НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
В. о. зав. кафедри

_____ Михайло НОВАТАРСЬКИЙ
(підпис)

“__” _____2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою “Комп’ютерні системи та мережі”

спеціальності 123 “Комп’ютерна інженерія”

на тему: Освітній веб-застосунок для навчання правилам поведінки під час загроз

Виконала : студентка 4 курсу, групи ІО-11
(шифр групи)

Лоханько Ксенія Вікторівна
(прізвище, ім’я, по батькові) (підпис)

Керівник ас. Пономаренко А.М.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) ас. Гончаренко О.О.
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент Коган А.В.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2025 р.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальність 123 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
В. о. зав. кафедри

_____ Михайло НОВАТАРСЬКИЙ
(підпис)

“__” _____ 2025 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт студента

Лоханько Ксенії Вікторівни

1. Тема проєкту *Освітній веб-застосунок для навчання правилам поведінки під час загроз*
керівник проекту ас. Пономаренко А.М.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «23» травня 2025 року № 1705-с .

2. Термін здачі студентом закінченого проєкту «02» червня 2025 р.

3. Вихідні дані до проєкту *технічна документація, теоретичні дані.*

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

Розділ 1. Аналіз предметної області та вимог до програмного забезпечення.

Розділ 2. Технології та засоби розробки програмного забезпечення.

Розділ 3. Розробка програмного забезпечення.

Розділ 4. Тестування та оцінка ефективності.

Розділ 5. Інструкція користувача.

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) структурна схема системи, структура бази даних, алгоритм дій програмного забезпечення.

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Гончаренко О.О		

7. Дата видачі завдання «14» січня 2025 р.

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	Затвердження теми проєкту	*11.01.2025*	
2.	Вивчення та аналіз завдання	*26.01.2025 - 01.05.2025*	
3.	Розробка архітектури та загальної структури системи	*15.05.2025 - 20.05.2025*	
4.	Розробка структур окремих підсистем	*21.05.2025 - 02.06.2025*	
5.	Програмна реалізація системи	*21.05.2025-02.06.2025*	
6.	Оформлення пояснювальної	*22.05.2025-04.06.2025*	
7.	Захист програмного продукту	*23.05.2025*	
8.	Передзахист	*03.06.2025*	
9.	Захист	*18.06.2025*	

Студент-дипломник _____ Ксенія ЛОХАНЬКО
(підпис)

Керівник проєкту _____ Артем ПОНОМАРЕНКО
(підпис)

АНОТАЦІЯ

У дипломній роботі реалізовано освітній веб-застосунок, що призначений для навчання учнів правилам поведінки під час надзвичайних ситуацій. Основною метою проєкту є підвищення рівня безпеки та обізнаності школярів за допомогою інтерактивного онлайн-середовища, яке також дозволяє вчителям ефективно контролювати навчальний процес.

У роботі проаналізовано наявні рішення у сфері безпеки та освіти, визначено їхні переваги й недоліки. Реалізовано веб-застосунок з функціональністю проходження уроків, тестування знань, моніторингу прогресу та адміністрування навчального контенту. У реалізації використано React для інтерфейсу та Firebase як бекенд-платформу.

Розроблений застосунок може бути використаний у школах для проведення уроків, а також для самостійного опрацювання тем. Рішення є доступним, гнучким та адаптованим до різних вікових груп.

ANNOTATION

The bachelor's thesis presents the development of an educational web application designed to teach students proper behavior during emergencies. The main goal of the project is to enhance students’ safety and awareness through an interactive online environment, which also enables teachers to effectively manage the learning process.

The work analyzes existing solutions in the fields of safety and education, highlighting their advantages and drawbacks. The implemented web application includes functionality for completing lessons, testing knowledge, tracking progress, and managing educational content. React was used for the frontend, while Firebase served as the backend platform.

The developed application can be used in schools during safety-related classes, as well as for independent study. The solution is accessible, flexible, and adapted for different age groups.

справки	Форма	*Значення*	*Найменування*	Кіл. листів	№ екземпляра	*Додаток*
			Документація загальна			
			Знову розроблена			
	А4	***ІАЛЦ.467200.002 ТЗ***	Освітній веб-застосунок для	4		
			навчання правилам			
			поведінки під час загроз			
			Технічне завдання			
	А4	***ІАЛЦ.467200.003 ПЗ***	Освітній веб-застосунок для	102		
			навчання правилам			
			поведінки під час загроз			
			Пояснювальна записка			
	А4	***ІАЛЦ.467200.004 Д1***	Освітній веб-застосунок для	1		
			навчання правилам			
			поведінки під час загроз			
			Структурна схема			
			системи			
	А4	***ІАЛЦ.467200.005 Д2***	Освітній веб-застосунок для	1		
			навчання правилам			
			поведінки під час загроз			
			Структура бази даних			
	А4	***ІАЛЦ.467200.006 Д3***	Освітній веб-застосунок для	1		
			навчання правилам			
			поведінки під час загроз			
			Алгоритм дій програмного			
			забезпечення			
	А4	***ІАЛЦ.467200.007 Д4***	Освітній веб-застосунок для	76		
			навчання правилам			
			поведінки під час загроз			
			Текст програмного коду			

					ІАЛЦ.467200.001 ОА		
Зм.	*Лист*	*№ докум.*	*Підпис*	*Дата*			
Розроб.		Лоханько К.В.			*Освітній веб-застосунок для навчання правилам поведінки під час загроз* Опис альбому	Літ.	Аркуш 1 / Аркушів 1
Перев.		Пономаренко А. М.					
Реценз.		Коган А.В.				***НТУУ “КПІ ” ФІОТ ІО-11***	
Н. контр.		Гончаренко О.О.					
Затвердив							

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Освітній веб-застосунок для навчання правилам поведінки під час загроз*»

Київ – 2025

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

Дане технічне завдання поширюється на розробку освітбого веб-застосунку для навчання правилам поведінки під час загроз.

Областю застосування цієї системи є використання у загальноосвітніх закладах — школах, ліцеях, гімназіях — а також для самостійного навчання учнів. Його можуть використовувати вчителі для організації занять із безпеки життєдіяльності, цивільного захисту, а також у рамках позакласної роботи або тренінгів.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки даної системи є виконання роботи кваліфікаційно-освітнього рівня «бакалавр комп'ютерної інженерії», затверджене факультетом “Інформатики та обчислювальної техніки” кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою та призначенням даної роботи є розробка інтерактивного веб-застосунку, який надає учням можливість навчатися правилам поведінки під час загроз, а вчителям — зручно організовувати навчальний процес і перевіряти знання.

4 ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки даного дипломного проекту є офіційні документації, публікації та статті в мережі Інтернет на дану тему, довідники по платформах дистанційного навчання, науково-технічна література.

5 ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до продукту, що розробляється

Розроблена система має виконувати такі вимоги:

- Простий і інтуїтивно-зрозумілий інтерфейс системи.
- Адаптивність для різних пристроїв (мобільні, планшети, ПК).
- Реєстрація та авторизація користувачів (вчителі / учні).
- Відображення індивідуального прогрессу користувача.
- Адмін-панель для вчителя з можливістю створення власних курсів.

5.2. Вимоги до програмного забезпечення

- ОС Windows, macOS чи Linux.
- Веб-браузер з підтримкою стандарту ECMAScript 6.
- Середовище розробки: PyCharm Community / Professional 2022.3+.
- Інструменти: Node.js 18+, Git, Firebase CLI, npm/yarn.

5.3. Вимоги до апаратної частини

- Центральний процесор: AMD Ryzen 5 3500U або аналогічний за продуктивністю (Intel Core i5 8-го покоління і вище)
- Оперативна пам’ять: не менше 2 ГБ
- Вільний простір на жорсткому диску або SSD: не менше 1 ГБ
- Наявність стабільного з’єднання з мережею Інтернет

6 ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Вивчення літератури	27.05.2025
Вивчення і аналіз існуючих аналогів.	30.05.2025
Розробка архітектури та загальної структури системи	4.06.2025
Розробка структур окремих частин системи	11.06.2025
Програмування та загальне налагодження системи.	16.06.2025
Виправлення помилок	17.06.2025
Підготовка звіту по результатах практики.	18.05.2025

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Освітній веб-застосунок для навчання правилам поведінки під час загроз*»

Київ – 2025

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД — база даних

НП — навчальний процес

НС — надзвичайні ситуації

API — (Application Programming Interface) інтерфейс прикладного програмування

CI/CD — (Continuous Integration / Continuous Delivery) безперервна інтеграція / доставка

CRUD — (Create, Read, Update, Delete) базові операції з даними

CSRF — (Cross-Site Request Forgery) міжсайтове підроблення запитів

DOM — (Document Object Model) об’єктна модель документа

HTML — (HyperText Markup Language) мова розмітки гіпертексту

HTTP — (HyperText Transfer Protocol) протокол передавання гіпертексту

HTTPS — (HyperText Transfer Protocol Secure) захищений протокол передавання гіпертексту

LMS — (Learning Management System) система керування навчанням

SDK — (Software Development Kit) набір інструментів для розробника

SPA — (Single Page Application) односторінковий застосунок

UID — (User Identifier) унікальний ідентифікатор користувача

URL — (Uniform Resource Locator) уніфікований локатор ресурсу

XSS — (Cross-Site Scripting) міжсайтове скриптування

ВСТУП

У сучасному світі, де ризики надзвичайних ситуацій зростають через природні катастрофи, техногенні аварії, терористичні загрози та військові конфлікти, особливого значення набуває підготовка населення до правильної поведінки у критичних умовах. Ця потреба стала ще більш актуальною у зв'язку з повномасштабним вторгненням Російської Федерації в Україну, що призвело до значного загострення безпекової ситуації та необхідності оперативного реагування на сигнали повітряної тривоги, обстріли, евакуацію та інші форми загроз.

Освіта з безпеки життєдіяльності має бути доступною, інтерактивною та ефективною, особливо для учнів — однієї з найуразливіших груп. У зв'язку з поширенням цифрових технологій та активним впровадженням дистанційного навчання, постає необхідність створення інструментів, що забезпечують якісне засвоєння правил поведінки під час НС. Одним із таких інструментів є освітній веб-застосунок, здатний стати ефективною платформою для ознайомлення з критично важливою інформацією.

Актуальність теми зумовлена як зростанням техногенних і природних загроз, так і військовими діями, що підсилюють потребу в системному навчанні та відпрацюванні алгоритмів дій у разі небезпеки. Існуючі ресурси часто є розрізненими, малодоступними або не адаптованими до навчального процесу.

Метою даної роботи є розробка веб-застосунку, що надає інтерактивний навчальний контент із правил поведінки під час загроз, систему оцінки знань та дозволяє вчителям і учням ефективно взаємодіяти у межах НП. Очікувані результати включають підвищення обізнаності учнів, скорочення часу на підготовку матеріалів для вчителів та забезпечення широкої доступності й зручності завдяки веб-формату.

РОЗДІЛ 1.
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Актуальність проблеми

1.1.1 Роль цифрових технологій у сфері освіти

У XXI столітті цифрові технології стали невіддільною частиною освітнього процесу, змінивши способи навчання та взаємодії вчителів і учнів. Впровадження електронного навчання, онлайн-платформ та мобільних застосунків забезпечило значний прорив у доступності освітніх матеріалів для широкої аудиторії. Завдяки цифровим технологіям, навчання стало можливим не тільки в авдиторіях, але й в будь-якому зручному місці, що значно підвищило рівень гнучкості у НП [2].

Онлайн-курси, вебінари та інші освітні платформи дозволяють учням та студентам здобувати нові знання та навички в зручний для них час і в комфортному темпі. Цифрові технології також дають змогу викладачам адаптувати матеріали до різних стилів навчання та індивідуальних потреб учнів, створюючи більш персоналізований підхід до навчання. Крім того, інтерактивні інструменти, такі як відеоуроки, тести та форуми для обговорень, сприяють глибшому засвоєнню матеріалу.

Роль веб-застосунків у цій сфері є незаперечною. Вони забезпечують прямий зв'язок між учнями та викладачами, даючи можливість для спільної роботи, обміну знаннями та зворотного зв'язку, незалежно від географічного положення учасників. Веб-платформи також забезпечують зручний доступ до матеріалів, тестів, відео та інших ресурсів, що значно підвищує ефективність НП.

1.1.2 Безпека та обізнаність під час загроз

В умовах сучасних глобальних загроз, зокрема у зв'язку з військовими конфліктами, надзвичайними ситуаціями чи природними катастрофами, особливо важливим аспектом освітнього процесу є підготовка громадян до дій у кризових ситуаціях. Підвищення обізнаності щодо поведінки під час загроз, таких як повітряні тривоги, обстріли, хімічні атаки чи природні катастрофи, є необхідною частиною освіти.

Цифрові технології можуть стати потужним інструментом у навчанні правил безпеки, надаючи доступ до інтерактивних курсів, відео та тренінгів, що дозволяють учням відпрацьовувати навички, які можуть бути життєво важливими в реальних умовах. Застосування симуляцій та віртуальних тренажерів дає змогу створити реалістичні сценарії для відпрацювання правильних дій під час загроз.

Освіта в сфері безпеки не обмежується лише теоретичними знаннями, вона включає розвиток практичних навичок, таких як евакуація, надання першої домедичної допомоги та першої психологічної допомоги, реагування на хімічні чи біологічні загрози. Використання цифрових платформ для навчання в цій сфері дозволяє формувати ці навички в умовах, які можна адаптувати під індивідуальні потреби та ситуації.

1.1.3 Вплив повномасштабного вторгнення на навчальний процес

Повномасштабне вторгнення Російської Федерації в Україну в 2022 році стало визначальним моментом, що змусило значну частину освітнього процесу перейти на дистанційне навчання [1]. Це змусило освітні установи адаптувати свої підходи до навчання, щоб зберегти процес навчання, навіть під час воєнних дій, обстрілів та інших небезпек. Враховуючи значну роль технологій у підтримці навчання, багато учнів та студентів були змушені

перейти до онлайн форматів, що дозволило продовжити НП без фізичної присутності у класах та зменшити ризики для життя та здоров’я.

У таких умовах зросла потреба в інтеграції елементів цивільної оборони, безпеки та психологічної підтримки в навчальні програми. Зокрема, важливим стало впровадження освітніх інструментів для навчання учнів та студентів правилам поведінки під час повітряних тривог, обстрілів, евакуацій та інших НС. Це стало нагальною необхідністю для забезпечення безпеки громадян, зокрема учнів та педагогів.

Веб-застосунки, розроблені спеціально для навчання у надзвичайних ситуаціях, мають величезний потенціал у забезпеченні безперервного навчання в умовах війни. Такі платформи можуть містити інформацію про поведінку під час загроз, психологічну допомогу, плани евакуації та інші важливі аспекти, що дозволяють мінімізувати ризики під час небезпеки. У цих умовах веб-застосунки, які поєднують освітній процес і навчання правилам цивільної оборони, можуть стати важливим елементом системи освіти, забезпечуючи безпеку та готовність до дій у разі НС.

Таким чином, розробка та впровадження освітніх веб-застосунків, які забезпечують не тільки доступ до знань, а й підготовку до поведінки в умовах загроз, є важливим кроком у напрямку адаптації системи освіти до сучасних викликів.

1.2 Аналіз існуючих програмних рішень

Сучасний ринок освітніх технологій стрімко розвивається, пропонуючи широкий спектр платформ та інструментів для підтримки НП в онлайн-форматі. В умовах цифровізації освіти зросла кількість веб-застосунків, орієнтованих на інтерактивне навчання, самоперевірку знань, організацію навчальних курсів та управління навчанням. Проте більшість із цих рішень сконцентровані переважно на передачі теоретичних знань і не мають фокусу

на розвиток практичних навичок поведінки в умовах надзвичайних ситуацій. Це створює нішу для розробки спеціалізованих платформ, які б інтегрували освітні та безпекові компоненти в єдиному рішенні.

У цьому підрозділі розглядаються три популярні освітні платформи — Kahoot!, Moodle та naurok.com.ua — з метою виявлення їхніх функціональних можливостей, переваг та обмежень щодо використання для формування навичок поведінки під час загроз.

1.2.1 Загальна характеристика наявних продуктів

Kahoot! — це популярна міжнародна платформа, яка дозволяє створювати інтерактивні вікторини, опитування, дискусії та навчальні ігри. Основний акцент у ній зроблено на гейміфікації освітнього процесу: учні можуть відповідати на запитання в режимі реального часу, змагаючись між собою за бали, що значно підвищує їхню мотивацію та залученість. Платформа активно використовується в школах, університетах і навіть у корпоративному навчанні. Основними перевагами Kahoot! є її простота у створенні та використанні контенту, висока інтерактивність і можливість колективного проходження вікторин у реальному часі. Однак, незважаючи на зручність, платформа орієнтована переважно на перевірку знань, а не на формування практичних навичок. Вона не передбачає інструментів для створення повноцінних навчальних курсів і має обмежену підтримку української мови, що є суттєвим недоліком у контексті створення національного освітнього продукту.

