML-документі є посилання на зовнішні ресурси, такі як CSS або JavaScript, браузер додатково завантажує ці файли та застосовує їх для оформлення сторінки або додавання функціональності.

Рисунок 2.1 — Вигляд документу HTML

Переваги HTML:

— легкість вивчення і використання: HTML має простий і інтуїтивно зрозумілий синтаксис, що робить її доступною навіть для новачків без спеціальної підготовки [10], навички написання базових HTML-документів можна опанувати за кілька днів;

— гнучкість у створенні структури, HTML дозволяє створювати як прості, так і складні багаторівневі структури сторінок, застосовуючи різноманітні теги, розробник може чітко організувати контент для зручного сприйняття користувачами;

— платформо незалежність, створені на HTML сторінки однаково коректно відображаються на різних операційних системах, таких як Windows, Linux,

macOS, Android та iOS. Це робить мову універсальним засобом для розробки веб-контенту;

— підтримка всіма браузерами, всі сучасні веб-браузери підтримують HTML, забезпечуючи сумісність сайтів та додатків на будь-яких пристроях.

— чистий та структурований код HTML дозволяє розробляти зрозумілий і легкий для підтримки код, що особливо важливо у великих проєктах з командною розробкою;

— Можливість інтеграції HTML-документи легко поєднуються з іншими веб-технологіями — CSS для стилізації, JavaScript для взаємодії, а також серверними мовами програмування, такими як PHP, Python, Ruby, що забезпечує динамічний та функціональний веб-контент.

Недоліки HTML:

— обмежена динамічність HTML сам по собі не може створювати динамічний контент або реагувати на дії користувача. Для цього необхідно використовувати додаткові технології, наприклад JavaScript або серверні мови;

— повторення коду при розробці великих сайтів багато елементів доводиться створювати вручну і окремо, що може призводити до дублювання коду і ускладнює підтримку;

— підтримка нових стандартів деякі браузери можуть не повністю підтримувати нові можливості та теги останніх версій HTML, що вимагає перевірки сумісності та застосування поліфілів або інших обхідних рішень.

У наведеному прикладі видно основні елементи які наведено на рис. 2.2:

— <!DOCTYPE html> — оголошення типу документа;

— <html> — кореневий елемент всього документа;

— <head> — розділ з метаданими, наприклад кодування та заголовок сторінки;

— <body> — основний вміст, який відображається на сторінці;

— заголовок <h1> і абзац <p> — приклади структурних тегів.

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <title>Приклад HTML-сторінки</title>
</head>
<body>
  <h1>Вітаємо на нашому сайті!</h1>
  <p>Це приклад простої HTML-сторінки.</p>
</body>
</html>
```

Рисунок 2.2 — Приклад структури HTML-документа

2.2 Каскадні таблиці стилів (CSS)

CSS (скорочення від англ. Cascading Style Sheets, що в перекладі означає каскадні таблиці стилів) — це спеціалізована мова опису зовнішнього вигляду елементів веб-сторінок. Вона використовується для візуального оформлення HTML-документів та надання їм привабливого, сучасного і зручного для сприйняття вигляду. Застосування CSS дозволяє задавати кольорову палітру тексту, встановлювати розмір і тип шрифтів, налаштовувати відображення фонових зображень, визначати відступи, а також створювати різноманітні графічні ефекти, анімації та адаптивні інтерфейси [11].

Завдяки можливостям CSS, розробник може адаптувати веб-інтерфейс для відображення як на традиційних комп’ютерах і ноутбуках, так і на планшетах чи смартфонах. Мова стилів відзначається простим синтаксисом, зрозумілим навіть для початківців, але водночас забезпечує гнучкий контроль над зовнішнім виглядом сторінки. CSS вважається невід’ємною частиною фронтенд-розробки.

Щоб комп’ютерна система або браузер могли розпізнати документ як файл зі стилями, він повинен мати відповідне розширення — .css. Це розширення

повідомляє, що вміст файлу містить інструкції з оформлення сторінки, а не її зміст (рис. 2.3).

Рисунок 2.3 — Вигляд документу CSS

Для того щоб стилі застосовувались до вмісту HTML-сторінки, необхідно підключити CSS-файл до HTML-документа. Це реалізується шляхом вставки відповідного HTML-тегу <link> у секцію <head>. Саме в цьому місці слід здійснювати підключення, оскільки браузер під час завантаження сторінки читає вміст зверху вниз і обробляє CSS до виводу основного контенту (рис. 2.4).

```
<link rel="stylesheet" href="css/style.css">
```

Рисунок 2.4 — Приклад підключення зовнішнього CSS-файлу

Мова CSS функціонує на основі так званих правил стилізації. Кожне таке правило складається з двох ключових частин: селектора і блоку оголошень. Селектор вказує на HTML-елемент, до якого буде застосовано стилі, а блок оголошень містить одну або кілька пар "властивість—значення", що визначають зовнішній вигляд цього елемента (рис. 2.5).

CSS-інструкції можуть бути вписані прямо у HTML-файл — у вигляді вбудованих або інлайнових стилів — або розміщені у зовнішньому файлі. Розділення стилів та структури вважається найкращою практикою у веб-розробці. Це забезпечує більшу гнучкість, зменшує дублювання коду та полегшує супровід проектів. У CSS для вибору елементів використовуються імена тегів, класи, ідентифікатори або вкладені селектори (рис. 2.6).

Рисунок 2.5 — Базова структура правила в CSS

```
.menu__item a {
    font-family: 'Open Sans Semibold';
    font-size: 20px;
    color: #212121;
}
```

Рисунок 2.6 — Приклад стилізації елемента за допомогою CSS

Використання CSS має низку важливих переваг. Воно дозволяє значно скоротити час розробки, адже один файл зі стилями можна підключити до кількох HTML-документів, забезпечуючи централізоване управління зовнішнім виглядом сайту [12]. Це також підвищує швидкість завантаження сторінок, оскільки стилі завантажуються лише один раз і зберігаються в кеші браузера, зменшуючи навантаження на мережу. Синтаксис CSS є зрозумілим і логічним, що полегшує процес навчання. Мова підтримує широкий функціонал: анімації, трансформації, адатину верстку, що дає великі можливості для реалізації сучасного веб-дизайну.

Окрім того, CSS забезпечує гнучкість і кросплатформенність — стилі адаптуються до екранів різних пристроїв, зокрема мобільних телефонів і планшетів.

Проте існують і певні недоліки CSS, які слід враховувати. Зокрема, це різниця в підтримці браузерами: не всі браузери однаково інтерпретують нові функції, особливо їх застарілі версії, що може призводити до проблем з відображенням. Також варто згадати про плутанину з версіями мови, адже наявність різних редакцій, таких як CSS2 чи CSS3, може ускладнити вибір інструментів та властивостей. Ще одним недоліком є відсутність механізмів захисту, адже CSS-код є відкритим і легко переглядається у браузері, що створює ризики несанкціонованого використання або копіювання стилів третіми особами.

2.3 Мова програмування JavaScript

JavaScript — це динамічна, об’єктно-орієнтована мова програмування, яка базується на прототипах.

Її застосовують для реалізації таких функцій, як анімовані графічні елементи, створення слайд-шоу, автозаповнення тексту в полях введення, інтерактивні елементи форм. Приклади використання JavaScript ми бачимо щодня: новинні стрічки, автоматичне оновлення контенту в соціальних мережах, таких як Twitter, або оновлення часу на пристрої [13]. Основне призначення JavaScript — створення динамічних веб-ресурсів та інтерактивних веб-додатків.

Окрім веб-браузерів, ця мова також активно використовується в розробці програмного забезпечення для роботи з апаратними компонентами та на серверній частині. Нижче розглянуто ключові напрямки, де використовується JavaScript:

Завдяки JavaScript користувач може взаємодіяти з веб-сторінкою без перезавантаження. Наведемо кілька прикладів функціональності:

— відображення або приховування елементів за натисканням кнопки;

— зміна кольору кнопки при наведенні курсору;

— реалізація слайдерів та галерей;

— масштабування зображень;

— демонстрація таймерів і зворотного відліку;

— відтворення відео та звуку безпосередньо на сторінці;

— створення анімацій.

Для створення таких застосунків JavaScript пропонує широку екосистему фреймворків. До найпопулярніших належать React, React Native, Angular, Vue. Для серверної логіки активно використовують середовище Node.js, що базується на рушії V8. Серед відомих компаній, які застосовують JavaScript — PayPal, LinkedIn, Netflix, Uber.

JavaScript використовується не лише у клієнтській частині. Наприклад, Node.js дає змогу реалізовувати серверні застосунки та створювати веб-сервери, завдяки чому JavaScript став повноцінною мовою для розробки повного циклу (full-stack).

За допомогою JavaScript можна створювати як браузерні ігри, так і окремі застосунки для настільних платформ. Це розширює можливості мови в галузі розваг.

Для написання коду на JavaScript достатньо простого текстового редактора. Найпопулярніші серед них: Atom, Sublime Text, WebStorm, Visual Studio Code, Notepad++, TextPad, Xcode. Вони забезпечують підсвічування синтаксису, що значно спрощує пошук помилок у коді.

Хоча для базового програмування достатньо редактора та браузера, професійна розробка часто вимагає додаткових інструментів:

— Git — система контролю версій, яка полегшує роботу в команді;

— Node — дає змогу виконувати JavaScript поза браузером;

— Gulp — автоматизує процес розробки;

— Babel — трансформує сучасний синтаксис у сумісний із більшістю браузерів;

— ESLint — допомагає виявити синтаксичні помилки та покращити якість коду.

Переваги JavaScript:

— підтримується більшістю сучасних браузерів, взаємодіє з CSS і сервером;

— знижене навантаження на сервер завдяки обробці на стороні клієнта;

— велика кількість бібліотек і плагінів полегшує розробку;

— простота синтаксису і компактність;

— зручний для взаємодії з елементами інтерфейсу: кнопками, формами, курсором;

— доступність для новачків — легкий в опануванні.

Недоліки JavaScript:

— обмежені можливості зчитування і збереження файлів задля безпеки;

— можливі помилки в обробці даних під час виконання;

— неможливість безпосереднього доступу до системних ресурсів пристрою;

— відкритий код може бути змінений зловмисниками, що становить ризик для безпеки.

2.4 JavaScript-фреймворки

Фреймворки JavaScript — це спеціалізовані бібліотеки, що містять заздалегідь підготовлені скрипти, які використовуються для реалізації типових завдань та функцій при розробці програмних продуктів на основі мови JavaScript.

Основне призначення фреймворків полягає у значному спрощенні процесу програмування та оптимізації розробки [14].

Переваги використання фреймворків JavaScript:

— є повністю безкоштовними для застосування;

— мають відкритий вихідний код, що дозволяє адаптувати їх під конкретні потреби, а також знижує ризики пов’язані з ліцензуванням;

— сприяють підвищенню продуктивності розробника — завдяки наявності готових модулів скорочується час написання коду вручну, покращується читабельність коду, а процес виявлення та усунення помилок відбувається значно ефективніше;

— підтримують модульну структуру проєкту, що забезпечує можливість

паралельної роботи кількох розробників над окремими частинами застосунку.

Одним із недоліків деяких фреймворків є обмежена підтримка з боку пошукових систем при індексації контенту, що може впливати на SEO-оптимізацію. Проте сучасні пошукові системи, зокрема Google, вже впровадили відповідні технології, які мінімізують цей недолік.

2.5 Особливості React

React.js — це відкрита JavaScript-бібліотека, призначена для розробки користувацьких інтерфейсів. Вона вирішує проблему часткового оновлення вмісту вебсторінки, що особливо актуально під час створення односторінкових застосунків (SPA) [15]. Розробка бібліотеки ведеться компаніями Facebook, Instagram та спільнотою незалежних розробників.

React.js дозволяє створювати масштабовані вебдодатки, в яких дані змінюються динамічно, без необхідності повного перезавантаження сторінки. Основна мета бібліотеки — забезпечити швидкість, простоту й масштабованість розробки. React відповідає за відображення інтерфейсу користувача (View) згідно з архітектурною моделлю MVC (Model-View-Controller), тому його можна ефективно поєднувати з іншими JavaScript-бібліотеками та фреймворками, такими як AngularJS [16]. Часто React використовують у зв'язці з Redux для керування станом застосунку.

Ключовою технологією React є використання віртуального DOM (Virtual DOM) — внутрішньої кеш-структури, яка зберігається в оперативній пам’яті. React порівнює поточний стан із попереднім та ефективно оновлює лише ті частини DOM, які зазнали змін. Цей процес називається примиренням (reconciliation). Розробник створює компоненти так, ніби вся сторінка перерендерюється при кожній зміні, але насправді бібліотека оновлює тільки змінені елементи, що значно підвищує продуктивність і знижує навантаження на браузер (рис. 2.7).

Ще однією особливістю є використання JSX — розширення JavaScript, яке дозволяє писати HTML-подібний синтаксис у коді. JSX компілюється у стандартні виклики методів бібліотеки React. JSX був натхненний технологією XHP, яку розроблено у Facebook для мови PHP. JSX дозволяє передавати атрибути, включно з користувацькими, до компонентів у вигляді пропсів (props).

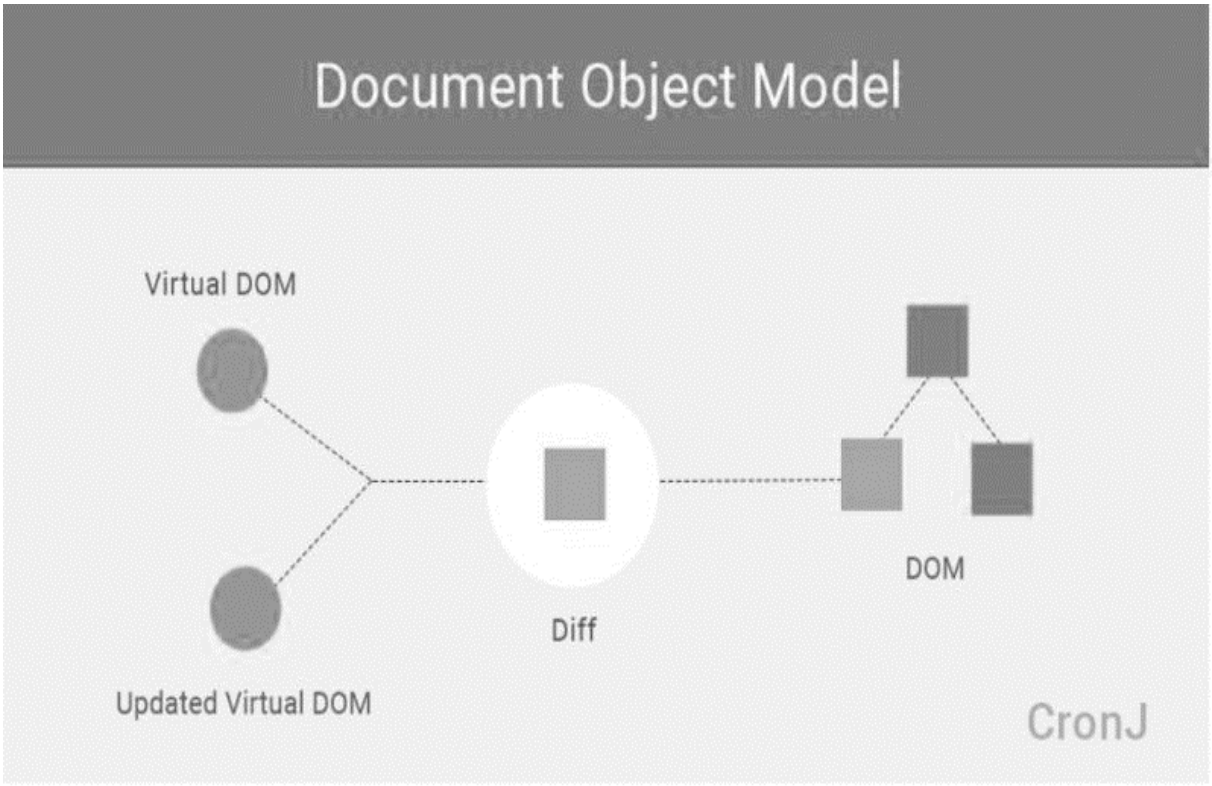

Рисунок 2.7 — Функціонування віртуального DOM

Ще однією особливістю є використання JSX — розширення JavaScript, яке дозволяє писати HTML-подібний синтаксис у коді. JSX компілюється у стандартні виклики методів бібліотеки React. JSX був натхненний технологією XHP, яку розроблено у Facebook для мови PHP. JSX дозволяє передавати атрибути, включно з користувацькими, до компонентів у вигляді пропсів (props).

React не обмежується лише відображенням HTML у браузері. Наприклад, Facebook використовує React для побудови динамічних графіків, Netflix та PayPal застосовують серверний рендеринг HTML за допомогою ізоморфної архітектурим [17].

React також реалізує методи життєвого циклу компонентів, які дозволяють розробнику обробляти дані в різні фази існування компонента (рис. 2.8).

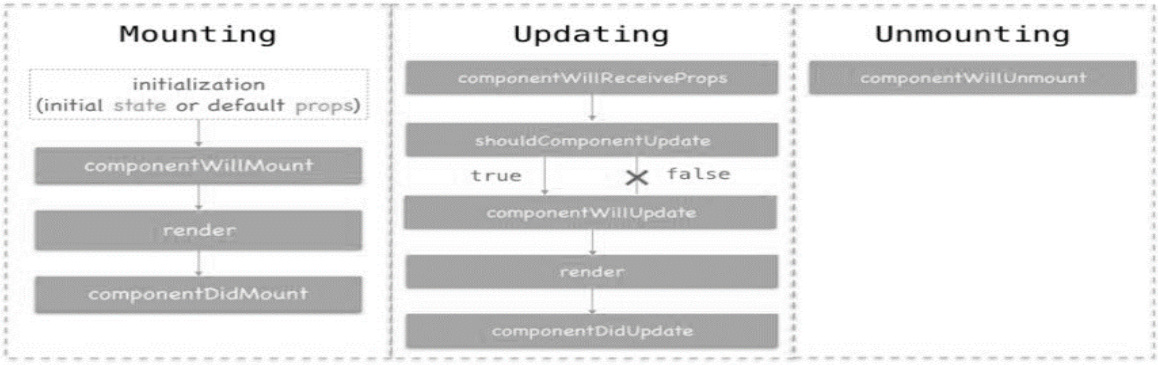

Рисунок 2.8 — Життєвий цикл в React

До основних методів життєвого циклу належать:

— shouldComponentUpdate — визначає, чи потрібно оновлювати компонент;

— componentWillMount — виконується перед монтуванням компонента у DOM;

— componentDidMount — виконується після рендерингу; часто використовується для ініціалізації даних або підключення до API;

— render — обов’язковий метод, який визначає вміст, що буде відображено на екрані.

React.js навмисно не надає повного набору інструментів для побудови всього додатка. Натомість він фокусується виключно на інтерфейсі, дозволяючи розробнику самостійно обирати засоби для обробки даних, взаємодії з API чи зберігання інформації.

Бібліотека була створена програмістом Джорданом Волке (Jordan Walke) у компанії Facebook під впливом концепцій XHP. Уперше React був представлений у 2011 році у стрічці новин Facebook, а пізніше — у блозі Instagram. Офіційний реліз відбувся у 2013 році на конференції JSConf US, а у 2015-му проєкт був відкритий для спільноти на конференції React.js Conf .

Для реалізації односпрямованого потоку даних, що відрізняється від двостороннього у AngularJS, Facebook запропонувала архітектуру Flux. У Flux дані передаються від дій (actions) через диспетчер (dispatcher) до сховищ (stores), а потім — до інтерфейсу користувача. Компоненти не змінюють пропси напряму, а викликають функції зворотного виклику, які створюють дії. Це дозволяє зберігати передбачуваність змін стану програми.

Найпоширенішою реалізацією Flux є Redux, який базується на концепції єдиного сховища (store), що зберігає увесь стан додатка, виступаючи як «єдине джерело істини».

2.6 Серверна платформа Node.js

Node.js — це кросплатформне середовище виконання з відкритим вихідним кодом, призначене для створення серверних і мережевих додатків. Програмні продукти, що розробляються на Node.js, реалізуються мовою JavaScript та можуть запускатися у середовищі Node.js на операційних системах OS X, Microsoft Windows та Linux.

Node.js логотип якого наведено на рисунку 2.9 містить багату колекцію вбудованих JavaScript-модулів, що значно спрощує розробку веб-додатків. Основною метою розробки середовища було забезпечення неблокуючої обробки запитів і введення/виведення даних, що дозволяє зберігати невеликий обсяг програми та забезпечити високу ефективність при роботі з великими обсягами даних у реальному часі, зокрема на розподілених системах.

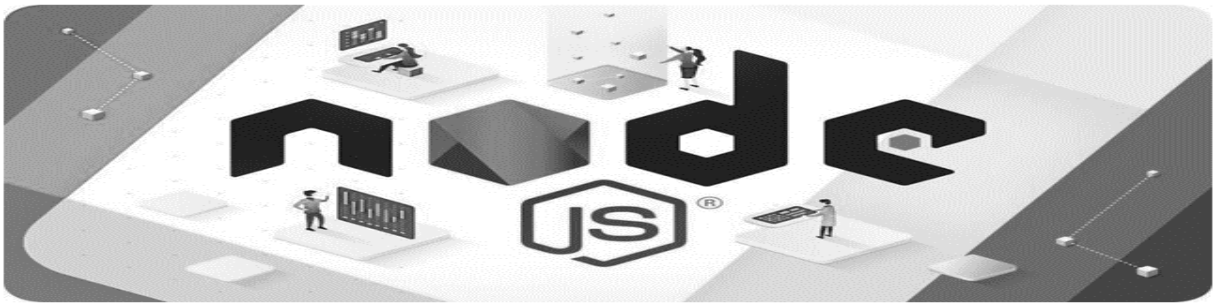

Рисунок 2.9 — Логотип Node.js

Основні особливості Node.js:

— асинхронність і орієнтація на події. Усі API Node.js працюють асинхронно, тобто неблокуючим чином. Після виклику API сервер негайно переходить до наступної операції, не очікуючи відповіді. Повідомлення про події дозволяє серверу обробити відповідь тоді, коли вона буде готова;

— висока швидкодія. Node.js базується на JavaScript-движку V8 від Google Chrome, що забезпечує дуже швидке виконання коду;

— однопотокова модель з масштабуванням. Попри використання лише одного потоку, цикл подій дозволяє Node.js обслуговувати велику кількість одночасних з'єднань. Це забезпечує високу масштабованість порівняно з традиційними серверними рішеннями, які створюють окремий потік для кожного запиту (наприклад, Apache HTTP Server);

— відсутність буферизації. Node.js не буферизує дані, а обробляє їх фрагментовано, передаючи одразу у вигляді потоків;

— ліцензування. Середовище Node.js поширюється за умовами відкритої ліцензії MIT, що сприяє його широкому використанню.

Node.js ідеально підходить для розробки масштабованих та продуктивних мережевих додатків, які потребують обробки великої кількості одночасних з’єднань з високою пропускною здатністю.

Разом з тим, Node.js менш ефективний у задачах, що пов’язані з інтенсивними обчисленнями або високим навантаженням на процесор. Такі операції можуть призводити до затримок та збоїв у роботі всього середовища.

Переваги Node.js:

— швидкість та зручність написання програмного коду;

— мультиплатформеність;

— доступ до широкого спектра безкоштовних інструментів;

— можливість повторного використання компонентів;

— висока продуктивність додатків.

Недоліки Node.js:

— надмірна гнучкість, що супроводжується частими оновленнями;

— залежність від зовнішніх бібліотек: видалення пакета з npm може призвести до непрацездатності багатьох проєктів, які його використовують.

2.7 Хмарний сервіс Firebase

Firebase — це платформа для розробки мобільних та вебзастосунків, створена з метою спростити роботу розробників, зокрема завдяки можливості обробки, передавання та зберігання даних у хмарному середовищі. Усі дані, що записуються у Firebase, синхронізуються між усіма клієнтами та узгоджуються в межах всієї системи .

У 2014 році компанія Google придбала Firebase у однойменного стартапу, після чого платформа стала одним із ключових інструментів компанії для хмарної розробки. Сьогодні Firebase інтегрований у хмарну інфраструктуру Google, а також підтримує взаємодію з численними відкритими бібліотеками та інструментами. Зображення головної сторінки Firebase наведено на рисунку 2.10.

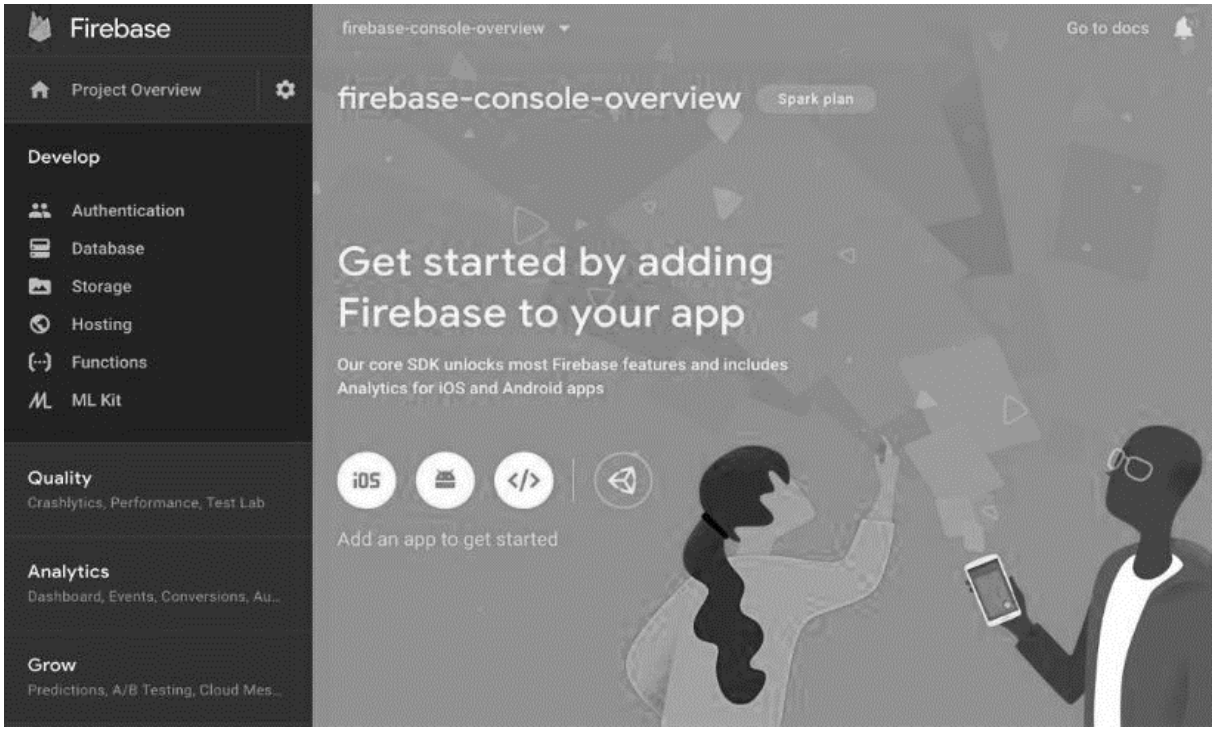

Рисунок 2.10 — Головна сторінка Firebase

Однією з причин популярності Firebase є його здатність забезпечувати постійну синхронізацію між локальними копіями даних і хмарним сховищем. Платформа дозволяє розподіляти навантаження між кількома пристроями, при цьому забезпечуючи цілісність та узгодженість даних завдяки зв’язку між центрами обробки даних. Кожен підключений пристрій, по суті, стає частиною хмари.

Firebase автоматично дублює локально збережені дані на сервери в хмарі, забезпечуючи їх узгодженість. Аналогічно, зміни в хмарному середовищі реплікуються на пристрої користувачів. Таким чином, розробники можуть обмінюватися даними з клієнтським застосунком шляхом запису інформації безпосередньо у хмару [18].

Крім того, у Firebase реалізовано підтримку безсервісної архітектури (serverless). Хмарні функції дозволяють автоматично запускати обробку при зміні бази даних або після першого входу користувача в систему. Такі функції можуть виконувати різні дії — від попередньої обробки зображень чи текстів до забезпечення додаткової узгодженості інформації або ініціації інших хмарних процесів.

Платформа підтримує автентифікацію за допомогою електронної пошти і пароля, номера телефону, а також через популярні соціальні мережі, зокрема Google, Facebook та Twitter. SDK Firebase забезпечує легку інтеграцію будь-якого з цих способів у застосунок. Ще однією ключовою функцією є база даних реального часу, яка дозволяє синхронізувати дані між клієнтами миттєво та працювати навіть в офлайн-режимі. Firebase також надає швидкий і надійний хостинг для вебзастосунків із глобальним кешуванням за допомогою мережі доставки контенту (CDN). Тестова лабораторія дозволяє перевіряти додатки на реальних і віртуальних пристроях, розміщених у дата-центрах Google. Окрім цього, платформа дає змогу легко надсилати push-повідомлення без складного налаштування сервера.

До переваг Firebase належать підтримка різних способів автентифікації,

зокрема через електронну пошту, Google, Facebook і GitHub; можливість зберігати та синхронізувати дані в реальному часі; наявність готових API для швидкої інтеграції; вбудовані засоби безпеки, що працюють на рівні вузлів даних; зручне хмарне сховище на базі Google Cloud Storage; підтримка статичного хостингу та масштабована інфраструктура, яка дозволяє адаптувати застосунок до будь-якого обсягу трафіку.

Разом із цим, у Firebase є й певні недоліки. Йдеться насамперед про обмежену гнучкість запитів через специфіку потокової моделі обробки даних. Також платформа не підтримує класичні реляційні бази даних, оскільки побудована на основі NoSQL-підходу, що може ускладнювати реалізацію складних зв’язків між даними.

2.8 Бібліотека Material UI

Material UI (MUI) — це CSS-фреймворк, що надає набір готових компонентів для React і ґрунтується на концепціях Material Design від компанії Google, які були вперше представлені у 2014 році. MUI дозволяє швидко створювати сучасні користувацькі інтерфейси для веб- та мобільних застосунків, забезпечуючи відповідність єдиному дизайну. Основною ідеєю Material Design є забезпечення послідовного користувацького досвіду незалежно від пристрою чи платформи, з якої відбувається взаємодія з продуктом.

Material Design охоплює чітко визначені принципи щодо типографіки, макетів, відступів, масштабування, колірної гами, зображень тощо, дозволяючи дизайнерам створювати ієрархічно структуровані, інтуїтивно зрозумілі й естетично привабливі інтерфейси.

Бібліотека MUI для React на сьогодні є однією з найпопулярніших серед розробників, має понад 76 тисяч зірочок на GitHub і вважається сучасним інструментом для UI-розробки. Завдяки попередньо стилізованим компонентам та підтримці налаштувань, вона дозволяє створювати стильні застосунки за

короткий час. MUI базується на Leaner Style Sheets (LESS) — розширенні для CSS, що спрощує розробку.

Використання Material UI має низку суттєвих переваг. Насамперед, бібліотека містить велику кількість готових компонентів інтерфейсу, які легко інтегруються в застосунок. Це дає змогу швидко створювати як функціональні, так і візуально привабливі елементи дизайну. Основою MUI є система Material Design — продумана концепція проектування, що включає детальні інструкції щодо використання компонентів і піктограм, які відповідають загальній стилістиці інтерфейсу. Однією з важливих функцій є гнучке налаштування теми: за допомогою інструментів, які надає бібліотека, можна змінювати палітри кольорів, властивості поверхні та інші параметри, забезпечуючи єдине стилістичне оформлення всіх елементів інтерфейсу [19].

Документація MUI є чітко структурованою, містить численні приклади використання, що значно полегшує процес навчання та впровадження цієї бібліотеки у проєкти. Крім того, бібліотека постійно оновлюється командою розробників, яка активно виправляє помилки й залучає спільноту до процесу розвитку, зокрема через щорічні опитування. Важливою перевагою також є активна спільнота користувачів — велика кількість доступних прикладів, форумів, навчальних матеріалів і відкритий код дозволяють швидко знаходити рішення та отримувати допомогу.

2.9 Розробницьке середовище WebStorm

Середовище розробки WebStorm (рис. 2.11) — це інтегроване середовище розробки (IDE), створене компанією JetBrains спеціально для фронтенд-розробки з використанням JavaScript, HTML і CSS. Воно побудоване на основі платформи IntelliJ IDEA та вирізняється високим рівнем функціональності для реалізації сучасних веб-проєктів.

WebStorm надає розробникам розширену підтримку мов JavaScript, TypeScript, HTML, CSS, а також платформ Node.js і сучасних фреймворків,

зокрема Angular, React, Vue.js і Meteor. IDE забезпечує інтелектуальне автодоповнення, розпізнавання помилок у процесі написання коду, навігацію, рефакторинг і підсвічування синтаксису — усе це значно спрощує процес розробки. Завдяки поєднанню зручності, гнучкості, широкого функціоналу та активної підтримки спільноти, Material UI є одним із найкращих рішень для створення інтерфейсів у сучасних React-застосунках.

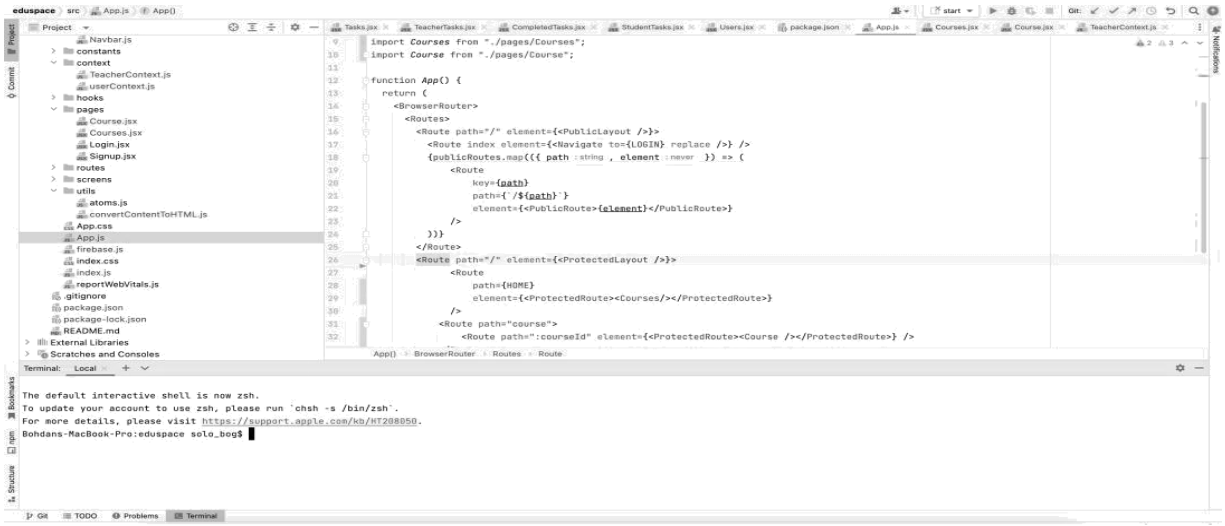

Рисунок 2.11 — Головне вікно IDE WebStorm

Однією з переваг WebStorm є можливість роботи з різними інструментами без необхідності додаткового встановлення сторонніх програм. Сотні вбудованих перевірок допомагають виявити потенційні помилки та негайно запропонувати варіанти їх виправлення, що сприяє підвищенню якості розробленого програмного забезпечення.