Moodle — це потужна і гнучка система управління навчанням (LMS) з відкритим кодом, яка широко використовується у всьому світі. Вона дозволяє викладачам створювати структуровані курси, додавати різноманітні типи завдань, такі як тести, форуми чи завантаження файлів, а також відслідковувати прогрес користувачів і формувати аналітичні звіти. До переваг Moodle належать висока масштабованість і адаптивність під потреби

навчального закладу, підтримка освітніх стандартів SCORM та LTI, а також можливість повної локалізації українською мовою. Водночас, ця система має досить складний інтерфейс, що може створювати труднощі для звичайних користувачів, а її налаштування потребує певних технічних знань та ресурсів. Крім того, Moodle орієнтований переважно на класичне академічне навчання і не враховує специфіку кризових ситуацій або підготовки до надзвичайних подій.

Naurok.com.ua — це українська освітня онлайн-платформа, орієнтована на вчителів, учнів та батьків. Вона пропонує великий обсяг навчального контенту: тести, інтерактивні завдання, відеоуроки та матеріали для самостійної підготовки. Платформа активно використовується в українських школах і має переваги у вигляді повністю україномовного контенту, широкої бази готових матеріалів та доступного безкоштовного функціоналу для зареєстрованих користувачів. Проте, незважаючи на функціональність, Naurok не містить спеціалізованого контенту з питань цивільної безпеки чи захисту в умовах загроз. Їй також бракує інтерактивних інструментів для моделювання ситуацій і тренування поведінки під час тривог чи обстрілів. Відсутня інтеграція з системами сповіщення та офлайн-практиками, що обмежує її потенціал як інструменту для навчання у сфері безпеки.

1.2.2 Порівняльний аналіз функціоналу аналогів

У таблиці 1.1 представлено порівняльний аналіз трьох популярних освітніх платформ — Kahoot!, Moodle та Naurok — за критеріями, що мають ключове значення для організації ефективного цифрового навчання, зокрема у контексті формування навичок поведінки в умовах надзвичайних ситуацій. Порівняння охоплює технічні характеристики, можливості адаптації до тематики загроз, доступність, гнучкість та рівень інтерактивності.

Таблиця 1.1 – порівняння освітніх веб-додатків

Критерій	Kahoot!	Moodle	Naurok
Мова інтерфейсу	Англійська, обмежена підтримка UA	Багатомовний, підтримує українську	Українська
Тип системи	Веб-сервіс	LMS (CMS)	Освітній портал
Платформа	Веб, мобільні застосунки	Веб, мобільні застосунки	Веб
Підтримка тестування	Так (вікторини)	Так	Так (тести)
Інтерактивність	Висока (гейміфікація)	Середня (форуми, відео, тести)	Середня
Можливість імпорту курсів	Часткова (CSV/Excel-файли запитань)	Повна (SCORM, IMS, Moodle backup)	Відсутня
Матеріали для навчання	Відсутні	Є повний курс-менеджмент	Матеріали лише для вчителів
Адаптація до теми загроз	Відсутня	Можлива вручну	Відсутня
Наявність офлайн-доступу	Ні	Частково (через плагіни)	Ні
Безкоштовність	Частково	Безкоштовний (open source)	Безкоштовний
Застосовність до навчання правилам поведінки під час загроз	Низька	Середня	Низька

1.2.3 Висновки щодо недоліків існуючих рішень

У результаті аналізу трьох популярних освітніх платформ — Kahoot!, Moodle та naurok.com.ua — було виявлено, що жодна з них не є повноцінним рішенням для навчання правилам поведінки під час загроз. Платформи Kahoot! та naurok.com.ua орієнтовані переважно на тестування знань, але не мають повноцінного функціоналу для створення навчальних курсів чи матеріалів саме для учнів. Moodle є найбільш гнучкою системою з широким функціоналом, можливістю створення курсів, тестів, імпорту матеріалів та підтримкою міжнародних стандартів. При цьому, базове користування платформою є достатньо інтуїтивним, але її повне налаштування, встановлення на сервер та адаптація під потреби навчального закладу можуть вимагати залучення фахівця з ІТ.

Таким чином, існує потреба у створенні спеціалізованого веб-застосунку, який буде поєднувати простоту у користуванні, підтримку мультимедійних навчальних матеріалів та інтерактивні тести для учнів — з акцентом на поведінку під час надзвичайних ситуацій.

1.3 Аналіз вимог до програмного забезпечення

У процесі розробки програмного забезпечення надзвичайно важливо на початковому етапі чітко сформулювати вимоги до майбутньої системи. Це дозволяє не лише забезпечити відповідність розробленого продукту очікуванням кінцевих користувачів, а й уникнути неоднозначностей під час реалізації функціоналу. Вимоги виступають основою для проєктування архітектури, вибору технологій, тестування та оцінки готовності продукту.

Загалом їх поділяють на дві основні категорії: функціональні та нефункціональні [3]. Функціональні вимоги визначають, які конкретні дії має виконувати система — які сервіси надаються користувачу, як відбувається

взаємодія з інтерфейсом, які сценарії використання підтримуються. Натомість нефункціональні вимоги стосуються якості роботи застосунку: швидкодії, надійності, безпеки, масштабованості, зручності у використанні тощо. Обидва типи вимог є критично важливими для створення ефективного, стабільного й корисного цифрового продукту.

1.3.1 Функціональні вимоги

Функціональні вимоги базуються на аналізі предметної області, наявних рішень та потреб цільової аудиторії — вчителів і учнів.

Основні функціональні можливості:

- Реєстрація та автентифікація користувачів
  - Користувачі можуть створювати облікові записи як учні або викладачі;
  - Вхід до системи здійснюється з перевіркою ролі та наданням доступу до відповідного функціоналу.
- Панель адміністратора/вчителя
  - Авторизовані викладачі можуть створювати, редагувати та видаляти власні навчальні курси;
  - Додавання уроків у форматі тексту та відео;
  - Створення тестових завдань з автоматичною перевіркою.
- Інтерфейс учня
  - Перегляд навчальних матеріалів у зручній послідовності;
  - Проходження тестів після кожного уроку або модуля;
  - Отримання результатів тестування.
- Управління прогресом навчання
  - Збереження результатів проходження уроків і тестів;
  - Показ прогресу користувача у курсі.
- Повідомлення та інструкції

– Виведення контекстних підказок для вчителів під час створення курсів, уроків та тестів;
– Сповіщення користувача про завершення уроків і результати.

Цей набір функціональностей забезпечує базову підтримку навчального процесу та сприяє формуванню знань і навичок поведінки під час загроз у шкільному середовищі.

1.3.2 Нефункціональні вимоги

Нефункціональні вимоги описують характеристики програмного забезпечення, які не пов’язані безпосередньо з його функціональністю, але мають важливе значення для якості, зручності використання, продуктивності, безпеки та сумісності застосунку.

Основні нефункціональні вимоги до освітнього веб-застосунку:

- Зручність використання (Usability)
  – Інтерфейс повинен бути інтуїтивно зрозумілим для учнів молодшого та середнього шкільного віку, а також для вчителів без технічного досвіду.
  – Навігація має бути логічною та послідовною.
- Доступність (Accessibility)
  – Застосунок має бути доступний через сучасні браузери з підтримкою стандартів HTML5 та ES6.
  – Інтерфейс має адаптуватися до різних розмірів екранів (від смартфонів до настільних ПК).
- Продуктивність (Performance)
  – Сторінки застосунку повинні завантажуватись не довше 2 секунд при середньому з'єднанні з Інтернетом.
  – Система повинна стабільно працювати при одночасному доступі до 100 користувачів.

- Безпека (Security)
  - Всі дані користувачів повинні передаватися через захищене з’єднання (HTTPS).
  - Має бути реалізована базова авторизація з розмежуванням прав доступу для вчителів і учнів.
  - Захист від несанкціонованого доступу до персональних даних та результатів тестування.
- Надійність (Reliability)
  - Застосунок повинен стабільно працювати протягом тривалого часу без збоїв.
- Масштабованість (Scalability)
  - Архітектура повинна дозволяти розширення функціональності або збільшення кількості користувачів без суттєвого перепроектування системи.
- Можливість супроводу та оновлення (Maintainability)
  - Код повинен бути структурованим, документованим та підтримуваним, що полегшить майбутні оновлення і розвиток системи.

ВИСНОВОК ДО РОЗДІЛУ 1

У першому розділі було проаналізовано предметну область, до якої належить освітній веб-застосунок, призначений для навчання правилам поведінки під час загроз. Встановлено актуальність розробки у контексті зростаючих потреб у безпечному та ефективному освітньому середовищі, зокрема в умовах повномасштабної війни в Україні.

Проведено аналіз популярних освітніх платформ Moodle, Kahoot!, naurok.com.ua, що дозволило виявити їхні переваги й недоліки, а також визначити ключові функції, які варто реалізувати в новому застосунку. Особливу увагу приділено адаптивності, інтерактивності та зручності використання в навчальному процесі.

Також у розділі було сформульовано вимоги до програмного забезпечення — як функціональні (створення курсів, проходження тестів, реєстрація користувачів тощо), так і нефункціональні (продуктивність, безпека, зручність, доступність та масштабованість). Це створює основу для ефективного проєктування та розробки системи в подальших етапах дипломної роботи.

РОЗДІЛ 2.
ТЕХНОЛОГІЇ ТА ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір мов програмування та фреймворків

Під час розробки веб-застосунку для навчання правилам поведінки під час надзвичайних ситуацій було обрано сучасний технологічний стек, орієнтований на швидку розробку, легке масштабування та простоту підтримки.

2.1.1 Клієнтська мова програмування

Основною мовою програмування для клієнтської частини є TypeScript — надмножина JavaScript, яка забезпечує статичну типізацію, підвищує безпеку коду та полегшує його супровід [8]. Статична типізація дозволяє виявляти помилки ще на етапі розробки, що є особливо важливим для проєктів із великою кількістю взаємодіючих модулів.

2.1.2 Побудова інтерфейсу

Для побудови користувацького інтерфейсу було використано бібліотеку React, яка дозволяє реалізувати компонентну архітектуру. Це дає змогу створювати повторно використовувані UI-елементи, спрощує управління станом застосунку та підвищує продуктивність за рахунок віртуального DOM. Переміщення між сторінками реалізується за допомогою бібліотеки React Router, що дозволяє створювати односторінковий застосунок зі швидкою та плавною навігацією.

Разом із перевагами використання React існують і певні недоліки. Зокрема, відсутність повної структури "з коробки" може ускладнити організацію проєкту, особливо на початковому етапі. Щоб уникнути хаосу в

кодовій базі, було прийнято рішення дотримуватись модульної структури проєкту з розділенням на компоненти, сторінки, сервіси та типи, що забезпечує масштабованість і легкість навігації.

Ще однією важливою задачею стало забезпечення стабільної продуктивності при збільшенні кількості компонентів і складності інтерфейсу. Для цього в проєкті було реалізовано чіткий поділ логіки між локальним і глобальним станом: локальний стан компонентів керується через useState такі як вибрані відповіді в тестах, стан завантаження, помилки тощо), а для глобального стану — таких як автентифікація користувача та оновлення прогресу — застосовується контекст API. Такий підхід дозволив уникнути зайвих перерендерів і зменшити навантаження на інтерфейс. Контекст також дав змогу ефективно передавати дані між компонентами без необхідності "пробросу" пропсів через кілька рівнів вкладеності, що особливо важливо в навчальному застосунку з великою кількістю залежних частин.

2.1.3 Управління стилями

Для оформлення інтерфейсу замість класичного CSS або зовнішніх бібліотек стилів було застосовано підхід із використанням інлайн-стилів, які задаються безпосередньо в компонентах. Це забезпечує кращий контроль над зовнішнім виглядом елементів, дозволяє динамічно змінювати стилі залежно від стану та контексту, а також спрощує реалізацію адаптивного дизайну. Зокрема, розмітка адаптується під розміри вікна браузера, що забезпечує зручність користування на мобільних пристроях.

2.1.4 Форматування контенту

Опис курсів і уроків створюється за допомогою редактора Quill, який підтримує широкий набір форматів для тексту. Викладачі можуть використовувати різноманітні інструменти для форматування тексту, такі як заголовки, списки, виділення, вставка посилань, зображень та багато іншого.

Для зручності в процесі введення надано інтерфейс з кнопками форматування та контекстними підказками, що дозволяють швидко застосовувати необхідні стилі. Це забезпечує гнучкість і зручність при створенні контенту без потреби в знанні Markdown-синтаксису.

2.2 Система керування базами даних

Під час розробки освітнього веб-застосунку, спрямованого на формування знань і навичок поведінки у надзвичайних ситуаціях, техногенних аваріях, терористичних загрозах та інших критичних умовах, було обрано Firebase Cloud Firestore як основну систему зберігання та обробки даних. Це хмарна документоорієнтована база даних, яка є частиною екосистеми Google Firebase [5] і надає широкі можливості для зберігання динамічного контенту, персоналізації користувацького досвіду та обробки інформації в режимі реального часу без потреби в окремому серверному середовищі.

Firestore належить до категорії NoSQL-баз даних [10] — тобто систем, які не використовують традиційні реляційні схеми з таблицями, рядками та стовпцями. Замість цього вона працює за моделлю документів і колекцій, де кожен документ має унікальний ідентифікатор і довільну структуру, що включає вкладені об'єкти та масиви. Документи групуються в колекції, які можуть містити підколекції, утворюючи гнучку, масштабовану структуру даних. Такий підхід дозволяє ефективно зберігати навчальний контент, відстежувати індивідуальний прогрес користувачів і підтримувати розвиток платформи без потреби жорстко визначеної схеми даних.

У межах цього проєкту Firestore використовується для зберігання інформації про курси, включаючи назву, опис, автора, список уроків та наявність фінального тесту; контенту уроків, такого як текстовий матеріал, інструкції, ілюстрації та посилання; тестових завдань і результатів їх проходження; даних користувачів, зокрема ролей, індивідуального прогресу,

обраних курсів; а також зв’язків між курсами, уроками, тестами та користувачами.

Серед основних переваг використання Firestore варто відзначити гнучкість структури зберігання, що дозволяє поступово додавати нові поля або типи контенту без ризику порушення цілісності існуючих даних. Завдяки підтримці синхронізації в реальному часі інтерфейс користувача оновлюється автоматично після зміни даних у базі, що особливо важливо для миттєвого відображення результатів тестування або змін у прогресі навчання. Інтеграція з Firebase Authentication забезпечує надійну автентифікацію та прив’язку даних до облікових записів користувачів, даючи змогу обмежувати доступ до окремих функцій на основі ролей. Firestore автоматично масштабується відповідно до навантаження, зберігаючи стабільну роботу застосунку незалежно від кількості користувачів. Офіційний JavaScript SDK дозволяє легко інтегрувати базу даних з інтерфейсом, створеним на React з використанням TypeScript, що прискорює розробку та зменшує затримки під час взаємодії з даними.

Водночас використання Firestore має певні обмеження, які були враховані під час проєктування системи. Зокрема, є обмеження на глибину вкладеності та кількість одночасних запитів, що може впливати на продуктивність при зростанні кількості користувачів. Щоб мінімізувати ці ризики, було реалізовано нормалізовану структуру бази даних, у якій дані поділено на окремі колекції з чіткими зв’язками за допомогою унікальних ідентифікаторів. Крім того, з огляду на обмеження Firebase Storage та високі витрати при зберіганні великих мультимедійних файлів, функцію завантаження відео було замінено на інтеграцію з зовнішніми відеохостингами. Зокрема, реалізовано підтримку вбудованих відео з YouTube, що дозволяє користувачам отримувати доступ до навчального відеоконтенту без навантаження на інфраструктуру платформи. Окрему увагу було приділено безпеці — для запобігання витоку чутливої інформації були

налаштовані правила доступу, які обмежують можливість читання та запису даних лише для автентифікованих користувачів з відповідними правами.

Таким чином, Firestore є не лише засобом зберігання даних, а й основою всієї архітектури освітнього застосунку. Його використання дозволило створити централізовану, гнучку та інтерактивну платформу, що забезпечує стабільну роботу, персоналізацію навчального процесу та ефективну взаємодію користувачів з навчальним контентом.