Інструменти навігації дозволяють легко переходити до визначення функцій, методів чи змінних, що особливо важливо при роботі з великими проєктами. IDE також підтримує налагодження коду як на стороні клієнта, так і на стороні сервера (Node.js), з можливістю встановлення точок зупину, покрокового виконання та перевірки значень змінних безпосередньо у середовищі розробки.

WebStorm легко інтегрується з популярними інструментами командного рядка, такими як Grunt, Gulp та npm, надаючи зручний інтерфейс для виконання задач збирання. Також підтримуються зовнішні лінтери: ESLint, TSLint, Stylelint,

JSHint, JSLint тощо. Окрім цього, функція локальної історії дозволяє відстежувати всі зміни в коді та повертатися до попередніх версій у разі потреби.

Основні переваги WebStorm:

— розвинена система рефакторингу коду;

— зручна навігація між файлами, класами та методами;

— інтелектуальне автодоповнення та підказки;

— аналіз коду в режимі реального часу;

— інтеграція із системами контролю версій (Git, SVN тощо) для миттєвого перегляду змін.

У межах дослідження були розглянуті інструменти та технології, що застосовуються під час створення програмного продукту «Освітня веб-платформа». Описано HTML, CSS, JavaScript, популярні JS-фреймворки, хмарну базу даних Firebase, бібліотеку Material UI, а також середовище розробки WebStorm. Кожен інструмент проаналізовано з точки зору функціональності, зручності використання та доцільності впровадження у проєкт.

3 РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ НАВЧАЛЬНИМ КОНТЕНТОМ

3.1 Опис системи

Процес розробки програмних продуктів починається з визначення вимог та вихідних даних. В результаті аналізу вимог формується специфікація програмного забезпечення, яка подається у вигляді текстових описів, структурних схем та діаграм. На етапі формування специфікацій створюють загальну модель предметної області та деталізують основні функції програмного продукту й його поведінку при взаємодії з оточенням.

Специфікація вимог до програмного забезпечення (англ. Software Requirements Specification, SRS) — це документ, який має на меті надати вичерпний опис розроблюваного продукту, включаючи його призначення, ключові бізнес-процеси, функціональність, основні параметри продуктивності та поведінкові характеристики [20]. По суті, SRS виступає своєрідною картою, що спрямовує весь процес розробки і допомагає всім учасникам залишатися на одному напрямку.

Зазвичай SRS затверджується наприкінці етапу проектування вимог — найранішого етапу розробки ПЗ. Документ охоплює як функціональні, так і нефункціональні вимоги. Функціональні вимоги визначають завдання системи та її компонентів (наприклад, резервування книг у бібліотечній системі), а нефункціональні — характеристики роботи системи, такі як безпека чи доступність.

Відповідно до стандарту IEEE 830-1998, визначаються методи та рекомендації щодо складання SRS, що допомагає замовникам чітко формулювати очікування, а розробникам — розуміти потреби клієнтів.

Окрім створення основи для успішної розробки через налагодження взаємодії між замовником і виконавцем, SRS має й інші переваги. Він дозволяє виявляти та виправляти помилки ще на стадії проектування, уникати

суперечностей у вимогах, а також взаємодіяти із зацікавленими сторонами для уточнення завдань.

Внесення змін на ранніх етапах розробки значно дешевше, ніж після початку програмування, оскільки пізніше це потребує значних витрат часу та ресурсів. Якісно складена SRS оптимізує процес розробки, запобігаючи дублюванню роботи та спрощуючи вирішення проблем.

Інша документація, як технічна, так і управлінська, базується на SRS, що забезпечує їх узгодженість і точність.

Документи SRS відрізняються залежно від проекту, оскільки методи розробки можуть варіюватися (водоспад, Agile тощо). Проте існує базова структура, яку зазвичай включають:

— вступ (мета, аудиторія, передбачуване використання, сфера застосування, сфера застосування, скорочення і терміни);

— загальний опис (потреби користувачів, припущення та обмеження);

— характеристика системи (функціональні вимоги, вимоги до зовнішніх інтерфейсів, особливості системи, нефункціональні вимоги).

Перший розділ дає загальну інформацію про продукт, його цілі, цільову аудиторію і сферу використання. Другий описує користувацькі потреби та можливі обмеження. Останній розділ конкретизує функціональні та нефункціональні вимоги.

Прикладом реалізації SRS є документ, що описує створення освітньої веб-платформи, яка покликана покращити навчальний процес, підвищити рівень інтерактивності та зацікавленості студентів. Для досягнення цієї мети необхідна чітка технічна документація, яку забезпечує специфікація вимог до програмного забезпечення (SRS).

Мета полягає у створенні освітньої платформи, яка дозволить викладачам створювати курси та завдання до них, а студентам — виконувати ці завдання, взаємодіяти з матеріалами та підтримувати комунікацію з викладачами. Основною аудиторією платформи є викладачі, які розміщують навчальні

матеріали, і студенти, які мають доступ до цих матеріалів через інтернет. Система є веб-додатком, призначеним для підтримки онлайн-навчання, подібно до існуючих рішень, таких як Google Classroom чи Moodle.

Після пандемії 2020 року значно зріс попит на дистанційні освітні інструменти. Платформа має на меті організувати ефективне онлайн-навчання та забезпечити контроль за успішністю студентів у конкурентному середовищі. Основні функції платформи включають можливість створення завдань, їх призначення студентам, перевірку виконання, виставлення оцінок і підтримання постійної комунікації між учасниками освітнього процесу.

У системі передбачено два типи користувачів: викладачі та студенти. Обидві категорії проходять авторизацію через реєстраційну форму або логін, після чого отримують доступ до дашборду з курсами, до яких вони належать. Веб-додаток реалізовано за допомогою JavaScript і бібліотеки React, а серверна частина працює на базі Firebase, що забезпечує функції авторизації, бази даних Firestore та зберігання файлів.

Інтерфейс системи простий та інтуїтивно зрозумілий, що робить його доступним для широкого кола користувачів без потреби у спеціальній підготовці. Реєстрація користувачів передбачає введення електронної пошти та пароля або вхід через обліковий запис Google. Система перевіряє коректність введених даних та повідомляє про помилки.

Після авторизації користувач отримує доступ до своїх курсів. Він може переглядати їхній список, переходити до конкретного курсу, приєднуватися до нового курсу за ID або створювати новий курс. Викладачі мають змогу публікувати оголошення, які одразу стають доступними студентам, а ті, своєю чергою, можуть залишати коментарі. Завдання створюються викладачем через окрему вкладку курсу, з можливістю додавання опису, дедлайну та прикріплення файлів. Студенти виконують завдання та завантажують їх через відповідну вкладку, після чого викладач може їх перевірити.

Нижче представлено табл. 3.1 з описом зовнішніх та нефункціональних вимог до програмного забезпечення. Вона охоплює характеристики інтерфейсів, вимоги до продуктивності, безпеки та якості роботи системи.

Таблиця 3.1 — Зовнішні та нефункціональні вимоги

Категорія	Тип	Опис
Зовнішні вимоги до інтерфейсу	Користувацький інтерфейс	Інтерфейс для взаємодії користувача з платформою через веббраузер
	Front-end	React версії 17.0
	Back-end	Firebase
	Інтерфейс з апаратним забезпеченням	Взаємодія програмного забезпечення з операційною системою та браузером користувача
	Операційна система	Windows
	Веббраузер	Підтримка HTML та JavaScript
Нефункціональні вимоги	Швидкодія	Завантаження додатка зі швидкістю не менше 1 Мб/с База даних повинна витримувати до 500 запитів на секунду
	Безпека	Захист усіх файлів та даних користувачів від несанкціонованого доступу
	Якість програмного забезпечення	Покриття тестами не менше 90% коду Індекс продуктивності за Google PageSpeed понад 90 балів

3.2 Проектування структури

Проаналізувавши постановку задач, реалізованих у розроблюваній системі, можна виділити ключові абстракції предметної області, які наведені в табл. 3.2.

Таблиця 3.2 — Абстракції системи

Абстракція	Опис
Користувач	Особа, яка ще не приєдналася до конкретного курсу, тобто не є ані студентом, ані викладачем.
Курс	Потік завдань, у межах якого відбувається взаємодія між учасниками: обговорення, виконання та оцінювання завдань.
Студент	Користувач, який приєднався до курсу за ідентифікатором (ID).
Викладач	Користувач, який створив власний курс.
Оголошення	Повідомлення для всіх учасників курсу, які може створювати лише викладач.
Завдання	Уся необхідна інформація для виконання завдання студентом.
Модальне вікно	Елемент інтерфейсу, що блокує роботу користувача до моменту його закриття.

На основі виділених абстракцій побудовано діаграму прецедентів, що відображає взаємодію об’єктів між собою (рис. 3.1). У табл. 3.3 розглянуто основні характеристики поведінки абстракції, які відображають її роль у процесах аналізу, моделювання та систематизації даних. Вона демонструє ключові аспекти взаємодії абстрактних моделей з реальними об'єктами та явищами, що важливо для забезпечення структурованого підходу до проектування та оцінки складних систем.

У контексті побудови моделі взаємодії системи з користувачами важливо виділити ключові поведінкові характеристики кожної абстракції. Це дозволяє формалізувати функціональні ролі в системі та чітко окреслити обов’язки користувачів у межах навчальної платформи. Такий підхід сприяє кращому

розумінню структури системи, спрощує процес проєктування інтерфейсів і дає змогу ефективно описати сценарії використання [21]. Ці характеристики є основою для побудови сценаріїв взаємодії, визначення функціональних вимог і моделювання поведінки системи в типових ситуаціях.

Таблиця 3.3 — Характеристики поведінки абстакції

Абстракція	Поведінка	Опис поведінки
Користувач	Вхід/реєстрація, участь у курсах	Користувач входить або реєструється в системі, переглядає доступні курси. Може створити курс як викладач або приєднатися до нього як студент.
Курс	Структуризація даних курсу	Об'єднує завдання, оголошення, коментарі, а також містить список учасників.
Студент	Коментування, здача завдань	Має доступ до оголошень викладача, може коментувати їх, переглядати список завдань та надсилати виконані роботи.
Викладач	Створення контенту та оцінювання	Створює завдання та оголошення, переглядає надіслані роботи студентів та виставляє оцінки.
Модальне вікно	Повідомлення про результати дій	З'являється при внесенні змін або видаленні об'єктів у системі.
Оголошення	Передача інформації	Відображається на сторінці курсу в розділі оголошень. Студенти можуть залишати коментарі.
Завдання	Постановка та виконання	Створюється викладачем, міститься у відповідному розділі курсу. Студенти надсилають виконані завдання, які згодом перевіряються.

Діаграма прецедентів (use case diagram) на рис. 3.1 відображає основних акторів системи та їх взаємодію з функціональністю веб-застосунку.

У системі виділено трьох головних акторів:

— користувач — має можливість зареєструватися або увійти в систему, після чого може обрати роль викладача або студента;

— викладач — після створення курсу отримує доступ до функцій управління курсом: створення завдань, публікація оголошень, перегляд робіт студентів і виставлення оцінок;

студент — може приєднатися до курсу, переглядати оголошення, коментувати їх, виконувати завдання та надсилати їх на перевірку.

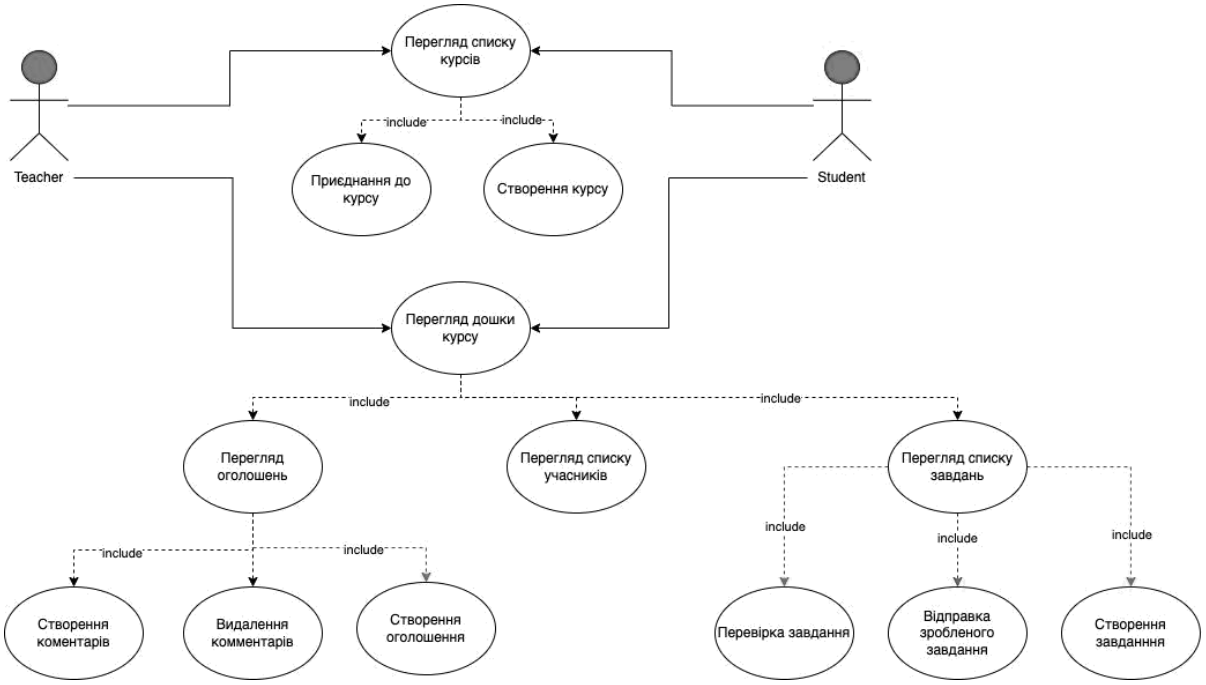

Рисунок 3.1 — Діаграма прецедентів веб-застосунку

Кожна дія користувача представлена як прецедент, що ілюструє логіку взаємодії між користувачами та системою. Така діаграма дозволяє чітко структурувати функціональні можливості програмного забезпечення, забезпечити повноту реалізації вимог та спростити процес подальшого проєктування інтерфейсів та реалізації логіки додатку.

3.3 Модель бази даних

База даних побудована на основі NoSQL Firestore Database, що є частиною хмарної платформи Firebase, розробленої компанією Google. Використання саме Firestore зумовлено вимогами до гнучкості, масштабованості та реального часу, що є критично важливими для систем управління навчальним контентом.

На відміну від реляційних баз даних, Firestore оперує моделлю даних документ-колекція, яка дозволяє ефективно зберігати та обробляти дані у вигляді JSON-подібних об’єктів. У контексті даного застосунку така структура дозволяє

швидко здійснювати читання, запис, оновлення та видалення інформації як у локальному кеші, так і в хмарному середовищі, забезпечуючи при цьому високу продуктивність та синхронізацію даних між клієнтами в режимі реального часу.

Архітектура бази даних повністю відповідає функціональній моделі системи, яка включає кілька основних сутностей: користувачів, курси, завдання, відповіді, повідомлення, коментарі тощо.

Короткий опис основних колекцій:

— users — містить документи з інформацією про всіх зареєстрованих користувачів. У кожному документі зберігаються такі поля, як ім’я, електронна пошта, роль (студент або викладач), UID та додаткові параметри;

— courses — зберігає дані про створені навчальні курси, включаючи назву, опис, автора (викладача), список учасників тощо;

— assignments — колекція із завданнями, що прив’язуються до певного курсу. Кожне завдання містить тему, інструкції, дедлайн та список надісланих рішень;

— submissions — відповіді студентів на конкретні завдання. Містять прикріплені файли, дату відправки, оцінку тощо;

— announcements — використовується для оголошень викладачів у рамках курсу. Оголошення мають текст, дату публікації, автора;

— comments — дозволяє користувачам залишати коментарі до оголошень або завдань, підтримуючи інтерактивну взаємодію в межах системи.

Важливою особливістю даної структури є глибока ієрархічна організація даних. Наприклад, завдання можуть містити підколекції з відповідями студентів, а коментарі можуть вкладатися в документи курсів або оголошень. Такий підхід дозволяє будувати зв’язки між сутностями без необхідності в складних JOIN-операціях, характерних для SQL.

Додатково реалізовано правила безпеки Firestore Security Rules, які обмежують доступ до певних даних на основі автентифікації користувача та його ролі в системі. Це гарантує високий рівень захисту персональної та навчальної

інформації, зокрема відповідає нефункціональним вимогам безпеки, визначеним у технічному завданні.

На рис. 3.2 представлена загальна схема логічної структури бази даних, яка демонструє взаємозв’язки між основними колекціями та підколекціями.

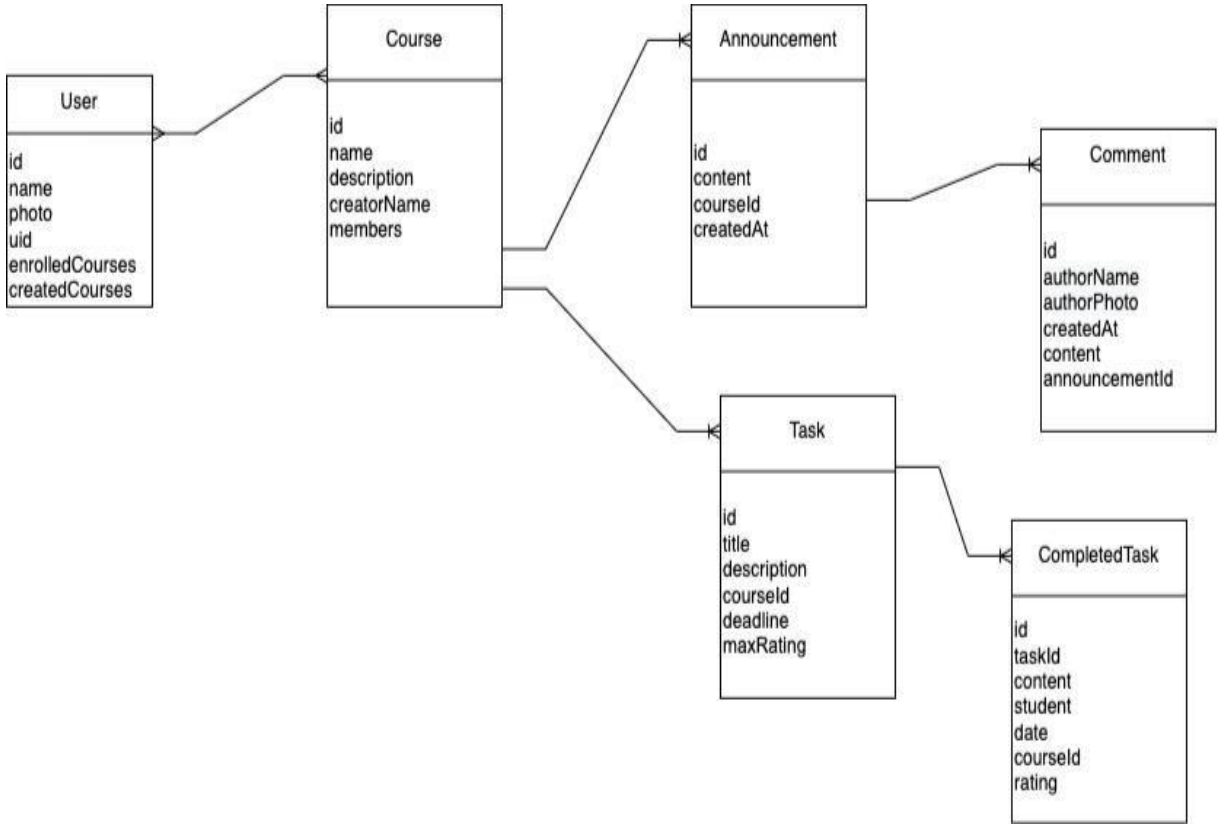

Рисунок 3.2 — Логічна структура бази даних

Така схема дозволяє чітко уявити, як організовані дані, як відбувається навігація між ними, та як забезпечується ефективна взаємодія з базою з боку фронтенду застосунку. Крім того, така структура полегшує масштабування системи та подальше розширення функціоналу без порушення логіки збереження даних.

У результаті, розроблена структура забезпечує гнучкість у роботі з даними, спрощує реалізацію CRUD-операцій і дозволяє уникнути дублювання інформації.

3.4 Розробка шаблону додатку

React.js є однією з найпопулярніших JavaScript-бібліотек, яка широко використовується для створення сучасних односторінкових вебзастосунків. Її переваги полягають у високій швидкодії, гнучкості компонентного підходу та підтримці віртуального DOM, що забезпечує швидке оновлення інтерфейсу без перезавантаження сторінки.

Попри всі переваги, початок роботи з React.js іноді може виявитися складним для новачків, оскільки вимагає конфігурації середовища розробки, зокрема налаштування транслятора JavaScript-коду (наприклад, Babel) і засобу збирання проєкту (Webpack). Щоб спростити цей процес, розробниками було створено готове рішення — інструмент автоматичного створення проєкту на базі React.

Зазначений інструмент дозволяє швидко розгорнути структуру проєкту з усіма базовими налаштуваннями. Його встановлення передбачає використання системи керування пакетами, після чого розробник може згенерувати шаблон програми й одразу перейти до розробки. Готовий шаблон включає усі основні файли, попередньо налаштовані для комфортної роботи, і дозволяє запустити локальний сервер, що автоматично оновлює сторінку при зміні коду.

Особливістю React є одностороння прив’язка даних, коли інформація передається лише в одному напрямку — від батьківського компонента до дочірнього. Це дозволяє більш точно контролювати логіку застосунку і спрощує пошук джерел змін або помилок. Крім того, компоненти в React організовуються як окремі модулі, кожен з яких може містити власний стан і набір властивостей, що сприяє повторному використанню коду.

У межах розробки клієнтської частини системи управління навчальним контентом було створено React-застосунок на основі вищезгаданого шаблону. Після його генерації було додано необхідні бібліотеки для підключення до серверної частини, зокрема інструменти взаємодії з хмарною платформою Firebase. Після встановлення відповідних бібліотек стала доступною інтеграція з

базою даних, можливість автентифікації користувачів, зчитування і запис інформації в режимі реального часу.

У результаті використання React.js дозволило створити динамічний, гнучкий та інтуїтивно зрозумілий інтерфейс, який ефективно реагує на дії користувача та забезпечує високу швидкість взаємодії з даними без потреби перезавантаження сторінки.

3.5 Опис основних програмних модулів

Розроблений веб-сервіс має чітко структуровану файлову організацію, що забезпечує зручність підтримки, масштабування та розширення проєкту. Фрагмент файлової структури розробленого веб-додатку зображено на рис. 3.3.

Основна структура каталогів та файлів подана нижче:

— public — директорія, що містить публічні ресурси, доступні напряму з браузера;

— src — основна папка з ресурсами веб-додатку. Містить такі підкаталоги та файли;

— index.js — головний JavaScript-файл, який підключається до HTML і виконує ініціалізацію React-додатку;

— context — папка з контекстами React, що дозволяють реалізувати глобальний стан застосунку;

— components — набір багаторазових React-компонентів, які використовуються для побудови інтерфейсу користувача;

— routes — реалізація маршрутизації між сторінками;

— App.css — файл стилів, що застосовуються до всього додатку;

— constants — каталог зі статичними значеннями, які використовуються в застосунку (наприклад, ролі користувачів, кольори тощо);

— hooks — набір власних (кастомних) React-хуків для повторного використання логіки;

— firebase.js — файл з конфігурацією Firebase та методами для аутентифікації й роботи з базою даних;

— App.js — головний компонент, з якого починається рендеринг усього інтерфейсу;

— utils — набір допоміжних функцій, що спрощують логіку окремих компонентів;

— index.css — глобальні стилі, які застосовуються до всього за стосунку;

— pages — окремі сторінки додатку (реєстрація, головна сторінка курсу тощо).

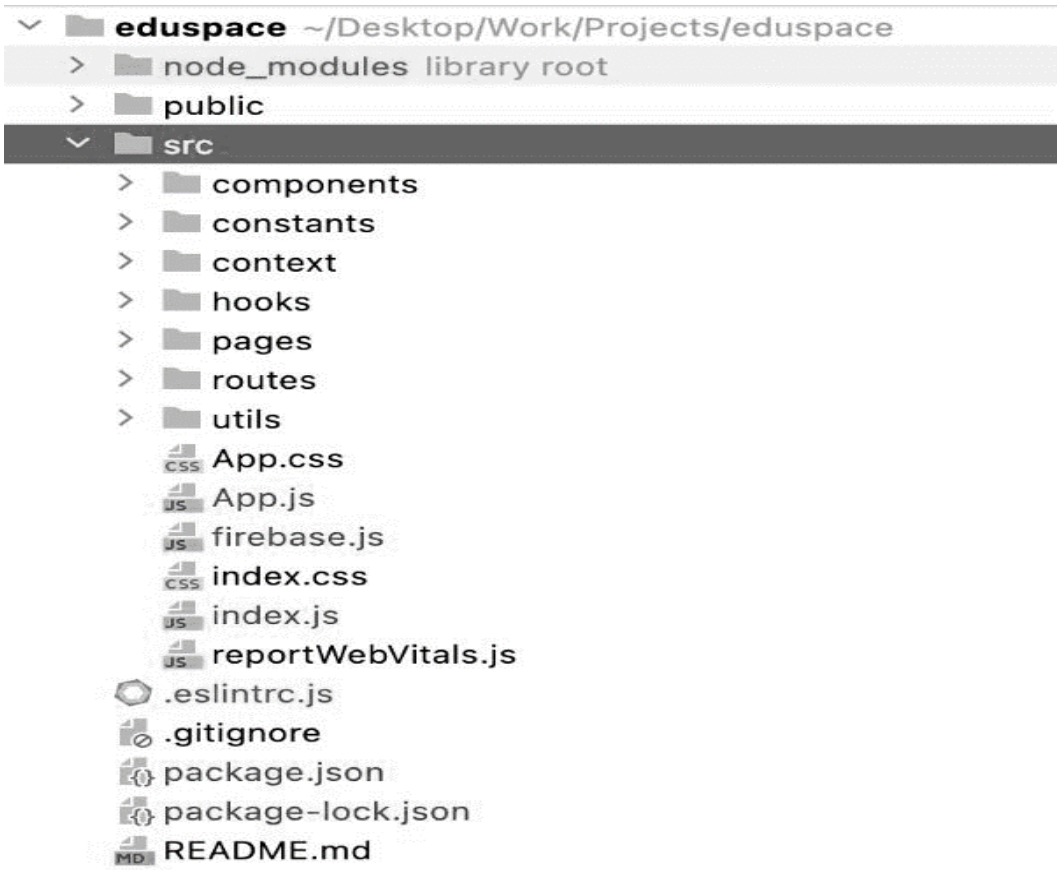

Рисунок 3.3 — Фрагмент файлової структури розробленого веб-додатку

Файл eslintrc.js відповідає за конфігурацію ESLint і допомагає дотримуватись єдиного стилю написання коду, запобігаючи синтаксичним і логічним помилкам. Файл .gitignore містить перелік директорій та файлів, які не

повинні потрапляти до репозиторію Git — це, наприклад, тимчасові або локальні налаштування середовища. package.json є ключовим конфігураційним файлом, у якому зберігається інформація про сам проєкт: його назва, версія, залежності, npm-скрипти, а також базові метадані. Файл package-lock.json, що створюється автоматично, фіксує точні версії встановлених пакетів, забезпечуючи стабільність та повторюваність збірки на різних машинах.

3.6 Логічні процеси системи

Для реалізації повного циклу користувацької взаємодії необхідно реалізувати кілька ключових логічних механізмів. Першим кроком є підключення до платформи Firebase, яка забезпечує сервіси аутентифікації та бази даних. Ініціалізація з'єднання з Firebase наведено в лістингу 3.1

Лістинг 3.1 — Ініціалізація з'єднання з Firebase