2.3 Середовище розробки (IDE, інструменти)

У процесі створення освітнього веб-застосунку для навчання правилам поведінки під час загроз було використано сучасне середовище розробки та набір інструментів, що забезпечують ефективну розробку, налагодження, візуалізацію та підтримку застосунку. Оскільки застосунок орієнтований на динамічну взаємодію з користувачем, підтримку курсів, уроків, тестів та відображення індивідуального прогресу, було обрано інструменти, що дозволяють зручно працювати з React, TypeScript, Firebase та інтерфейсною логікою.

Основним середовищем розробки виступає PyCharm Professional — інтегроване середовище розробки від JetBrains, яке, попри свою основну орієнтацію на Python, також має повну підтримку фронтенд-розробки: React, HTML, CSS, Tailwind та JavaScript/TypeScript. PyCharm забезпечує високу зручність у роботі з проєктом завдяки автодоповненню, статичному аналізу коду, потужному редактору, навігації по структурі проєкту, а також інтеграції з Git та консоллю. Вибір PyCharm був обумовлений попереднім досвідом роботи з цим середовищем, а також його широкими можливостями для налаштування і розширення функціоналу.

Для керування версіями коду було використано систему контролю версій Git у поєднанні з GitHub. Такий підхід дозволив забезпечити історію

змін проєкту, створення окремих гілок для реалізації нових функцій, а також зберігати резервні копії на віддаленому репозиторії.

Візуальна частина застосунку (іконки, логотип, аватар користувача, стилізовані ілюстрації для курсів або тестів) створювалась у графічному редакторі Figma — сучасному інструменті для дизайну інтерфейсів, який підтримує створення адаптивних макетів, компонентів та прототипів. Figma дозволяє швидко експортувати графіку у форматах, придатних для інтеграції у React-застосунок. Оскільки застосунок має освітнє спрямування, візуальна складова відіграє важливу роль у залученні користувачів та забезпеченні інтуїтивно зрозумілого інтерфейсу.

Для тестування взаємодії з Firebase Firestore, перевірки коректності HTTP-запитів, авторизації та доступу до ресурсів застосовувався Postman. Це дозволяло на ранніх етапах перевірити коректність передачі даних між клієнтською частиною і сервером Firebase, особливо у випадках створення курсів, оновлення уроків, збереження результатів тестування тощо.

Окрім цього, під час розробки активно використовувались DevTools браузера Google Chrome. З їх допомогою здійснювалось налагодження JavaScript/TypeScript-коду, перевірка структури DOM, аналіз CSS-стилів, тестування адаптивності інтерфейсу на різних розмірах екранів та оцінка продуктивності застосунку.

Публікація готового застосунку та його оновлення здійснювались за допомогою Firebase Hosting — хмарного рішення для розміщення статичних вебзастосунків. Деплой проводився через Firebase CLI, що дозволяло швидко викладати зміни на продакшн-сервер без необхідності налаштування власного хостингу або FTP-доступу.

Таблиця 2.1 Зведена таблиця використаних інструментів:

Компонент	Інструмент / Технологія
Середовище розробки	PyCharm Professional
Система контролю версій	Git + GitHub
Графічні матеріали	Figma
Тестування HTTP-запитів	Postman
Інструменти відладки	DevTools (Google Chrome)
Хостинг	Firebase Hosting

Таким чином, вибраний набір інструментів дозволив забезпечити повний цикл розробки — від створення структури коду і дизайну до тестування і розгортання освітнього застосунку, призначеного для навчання правилам поведінки в умовах загроз.

2.4 Вибір архітектури застосунку

Під час вибору архітектури освітнього веб-застосунку для навчання правилам поведінки під час загроз було проаналізовано кілька можливих варіантів: класична клієнт-серверна архітектура із власним сервером, мікросервісна модель із окремими API та хмарними функціями, а також безсерверний (serverless) підхід. З урахуванням обмеженого бюджету, швидкості розгортання, вимог до масштабованості та підтримки реального часу, було обрано безсерверну архітектуру на базі хмарної платформи Firebase [4] у поєднанні з клієнтським застосунком на React + TypeScript.

Такий підхід дозволяє значно спростити інфраструктуру: основна бізнес-логіка виконується на клієнтському рівні в браузері користувача, а роль серверної частини виконує хмарна платформа Firebase, яка забезпечує готові

сервіси для зберігання даних, аутентифікації, розгортання застосунку та контролю доступу.

Клієнтська частина застосунку реалізована з використанням сучасних вебтехнологій — фреймворку React у поєднанні з мовою програмування TypeScript. Завдяки використанню бібліотеки React Router реалізовано односторінкову архітектуру SPA [7], яка забезпечує плавну навігацію між розділами без перезавантаження сторінки. Компонентна структура застосунку дозволила ізолювати функціональність різних частин інтерфейсу — уроків, тестів, курсу, результатів тощо — та легко масштабувати систему в майбутньому. Всі компоненти взаємодіють із локальним станом через хуки React (useState, useEffect) або спеціально створені кастомні хуки, які інкапсулюють повторювану логіку.

Ключовим архітектурним рішенням стало використання сервісного шару — окремих TypeScript-файлів, які відповідають за взаємодію з Firebase. Наприклад, логіка оновлення прогресу чи отримання результатів тестів реалізована в окремих сервісах progressService.ts, quizzesService.ts тощо, які централізують роботу з базою даних. Такий підхід дозволив уникнути дублювання коду в компонентах, покращити структуру програми та спростити її супровід.

Уся серверна логіка делегована хмарній платформі Firebase, яка в даному випадку виступає як Backend-as-a-Service. Для зберігання даних використовується Firestore — документно-орієнтована NoSQL база даних, що дозволяє зберігати інформацію про курси, уроки, тести, користувачів та їхній прогрес. Аутентифікація користувачів реалізована через Firebase Authentication із підтримкою реєстрації, входу, виходу та підтвердження електронної пошти. Увесь клієнтський інтерфейс після збірки автоматично розгортається на Firebase Hosting [6], що дозволяє максимально швидко публікувати нові версії без налаштування власного сервера.

Особливу увагу приділено контролю доступу до даних. Для цього використовуються Firebase Security Rules, які дозволяють перевіряти права користувача відповідно до його ролі. Наприклад, лише викладач має право створювати або редагувати курси, тоді як звичайний користувач може лише переглядати їх та проходити. Всі запити до Firestore перевіряються цими правилами, що забезпечує базовий рівень безпеки навіть за умови клієнтської реалізації логіки. Крім того, завдяки використанню Firebase SDK з підтримкою оновлень у реальному часі, застосунок може автоматично реагувати на зміни даних у базі, наприклад, відображати нові уроки або оновлений прогрес без перезавантаження сторінки.

Разом із перевагами serverless-підходу застосунок зіткнувся з певними обмеженнями. Зокрема, обмеженою є можливість реалізації складної серверної логіки, такої як автоматичне надсилання листів, генерація звітів або перевірки складних умов доступу. У рамках цього проєкту для обходу такого обмеження частина складної логіки, як-от перевірка прав на редагування курсів, була реалізована безпосередньо у клієнтському коді з додатковим захистом через Firebase Security Rules. Такий гібридний підхід дозволяє зберегти зручність і простоту безсерверної моделі, водночас мінімізуючи ризики несанкціонованого доступу.

Загалом обрана архітектура забезпечує високу гнучкість, масштабованість та стабільність освітнього застосунку. Завдяки використанню сучасних інструментів, таких як React, TypeScript та Firebase, вдалося реалізувати функціональний веб-застосунок без необхідності налаштовувати традиційний сервер. Це дозволило зосередитися на створенні якісного навчального контенту, логіки уроків і тестів, а також зручного та інтуїтивного інтерфейсу.

ВИСНОВОК ДО РОЗДІЛУ 2

У цьому розділі було обґрунтовано вибір мов програмування, фреймворків та технологій, які були використані для реалізації веб-застосунку. Використання сучасного стеку на основі TypeScript, React, Tailwind CSS та Firebase дало змогу досягти високої продуктивності, гнучкості й масштабованості системи.

Було проаналізовано особливості роботи з базою даних Firestore, визначено ключові колекції та їхню взаємодію, а також представлено загальну архітектуру даних. Особливу увагу приділено організації структури компонентів, зв’язкам між курсами, уроками, тестами та прогресом користувача.

Також описано інструменти, які використовувалися під час розробки, зокрема середовище PyCharm Professional для написання коду та Figma — для створення візуальних компонентів інтерфейсу, таких як аватари, іконки та інші графічні елементи.

Таким чином, обрані технологічні рішення повністю відповідають вимогам до створення сучасного освітнього веб-застосунку, забезпечують зручність розробки, підтримку й подальше масштабування системи.

РОЗДІЛ 3.
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура застосунку

Розроблене програмне забезпечення освітнього веб-застосунку реалізовано за принципом клієнт-серверної архітектури з використанням підходу Serverless, що дозволяє спростити процес розгортання, зменшити витрати на обслуговування та забезпечити гнучкість у масштабуванні. Архітектура клієнт-серверної взаємодії освітнього застосунку зображено на рис. 3.1.

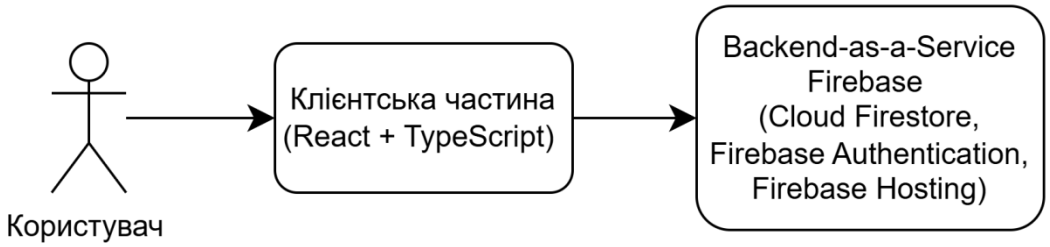


Рисунок 3.1 – Архітектура клієнт-серверної взаємодії

3.1.1 Компоненти системи та їх взаємозв'язки

Основна бізнес-логіка застосунку розроблена на клієнтській стороні з використанням бібліотеки React та мови програмування TypeScript, що забезпечує чітку типізацію, модульність і зручність у підтримці коду. Вся взаємодія з даними, авторизацією та навігацією відбувається безпосередньо у фронтенді, що виконується у веб-браузері користувача.

Структура клієнтської частини побудована на компонентному підході та розбита на модулі відповідно до функціональної структури. Основними складовими клієнтської частини є такі директорії: components/, pages/, services/, hooks/, utils/, а також assets/, public/ і файли App.tsx / main.tsx.

У директорії components/ зосереджені багаторазові інтерфейсні компоненти, які забезпечують логіку відображення і взаємодії з користувачем. Структура директорії components/ зображена на рис. 3.2. Наприклад, компоненти CourseCard та CourseCardMini відображають курси у вигляді карток, тоді як CourseDetails, LessonList і QuizList відповідають за детальне представлення змісту курсу, уроків та тестів відповідно. Компоненти авторизації LoginForm, RegisterForm, LogoutButton та GoogleLoginButton забезпечують взаємодію з Firebase Authentication. Крім того, Header, Footer, UserInfo і LearningProgress — це елементи загального інтерфейсу користувача, що відповідають за постійні частини макету застосунку. Компонент BackToCourseButton використовується для зручної навігації між розділами курсу.

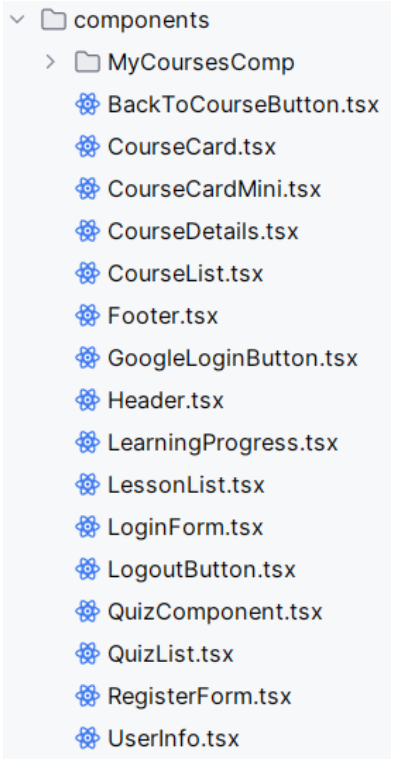


Рисунок 3.2 – Структура директорії components/

Окремим функціональним блоком є підмодуль MyCoursesComp/, який містить компоненти для створення та редагування навчального контенту (рис. 3.3). Зокрема, CourseForm, LessonForm, QuizForm та EditCourseModal

реалізують відповідні форми для керування курсами та їх наповненням, а компоненти CourseList і CourseContentList дозволяють переглядати доступні курси та матеріали до них.

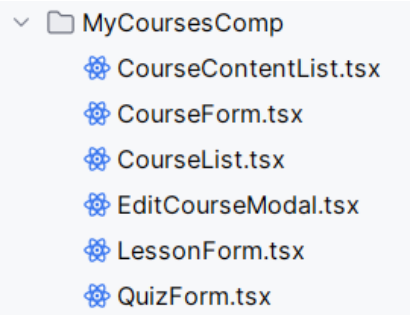


Рисунок 3.3 – Структура директорії MyCoursesComp/

У папці pages/ зосереджені всі основні сторінки освітнього веб-застосунку, кожна з яких відповідає окремому маршруту в React Router. Структуру директорії можна переглянути на рис. 3.4. Сторінка Home є головною та містить вступну інформацію про платформу. CoursesPage відображає перелік доступних курсів, а при переході на CoursePage користувач переглядає повну інформацію про обраний курс, зокрема уроки та тести. LessonPage використовується для перегляду вмісту конкретного уроку, а QuizPage — для проходження пов’язаних з курсом тестів. Сторінки Login і Register реалізують функціонал входу та реєстрації з підтримкою Firebase Authentication. MyCoursesPage призначена для викладачів і дозволяє створювати, редагувати та переглядати власні курси. Сторінка Profile містить інформацію про користувача, його роль, ім’я, email та навчальний прогрес. Загальна структура сторінок демонструє модульність і чіткий поділ відповідальностей у межах клієнтської частини за стосунку.

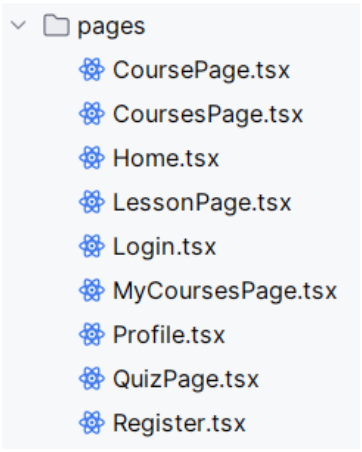


Рисунок 3.4 – Структура директорії pages/

Взаємодія з базою даних та хмарними сервісами Firebase реалізована через директорію services/, де кожен файл інкапсулює логіку обробки відповідної сутності (рис. 3.5). Наприклад, у coursesService.ts, lessonsService.ts, quizzesService.ts реалізовані CRUD-операції для курсів, уроків і тестів відповідно. Файли usersService.ts, progressService.ts відповідають за роботу з користувачами та їхнім навчальним прогресом. Файл mycoursesService.ts містить бізнес-логіку для взаємодії з курсами, створеними викладачем, зокрема функції для створення, оновлення, видалення курсів та їх компонентів, а також обробки відповідних запитів до Firestore. Конфігурація доступу до Firebase зберігається в окремому файлі firebaseConfig.ts.

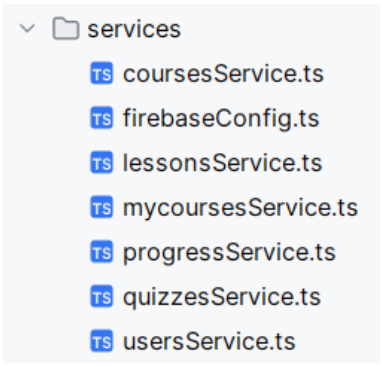


Рисунок 3.5 – Структура директорії services/

Для спрощення роботи з контекстом автентифікації використовується кастомний хук useAuth.ts (рис. 3.6), що дозволяє отримати поточного користувача, перевірити його роль і статус входу в систему. У директорії utils/

зберігаються допоміжні функції, зокрема calculateCourseProgress.ts — для обрахунку прогресу користувача в курсі (рис. 3.7).

```
// Хук для аутентифікації користувача
export const useAuth = () => {
  const [user, setUser] = useState<User | null>(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const unsubscribe = onAuthStateChanged(auth, (firebaseUser) => {
      setUser(firebaseUser);
      setLoading(false);
    });

    return () => unsubscribe();
  }, []);

  return { user, loading };
};
```

Рисунок 3.6 – Наповнення файлу useAuth.ts

```
export const calculateCourseProgress = (
  lessonsCompleted: string[],
  testsPassed: string[],
  courseLessons: string[],
  courseQuizzes: string[]
): number => {
  // Фільтруємо лише ті уроки, які є в курсі
  const validCompletedLessons = lessonsCompleted.filter(lessonId =>
    courseLessons.includes(lessonId)
  );

  // Фільтруємо лише ті квізи, які є в курсі
  const validCompletedQuizzes = testsPassed.filter(quizId =>
    courseQuizzes.includes(quizId)
  );

  const totalTasks = courseLessons.length + courseQuizzes.length;
  const completedTasks = validCompletedLessons.length + validCompletedQuizzes.length;

  return totalTasks ? Math.round((completedTasks / totalTasks) * 100) : 0;
};
```

Рисунок 3.7 – Наповнення файлу calculateCourseProgress.ts

Директорія public/ містить файли, доступні напряму з веб-сервера, включно з 404.html, аватарами користувачів тощо. Структуру директорії public/ можна переглянути на рис. 3.8.