```
const firebaseConfig = {
  apiKey: "AIzaSyCkcUTPKzWk5u9rLcHen2W3t8Lna8mwY",
  authDomain: "eduspace-15506.firebaseapp.com",
  projectId: "eduspace-15506",
  storageBucket: "eduspace-15506.appspot.com",
  messagingSenderId: "895198289171",
  appId: "1:895198289171:web:4e0156630f43f1bb890ee6"};
const app = firebase.initializeApp(firebaseConfig);
const auth = app.auth();
```

Далі створюється функція, яка дозволяє новому користувачеві зареєструвати обліковий запис за допомогою пошти та паролю (лістинг 3.2)..

Зареєстрований користувач має змогу авторизуватись, використовуючи ті ж самі облікові дані (лістинг 3.3).

Лістинг 3.2 — Реалізація процесу реєстрації нового користувача

```
const signUpWithEmailAndPassword = async (
  email,
  password,
  firstName,
  lastName,
  onError,
) => {
  try {
```

```
    const { user } = await auth.createUserWithEmailAndPassword(email, password);
    const tempUser = { ...user, displayName: `${firstName} ${lastName}` };
    await checkUser(tempUser);
  } catch (err) {
    onError(err.message);
  }
};
```

Лістинг 3.3 — Вхід у систему для існуючого користувача

```
const [login, , loading, signInError] = useSignInWithEmailAndPassword(auth);
const handleSubmit = (e) => {
  e.preventDefault();
  setError(null);
  login(email, password);
};
```

Крім того, користувачі можуть проходити авторизацію через акаунт Google, що значно пришвидшує процес (лістинг 3.4).

Лістинг 3.4 — Аутентифікація через Google OAuth

```
const googleProvider = new firebase.auth.GoogleAuthProvider();
// Sign in and check or create account in firestore
const signInWithGoogle = async (onError) => {
  try {
    const response = await auth.signInWithPopup(googleProvider);
    await checkUser(response.user);
  } catch (err) {
    onError(err.message);
  }
};
```

Після входу до системи користувач може або створити новий навчальний курс, або приєднатись до вже існуючого (лістинг 3.5 — 3.6).

Лістинг 3.5 — Механізм приєднання до існуючого курсу

```
const joinClass = async (courseId) => {
  try {
    setIsJoinModalOpen(false);
    const classRef = await db.collection('courses').doc(courseId).get();
    if (!classRef.exists) {
      return alert('Class doesn\'t exist, please provide correct ID');
    }
    await classRef.ref.collection('users').add({
      name: user.name,
      joiningDate: moment().format('MMM Do YY'),
      courseId,
```

```
      studentUid: user.uid,
    });
    const classData = await classRef.data();
    await user.ref.collection('enrolledCourses').add({
      creatorName: classData.creatorName,
      id: courseId,
      name: classData.name,
      description: classData.description,
    });
    alert(`Enrolled in ${classData.name} successfully!`);
  } catch (err) {
    console.error(err);
    alert(err.message);
  }
};
```

Лістинг 3.6 — Створення нового навчального курсу

```
const createClass = async (courseName, courseDescription) => {
  setIsCreateModalOpen(false);
  try {
    const {
      id
    } = await db.collection('courses').add({
      creatorName: user.name,
      name: courseName,
      description: courseDescription,
      users: [],
    });
    await user.ref.collection('createdCourses').add({
      id,
      name: courseName,
      description: courseDescription,
    });
  } catch (err) {
    alert(`Cannot create class - ${err.message}`);
  }
};
```

Викладач, перебуваючи на спеціально розробленій сторінці свого курсу, має змогу легко та оперативно публікувати важливі оголошення для всіх учасників, що дозволяє ефективно інформувати студентів про зміни, завдання або актуальні події, як це детально реалізовано у лістингу 3.7.

Лістинг 3.7 — Компонент створення оголошень

```
const addPost = async (editorState) => {
  await db.collection('posts').add({
```

```
    content: convertContentToHTML(editorState),
    courseId,
    createdAt: firebase.firestore.FieldValue.serverTimestamp(),
  });
```

Оголошення можна коментувати, а також, за потреби, видаляти (лістинг 3.8).

Лістинг 3.8 — Функції коментування і видалення повідомлень

```
const addComment = async (editorState) => {
  await db.collection('comments').add({
    content: convertContentToHTML(editorState),
    postId: post.id,
    createdAt: firebase.firestore.FieldValue.serverTimestamp(),
    authorName: user.name,
    authorPhoto: user.photo,
  });
};
const deletePost = async () => {
  await db.collection('posts').doc(post.id).delete();
  const tempComments = await db.collection('comments')
    .where('postId', '==', post.id).get();
  tempComments.docs.forEach((doc /*: QueryDocumentSnapshot<DocumentData>*/ ) => {
    doc.ref.delete();
  });
};
```

Подальшим етапом у розвитку можливостей платформи для освітнього процесу є реалізація функціоналу, що надає викладачеві повний контроль над створенням та налаштуванням навчальних завдань для студентів, забезпечуючи гнучкість у формуванні навчального плану, як це імплементовано у лістингу 3.9.

Студенти надсилають свої відповіді на завдання через форму (лістинг 3.10).

Після чого викладач може оцінити подану роботу (лістинг 3.11).

Лістинг 3.9 — Додавання нового завдання

```
const createTask = async () => {
  try {
    await db.collection('tasks').add({
      name,
      maxRating: parseInt(maxRating, radix = 10),
      description: convertContentToHTML(description),
      deadline,
      courseId,
```

```
    });
    setName('');
    setDeadline(null);
    setMaxRating(30);
    setDescription(EditorState.createEmpty());
    handleClose();
  } catch (err) {
    alert(err.message);
  }
```

Лістинг 3.10 — Передача студентом виконаного завдання

```
const sendWork = (taskId) => {
  try {
    db.collection('completedTasks').add({
      taskId,
      student: user.name,
      studentUid: user.uid,
      date: firebase.firestore.FieldValue.serverTimestamp(),
      content: convertContentToHTML(content),
      courseId,
    });
    setContent(EditorState.createEmpty());
    setIsModalOpen(false);
  } catch (e) {
    alert(e.message);
  }
};
```

Лістинг 3.11 — Система оцінювання відповідей студентів