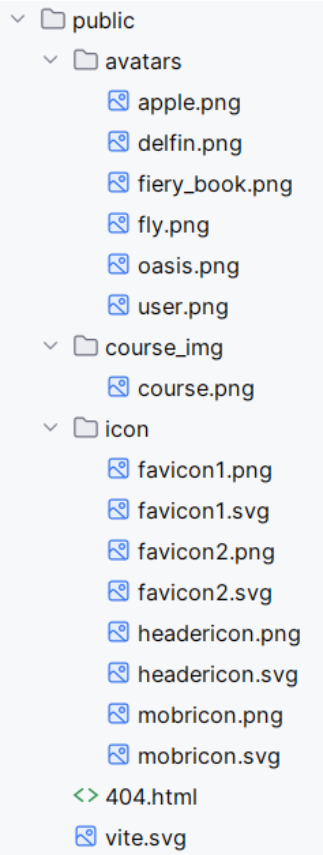


Рисунок 3.8 – Структура директорії services/

Файли main.tsx та App.tsx є точкою входу в застосунок. У файлі main.tsx здійснюється ініціалізація React-додатку, підключення глобальних стилів через index.css та обгортання застосунку в компонент BrowserRouter для забезпечення маршрутизації. Саме тут відбувається рендеринг кореневого компонента App у DOM. Файл App.tsx реалізує основну структуру застосунку: маршрутизацію за допомогою React Router, компонування сторінок (Header, main, Footer), та стилізацію за допомогою об’єктів CSSProperties. Структура дозволяє легко масштабувати застосунок за рахунок централізованого управління навігацією та розташування загальних елементів інтерфейсу.

Усі ці компоненти взаємодіють між собою через механізми React Context, Props та внутрішні сервіси, а обмін даними з серверною частиною (Firebase) здійснюється через SDK поверх REST API.

Серверна частина застосунку базується на хмарних сервісах Firebase, що виступають як BaaS, надаючи готові рішення для зберігання даних, аутентифікації та хостингу та обробки запитів. Взаємозв'язок між клієнтською і серверною частинами здійснюється за допомогою Firebase SDK — спеціальної бібліотеки, що надає інтерфейс для взаємодії з хмарними сервісами. Клієнт надсилає запити до Firebase безпосередньо з браузера, а SDK виконує всі необхідні операції поверх REST API.

Наприклад, при вході в систему клієнтський застосунок звертається до Firebase Authentication для перевірки облікових даних. У разі успішної автентифікації, доступ до даних у Firestore відбувається відповідно до ролі користувача, що дозволяє реалізувати контроль доступу на рівні клієнта з урахуванням правил безпеки Firebase.

3.1.2 Потоки даних у системі

Потоки даних у системі освітнього веб-застосунку визначають, як інформація передається між компонентами клієнтської та серверної частини. Основним каналом комунікації виступає Firebase SDK, який надає доступ до сервісів Firebase у браузері.

Після запуску застосунку користувач взаємодіє з інтерфейсом, де за допомогою React динамічно відображається доступний неавторизованим користувачам контент (курси, уроки, тести). У разі реєстрації чи входу в систему ініціюється запит до Firebase Authentication. Після успішної аутентифікації система перевіряє роль користувача — викладач чи учень, що визначає рівень доступу до функціоналу.

У разі, якщо користувач авторизований як викладач, йому доступні як проходження курсів на платформі, так і функції для створення та редагування

власних курсів, уроків та тестів. Усі ці дії призводять до запису відповідних документів у Firestore.

З іншого боку, якщо користувач авторизований як студент, його доступ обмежується лише переглядом курсу, проходженням уроків та виконанням тестів. Результати тестування автоматично зберігаються у Firestore, що дозволяє користувачу бачити свій результат та відслідковувати прогрес проходження певного курсу.

Передача даних відбувається у режимі реального часу — Firestore підтримує реактивні оновлення, що дозволяє системі миттєво відображати зміни. Наприклад, після того як користувач завершує тест, його результат миттєво оновлюється в Firestore, що дозволяє йому отримати актуальну інформацію про прогрес порходження курсу.

Завдяки використанню хмарної інфраструктури Firebase система не потребує окремого бекенд-сервера, а вся логіка обробки запитів, безпеки та зберігання делегується на керовані сервіси.

Отже, потоки даних у системі рухаються від клієнтської частини до серверної через Firebase SDK, обробляються в Firebase Authentication та Firestore, забезпечуючи надійне зберігання та обробку інформації з можливістю динамічного оновлення даних у реальному часі для всіх учасників освітнього процесу.

3.2 Фронтенд-розробка

У межах фронтенд-розробки було створено зручний та інтуїтивно зрозумілий інтерфейс користувача, який забезпечує доступ до основного функціоналу освітнього веб-застосунку. Основні етапи включали реалізацію навігаційної структури, побудову сторінок, стилізацію елементів інтерфейсу та адаптацію під різні типи пристроїв.

3.2.1 Технології інтерфейсу користувача

Для побудови інтерфейсу було обрано React — популярну JavaScript-бібліотеку, яка дозволяє створювати компонентно-орієнтовані динамічні односторінкові застосунки. Її основними перевагами є гнучкість, висока продуктивність, активна спільнота та можливість повторного використання компонентів, що значно спрощує підтримку та масштабування застосунку.

Для реалізації маршрутизації між сторінками застосунку було використано React Router, який забезпечує керовану навігацію без перезавантаження сторінки, що є критичним для SPA-підходу.

Щодо стилізації, було застосовано підхід інлайн-стилів, який передбачає створення об’єктів стилів безпосередньо у JavaScript-коді компонентів. Такий підхід дозволяє легко керувати стилями на основі стану компонента, використовувати змінні для темізації та реюзу стилів, зменшити залежність від зовнішніх CSS-файлів.

Це забезпечило високу гнучкість у розробці та швидке реагування інтерфейсу на дії користувача. Окрім цього, інтерфейс був адаптований під різні розміри екранів і пристрої, що покращує досвід користувача на мобільних телефонах і планшетах.

3.2.2 Реалізація сторінок та навігації

У межах розробки інтерфейсу користувача було реалізовано ключові сторінки, які забезпечують повний функціонал та логіку взаємодії користувача з освітнім веб-застосунком. Нижче подано опис основних сторінок із деталями реалізації, адаптивності та прикладами інтерфейсних рішень.

Навігація між сторінками здійснюється за допомогою бібліотеки React Router, що забезпечує відображення відповідного вмісту залежно від маршруту без перезавантаження браузера. Фрагмент коду, що ілюструє реалізацію маршрутизації в головному компоненті App.tsx, наведено на рис. 3.9.

```
function App() {  Show usages  Ksenia Lohanko
  return (
    <div style={styles.container}>
        <Header />
        <main style={styles.main}>
          <Routes>
            <Route path="/" element={<Home />} />
            <Route path="/profile" element={<Profile />} />
            <Route path="/login" element={<Login />} />
            <Route path="/register" element={<Register />} />
            <Route path="/my-courses" element={<MyCoursesPage />} />
            <Route path="/courses" element={<Courses />} />
            <Route path="/courses/:courseId" element={<CoursePage/>} />
            <Route path="/courses/:courseId/lessons/:lessonId" element={<LessonPage />} />
            <Route path="/courses/:courseId/quizzes/:quizId" element={<QuizPage />} />
          </Routes>
        </main>
        <Footer />
    </div>
  );
}
```

Рисунок 3.9 – Реалізація маршрутизації

Хедер (Header)

Хедер є універсальним елементом інтерфейсу, який відображається на всіх сторінках за стосунку (рис. 3.10). Він містить логотип, назву платформи, а також навігаційні посилання: «Головна», «Курси», «Мої курси», «Кабінет» та кнопку «Вхід» або «Вийти» залежно від стану автентифікації користувача.

Рисунок 3.10 – Хедер

Сторінка Кабінет користувача є важливою частиною застосунку, що дозволяє користувачам переглядати та редагувати свої особисті дані, відстежувати прогрес у курсах, а також надає можливість видалення акаунту. Для реалізації цього компонента було створено кілька важливих функцій та підкомпонентів.

Почнемо з того, що для завантаження даних користувача на сторінці використовуються хуки, зокрема useEffect. Цей хук ініціалізує завантаження даних про користувача після того, як компонент монтується. Для відстеження

автентифікації користувача використовується метод onAuthStateChanged з Firebase, який перевіряє, чи користувач авторизований. Якщо так, викликається функція getUserData, яка отримує інформацію про користувача з Firestore, а також перевіряє, чи є помилки під час цього процесу. Реалізація завантаження даних користувача за допомогою події зміни автентифікації в Firebase зображено на рис. 3.11

```
useEffect(() => {
  const unsubscribe = auth.onAuthStateChanged(async (currentUser) => {
    setLoading(true);
    try {
      if (!currentUser) {
        console.warn("User not found");
        setUser(null);
      } else {
        const data = await getUserData(currentUser.uid);
        setUser(data || null);
      }
    } catch (error) {
      console.error("Error loading user data:", error);
      setUser(null);
    } finally {
      setLoading(false);
    }
  });

  return () => unsubscribe();
}, []);
```

Рисунок 3.11 – Реалізація завантаження даних користувача

Дані користувача відображаються за допомогою компонента UserInfo, що включає ім'я користувача, емейл, статус (роль), а також можливість змінити аватар. У фрагменті реалізовано логіку збереження та скасування змін профілю користувача (рис. 3.12). Для цього використовуються копії початкових значень initialDisplayName та initialPhotoURL. При збереженні дані оновлюються в Firebase Authentication методом updateProfile і Firestore методом updateDoc. Якщо користувач скасовує редагування, відбувається відновлення початкових значень. Окрім того, користувач може обирати новий аватар із доступних варіантів, що зберігаються локально.

```
// Зберігаємо копії початкових значень для скасування
const [initialDisplayName, setInitialDisplayName] = useState(user?.displayName || "");
const [initialPhotoURL, setInitialPhotoURL] = useState(user?.photoURL || avatars[0]);

const [isEditing, setIsEditing] = useState(false);
const [error, setError] = useState<string>("");
const handleSave = async () => {  Show usages  Ksenia Lohanko
  try {
    if (auth.currentUser) {
      await updateProfile(auth.currentUser, { displayName, photoURL });

      const userRef = doc(db, "users", auth.currentUser.uid);
      await updateDoc(userRef, { displayName, photoURL });

      setIsEditing(false);
      // Зберігаємо нові початкові значення після збереження
      setInitialDisplayName(displayName);
      setInitialPhotoURL(photoURL);
    }
  } catch (err) {
    console.error("Error updating profile:", err);
    setError("Не вдалося оновити профіль.");
  }
};

const handleCancel = () => {  Show usages  Ksenia Lohanko
  setDisplayName(initialDisplayName); // Відновлюємо початкові значення
  setPhotoURL(initialPhotoURL);       // Відновлюємо початкові аватарки
  setIsEditing(false);
};
```

Рисунок 3.12 – Реалізація збереження та скасування змін профілю користувача

Що стосується прогресу в курсах, використовується компонент LearningProgress, де кожен курс має свою картку з компоненту CourseCardMini з відсотком прогресу (рис. 3.13), а також можливість переходу до самого курсу для детальнішого перегляду. Прогрес відображається на основі даних, збережених у Firestore.

```
return (
  <div>
    <h2>Мої курси</h2>
    <div style={{ display: "flex", flexWrap: "wrap", gap: "20px" }}>
      {userCourses.length === 0 ? (
        <p>У вас немає активних курсів.</p>
      ) : (
        userCourses.map((course) => (
          <CourseCardMini
            key={course.courseId}
            course={course}
            progress={course.progressPercentage}
            onClick={() => handleCourseClick(course.courseId)}
          />
        ))
      )}
    </div>
  </div>
);
```

Рисунок 3.13 – Реалізація відображення прогресу користувача

Важливою частиною сторінки є також функція видалення акаунту. Користувач може натискати кнопку "Видалити акаунт", після чого відкривається модальне вікно, де він підтверджує своє рішення. Після підтвердження викликаються методи для видалення користувача з Firestore та Firebase Auth, використовуючи функції deleteDoc та deleteUser відповідно. Після успішного видалення акаунту користувач буде перенаправлений на головну сторінку.

Для обробки помилок на сторінці передбачено відображення повідомлень про помилки в разі виникнення проблем із завантаженням даних або оновленням профілю. Якщо дані ще не завантажено, на екрані відображатиметься індикатор "Завантаження...". У разі помилки під час завантаження або редагування профілю користувач побачить відповідне повідомлення про помилку, наприклад, "Не вдалося оновити профіль" чи "Не вдалося видалити акаунт".

Сторінка перегляду всіх курсів (рис. 3.14) реалізована як головна точка входу для користувачів у модулі навчального контенту. Основна логіка

реалізована у компоненті CoursesPage, який відображає інтерфейс пошуку та вибору категорій, а також компонент CourseList, який відповідає за завантаження курсів із Firestore, їхню фільтрацію (рис. 3.15) та відображення (рис. 3.16).

```
return (<div style={styles.pageContainer}>
        <h1>Курси</h1>

        <div style={styles.filtersContainer}>
            <input
                type="text"
                placeholder="Пошук курсу..."
                value={searchTerm}
                onChange={handleSearchChange}
                style={styles.searchInput}
            />

            <select
                value={selectedCategory}
                onChange={handleCategoryChange}
                style={styles.select}
            >
                <option value="">Всі категорії</option>
                {categories.map((category) => (<option key={category} value={category}>
                        {category}
                    </option>))}
            </select>
        </div>

        <CourseList searchTerm={searchTerm} selectedCategory={selectedCategory}/>
    </div>);
```

Рисунок 3.14 – Реалізація сторінки перегляду курсів

```
const courseData = await getCourses();

// Фільтрація курсів за категорією
const filteredCourses = courseData.filter(course =>
  (selectedCategory === "" || course.category.includes(selectedCategory)) &&
  (searchTerm === "" || course.title.toLowerCase().includes(searchTerm.toLowerCase()))
);

setCourses(filteredCourses);
```

Рисунок 3.15 – Реалізація фільтрації курсів за категоріеб та назвою

```
    return (
      <div>
        <h2>Список курсів</h2>
        <div style={{ display: "grid", gridTemplateColumns: "repeat(auto-fill, minmax(250px, 1fr))", gap: "20px" }}>
          {courses.map((course) => (
            <CourseCard
              key={course.courseId}
              course={course}
              progress={progressData[course.courseId] || 0}
              onClick={() => handleCourseClick(course.courseId)}
            />
          ))}
        </div>
      </div>
    );
```

Рисунок 3.16 – Реалізація відображення списку курсів

У компоненті CoursesPage при першому завантаженні виконується запит до бази даних для отримання унікального списку категорій на основі даних усіх курсів , після чого вони виводяться у випадаючому списку для фільтрації. Пошук реалізовано через контрольований інпут, що фільтрує курси за назвою. Реалізацію отримання унікальних категорій курсів та пошуку і фільтрації наведено на рис. 3.17.

```
useEffect(() => {
  const loadCategories = async () => {  Show usages  Ksenia Lohanko *
    try {
      const courses: Course[] = await getCourses();
      // Витягуємо категорії з кожного курсу
      const allCategories = courses.flatMap(course =>
        course.category.map((cat: string) => cat.trim())
      );
      // Використовуємо Set для видалення дублікатів і конвертуємо в масив
      const uniqueCategories = Array.from(new Set(allCategories));
      setCategories(uniqueCategories);
    } catch (error) {
      console.error("Не вдалося завантажити категорії", error);
    }
  };

  loadCategories();
}, []);

const handleSearchChange = (event: React.ChangeEvent<HTMLInputElement>) => {  Show usages  Ksenia Lohanko
  setSearchTerm(event.target.value);
};

const handleCategoryChange = (event: React.ChangeEvent<HTMLSelectElement>) => {  Show usages  Ksenia Lohanko
  setSelectedCategory(event.target.value);
};
```

Рисунок 3.17 – Реалізація отримання категорій, пошуку і фільтрації курсів

Компонент CourseList виконує завантаження відфільтрованого списку курсів з урахуванням обраної категорії та пошукового запиту, а також – якщо користувач авторизований – завантажує прогрес проходження по кожному курсу, використовуючи функцію getUserData(user.uid). Прогрес зберігається у вигляді відсотка та відображається через візуальний елемент прогресбару.

Кожен курс відображається у вигляді картки через компонент CourseCard, який містить зображення, назву, опис, категорії у вигляді бейджів та прогресбар. При натисканні на картку виконується перенаправлення на сторінку перегляду курсу за допомогою navigate("/courses/" + courseId).

Сторінка курсу CoursePage показує розширену інформацію про конкретний курс: велике зображення зверху, назва, розгорнутий опис, категорії, прогресбар, а також список усіх уроків. Під прогресбаром уроків відображається кнопка «Почати/Продовжити курс», яка веде на наступний непройдений урок. Уроки згруповано у вертикальний список із позначками виконання — ті, що завершені, мають спеціальну позначку (рис. 3.18). Окремий блок — фінальний тест. Його кнопка стає активною лише після завершення всіх уроків (рис. 3.19), що перевіряється через стан виконання. Фрагмент коду, що відповідає за умовне відображення фінального тесту після завершення всіх уроків курсу зображено на рис. 3.20.

Рисунок 3.18 – Позначення пройденого уроку

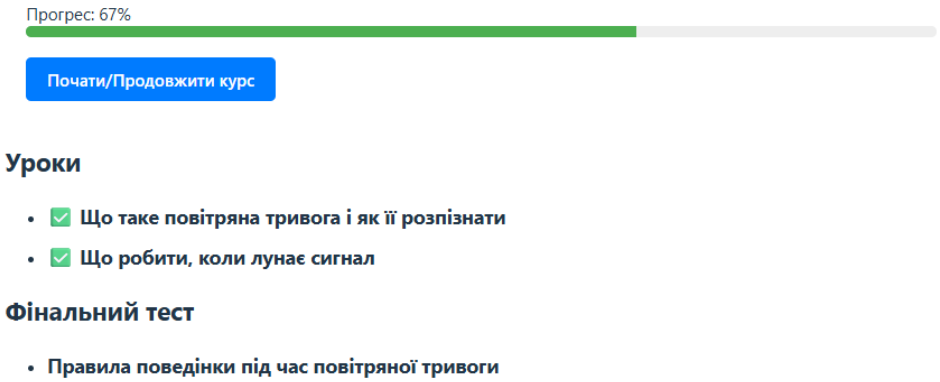


Рисунок 3.19 – Активний фінальний тест після проходження всіх уроків

```
{finalQuiz && (
  <>
    <h2>Фінальний тест</h2>
    {userProgress.lessonsCompleted.length === lessons.length ? (
      <QuizList
        quizzes={[finalQuiz]}
        completedQuizzes={userProgress.testsPassed}
      />
    ) : (
      <p>Пройдіть усі уроки, щоб отримати доступ до фінального тесту.</p>
    )}
  </>
)}
```

Рисунок 3.20 – Реалізація умовного відображення фінального тесту

Сторінка уроку LessonPage є ключовою одиницею контенту в межах курсу (рис. 3.21). Вона відкривається після вибору певного уроку зі списку на сторінці курсу або натискання кнопки «Продовжити курс». Урок містить назву, текстовий контент, де підтримується базове форматування у стилі Markdown: жирний текст, заголовки, списки, а також мультимедійний вміст. Медіа в уроці реалізовано через вставки посилань. Якщо в полі videoUrl вказане стандартне або коротке YouTube-посилання, воно автоматично конвертується у формат <iframe> для вбудованого перегляду відео безпосередньо на сторінці. Фрагмент коду на рис. 3.22 демонструє функцію getYoutubeEmbedUrl, яка перетворює звичайне або скорочене YouTube-посилання у формат для вставки відео через тег <iframe>. Текст уроку виводиться за допомогою HTML-розмітки dangerouslySetInnerHTML, що дозволяє викладачам створювати структурований, зручний для сприйняття матеріал з базовими стилями.

```
return (
    <div style={styles.pageContainer}>
        <BackToCourseButton/>
        {lesson && (
            <>
                <h1 style={styles.title}>{lesson.title}</h1>

                <div style={styles.content} dangerouslySetInnerHTML={{__html: lesson.content}}/>

                {lesson.videoUrl && (
                    <div style={styles.videoContainer}>
                        <iframe
                            width="100%"
                            height="400px"
                            src={getYoutubeEmbedUrl(lesson.videoUrl)}
                            title={lesson.title}
                            allowFullScreen
                        />
                    </div>
                )}

                {quiz && !quizCompleted && (
                    <div style={styles.quizWrapper}>
                        <QuizComponent
                            quiz={quiz}
                            selectedOption={selectedOption}
                            setSelectedOption={setSelectedOption}
                            onSubmitQuiz={handleSubmitQuiz}
                        />
                    </div>
                )}

                {quizCompleted && (
                    <button style={styles.completeButton} onClick={handleCompleteLesson}>
                        Завершити урок
                    </button>
```

Рисунок 3.21 – Реалізація інтерфейсу сторінки уроку

```
const getYoutubeEmbedUrl = (url: string): string => {  Show usages  new *
    const match = url.match(/(?:https?:\/\/)?(?:www\.)?youtube\.com\/watch\?v=([^&]+)/);
    if (match && match[1]) {
        return `https://www.youtube.com/embed/${match[1]}`;
    }
    const shortMatch = url.match(/(?:https?:\/\/)?youtu\.be\/([^?&]+)/);
    if (shortMatch && shortMatch[1]) {
        return `https://www.youtube.com/embed/${shortMatch[1]}`;
    }
    return url;
};
```

Рисунок 3.22 – Реалізація функції getYoutubeEmbedUrl

Таким чином, сторінка уроку не лише подає навчальний матеріал, а й інтерактивно перевіряє засвоєння знань, забезпечуючи логічний контрольний етап перед переходом далі.

Окрему увагу приділено інтегрованому компоненту тестування QuizComponent.tsx, який реалізований як багаторазовий, універсальний компонент перевірки знань і може бути використаний як на сторінці звичайного уроку, так і на сторінці фінального тесту. Компонент приймає дані тесту, поточний вибір відповідей, функцію для зміни стану setSelectedOption, та обробник підтвердження onSubmitQuiz. Він підтримує вибір одного варіанту на запитання, відображення правильної/неправильної відповіді після надсилання, а також скидання відповідей.

На рисунку 3.23 зображено частину логіки компонента, що відповідає за обробку вибору варіантів відповіді, перевірку правильності, керування станом відправлення (isSubmitted), а також за методи handleSubmit та handleReset, які дозволяють підтвердити або скинути тест відповідно.

```tsx
interface QuizComponentProps {
  quiz: Quiz;
  selectedOption: { [key: string]: string };
  setSelectedOption: React.Dispatch<React.SetStateAction<{ [key: string]: string }>>;
  onSubmitQuiz: () => void;
}

const QuizComponent: React.FC<QuizComponentProps> = ({ quiz, selectedOption, setSelectedOption, onSubmitQuiz }) => {
  const [isSubmitted, setIsSubmitted] = useState(false);

  const handleOptionChange = (question: Question, option: string) => {
    setSelectedOption((prev) => ({
      ...prev,
      [question.question]: option,
    }));
  };

  const renderAnswerFeedback = (selected: string, correct: string) => {
    if (!isSubmitted) return null;
    if (!selected) return null;


    return selected === correct ? (
      <span style={{ color: 'green', marginLeft: '10px' }}>✓</span>
    ) : (
      <span style={{ color: 'red', marginLeft: '10px' }}>✗</span>
    );
  };

  const handleSubmit = () => {
    setIsSubmitted(true);
    onSubmitQuiz();
  };

  const handleReset = () => {
    setIsSubmitted(false);
    setSelectedOption({});
  };
```

Рисунок 3.23 – Реалізація логіки вибору варіантів і обробки тесту

На рисунку 3.24 показано розмітку, яку повертає компонент: виведення заголовка тесту, запитань та варіантів відповідей, а також відображення позначки «✔» або «✘» після надсилання відповідей, що забезпечує візуальний зворотний зв’язок користувачу.

```
return (
  <div style={styles.quizContainer}>
    <h2>{quiz.title}</h2>
    {quiz.questions.map((q, index) => (
      <div key={index} style={styles.questionBlock}>
        <div style={styles.questionHeader}>
          <p style={styles.questionText}>{q.question}</p>
          {/* Відображаємо позначку поруч з питанням */}
          {renderAnswerFeedback(selectedOption[q.question], q.correct)}
        </div>
        <ul>
          {q.options.map((option, i) => (
            <li key={i}>
              <input
                type="radio"
                name={q.question}
                value={option}
                checked={selectedOption[q.question] === option}
                onChange={() => handleOptionChange(q, option)}
                disabled={isSubmitted}
              />
              {option}
            </li>
          ))}
        </ul>
      </div>
    ))}
    <div>
      <button style={styles.completeButton} onClick={handleSubmit}>
        Підтвердити
      </button>
      {isSubmitted && (
        <button style={styles.resetButton} onClick={handleReset}>
          Скинути тест
        </button>
      )}
    </div>
  </div>
);
```

Рисунок 3.24 – Реалізація інтерфейсу тесту

Якщо у властивості lesson.quizId задано ідентифікатор тесту, QuizComponent автоматично рендериться в межах уроку. У цьому випадку тест є обов'язковим для проходження перед завершенням уроку: кнопка «Завершити урок» стає доступною лише після правильної відповіді на всі запитання тесту. Така логіка реалізована через умовний рендеринг у React — компонент перевіряє, чи встановлений стан quizCompleted === true, перш ніж показати кнопку. Процес перевірки відповідей винесено до функції handleSubmitQuiz (рис. 3.25), де відбувається порівняння вибраних

користувачем опцій з правильними відповідями, і лише за повного збігу тест вважається успішно завершеним.

```
const handleSubmitQuiz = async () => {  Show usages  new *
    if (!quiz || !userId || !courseId || !lessonId) return;

    const correctAnswers = quiz.questions.filter(
        (q) => selectedOption[q.question] === q.correct
    );

    if (correctAnswers.length === quiz.questions.length) {
        setQuizCompleted(true);
        console.log('Тест завершено успішно');

        await updateUserProgress(userId, courseId, lessonId, quiz.quizId ?? undefined);
    } else {
        setQuizCompleted(false);
        console.log('Тест не пройдено');
    }
};
```

Рисунок 3.25 – Реалізація функції handleSubmitQuiz

Сторінка фінального тесту призначена для завершення навчального курсу після проходження всіх уроків. Вона містить інтерактивний компонент тесту QuizComponent, який відображає запитання, варіанти відповідей та дозволяє користувачу надсилати свої відповіді. Після натискання кнопки підтвердження onSubmitQuiz, відповіді перевіряються, і якщо всі правильні — прогрес оновлюється, а користувач бачить повідомлення про успішне завершення курсу з кнопкою повернення на сторінку курсу.

Кожна сторінка уроку та тесту також містить кнопку «Повернутися до курсу», що забезпечує швидку навігацію для користувача. Таким чином, інтерфейс курсу передбачає повний цикл навчання — від послідовного ознайомлення з контентом до перевірки знань і підсумкового тестування з прозорими умовами переходу між етапами.

Сторінка MyCoursesPage, де викладач може переглядати власні курси, редагувати їх або створювати нові, складається з двох частин. Перша частина — форма створення курсу, реалізована через компонент CourseForm. Цей компонент дозволяє викладачу створити повноцінний навчальний курс, що включає загальну інформацію, список уроків та фінальний тест. З точки зору

інтерфейсу, CourseForm відображається у вигляді окремої секції, що містить поля для заповнення назви курсу, контенту з підтримкою простого форматування на зразок Markdown, а також кнопки для створення уроків і тестів. Ключовим елементом форми є можливість додавати уроки безпосередньо під час створення курсу — для цього передбачено кнопку "Створити урок", яка відкриває модальне вікно або вбудований компонент для введення даних уроку. Усередині кожного уроку також можна створити тест — цей функціонал реалізовано так, щоб не ускладнювати загальну логіку, завдяки використанню типу Omit<Quiz, 'quizId' | 'courseId'>. У кінці форми передбачено кнопку для створення фінального тесту, що додається як окремий блок до курсу. Вся інформація про курс, включаючи його структуру з уроками та тестами, зберігається у відповідному стані форми, який після натискання на кнопку "Створити курс" передається до сервісного методу createFullCourse(...) (рис. 3.26), що відповідає за збереження даних у базу даних Firebase.

```
export const createFullCourse = async (teacherId: string,    Show usages   Ksenia Lohanko *
    courseData: Omit<Course, "courseId" | "lessons" | "quiz_id">, lessonsData:
    LessonWithOptionalQuiz[], finalQuizData?: Omit<Quiz, "quizId" | "courseId">): Promise<Course> => {
    const batch = writeBatch(db);
    const courseRef = doc(collection(db, "courses"));
    const courseId = courseRef.id;
    const lessonRefs = lessonsData.map(() => doc(collection(db, "lessons")));
    const lessonIds: string[] = [];
    let finalQuizId: string | undefined;
    if (finalQuizData) {
        const quizRef = doc(collection(db, "quizzes"));
        finalQuizId = quizRef.id;
        batch.set(quizRef, {...finalQuizData, courseId, teacherId, createdAt: new Date().toISOString(),});
    }
    lessonsData.forEach((lesson, index) => {
        const lessonRef = lessonRefs[index];
        const lessonId = lessonRef.id;
        lessonIds.push(lessonId);
        let quizId: string | undefined;
        if (lesson.quiz) {
            const quizRef = doc(collection(db, "quizzes"));
            quizId = quizRef.id;
            batch.set(quizRef, {...lesson.quiz, courseId, teacherId, createdAt: new Date().toISOString(),});
        }
        batch.set(lessonRef, {...lesson, courseId, teacherId, order: index, quizId, createdAt: new Date().toISOString(),});
    });
    const fullCourse: Course = {...courseData, courseId, lessons: lessonIds, quiz_id: finalQuizId, teacherId,
        createdAt: new Date().toISOString(),};
    batch.set(courseRef, fullCourse);
    await batch.commit();
    return fullCourse;
};
```

Рисунок 3.26 – Реалізація методу createFullCourse

З технічної точки зору, CourseForm має просту внутрішню структуру, поділену на окремі підкомпоненти для полегшення підтримки. Для додавання нового уроку використовується компонент LessonForm, а для створення тестів в уроках і фінального опитування —QuizForm. Таке розділення дозволяє легко керувати станом, виконувати перевірку введених даних та оновлювати список елементів.

Друга частина сторінки MyCoursesPage — це перегляд, редагування та видалення існуючих курсів, реалізована через компоненти CourseList та EditCourseModal. Компонент CourseList відповідає за відображення списку курсів, що належать авторизованому викладачу. Він отримує дані про курси з бекенду, а також приймає колбеки onEdit та onDelete, які викликаються при натисканні відповідних кнопок. Візуально курси представлені у вигляді сітки карток, де кожна картка містить обкладинку, назву, короткий опис і дві кнопки — "Редагувати" та "Видалити". При спробі видалити курс спрацьовує стандартне підтвердження window.confirm, що є простим але ефективним UX-підходом для запобігання випадковому видаленню. Якщо дані ще завантажуються або курсів немає — відображається відповідне повідомлення. Картки також передбачають fallback для зображення, якщо посилання не працює. Реалізація цього модулю наведена на рис. 3.27.

```
return (
  <div style={styles.container}>
    <h3 style={styles.heading}>Список курсів</h3>

    {isLoading ? (
      <p style={styles.message}>Завантаження курсів...</p>
    ) : courses.length === 0 ? (
      <p style={styles.message}>Курсів ще немає.</p>
    ) : (
      <div style={styles.grid}>
        {courses.map((course) => (
          <div key={course.courseId} style={styles.card}>
            <img
              src={course.image_url || "/course_img/course.png"}
              alt={course.title}
              style={styles.image}
              onError={(e) => {
                e.currentTarget.src = "/course_img/course.png";
              }}
            />

            <div style={styles.cardContent}>
              <h4 style={styles.title}>{course.title}</h4>
              <p style={styles.description}>{course.short_description}</p>
              <div style={styles.buttonGroup}>
                <button onClick={() => onEdit(course)} style={styles.editButton}>Редагувати</button>
                <button onClick={() => handleDelete(course.courseId)} style={styles.deleteButton}>Видалити</button>
              </div>
            </div>
          </div>
        ))}
      </div>
    )}
  </div>
);
```

Рисунок 3.27 – Реалізація відображення власних курсів викладача

Для редагування курсу використовується модальне вікно EditCourseModal, яке відкривається при натисканні кнопки "Редагувати". Усередині модального вікна формується окрема форма, що ініціалізується даними з обраного курсу, включаючи назву, короткий опис, повний опис, список категорій у вигляді одного рядка з розділенням через кому, список уроків і фінальний тест. Створення такої форми не просто копіює логіку CourseForm, а включає власну перевірку помилок при надсиланні. Ключовою особливістю цього компонента є інтеграція з CourseContentList — підкомпонентом, який на основі lessonIds та quizId отримує реальні назви уроків і тесту через асинхронні виклики до getLessonById та getQuizById. Це

дозволяє викладачу бачити актуальний вміст курсу та видаляти непотрібні елементи без перезавантаження сторінки.