```
const evaluate = () => {
  try {
    db.collection('completedTasks').doc(id).update({
      rating: newRating,
    });
    setIsModalOpen(false);
  } catch (e) {
    alert(e.message);
```

3.7 Проектування користувацького інтерфейсу

Інтерфейс веб-застосунку має уніфікований вигляд для всіх сторінок і включає такі елементи: палітру кольорів, панель навігації, форму реєстрації, панель курсів і управління завданнями.

Початковий екран — це сторінка для реєстрації нових користувачів (див. рис.3.4).

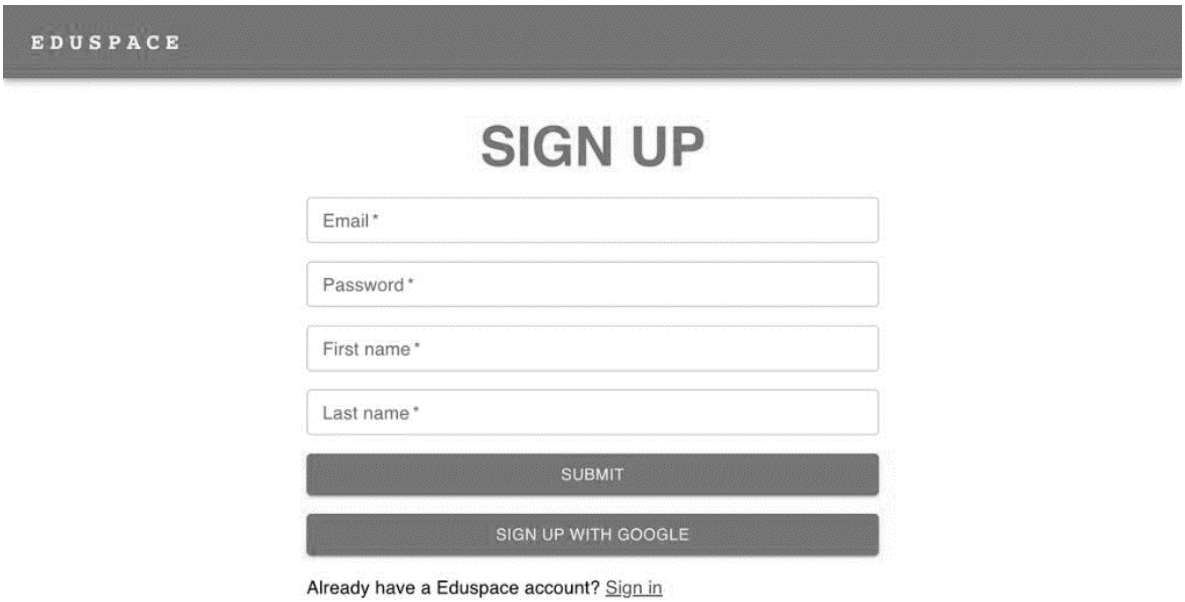

Рисунок 3.4 — Інтерфейс сторінки реєстрації

У разі помилок, наприклад, якщо пошта вже зареєстрована, з’являється відповідне повідомлення (див. рис.3.5).

Рисунок 3.5 — Повідомлення про реєстраційну помилку

Зареєстровані користувачі можуть авторизуватись (див. рис.3.6).

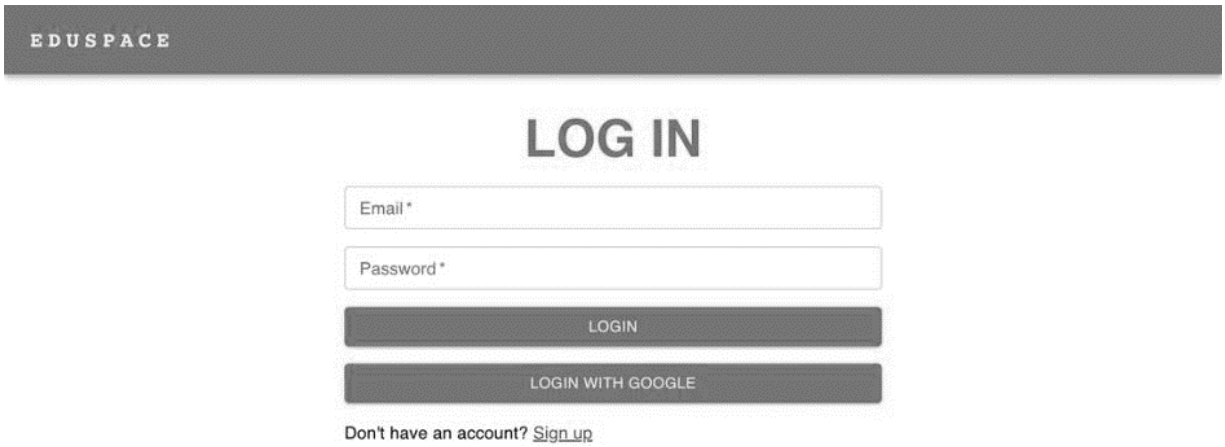

Рисунок 3.6 — Інтерфейс логіну

При введенні невірних даних система попереджає користувача (див. рис.3.7).

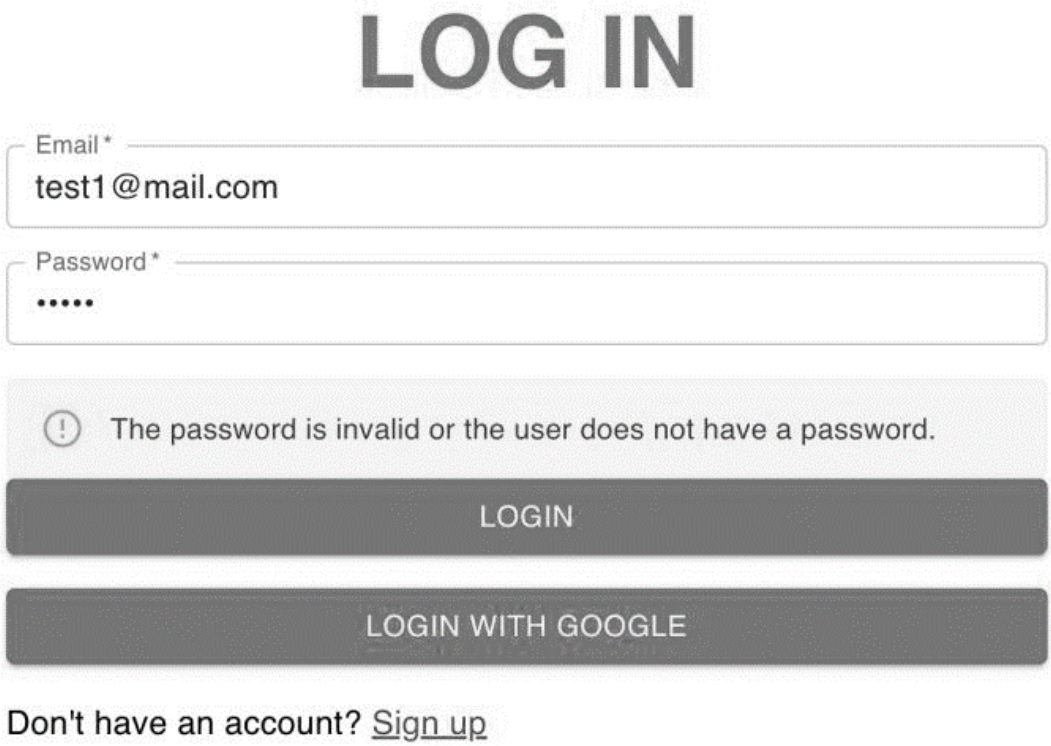

Рисунок 3.7 — Вивід помилки при неправильному логіні

Після входу користувач бачить перелік курсів, у яких бере участь (див. рис.3.8).

Кнопки для створення чи приєднання до курсів відкривають відповідні модальні вікна (див. рис.3.9-3.10).

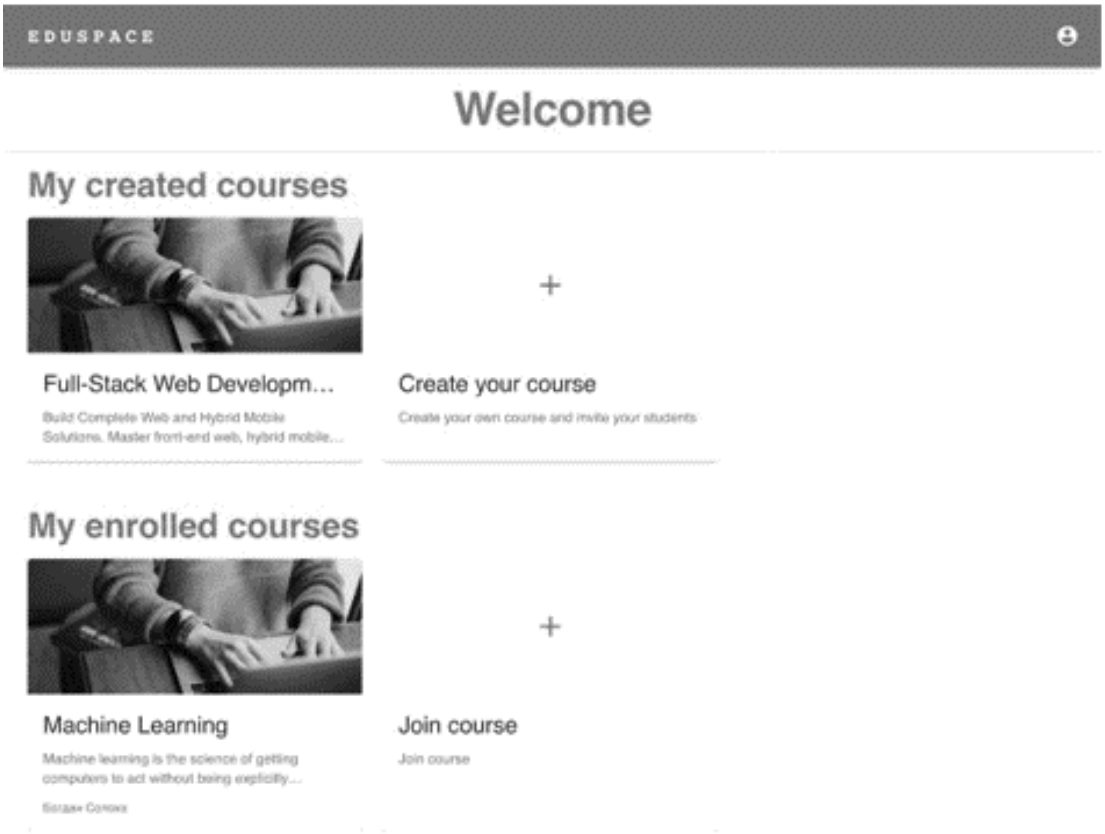

Рисунок 3.8 — Панель управління курсами користувача

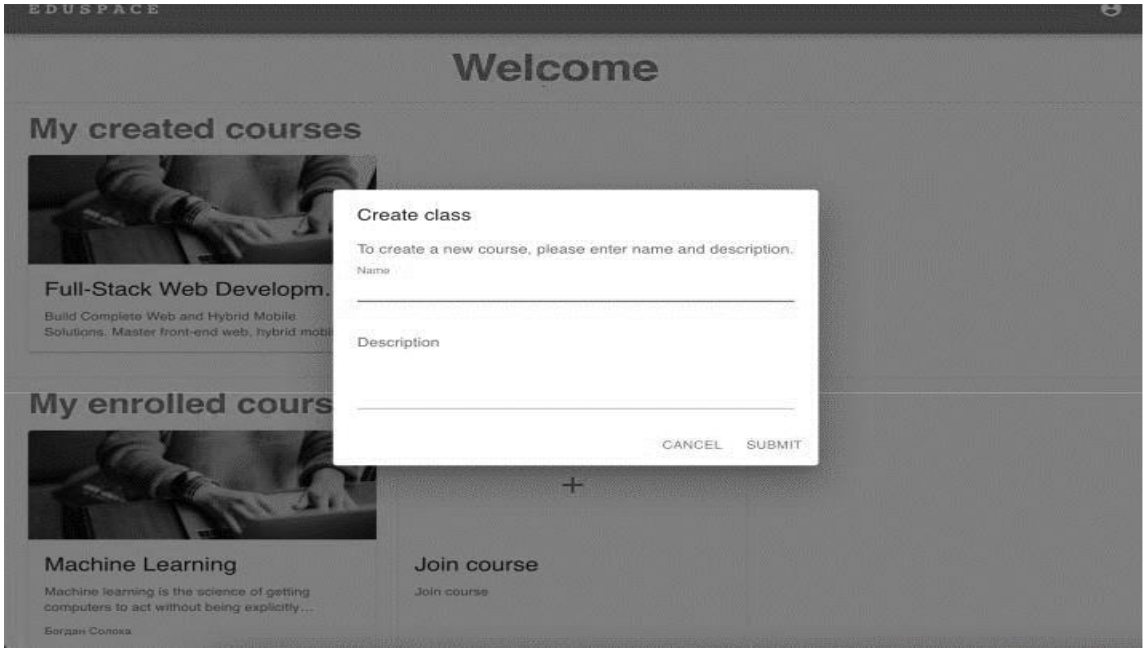

Рисунок 3.9 — Діалог створення нового курсу

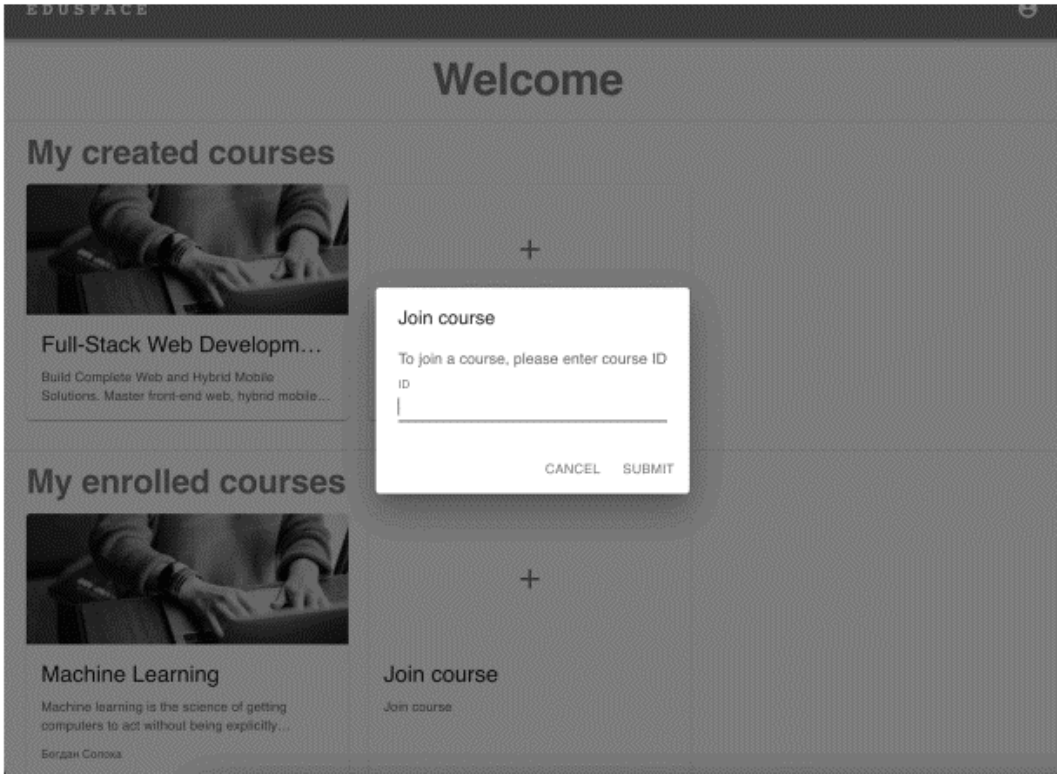

Рисунок 3.10 — Діалог приєднання до наявного курсу

Перехід до конкретного курсу дозволяє взаємодіяти з його вмістом (див. рис.3.11).

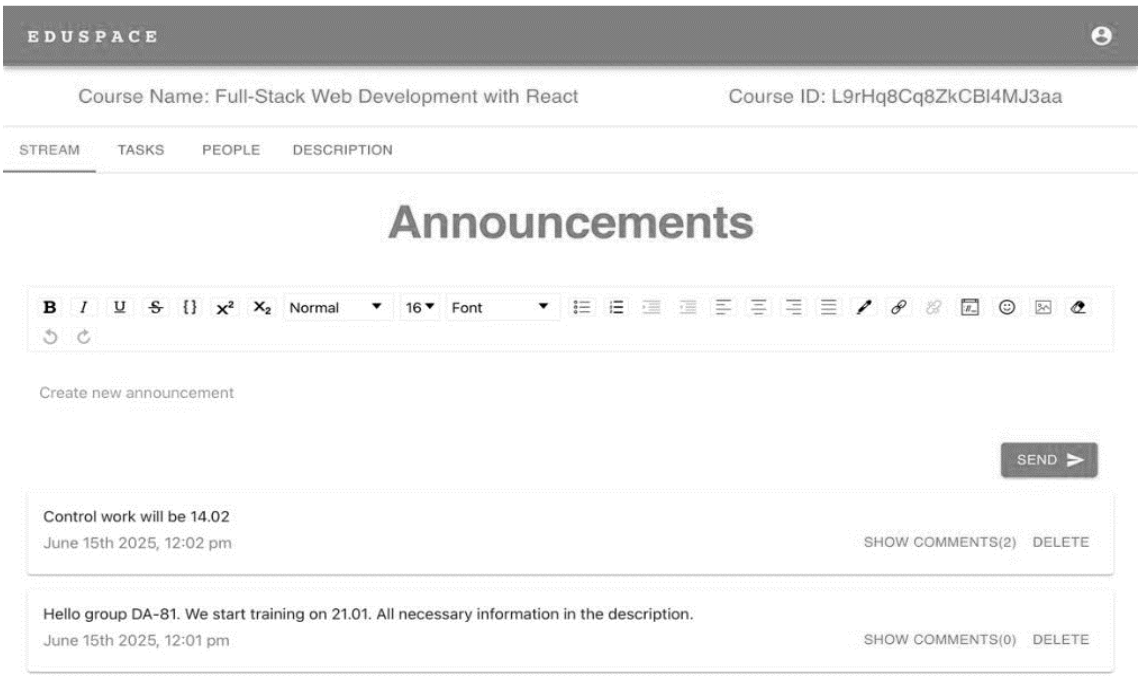

Рисунок 3.11 — Інтерфейс курсу для викладача

На вкладці «Потік» викладач може публікувати оголошення, а студенти — залишати коментарі (див. рис.3.12).

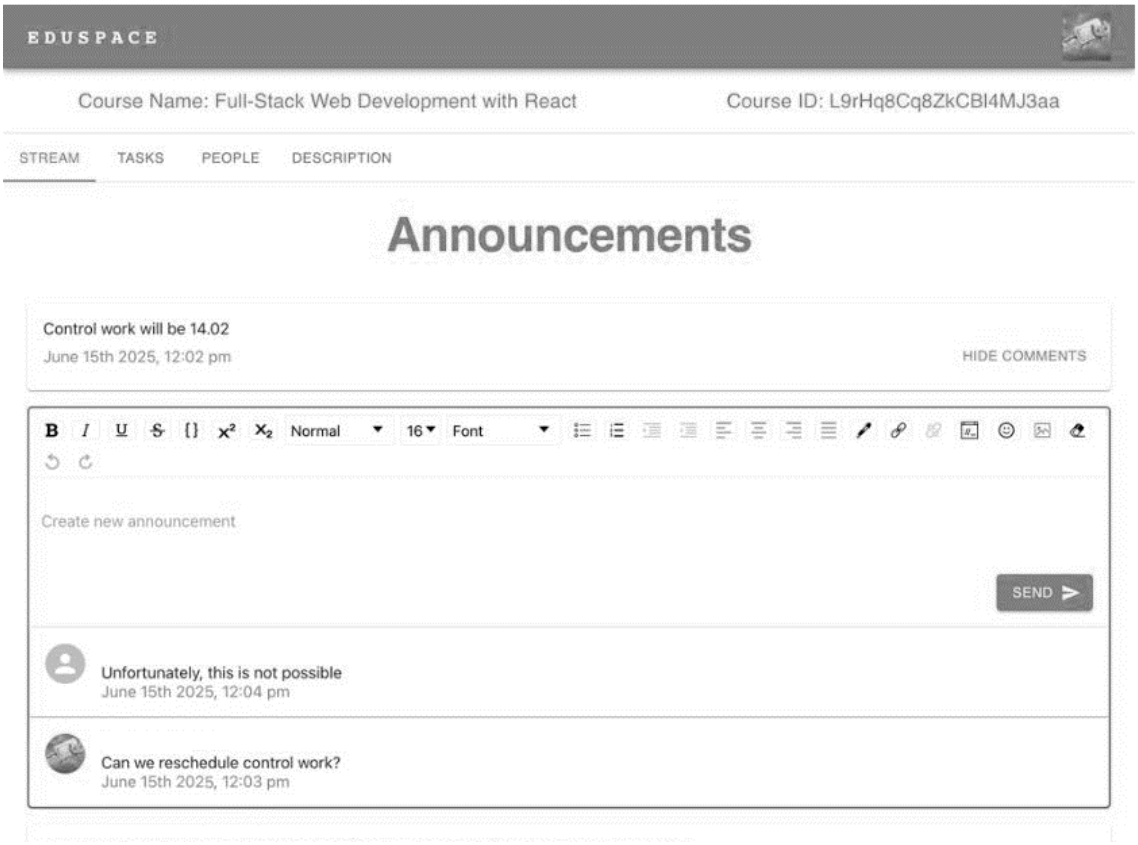

Рисунок 3.12 — Інтерфейс потоку оголошень для студента

Вкладка «Завдання» містить список завдань, які створює викладач (див. рис.3.13).

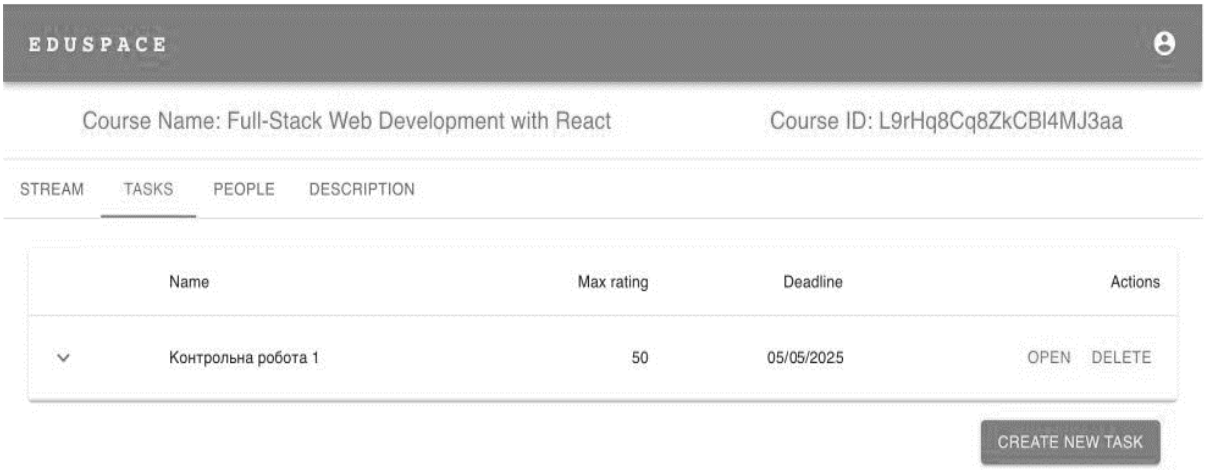

Рисунок 3.13 — Панель завдань для викладача

При створенні нового завдання відкривається форма з параметрами (див. рис.3.14).

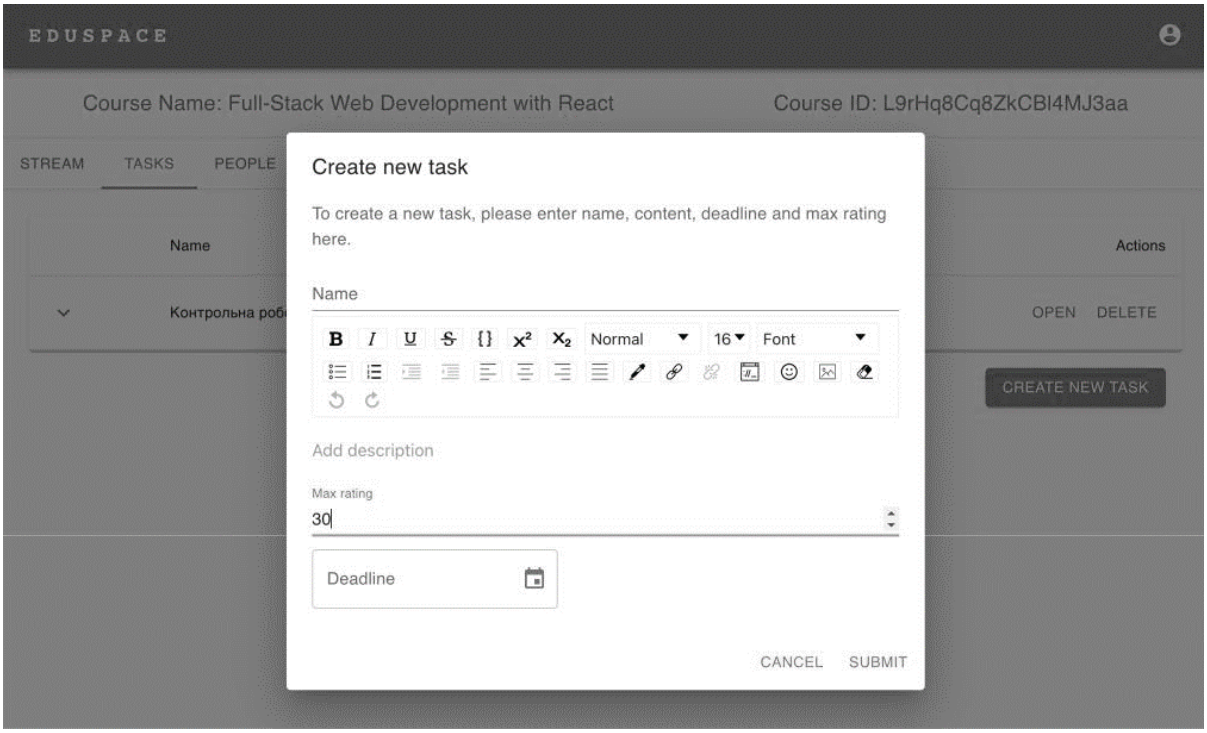

Рисунок 3.14 — Діалог створення нового завдання

Після цього завдання відображається у студентів (див. рис.3.15).

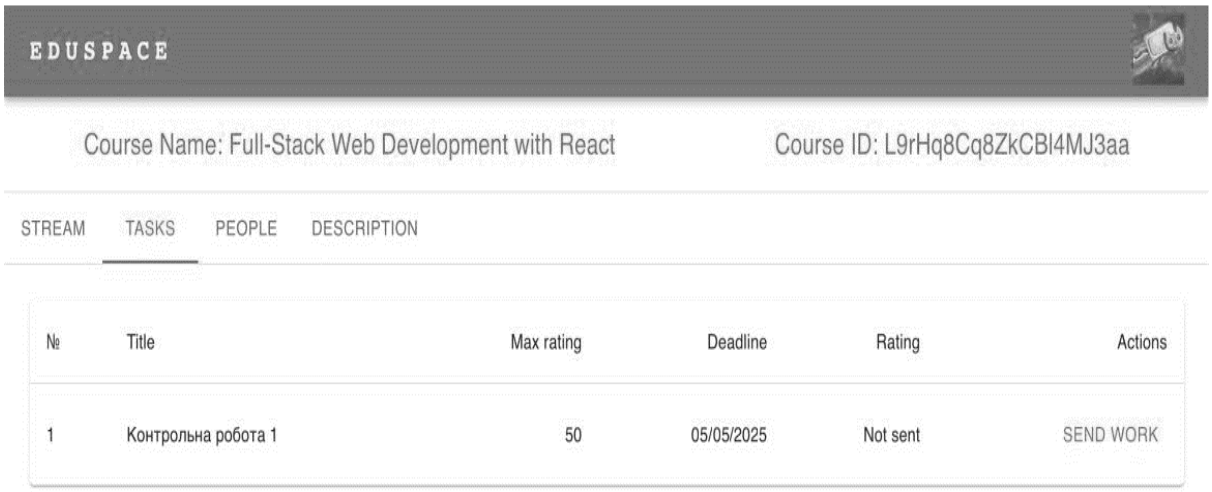

Рисунок 3.15 — Список завдань, доступних студенту

Студент відкриває завдання, читає опис і надсилає відповідь (див. рис.3.16).

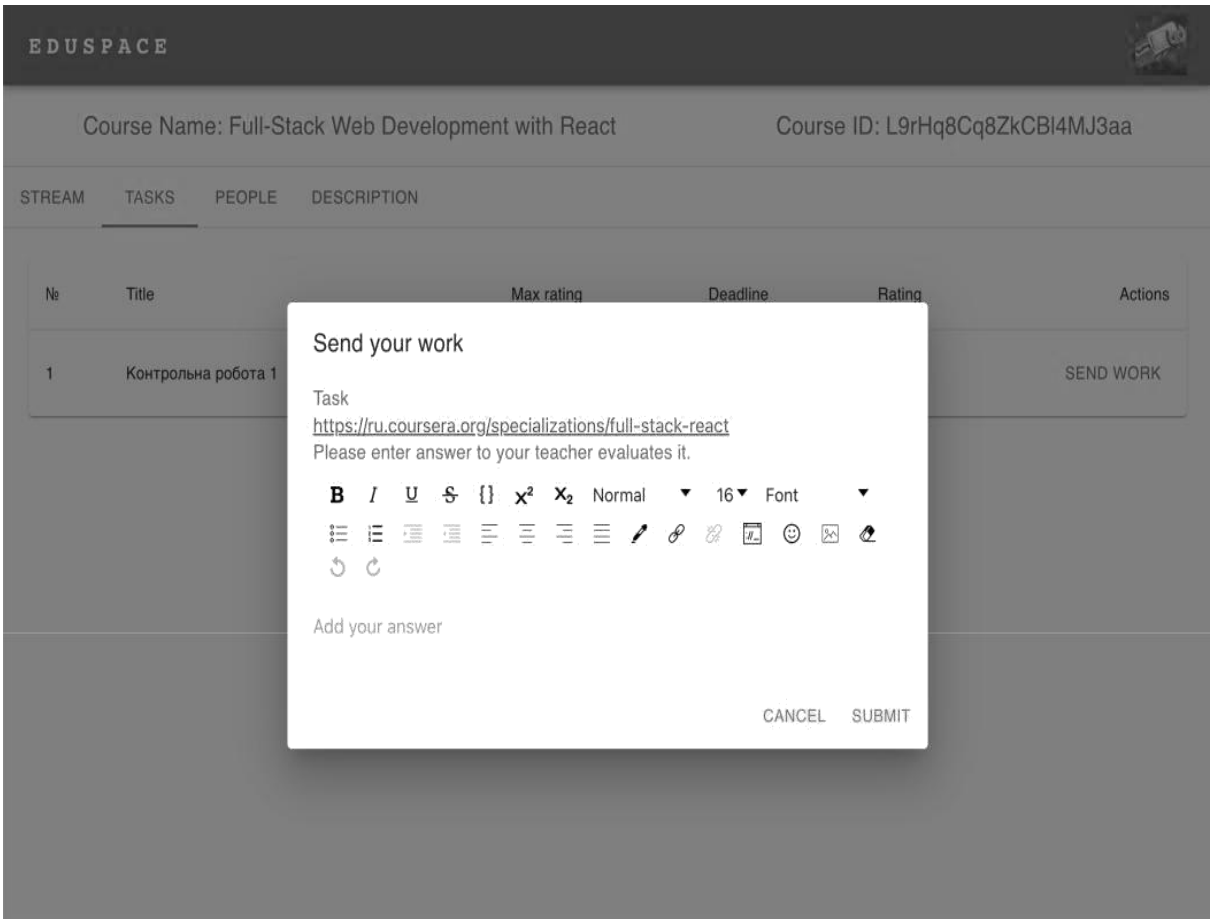

Рисунок 3.16 — Форма відповіді на завдання

Надіслана робота зберігається в системі (див. рис.3.17).

Рисунок 3.17 — Відображення поданої роботи

Викладач може переглядати та оцінювати ці відповіді (див. рис.3.18).

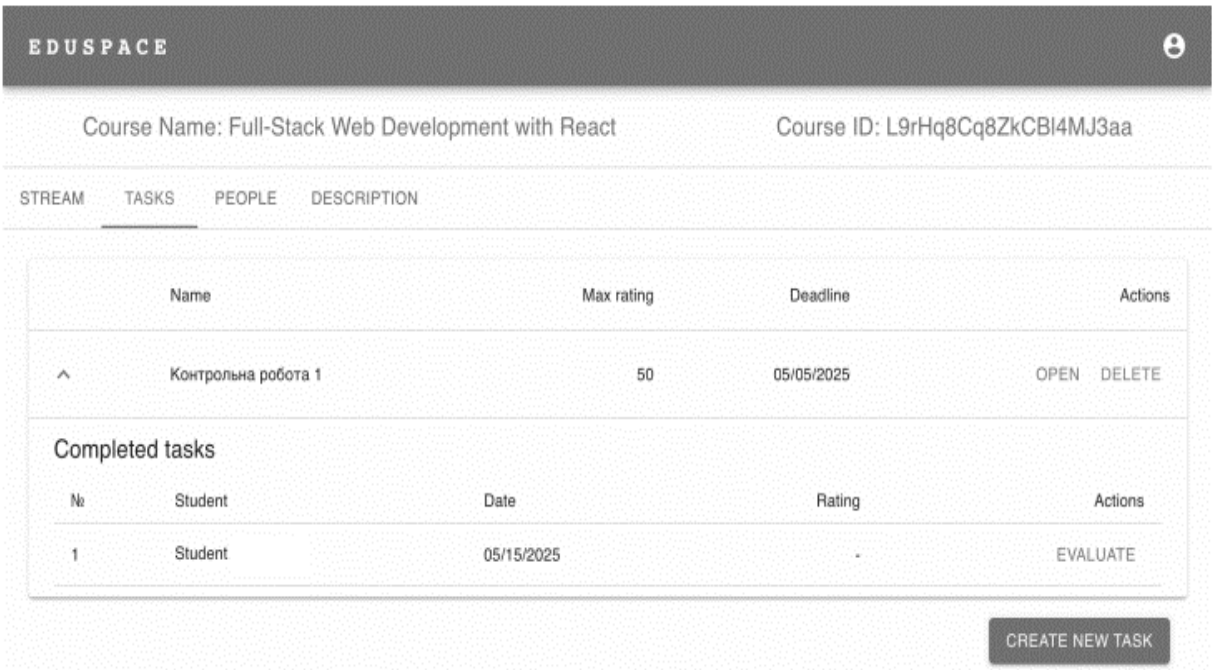

Рисунок 3.18 — Список відповідей на перевірку

Процес оцінювання реалізовано через спеціальне вікно (див. рис.3.19).

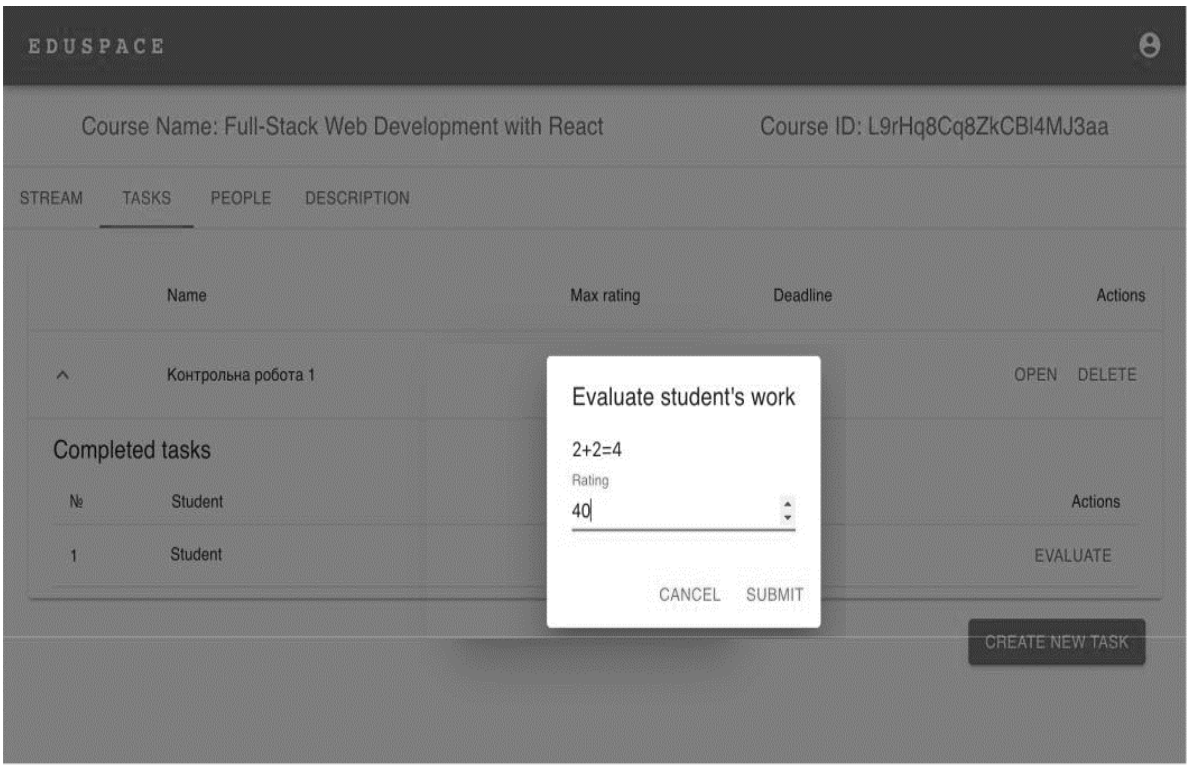

Рисунок 3.19 — Вікно перевірки і виставлення оцінки

Оцінки відображаються одразу в списку (див. рис.3.20).

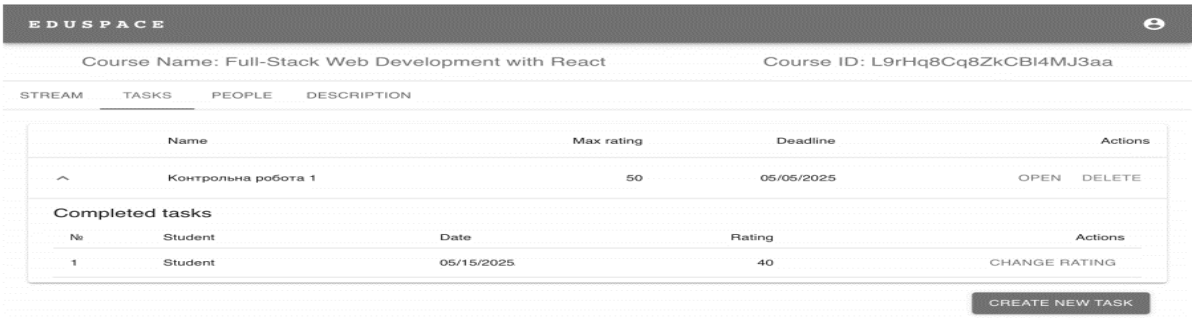

Рисунок 3.20 — Список перевірених робіт

Студент бачить свою оцінку в особистому кабінеті (див. рис.3.21).

Рисунок 3.21 — Інтерфейс перегляду оцінки студентом

На вкладці «Люди» можна переглядати список учасників курсу (див. рис.3.22).

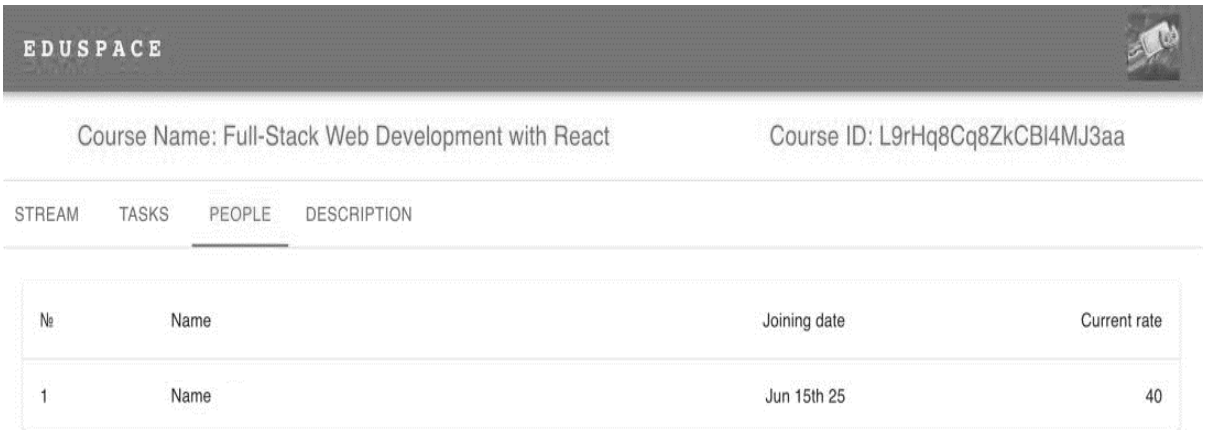

Рисунок 3.22 — Список студентів курсу з рейтингами

У підсумку можна зазначити, що для реалізації системи управління навчальним контентом у вигляді веб-додатку було використано сучасні інструменти та технології, зокрема мову гіпертекстової розмітки HTML, каскадні таблиці стилів CSS, бібліотеку React, компонентну бібліотеку Material UI, хмарну платформу Firebase та інтегроване середовище розробки WebStorm. Функціональні можливості системи були реалізовані відповідно до попередньо визначених і проаналізованих вимог користувачів.

Розроблена система управління навчальним контентом орієнтована на потреби освітньої сфери та призначена для організації структури навчальних матеріалів, обліку виконання завдань, моніторингу успішності студентів і забезпечення зручного доступу до актуальної інформації. Завдяки широкому функціоналу система сприяє підвищенню ефективності навчального процесу та оптимізації взаємодії між учасниками освітнього середовища.

Програмний продукт вирізняється доступністю, простотою налаштування, а також інтуїтивно зрозумілим і сучасним інтерфейсом, що робить його зручним для використання як викладачами та студентами, так і адміністративним персоналом закладів освіти.

3.8 Тестування та оптимізація системи

Тестування розробленої системи управління навчальним контентом проводилося з метою перевірки її функціональної коректності, стабільності роботи, зручності для кінцевого користувача, а також відповідності реалізації до технічних вимог, зазначених у технічному завданні. Основну увагу було зосереджено на перевірці інтерфейсних елементів, функцій керування курсами та навчальними матеріалами, обробки даних у базі Firestore, а також системи авторизації з підтримкою ролей користувачів.

Методика тестування полягала у проведенні ручного, сценарного тестування без застосування спеціалізованих фреймворків автоматизації, оскільки розроблена система перебуває на етапі дослідного впровадження, і такий підхід

дозволяє швидко виявити найпоширеніші логічні помилки. Перевірку функціоналу здійснювали на типових пристроях користувачів: зокрема, тестування проводилося на ноутбуці з операційною системою Windows 10 та веббраузерами Google Chrome і Mozilla Firefox, а також на мобільних пристроях з операційною системою Android та планшеті з iOS. Такий підхід дозволив змоделювати реальні умови експлуатації системи в межах різних пристроїв і розмірів екранів.

У процесі тестування було перевірено працездатність таких основних компонентів, як система реєстрації та авторизації користувачів з розмежуванням прав доступу, створення та редагування навчальних курсів, додавання та перегляд матеріалів, доступ до яких залежить від ролі користувача. Також тестувалася стабільність збереження даних у Firestore, включаючи перевірку на випадок втрати з’єднання, повторного входу до системи та оновлення сторінки. Усі функціональні модулі показали стабільну роботу, інтерфейс коректно реагував на дії користувача, а дані коректно оновлювались у реальному часі завдяки використанню реактивного підходу та Firebase SDK.

Важливим етапом стала оптимізація системи. На рівні інтерфейсу було застосовано методи мемоізації компонентів за допомогою React.memo та хуків useCallback, useMemo, що дозволило уникнути зайвого перерендеру та підвищити загальну продуктивність додатку. Компоненти, які мають велику вагу або рідко використовуються, завантажуються динамічно (через React.lazy та Suspense), що сприяє швидшому відображенню початкової сторінки. Також було оптимізовано роботу із запитами до Firebase: зменшено кількість зчитувань, додано кешування на стороні клієнта, а для складних компонентів реалізовано відображення індикаторів завантаження. Інтерфейс було адаптовано до різних розмірів екранів за допомогою можливостей бібліотеки Material UI, а також забезпечено темну тему для зручності користувачів.

У ході тестування було виявлено кілька дрібних недоліків, зокрема некоректне оновлення інтерфейсу після видалення курсу у деяких браузерах та

затримка в оновленні списку курсів при великій кількості записів. Для вирішення цих проблем було реалізовано примусове оновлення стану через хук useEffect та впроваджено пагінацію на стороні клієнта. Також було виявлено проблему зі спробою авторизації після введення неправильних даних: система не оновлювала інтерфейс без перезавантаження. Цей недолік було усунуто шляхом обробки помилок через зворотні колбеки Firebase.

У результаті проведеного тестування та оптимізації можна зробити висновок, що система управління навчальним контентом функціонує стабільно, має високу продуктивність та готова до реального використання. Реалізовані механізми обробки даних, адаптивний інтерфейс, а також чітко побудована архітектура компонентів дозволяють забезпечити ефективне та зручне управління освітніми матеріалами для викладачів, адміністраторів і студентів. У подальшому можливе доповнення системи модулем автоматизованого тестування з використанням Jest або React Testing Library для ще більшої надійності та контролю якості.

ВИСНОВКИ

У процесі виконання роботи було розроблено систему управління навчальним контентом у вигляді веб-додатку. Система підтримує два типи користувачів: студент і викладач, кожен з яких має доступ до індивідуального інтерфейсу з відповідними функціями.

Інтерфейс викладача включає такі можливості, як створення та видалення навчальних завдань, розміщення оголошень, контроль виконання завдань і перевірка поданих робіт. Інтерфейс студента забезпечує перегляд призначених завдань, їх виконання, доступ до оголошень та можливість залишати коментарі.

Вся інформація про користувачів і навчальні ресурси зберігається у хмарній базі даних Firebase, що працює в режимі реального часу. Це забезпечує надійність збереження даних, їхню доступність з будь-якого пристрою та миттєве оновлення контенту. Застосування хмарної технології дозволяє уникнути ризиків, пов’язаних із фізичним зберіганням даних на пристроях. Було проведено аналіз наявних технологій для створення веб-сервісу, в результаті чого були обрані ефективні та зручні у впровадженні інструменти: середовище WebStorm для розробки інтерфейсу, бібліотека React для побудови компонентів і обробки подій, а також Material UI для забезпечення сучасного вигляду застосунку.

У результаті реалізовані такі ключові завдання:

— розроблено веб-інтерфейс із поділом функціоналу для різних ролей користувачів;

— впроваджено механізми взаємодії з навчальним контентом;

— забезпечено стабільну роботу системи завдяки використанню надійних технологій збереження даних.

Розроблена система є гнучкою, доступною, інтуїтивно зрозумілою та може бути легко адаптована до потреб різних освітніх установ. У перспективі проєкт планується розширювати: інтегрувати додаткові модулі, покращувати візуалізацію, а також впроваджувати нові інструменти та технології відповідно до актуальних вимог користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Комп’ютерна система управління навчальним контентом/ Гресько М. Р., Снігур А. В. // Матеріали LIV наукової-технічної конференції підрозділів (Вінниця, 2025 р.) [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/25174

2. The Evolution and Diffusion of Learning Management Systems: The Case of Canvas LMS [Електронний ресурс]. — Режим доступу: https://ohiostate.pressbooks.pub/drivechange/chapter/the-evolution-and-diffusion-of-learning-management-systems-the-case-of-canvas-lms/ — Дата звернення: 05.05.2025.

3. The Evolution of Learning Management Systems (LMS) [Електронний ресурс]. — Режим доступу: https://spekit.com/blog/the-evolution-of-learning-management-systems/ — Дата звернення: 05.05.2025.

4. WHAT IS A LEARNING MANAGEMENT SYSTEM? AND WHY DO I NEED ONE? [Електронний ресурс] // Shareknowledge. — Режим доступу: https://www.shareknowledge.com/blog/what-learning-management-system-and-why-do-i-need-one — Дата звернення: 05.05.2025.

5. History of LMS [Електронний ресурс] // Easy LMS. — Режим доступу: https://www.easy-lms.com/knowledge-center/lms-center/history-of-lms/item10401 — Дата звернення: 05.05.2025.

6. Software requirements specification [Електронний ресурс] // Wikipedia. — Режим доступу: https://en.wikipedia.org/wiki/Software_requirements_specification — Дата звернення: 05.05.2025.

7. SRS for eLearning Management System [Електронний ресурс] // Belitsoft. — Режим доступу: https://belitsoft.com/custom-elearning-development/srs-for-e-learning-management-system — Дата звернення: 05.05.2025.

8. How To Create Software Requirements Specification And Improve Your Software Development Process [Електронний ресурс] // DesignRush. — Режим

доступу: https://www.designrush.com/agency/software-development/trends/software-requirements-specification — Дата звернення: 05.05.2025.

9. IEEE recommended practice for software requirements specifications (IEEE Std 830-1998). — New York: The Institute of Electrical and Electronics Engineers, Inc., 1998. — 40 p. — ISBN 0-7381-0332-2.

10. Single-page application [Електронний ресурс] // Wikipedia. — Режим доступу: https://en.wikipedia.org/wiki/Single-page_application — Дата звернення: 05.05.2025.

11. Client—server model [Електронний ресурс] // Wikipedia. — Режим доступу: https://en.wikipedia.org/wiki/Client—server_model — Дата звернення: 05.05.2025.

12. Must-Know Concepts of Modern Web Architecture [Електронний ресурс] // Better Programming. — Режим доступу: https://betterprogramming.pub/10-must-know-concepts-of-modern-web-architecture-9ecbefef8bc — Дата звернення: 05.05.2025.

13. Web application vs. website: finally answered [Електронний ресурс] // ScienceSoft. — Режим доступу: https://www.scnsoft.com/blog/web-application-vs-website-finally-answered — Дата звернення: 05.05.2025.

14. Firebase Documentation [Електронний ресурс] // Firebase. — Режим доступу: https://firebase.google.com/docs — Дата звернення: 05.05.2025.

15. React Firebase Hooks [Електронний ресурс] // GitHub. — Режим доступу: https://github.com/CSFrequency/react-firebase-hooks — Дата звернення: 05.05.2025.

16. React — Вікіпедія [Електронний ресурс]. — Режим доступу: https://uk.wikipedia.org/wiki/React — Дата звернення: 05.05.2025.

17. React — A JavaScript library for building user interfaces [Електронний ресурс] // React. — Режим доступу: https://reactjs.org/ — Дата звернення: 05.05.2025.

18. MUI: The React component library you always wanted [Електронний ресурс] // Material UI. — Режим доступу: https://mui.com/ — Дата звернення: 05.05.2025.

19. CS303 Digital Design Lab1: вебсайт. URL: https://ee.ius.edu.ba/sites/default/files/u747/cs303_f19_lab1.pdf (дата звернення: 05.05.2025).

20. Проблемно і функціонально орієнтовані комп’ютерні системи та мережі: вебсайт. URL: http://dspace.nbuv.gov.ua/bitstream/handle/123456789/12005/09-Petrenko.pdf?sequence=1 (дата звернення: 05.05.2025).

21. Zynq-7000 SoCDataSheet: Overview: вебсайт. URL: https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf (дата звернення: 05.05.2025).

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
«21» березня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання бакалаврської кваліфікаційної роботи
«Система управління навчальним контентом»
08-54. БКР.039.00.000 ТЗ

Науковий керівник: к.т.н., доц. каф. ОТ
Снігур А.В.

Студент групи 1СП-21б
Гресько М.Р.

м. Вінниця — 2025

1 Підстава для використання бакалаврської кваліфікаційної роботи (БКР)

1.1 Сучасні освітні процеси активно впроваджують цифрові технології для забезпечення ефективного дистанційного та змішаного навчання. Одним із ключових елементів таких технологій є системи управління навчальним контентом, які дозволяють організовувати, зберігати, оновлювати та розповсюджувати навчальні матеріали у зручному для користувача форматі.

1.2 Наказ про затвердження теми кваліфікаційної роботи.

2 Мета БКР і призначення розробки

2.1 Мета роботи — розробка веб-системи управління навчальним контентом, яка дозволяє створювати навчальні курси, додавати до них структуровані матеріали, редагувати й організовувати контент у межах кожного курсу.

2.2 Призначення розробки — розробка програмного забезпечення для кіберфізичного комплексу збору даних на базі платформи Arduino, який виконує автоматичне визначення та обхід перешкоди.

3 Вихідні дані для виконання БКР

3.1 Вивчення існуючих систем управління навчальним контентом.

3.2 Аналіз вимог до функціоналу системи з боку користувачів.

3.3 Проєктування структури бази даних для зберігання курсів, модулів.

3.4 Розробка користувацького інтерфейсу.

3.5 Реалізація основного функціоналу системи.

3.6 Проведення тестування системи та збір відгуків від потенційних користувачів.

3.7 Аналіз результатів тестування, оптимізація функцій і підготовка звіту.

4. Вимоги до виконання БКР

Головна вимога — створити веборієнтовану систему управління навчальним контентом, яка реалізує базовий функціонал.

5 Етапи БКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

№ етапу	Назва етапу	Термін виконання	Очікувані результати
1	Аналіз існуючих платформ та середовища розробки	21.03.25 — 30.03.25	Аналітичний огляд
2	Реалізація та тестування	31.03.25 — 13.04.25	Розділ 2
3	Оптимізація та перспективи розвитку	14.04.25 — 27.04.25	Розділ 3
4	Оформлення пояснювальної записки, графічного матеріалу та презентації	28.04.25 — 11.05.25	Пояснювальна записка, графічний матеріал та презентація

6 Матеріали, що подаються до захисту БКР

До захисту подаються: пояснювальна записка БКР, графічні матеріали, протокол попереднього захисту на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до БКР українською та іноземною мовами, довідка про відповідність оформлення вимогам.

7 Порядок контролю виконання та захисту БКР

Контроль виконання етапів здійснюється науковим керівником згідно з встановленими термінами. Захист БКР проходить на засіданні екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлення та порядок виконання БКР

8.1 При оформленні БКР використовуються:

— ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 123 «Комп'ютерна інженерія» (освітня програма «Системне програмування») — кафедра обчислювальної техніки, ВНТУ 2023.

8.2 Порядок виконання БКР викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21»

ДОДАТОК Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Система управління навчальним контентом