Компонент CourseContentList використовує локальний стан для зберігання назв уроків і фінального тесту. Завдяки useEffect, після кожної зміни у списку lessonIds або quizId, він заново підтягує відповідні дані, гарантуючи, що відображається свіжа інформація. Видалення елементів із курсу синхронно оновлює стан як у EditCourseModal, так і в CourseContentList, забезпечуючи узгодженість UI. Візуально це реалізовано через прості списки з кнопками "Видалити", а при завантаженні відображається підпис "Завантаження...".

З загальноархітектурної точки зору, ці компоненти добре розділені за відповідальністю — CourseList рендерить відображення списку, EditCourseModal відповідає за логіку редагування, а CourseContentList — за інтерактивне відображення вмісту курсу. Це дає змогу підтримувати проект масштабованим і легко тестованим.

3.2.3 Стилізація та адаптивність

У процесі розробки освітнього веб-застосунку для навчання правилам поведінки під час загроз особлива увага приділялася візуальному оформленню інтерфейсу користувача та забезпеченню адаптивності для різних типів пристроїв. Застосунок створювався з урахуванням потреб як десктопних, так і мобільних користувачів, тому усі компоненти проєкту були спроєктовані так, щоб зручно виглядати на будь-якій роздільній здатності екрана.

Стилізація реалізована переважно за допомогою інлайн-стилів, що описуються у вигляді об’єктів JavaScript безпосередньо в компонентах React. Наприклад, у компоненті CourseList, який відповідає за відображення списку курсів викладача, стилі для карток курсів, кнопок, зображень та інших елементів оформлення задаються централізовано через об’єкт styles (рис. 3.28). Це дозволяє зберігати усі стилі в одному місці, забезпечуючи легкість у

підтримці та зміні оформлення, а також уникати потенційних конфліктів і дублювання CSS-класів. Такий підхід також підвищує модульність компонентів і спрощує структуру проєкту, особливо на етапі швидкої розробки функціональності коли змінюються або експериментально тестуються елементи інтерфейсу.

```
const styles: { [key: string]: CSSProperties } = {
  container: { marginTop: "24px", marginBottom: "24px"},
  heading: {
    fontSize: "20px",
    fontWeight: 600,
    marginBottom: "16px"
  },
  message: { color: "#6b7280"},
  grid: {
    display: "grid",
    gap: "24px",
    gridTemplateColumns: "repeat(auto-fill, minmax(280px, 1fr))"
  },
  card: {
    border: "1px solid #ccc",
    borderRadius: "8px",
    overflow: "hidden",
    backgroundColor: "#fff",
    boxShadow: "0 2px 4px rgba(0,0,0,0.1)",
    display: "flex",
    flexDirection: "column",
    transition: "box-shadow 0.3s"
  },
  image: {
    width: "100%",
    height: "200px",
    objectFit: "cover"
  },
```

Рисунок 3.28 – Реалізація стилізації у компоненті CourseList

Реалізовано адаптивну мобільну версію меню хедера, яка активується при зменшенні ширини вікна — з’являється бургер-іконка, натискання на яку відкриває вертикальний список посилань (рис. 3.29). Для цього застосовується метод window.innerWidth, що перевіряється у момент монтування компонента за допомогою React-хука useEffect. Додатково підписується слухач події resize, який у реальному часі відстежує зміну ширини вікна, динамічно перемикаючи режим між мобільним та десктопним. Такий підхід дозволяє не покладатися на

класичні CSS-медіа-запити, а гнучко керувати відображенням елементів через змінні стану у React-компоненті. Реалізацію даної частини показано на рис. 3.30, де видно логіку перемикання вмісту на основі поточної ширини екрана.

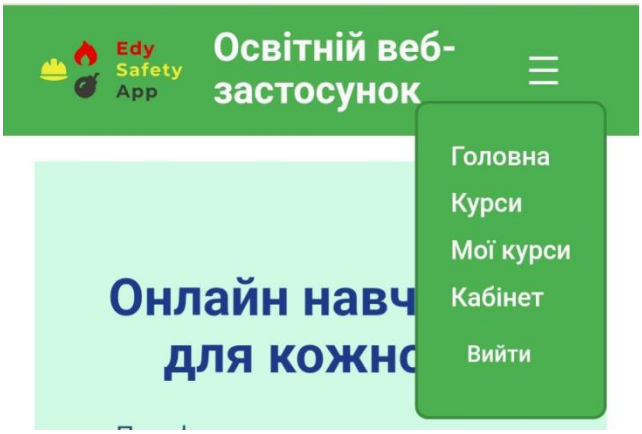


Рисунок 3.29 – Адаптивна мобільна версія хедера

```
useEffect(() => {
  const handleResize = () => {  Show usages  Ksenia Lohanko
    setIsMobile(window.innerWidth <= 970);
    if (window.innerWidth > 970) {
      setIsOpen(false);
    }
  };
  window.addEventListener("resize", handleResize);
  return () => window.removeEventListener("resize", handleResize);
}, []);
```

Рисунок 3.30 – Реалізація адаптивного перемикання через подію resize.

Всі компоненти застосунку, включно з CourseList, EditCourseModal, CourseForm, були оформлені у подібному стилі: стилі визначаються окремими об'єктами styles всередині файлів і призначаються через атрибут style={...}. Для адаптивного компонування елементів використовуються гнучкі властивості Flexbox, що дозволяє автоматично адаптувати компонування блоків на сторінках з курсами, уроками та формами редагування під різні розміри екранів без потреби у складному CSS. Наприклад, у списку курсів при вузькій ширині картки автоматично вишиковуються вертикально, а не горизонтально, що значно підвищує зручність перегляду на смартфонах (рис. 3.31).

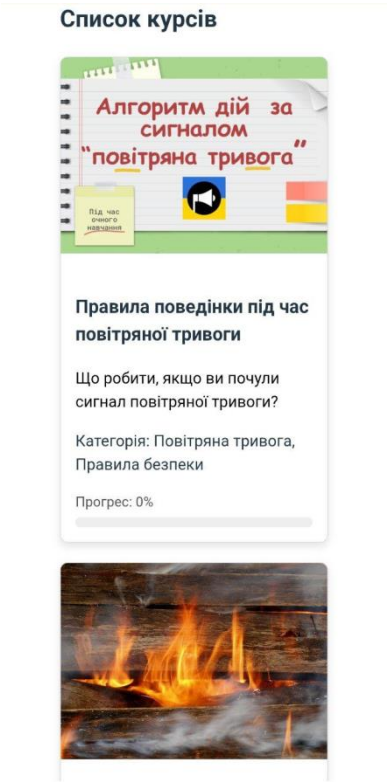


Рисунок 3.31 – Адаптивна мобільна версія списку курсів

Таким чином, застосунок поєднує в собі простий у підтримці підхід до стилізації та ефективну реалізацію адаптивності, що забезпечує комфортну роботу як з комп'ютера, так і з мобільного пристрою, без залучення додаткових бібліотек або фреймворків для UI.

3.3 Бекенд та робота з БД

Розробка бекенду освітнього веб-застосунку базується на хмарному рішенні Firebase, яке поєднує в собі базу даних, автентифікацію користувачів, хостинг та інші інструменти. Основна частина логіки бекенду реалізована через Firestore — NoSQL-документно-орієнтовану базу даних, що дозволяє зберігати дані у вигляді колекцій та документів. Робота з Firestore відбувається безпосередньо з клієнтської частини за допомогою офіційної бібліотеки firebase/firestore, що дозволяє виконувати як базові CRUD-операції, так і складні запити.

3.3.1 Організація структури бази даних

Структура бази даних Firebase Firestore у освітньому веб-застосунку побудована з урахуванням логічних колекцій, які відповідають основним сутностям застосунку, що забезпечує зручність у зберіганні та отриманні даних. Основними колекціями є courses, lessons, quizzes та users. Кожна з цих колекцій складається з окремих документів, кожен з яких має унікальний ідентифікатор. Загальна структура бази даних зображена на рис. 3.32.

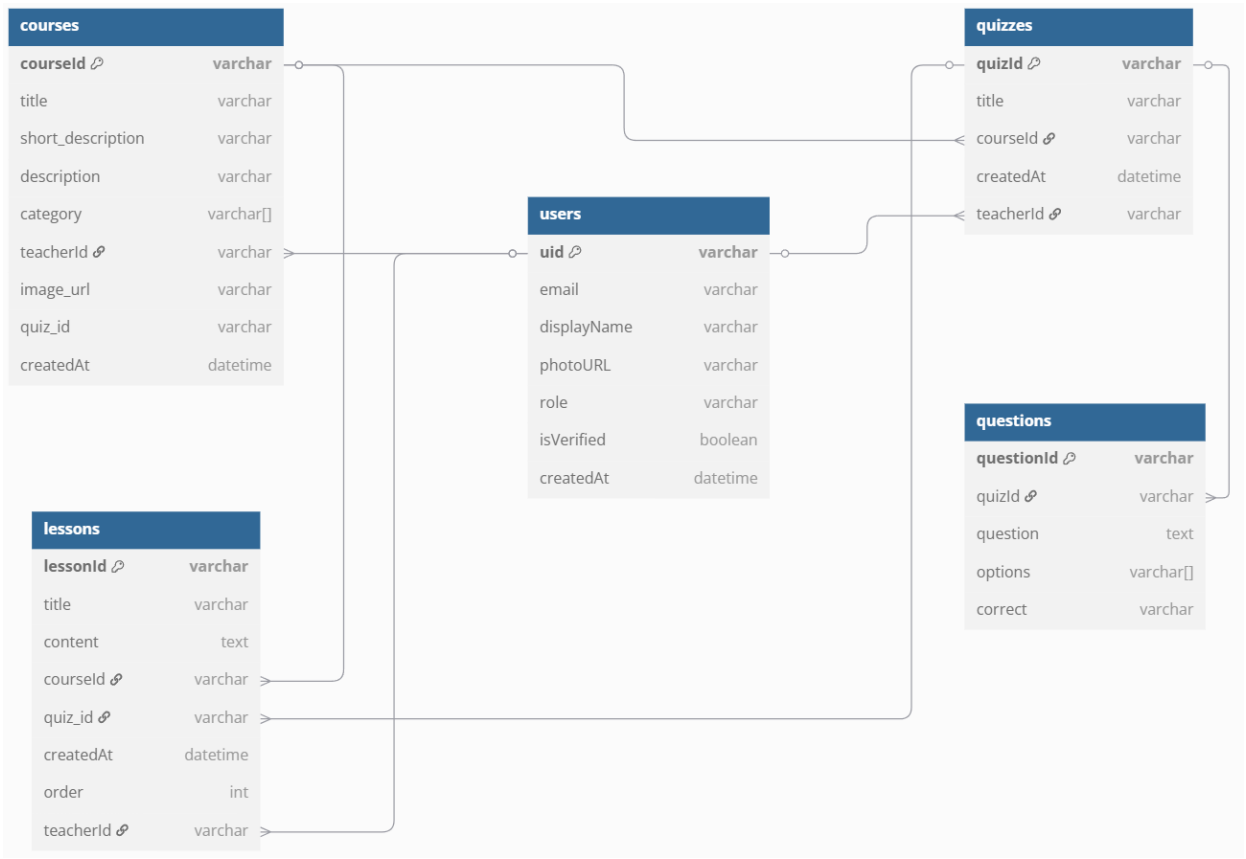


Рисунок 3.32 – Структура бази даних

Колекція users в базі даних зберігає інформацію про всіх користувачів освітнього веб-застосунку. Кожен документ цієї колекції ідентифікується за унікальним uid, який надається Firebase Auth під час реєстрації або авторизації користувача. Документ кожного користувача містить кілька полів, серед яких displayName — ім'я користувача, яке буде відображатися на сайті; email — електронна пошта користувача, яка також може бути використана для входу в систему; photoURL — посилання на фото або аватар користувача . Крім того,

зберігається поле role, яке визначає роль користувача в системі: "викладач" або "учень", що є важливим для контролю доступу до різних функцій застосунку.

Ще одне важливе поле — це progress, яке зберігає прогрес користувача по кожному курсу, до якого він зарахований. Це поле є об'єктом, де ключами є унікальні ідентифікатори курсів (courseId), а значеннями — дані про прогрес користувача, зокрема, масив lessonsCompleted (уроки, які користувач пройшов), масив testsPassed (тести, які користувач успішно склав), а також загальний відсоток виконання курсу через поле progressPercentage. Також в документі зберігається поле createdAt, яке вказує на дату та час реєстрації користувача в системі. Для верифікації електронної пошти може бути використано поле isVerified, що вказує на те, чи підтвердив користувач свою пошту. Така структура дозволяє не тільки зберігати основну інформацію про користувача, але й детально відслідковувати його прогрес у навчанні та керувати доступом в залежності від ролі користувача. Приклад документу з колекції users наведено на рис. 3.33.

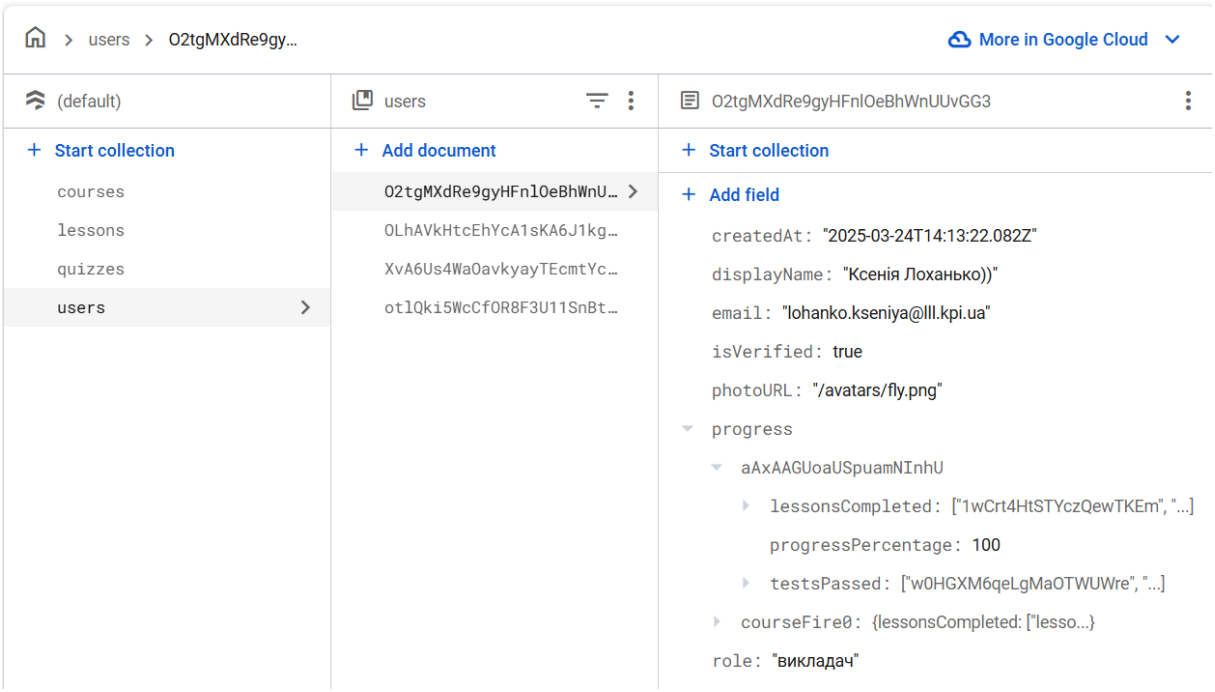


Рисунок 3.33 – Структура колекції users у базі даних Firestore.