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ _____кафедра обчислювальної техніки_____
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism ___3___%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____Азаров О.Д., зав. каф. ОТ _____ _____
(прізвище, ініціали, посада) (підпис)

____Крупельницький Л.В., доц. каф ОТ_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ ___Захарченко С.М.____
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ ____ к.т.н., доц. кафедри ОТ Снігур А.В._____
(підпис) (прізвище, ініціали, посада)

Здобувач _____ _____Гресько М.Р_____
(підпис) (прізвище, ініціали)

ДОДАТОК В

Ілюстративна частина

СИСТЕМА УПРАВЛІННЯ НАВЧАЛЬНИМ КОНТЕНТОМ

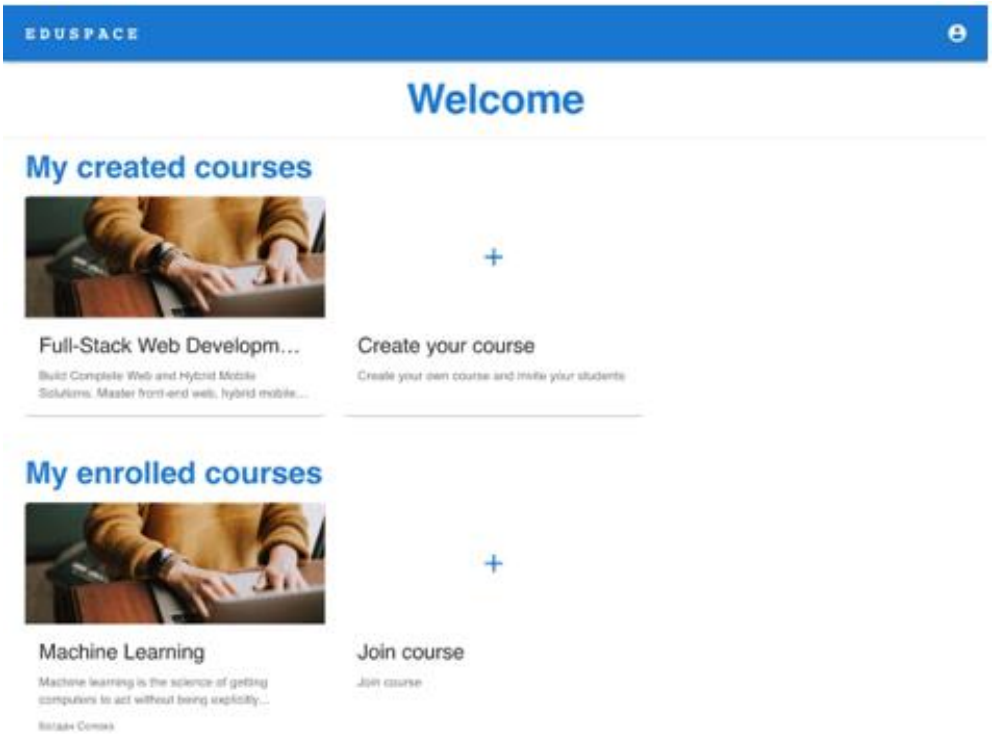

Рисунок В.1— Панель управління курсами користувача.

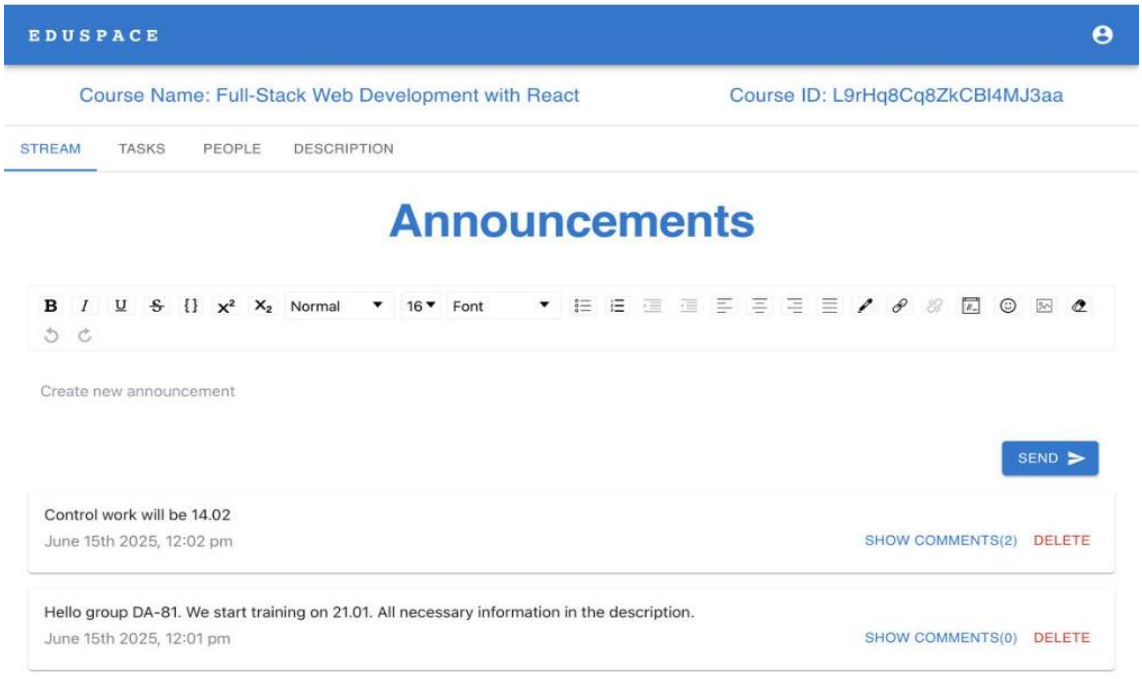

Рисунок В.2 — Інтерфейс курсу для викладача

ДОДАТОК Г

Лістинг програмного забезпечення

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals'; import {UserProvider} from './context/userContext';
import '../node_modules/react-draft-wysiwyg/dist/react-draft-wysiwyg.css';
ReactDOM.render(
<React.StrictMode>
<UserProvider>
<App />
</UserProvider>
</React.StrictMode>,
document.getElementById('root'),
);
// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals reportWebVitals();
import React from 'react';
import {BrowserRouter, Link, Navigate, Route, Routes} from 'react-router-dom'; import PublicLayout
from './components/templates/PublicLayout';
import {HOME, LOGIN} from './constants/routes'; import {publicRoutes} from './routes/routes';
import PublicRoute from './routes/PublicRoute';
import ProtectedLayout from './components/templates/ProtectedLayout'; import ProtectedRoute from
'./routes/ProtectedRoute'; import Courses from './pages/Courses';
import Course from './pages/Course';
function App() {
return (
<BrowserRouter>
<Routes>
<Route path="/" element={<PublicLayout />}>
<Route index element={<Navigate to={LOGIN} replace />} /> {publicRoutes.map(({path, element})
=> (
<Route
key={path}
path={`/${path}`}
element={<PublicRoute>{element}</PublicRoute>} />
))}
</Route>
<Route path="/" element={<ProtectedLayout />}> <Route
path={HOME}
element={<ProtectedRoute><Courses/></ProtectedRoute>} />
<Route path="course">
<Route path=":courseId" element={
<ProtectedRoute><Course /></ProtectedRoute>
} />
</Route>
```

```
</Route>
<Route path="*" element={<Link to="login">Go to log in</Link>} /> </Routes>
</BrowserRouter>
);
}
export default App;
import firebase from 'firebase';
const firebaseConfig = {
apiKey: 'AIzaSyCkcUIPkZWnk5ur4-lcHen2WJ18Lha8mmY',
authDomain: 'eduspace-15506.firebaseapp.com',
projectId: 'eduspace-15506',
storageBucket: 'eduspace-15506.appspot.com',
messagingSenderId: '895198289171',
appId: '1:895198289171:web:4e0156630f43f1bb890ee6', };
const app = firebase.initializeApp(firebaseConfig); const auth = app.auth();
const db = app.firestore();
const checkUser = async (user) => {
const querySnapshot = await db
.collection('users')
.where('uid', '==', user.uid)
.get();
if (querySnapshot.docs.length === 0) {
// create a new user
await db.collection('users').add({
uid: user.uid,
name: user.displayName,
photo: user.photoURL,
});
}
};
const googleProvider = new firebase.auth.GoogleAuthProvider();
// Sign in and check or create account in firestore const signInWithGoogle = async (onError) => {
try {
const response = await auth.signInWithPopup(googleProvider); await checkUser(response.user);
} catch (err) { onError(err.message);
}
};
const signUpWithEmailAndPassword = async ( email,
password,
firstName,
lastName,
onError,
)=>{
try {
const {user} = await auth.createUserWithEmailAndPassword(email, password); const tempUser =
{...user, displayName: `${firstName} ${lastName}`}; await checkUser(tempUser);
} catch (err) {
onError(err.message);
}
};
const logout = () => {
```

```
auth.signOut();
};
export {app, auth, db, signInWithGoogle, signUpWithEmailAndPassword, logout};
import React from 'react';
import LoginForm from '../components/organisms/LoginForm';
function Login() {
return (
<LoginForm />
);
}
export default Login;
import React from 'react';
import SignupForm from '../components/organisms/SignupForm';
function Login() {
return (
<SignupForm />
);
}
export default Login;
import React, {useEffect, useState} from 'react'; import {
Container, Divider, Grid,
} from '@mui/material';
import H1 from '../components/atoms/H1';
import H2 from '../components/atoms/H2';
import CourseCard from '../components/molecules/CourseCard'; import AddCourseCard from
'../components/molecules/AddCourseCard'; import {db} from '../firebase';
import DialogModal from '../components/organisms/DialogModal'; import useUser from
'../hooks/useUser';
import JoinCourseModal from '../components/organisms/JoinCourseModal'; import moment from
'moment';
function Courses() {
const [isCreateModalOpen, setIsCreateModalOpen] = useState(false); const [isJoinModalOpen,
setIsJoinModalOpen] = useState(false); const {user} = useUser();
const [enrolledCourses, setEnrolledCourses] = useState([]);
const [createdCourses, setCreatedCourses] = useState([]); const [loading, setLoading] =
useState(false);
const fetchCourses = async () => {
try {
setLoading(true);
await user.ref.collection('enrolledCourses')
.onSnapshot((snapshot) => {
const courses = snapshot?.docs.map((doc)=>{ return doc.data();
});
setEnrolledCourses(courses);
});
await user.ref.collection('createdCourses')
.onSnapshot((snapshot) => {
const courses = snapshot?.docs.map((doc)=>{ return doc.data();
});
setCreatedCourses(courses);
});
```

```
} catch (error) { console.error(error.message);
} finally {
setLoading(false);
}
};
useEffect(() => {
fetchCourses();
}, [user]);
const joinClass = async (courseId) => { try {
setIsJoinModalOpen(false);
const classRef = await db.collection('courses').doc(courseId).get(); if (!classRef.exists) {
return alert(`Class doesn't exist, please provide correct ID`);
}
await classRef.ref.collection('users').add({ name: user.name,
joiningDate: moment().format('MMM Do YY'), courseId,
studentUid: user.uid,
});
const classData = await classRef.data();
await user.ref.collection('enrolledCourses').add({ creatorName: classData.creatorName,
id: courseId,
name: classData.name,
description: classData.description,
});
alert(`Enrolled in ${classData.name} successfully!`);
} catch (err) { console.error(err);
alert(err.message);
}
};
const createClass = async (courseName, courseDescription) => { setIsCreateModalOpen(false);
try {
const {id} = await db.collection('courses').add({
creatorName: user.name,
name: courseName,
description: courseDescription,
users: [],
});
await user.ref.collection('createdCourses').add({ id,
name: courseName,
description: courseDescription,
});
} catch (err) {
alert(`Cannot create class — ${err.message}`);
}
};
if (loading) return 'Loading';
if (!user) return 'User not found';
return (
<div>
<H1 sx={{textAlign: 'center', margin: '16px 0'}}> Welcome,
{' '}
{user.name}
```

```
</H1>
<Divider />
<Container>
<H2 sx={{margin: '12px 0'}}>My created courses</H2> <Grid container spacing={2}>
{createdCourses && createdCourses.map(({name, description, id}) => ( <Grid item xs={6} md={4}
lg={3} key={id}>
<CourseCard title={name} description={description} id={id} /> </Grid>
))}
<Grid item xs={6} md={4} lg={3} key="create"> <AddCourseCard
title="Create your course"
description="Create your own course and invite your students"
onClick={() => setIsCreateModalOpen(true)}
/>
</Grid>
</Grid>
</Container>
<Divider sx={{marginTop: 4}} />
<Container>
<H2 sx={{margin: '12px 0'}}>My enrolled courses</H2> <Grid container spacing={2}>
{enrolledCourses && enrolledCourses.map( ({name, description, id, creatorName}) => (
<Grid item xs={6} md={4} lg={3} key={id}> <CourseCard
title={name}
description={description}
author={creatorName}
id={id}/>
</Grid>
))}
<Grid item xs={6} md={4} lg={3} key="join"> <AddCourseCard
title="Join course"
description="Join course"
onClick={()=>setIsJoinModalOpen(true)}/> </Grid>
</Grid>
</Container>
<JoinCourseModal
title="Join course"
label="ID"
isOpen={isJoinModalOpen}
handleClose={()=>setIsJoinModalOpen(false)} handleSubmit={joinClass}
/>
<DialogModal
title="Create class"
label="Name"
isOpen={isCreateModalOpen}
handleClose={()=>setIsCreateModalOpen(false)} handleSubmit={createClass}
/>
</div>
);
}
export default Courses;
import React, {useEffect, useState} from 'react';
import {Box, Tab, Tabs, Typography} from '@mui/material'; import {db} from '../firebase';
```

```
import {useParams} from 'react-router-dom'; import useUser from '../hooks/useUser';
import Users from '../components/organisms/Users'; import Stream from
'../components/organisms/Stream'; import Tasks from '../components/organisms/Tasks'; import
{TeacherContext} from '../context/TeacherContext'; import {node, number} from 'prop-types';
function TabPanel(props) {
const {children, value, index, ...other} = props;
return (
<div
role="tabpanel"
hidden={value !== index}
id={`simple-tabpanel-${index}`} aria-labelledby={`simple-tab-${index}`} {...other}
>
{value === index && (
<Box sx={{p: 3}}>
{children}
</Box>
)}
</div>
);
}
TabPanel.propTypes = {
children: node.isRequired,
value: number.isRequired,
index: number.isRequired,
};
function a11yProps(index) {
return {
'id': `simple-tab-${index}`,
'aria-controls': `simple-tabpanel-${index}`, };
}
const Course = () => {
const [isTeacher, setIsTeacher] = useState(false); const [value, setValue] = React.useState(0); const
handleChange = (event, newValue) => {
setValue(newValue);
};
const [classData, setClassData] = useState({});
const {user} = useUser();
const {courseId} = useParams();
useEffect(() => {
db.collection('courses')
.doc(courseId)
.onSnapshot((snapshot) => {
const data = snapshot.data();
console.log(user);
setClassData(data);
});
user.ref.collection('createdCourses').where('id', '==', courseId)
.onSnapshot((snapshot)=>{
if (snapshot.docs.length > 0) setIsTeacher(true); else setIsTeacher(false);
});
}, [courseId]);
```

```
return (
<TeacherContext.Provider value={isTeacher}> <Box sx={{width: '100%'}}>
<Box sx={
{display: 'flex',
justifyContent: 'space-around',
borderBottom: 1,
borderColor: 'divider',
padding: '16px 0',
}
}>
<Typography variant="h6" color="primary"> Course Name: {classData?.name}
</Typography>
<Typography variant="h6" color="primary"> Course ID: {courseId}
</Typography>
</Box>
<Box sx={{borderBottom: 1, borderColor: 'divider'}}> <Tabs
value={value}
onChange={handleChange}
aria-label="basic tabs example"
>
<Tab label="Stream" {...a11yProps(0)} />
<Tab label="Tasks" {...a11yProps(1)} />
<Tab label="People" {...a11yProps(2)} />
<Tab label="Description" {...a11yProps(3)} />
</Tabs>
</Box>
<TabPanel value={value} index={0}>
<Stream />
</TabPanel>
<TabPanel value={value} index={1}>
<Tasks/>
</TabPanel>
<TabPanel value={value} index={2}>
<Users />
</TabPanel>
<TabPanel value={value} index={3}>
{classData.description}
</TabPanel>
</Box>
</TeacherContext.Provider>
);
};
export default Course;
import {useContext} from 'react';
import {UserContext} from '../context/userContext';
const useUser = () => {
const {user, setUser} = useContext(UserContext); return {user, setUser};
};
export default useUser;
import React, {useEffect, useState, createContext} from 'react'; import {useAuthState} from 'react-
firebase-hooks/auth';
```

```
import {auth, db} from '../firebase';
import {node} from 'prop-types';
export const UserContext = createContext({ user: null,
setUser: ()=>{},
});
export const UserProvider = ({children}) => { const [user, setUser] = useState(null); const
[authUserAuth] = useAuthState(auth);
useEffect(() => {
if (authUserAuth) {
const unsubscribe = db.collection('users')
.where('uid', '==', authUserAuth.uid)
.onSnapshot((snapshot) => {
const docRef = snapshot?.docs[0];
setUser({...docRef.data(), ref: docRef.ref}); });
return () => {
unsubscribe();
};
} else {
setUser(null);
}
}, [authUserAuth]);
return (
<UserContext.Provider
value={{user, setUser}}
>
{children}
</UserContext.Provider>
);
};
UserProvider.propTypes = {
children: node.isRequired,
};
import {createContext} from 'react';
export const TeacherContext = createContext(false);
export const LOGIN = '/login';
export const SIGNUP = '/signup';
export const HOME = '/home';
export const COURSE = '/course/:courseId';
import React from 'react';
import {
Box,
} from '@mui/material';
import {Outlet} from 'react-router-dom';
import PublicHeader from '../organisms/PublicHeader'; import Container from '../atoms/Container';
function PublicLayout() {
return (
<div>
<PublicHeader />
<Container>
<Box sx={{
mx: 'auto',
```

```
mt: 4,
maxWidth: 500,
}}
>
<Outlet />
</Box>
</Container>
</div>
);
}
export default PublicLayout;
import React from 'react';
import {
Outlet,
} from 'react-router-dom';
import ProtectedHeader from '../organisms/ProtectedHeader';
function ProtectedLayout() {
return (
<div>
<ProtectedHeader />
<Outlet />
</div>
);
}
export default ProtectedLayout;
/* eslint-disable max-len */
import * as React from 'react';
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody'; import TableCell from
'@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer'; import TableHead from
'@mui/material/TableHead';
import TableRow from '@mui/material/TableRow'; import Paper from '@mui/material/Paper'; import
{db} from '../../firebase';
import {useEffect, useState} from 'react'; import {useParams} from 'react-router-dom';
export default function Users() {
const [users, setUsers] = useState([]);
const [completedTasks, setCompletedTasks] = useState([]);
const {courseId} = useParams();
useEffect(async ()=>{
const tempUsers = await db.collection(`courses/${courseId}/users`).get();
setUsers(tempUsers.docs.map((doc)=>doc.data()));
const tempCompletedTasks = await
db.collection('completedTasks').where('courseId', '==', courseId).get();
setCompletedTasks(tempCompletedTasks);
}, [courseId]);
const getCurrentRating = (studentUid) => { let rating = 0;
completedTasks.docs?.filter((doc)=>doc.data().studentUid ===
studentUid).forEach((task)=>{
rating+=parseInt(task.data()?.rating);
});
```

```
return rating;
};
return (<>
{users.length === 0 ? <p style={{textAlign: 'center', color: 'gray', marginBottom: '12px'}}>No
students.<br/> Share the course ID with your students so they can join</p> :
<TableContainer component={Paper}>
<Table sx={{minWidth: 650}} aria-label="simple table"> <TableHead>
<TableRow>
<TableCell>№</TableCell>
<TableCell>Name</TableCell>
<TableCell align="right">Joining date</TableCell> <TableCell align="right">Current
rate</TableCell>
</TableRow>
</TableHead>
<TableBody>
{users.map((user, index) => (
<TableRow
key={user.name}
sx={{'&:last-child td, &:last-child th': {border: 0}}}
>
<TableCell>
{index+1}
</TableCell>
<TableCell component="th" scope="row">
{user.name}
</TableCell>
<TableCell
align="right">{user.joiningDate}</TableCell> <TableCell
align="right">{getCurrentRating(user.studentUid)}</TableCell>
</TableRow>
))}
</TableBody>
</Table>
</TableContainer>}
</>
);
}
/* eslint-disable react/prop-types,max-len */ import React, {useEffect, useState} from 'react'; import
{db} from '../../firebase';
import TableRow from '@mui/material/TableRow'; import TableCell from '@mui/material/TableCell';
import IconButton from '@mui/material/IconButton';
import KeyboardArrowUpIcon from '@mui/icons-material/KeyboardArrowUp';
import KeyboardArrowDownIcon from '@mui/icons-material/KeyboardArrowDown';
import moment from 'moment';
import {
Button,
Dialog,
DialogActions,
DialogContent,
DialogContentText,
DialogTitle,
```

```
} from '@mui/material';
import Collapse from '@mui/material/Collapse'; import Box from '@mui/material/Box';
import CompletedTasks from '../molecules/CompletedTasks'; import {useParams} from 'react-router-
dom';
import TableContainer from '@mui/material/TableContainer'; import Paper from
'@mui/material/Paper'; import Table from '@mui/material/Table';
import TableHead from '@mui/material/TableHead'; import TableBody from
'@mui/material/TableBody';
import CreateTaskModal from '../molecules/CreateTaskModal';
const TeacherTask = ({task}) => {
const [open, setOpen] = React.useState(false);
const [isDeleteModalOpen, setIsDeleteModalOpen] = useState(false);
const [isDescriptionModalOpen, setIsDescriptionModalOpen] = useState(false); const deleteTask =
async () => {
await db.collection('tasks').doc(task.id).delete(); const tempCompletedTasks = await
db.collection('completedTasks').where('taskId', '==', task.id).get();
tempCompletedTasks.docs.forEach((doc)=>{
doc.ref.delete();
});
};
return (
<React.Fragment>
<TableRow sx={{'& > *': {borderBottom: 'unset'}}}>
<TableCell>
<IconButton
aria-label="expand row"
size="small"
onClick={() => setOpen(!open)}
>
{open ? <KeyboardArrowUpIcon /> : <KeyboardArrowDownIcon />} </IconButton>
</TableCell>
<TableCell component="th" scope="row">
{task.name}
</TableCell>
<TableCell align="right">{task.maxRating}</TableCell>
<TableCell align="right"> {moment(task.deadline?.toDate() || 0).format('L')}</TableCell>
<TableCell align="right">
<Button variant="text"
onClick={()=>setIsDescriptionModalOpen(true)}>Open</Button> <Button variant="text"
color="error"
onClick={()=>setIsDeleteModalOpen(true)}>Delete</Button> </TableCell>
</TableRow>
<TableRow>
<TableCell style={{paddingBottom: 0, paddingTop: 0}} colSpan={6}> <Collapse in={open}
timeout="auto" unmountOnExit>
<Box sx={{margin: 1}}>
<CompletedTasks taskId={task.id} maxRating={task.maxRating} deadline={task.deadline}/>
</Box>
</Collapse>
```