Колекція courses в базі даних зберігає документи, які описують різноманітні курси в системі. Кожен документ курсу містить кілька важливих полів, що дозволяють описати його основні характеристики. Поле title визначає назву курсу, яка є основною інформацією для користувачів при пошуку та перегляді курсу. Опис курсу зберігається в полі description, що дає змогу більш детально розповісти про зміст курсу, його мету та загальні характеристики. Для зручності, у полі short_description надається короткий опис курсу, який може бути використаний для попереднього перегляду на сторінках списку курсів або на головних екранах застосунку.

Курс може належати до певної категорії, яка вказується в полі category, що дозволяє сортувати курси за темами, такими як правила безпеки, пожежа, повітряна тривога тощо. Для кожного курсу передбачено поле image_url, яке зберігає посилання на зображення, що ілюструє курс. Це зображення допомагає візуально виділити курс серед інших та робить інтерфейс більш привабливим для користувачів.

Поле lessons містить масив, що включає лише lessonId[] — посилання на уроки, що входять до складу курсу. Це дозволяє курсу містити кілька уроків, при цьому не дублюючи їхню інформацію в кожному курсі, що полегшує управління даними та сприяє нормалізації бази даних. Поле quiz_id зберігає посилання на фінальний тест курсу, який учні мають пройти в кінці навчання для оцінки своїх знань. Також у документі курсу містяться поля createdAt, що вказує на дату створення курсу, та teacherId, яке містить ідентифікатор викладача, що є автором курсу. Приклад документу з колекції courses наведено на рис. 3.34.

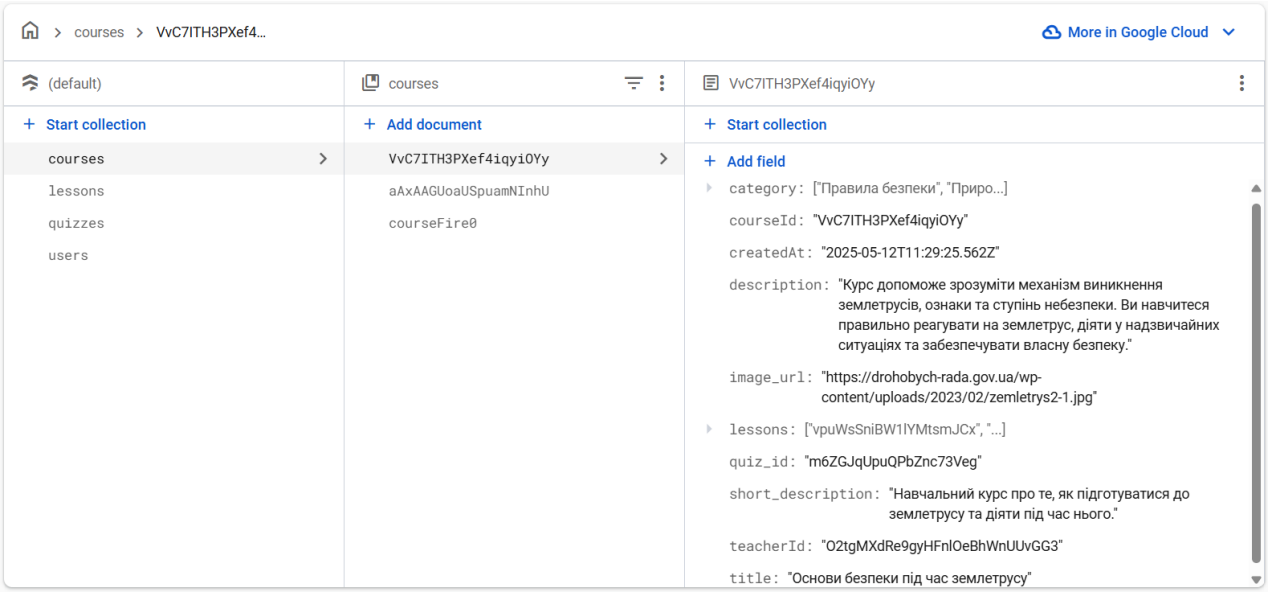


Рисунок 3.34 – Структура колекції courses у базі даних Firestore.

Така організація дозволяє значно підвищити гнучкість і зручність роботи з курсами, оскільки однакові уроки та тести можуть бути використані в кількох курсах, не дублюючи дані, що зменшує обсяг інформації в базі та покращує її структуру.

Колекція lessons у Firestore містить окремі навчальні одиниці, які є складовими певного курсу. Кожен документ у цій колекції представляє собою один урок і має унікальний ідентифікатор lessonId. Уроки є центральними елементами освітнього процесу, оскільки вони містять теоретичний або практичний матеріал, з яким користувач взаємодіє під час проходження курсу. Основне текстове наповнення уроку зберігається в полі content, що підтримує форматований текст і може містити структуровану інформацію, таку як заголовки, списки, підсвічування важливих фрагментів тощо, що підвищує зручність сприйняття навчального матеріалу.

Назва уроку зазначається в полі title, і вона відображається на сторінках курсу або в загальному переліку уроків. Поле courseId є посиланням на той курс, до якого належить даний урок. Це дозволяє будувати зв’язки між курсом та його вмістом і забезпечує структурованість у навігації по навчальному матеріалу. Якщо до уроку прив'язано тест, то його ідентифікатор зберігається

в полі quiz_id. Таким чином, після вивчення матеріалу учень може одразу пройти перевірку знань у вигляді тесту, що покращує інтерактивність та ефективність засвоєння інформації.

Поле createdAt зберігає дату і час створення уроку у форматі рядка ISO, що дозволяє відстежувати хронологію створення контенту. Для впорядкування уроків у межах одного курсу використовується числове поле order, яке задає послідовність відображення уроків. Завдяки цьому навіть при динамічному редагуванні курсу (додаванні чи переміщенні уроків) порядок їх проходження залишається зрозумілим і контрольованим. Крім того, у полі teacherId зберігається ідентифікатор викладача, який створив цей урок, що дозволяє реалізувати авторське право та контроль доступу до редагування.

Загалом, структура колекції lessons дозволяє гнучко керувати освітнім контентом, розділяючи його на логічні модулі, які зручно редагувати, компонувати та перевикористовувати в різних курсах. Приклад документу з колекції lessons можна переглянути на рис. 3.35.

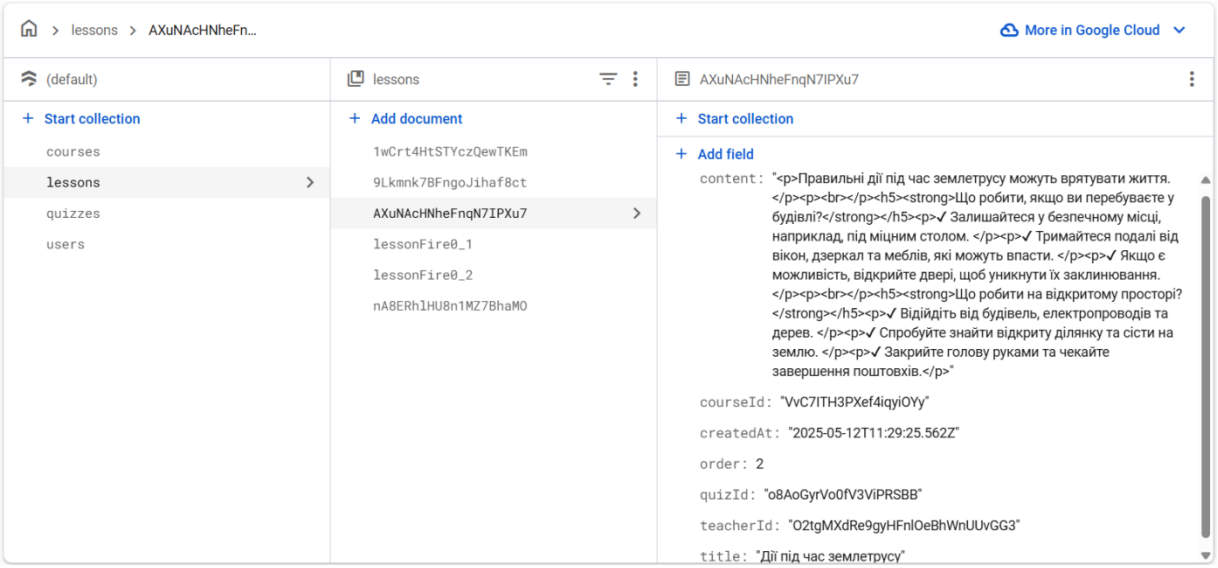


Рисунок 3.35 – Структура колекції lessons у базі даних Firestore.

Колекція quizzes у Firestore зберігає окремі тестові блоки, які можуть бути або частиною конкретного уроку, або виконувати роль фінального тесту всього курсу. Кожен документ у цій колекції репрезентує один тест і має

унікальний ідентифікатор quizId. Основне призначення цієї колекції — забезпечити структуровану перевірку знань учня після проходження теоретичного матеріалу.

Тест має поле title, яке містить його назву та може бути використане для ідентифікації або виведення у списку доступних тестів. Через поле courseId встановлюється зв'язок між тестом і курсом, у межах якого він використовується. Якщо тест є частиною уроку, зв’язок може також здійснюватися опосередковано через quiz_id у документі уроку, що дозволяє одній і тій самій тестовій структурі бути прив’язаною як до курсу в цілому (фінальний тест), так і до окремого уроку (тематичне завдання).

Основна функціональність тесту реалізується через масив questions, у якому кожен елемент — це об'єкт із полями question (текст запитання), options (масив варіантів відповіді) та correctIndex (індекс правильної відповіді в масиві options). Така структура є простою й ефективною для зберігання базових тестових завдань із вибором однієї правильної відповіді, що легко обробляється як на боці клієнта, так і на сервері.

Дата створення тесту зберігається в полі createdAt у вигляді рядка у форматі ISO, що дозволяє відстежувати хронологію створення й оновлення тестів. Поле teacherId визначає автора тесту, тобто користувача з роллю викладача, який створив даний тест. Це забезпечує контроль прав доступу та редагування, дозволяючи змінювати лише власні матеріали.

Завдяки цій структурі тести в системі можуть легко комбінуватись із різними курсами та уроками, забезпечуючи масштабованість та перевикористання контенту без дублювання. Такий підхід відповідає принципам нормалізації бази даних і дає змогу розширювати систему з мінімальними затратами. Приклад документу з колекції quizzes наведено на рис. 3.36.

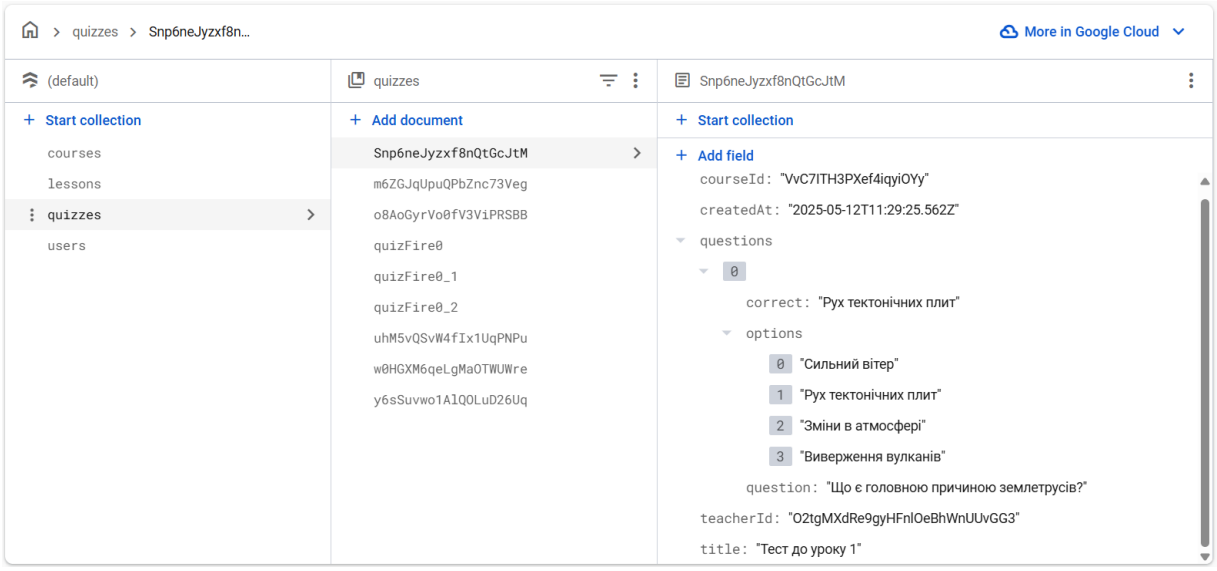


Рисунок 3.36 – Структура колекції quizzes у базі даних Firestore.

3.3.2 CRUD-операції та обробка запитів

Усі CRUD-операції для створення, читання, оновлення та видалення даних в освітньому веб-застосунку реалізовані через окремі сервісні функції, які інкапсулюють прямий доступ до бази даних Firestore. Для кожної основної сутності, такої як курс, урок, тест або користувач, створено відповідний сервіс — coursesService, lessonsService, quizzesService, userService. Ці сервіси є єдиним джерелом взаємодії з Firestore, завдяки чому досягається чітке розділення відповідальностей між логікою даних і компонентами інтерфейсу. Компоненти React не мають прямого доступу до бази даних, а взаємодіють із нею лише через виклик сервісних функцій, що дозволяє централізовано керувати запитами, обробкою помилок та оптимізацією.

Наприклад, одна з ключових функцій — createFullCourse(...) — дозволяє створити курс одночасно з усіма його уроками та фінальним тестом в межах єдиної транзакції або послідовної асинхронної операції (див. рис. 3.32). Це значно спрощує логіку на фронтенді та дозволяє розробнику працювати з комплексною структурою курсу як з єдиним об'єктом. Такий підхід зменшує ризик часткового збереження даних і гарантує, що всі елементи курсу будуть

створені лише в тому разі, якщо успішно створені всі залежні сутності такі як уроки, тести.

Операції читання, такі як getCoursesByTeacher (рис. 3.37), або видалення, наприклад deleteCourse (рис. 3.38), також реалізовані в межах відповідних сервісів і забезпечують не лише базовий доступ до даних, а й додаткову логіку, зокрема перевірку прав доступу (чи є користувач автором курсу), завантаження пов'язаних уроків чи тестів, обробку залежностей тощо.

```
export const getCoursesByTeacher = async (teacherId: string): Promise<Course[]> => {
    const q = query(collection(db, "courses"), where("teacherId", "==", teacherId));
    const snapshot = await getDocs(q);
    return snapshot.docs.map((doc) => ({courseId: doc.id, ...doc.data()} as Course));
};
```

Рисунок 3.37 – Реалізація методу getCoursesByTeacher

```
export const deleteCourse = async (courseId: string) => {
    try {
        // 1. Видаляємо всі уроки з цим courseId
        const lessonsQuery = query(collection(db, "lessons"), where("courseId", "==", courseId));
        const lessonsSnapshot = await getDocs(lessonsQuery);
        const lessonDeletions = lessonsSnapshot.docs.map(docSnap => deleteDoc(doc(db, "lessons", docSnap.id)));
        await Promise.all(lessonDeletions);
        console.log(`Видалено ${lessonDeletions.length} урок(ів), пов'язаних із курсом.`);

        // 2. Видаляємо всі тести з цим courseId
        const quizzesQuery = query(collection(db, "quizzes"), where("courseId", "==", courseId));
        const quizzesSnapshot = await getDocs(quizzesQuery);
        const quizDeletions = quizzesSnapshot.docs.map(docSnap => deleteDoc(doc(db, "quizzes", docSnap.id)));
        await Promise.all(quizDeletions);
        console.log(`Видалено ${quizDeletions.length} тест(ів), пов'язаних із курсом.`);

        // 3. Нарешті видаляємо сам курс
        const courseRef = doc(db, "courses", courseId);
        await deleteDoc(courseRef);
        console.log(`Курс з ID ${courseId} успішно видалено!`);
    } catch (error) {
        console.error("Помилка при видаленні курсу та пов'язаних даних:", error);
    }
};
```

Рисунок 3.38 – Реалізація методу deleteCourse

Всі функції сервісів виконуються асинхронно з використанням синтаксису async/await, що забезпечує чистий та читабельний код. У разі помилок, зокрема відсутності з'єднання, помилок авторизації або відмови

Firestore, реалізовано обробку виключень try/catch, яка дозволяє не тільки вивести користувачу повідомлення про помилку, а й запобігти непередбаченому завершенню роботи додатку. У випадку недоступності сервера або втрати мережі застосунок не зависає, а повертає контроль з відповідним повідомленням, що дозволяє реалізовувати стабільну, fault-tolerant архітектуру.

Завдяки такій структурі CRUD-операцій забезпечується гнучкість і масштабованість системи — нові функції або зміни в логіці доступу до даних можна впроваджувати централізовано, без потреби змінювати логіку в кожному окремому компоненті інтерфейсу.

3.3.3 Авторизація та автентифікація

Система автентифікації (рис. 3.39) в освітньому веб-застосунку реалізована на основі Firebase Authentication — потужного та безпечного сервісу, що дозволяє швидко інтегрувати механізми реєстрації, входу та управління користувачами без необхідності створення власного бекенду. У межах проєкту реалізовано автентифікацію за email і паролем, що є класичним, інтуїтивно зрозумілим і достатньо безпечним способом входу для більшості користувачів, а також вхід за допомогою акаунту Google.

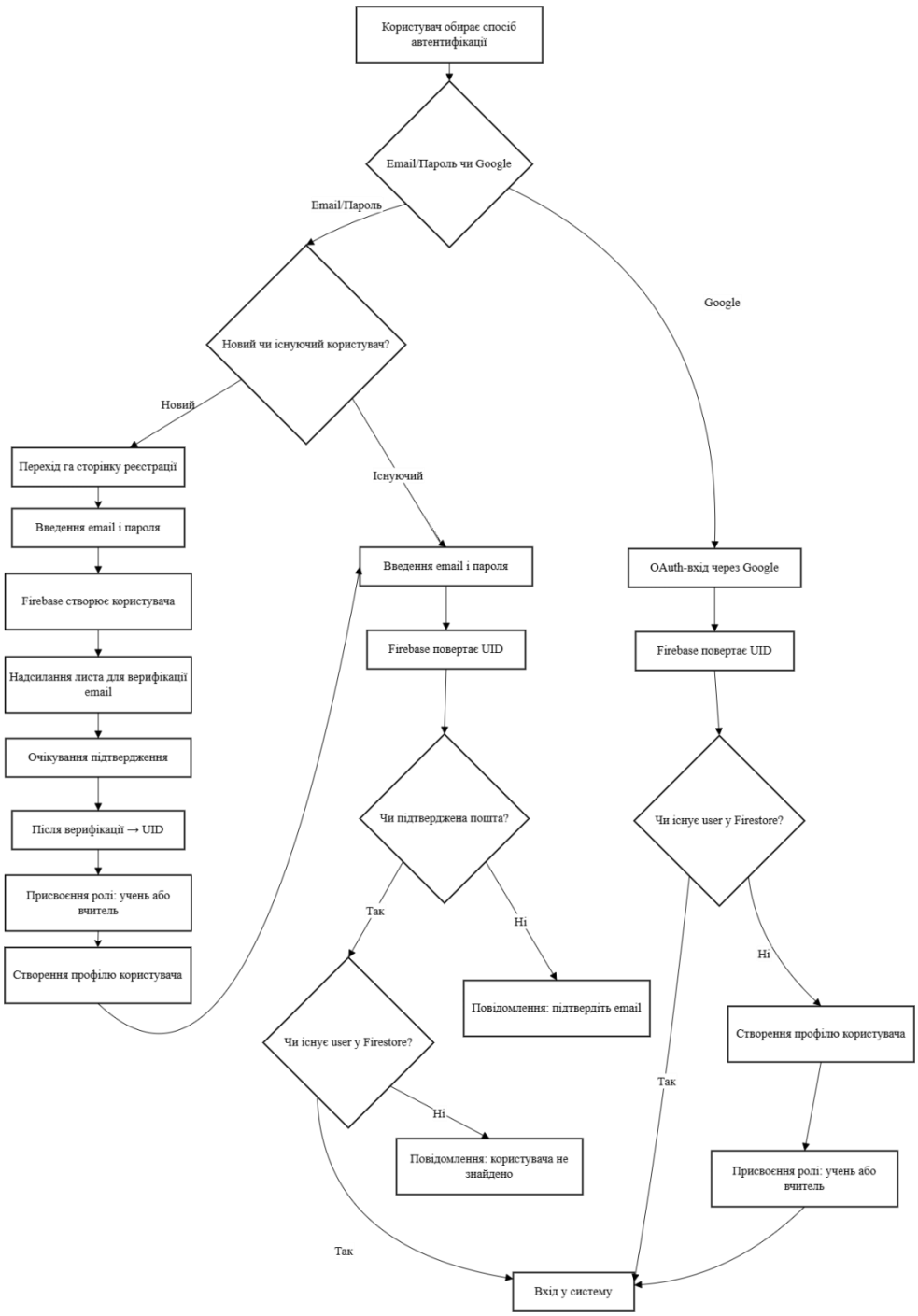


Рисунок 3.39 – Схема автентифікації користувача

Після створення облікового запису через Firebase Authentication користувач автоматично отримує унікальний UID — ідентифікатор, який використовується для подальшого зв’язку з його профілем у Firestore. У Firestore в колекції users створюється окремий документ, де, крім UID, зберігаються такі поля, як email, displayName, а також спеціальне поле role, що

визначає тип користувача — зокрема, викладач або учень. Це дозволяє системі розмежовувати доступ до функціоналу відповідно до ролі.

Авторизація реалізована на клієнтському боці — після входу в систему, застосунок запитує роль користувача з колекції users і, залежно від неї, динамічно адаптує інтерфейс. Схема рольової авторизації наведена на рис. 3.40. Наприклад, якщо роль — вчитель, то в інтерфейсі з’являються можливості для створення та редагування курсів, додавання уроків, налаштування тестів, а також доступ до аналітики чи перевірки прогресу учнів. Для учня ці можливості приховані, та доступний інтерфейс проходження курсів і тестів. Неавторизовані користувачі можуть лише переглядати загальнодоступні курси та відкривати окремі уроки, але їхній прогрес не фіксується, і доступ до проходження тестів обмежений. Такий підхід дозволяє розширити потенційну аудиторію та дає можливість ознайомлення із платформою без обов’язкової реєстрації.

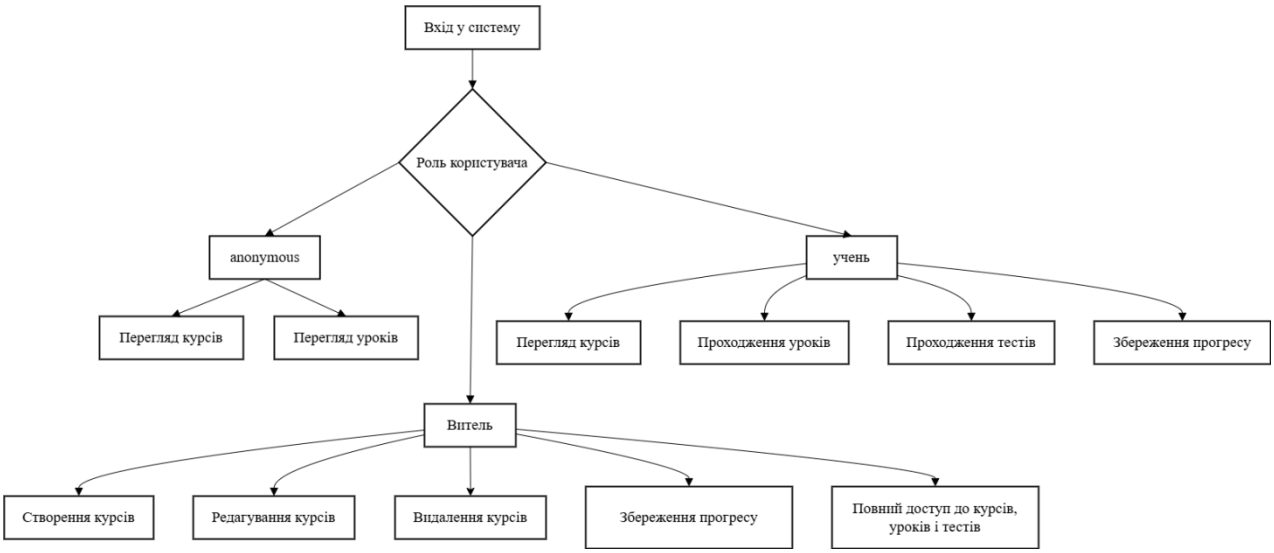


Рисунок 3.40 – Схема рольової авторизації

У випадку оновлення ролі (наприклад, коли адміністратор вручну змінює роль користувача у Firestore), система під час наступного входу автоматично підтягує актуальні права доступу. Таким чином, рольова модель лишається гнучкою і легко підтримується без складної серверної логіки.

Firebase у цьому випадку виконує всі ключові функції бекенду — обробку автентифікації, зберігання інформації про користувачів, підтримку сесій, захист від несанкціонованого доступу. Це дозволило повністю зосередитися на розробці бізнес-логіки та інтерфейсу, значно скоротивши час на створення MVP-версії застосунку. У поєднанні з Firebase Security Rules можна забезпечити надійний контроль доступу навіть без використання окремого Node.js-сервера.

3.4 Розгортання та CI/CD

З метою забезпечення доступності освітнього веб-застосунку для користувачів та автоматизації процесів розгортання і оновлення, було реалізовано інфраструктуру безперервної інтеграції та доставки (CI/CD) [9], засновану на хмарних сервісах Firebase та GitHub Actions. Такий підхід дозволяє забезпечити надійність, швидкість та повторюваність процесу публікації нових версій застосунку, мінімізуючи ручні втручання.

3.4.1 Використання хостинг-платформи

Для розгортання застосунку використовується Firebase Hosting, що надає швидкий і надійний хостинг для фронтенд-застосунків. Ця платформа створена спеціально для SPA і дозволяє забезпечити стабільну та безпечну роботу користувацького інтерфейсу.

Firebase Hosting автоматично налаштовує HTTPS для всіх підключень, що гарантує шифрування трафіку та захист від атак типу “man-in-the-middle”. Крім того, відбувається оптимізація статичних файлів, що прискорює їх завантаження навіть при повільному інтернет-з’єднанні.

На рисунку 3.41 наведено панель керування деплоєм застосунку через Firebase Hosting. На скріншоті зображено поточний активний реліз застосунку, історію попередніх релізів, а також прив’язані доменні імена. Інтерфейс

дозволяє швидко аналізувати статуси розгортань, виконувати відкат (rollback) у разі потреби, і керувати доменами — усе це через зручну веб-панель без необхідності ручної конфігурації.

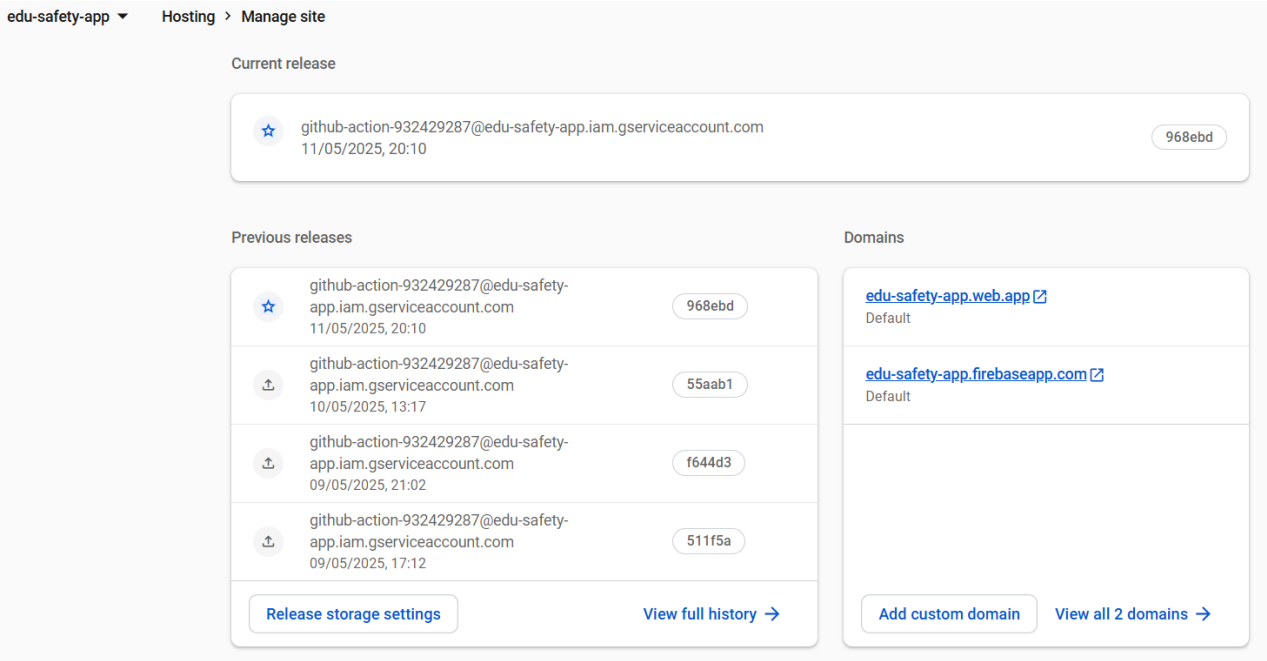


Рисунок 3.41 – керування Firebase Hosting

Розгортання відбувається за допомогою Firebase CLI, яка дозволяє виконувати локальні білди та деплой через одну команду, або інтегрувати ці дії в CI/CD-процес для повної автоматизації.

3.4.2 Налаштування деплою через GitHub Actions

Щоб забезпечити постійну актуальність продуктивного середовища та мінімізувати потребу у ручному втручанні, було впроваджено інфраструктуру безперервної інтеграції та доставки (CI/CD), засновану на використанні GitHub Actions.

CI/CD-процес налаштований таким чином, що при кожному пуші в основну гілку репозиторію (main) GitHub автоматично запускає пайплайн, який виконує повний цикл: від білду до деплою застосунку на Firebase Hosting. Процес складається з кількох етапів: спочатку виконується білд застосунку за допомогою команди npm run build, що створює оптимізовану

продакшн-версію. Далі відбувається автоматичне завантаження зібраних файлів на сервер хостингу Firebase. Це дозволяє негайно вивантажити останню версію SPA-застосунку в хмару, відкриваючи користувачам доступ до оновленого функціоналу без затримок. Таким чином забезпечується безперервне оновлення та стабільність продакшн-середовища. CI/CD-процес деплою застосунку на Firebase Hosting зображено на рис. 3.42.

Рисунок 3.42 – CI/CD-процес деплою застосунку

Для реалізації CI/CD використовуються два окремі workflow-файли: firebase-hosting-merge.yml (рис. 3.43) і firebase-hosting-pull-request.yml (рис. 3.44). Перший з них (firebase-hosting-merge.yml) запускається автоматично

після виконання злиття змін до гілки main. Він виконує встановлення залежностей, білд проєкту, аутентифікацію через Firebase та подальший деплой на Firebase Hosting. Це забезпечує, що лише протестовані й перевірені зміни потрапляють у продуктивне середовище. Другий файл (firebase-hosting-pull-request.yml) спрацьовує при створенні pull request до основної гілки. Він виконує лише білд, без автоматичного деплою, що дозволяє протестувати нові зміни ізольовано, не впливаючи на роботу живого сервісу. Така структура забезпечує гнучкість і надійність процесу розгортання, підтримуючи розробку на стабільній основі. На рис. 3.45 показано виконання автоматичного workflow деплою в GitHub Actions.

```yaml
# This file was auto-generated by the Firebase CLI
# https://github.com/firebase/firebase-tools

name: Deploy to Firebase Hosting on merge
on:
  push:
    branches:
      - main
jobs:
  build_and_deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - run: npm ci && npm run build
      - uses: FirebaseExtended/action-hosting-deploy@v0
        with:
          repoToken: ${{ secrets.GITHUB_TOKEN }}
          firebaseServiceAccount: ${{ secrets.FIREBASE_SERVICE_ACCOUNT_EDU_SAFETY_APP }}
          channelId: live
          projectId: edu-safety-app
```

Рисунок 3.43 – Workflow-файл firebase-hosting-merge.yml

```yaml
# This file was auto-generated by the Firebase CLI
# https://github.com/firebase/firebase-tools

name: Deploy to Firebase Hosting on PR
on: pull_request
permissions:
  checks: write
  contents: read
  pull-requests: write
jobs:
  build_and_preview:
    if: ${{ github.event.pull_request.head.repo.full_name == github.repository }}
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - run: npm ci && npm run build
      - uses: FirebaseExtended/action-hosting-deploy@v0
        with:
          repoToken: ${{ secrets.GITHUB_TOKEN }}
          firebaseServiceAccount: ${{ secrets.FIREBASE_SERVICE_ACCOUNT_EDU_SAFETY_APP }}
          projectId: edu-safety-app
```

Рисунок 3.44 – Workflow-файл firebase-hosting-pull-request.yml

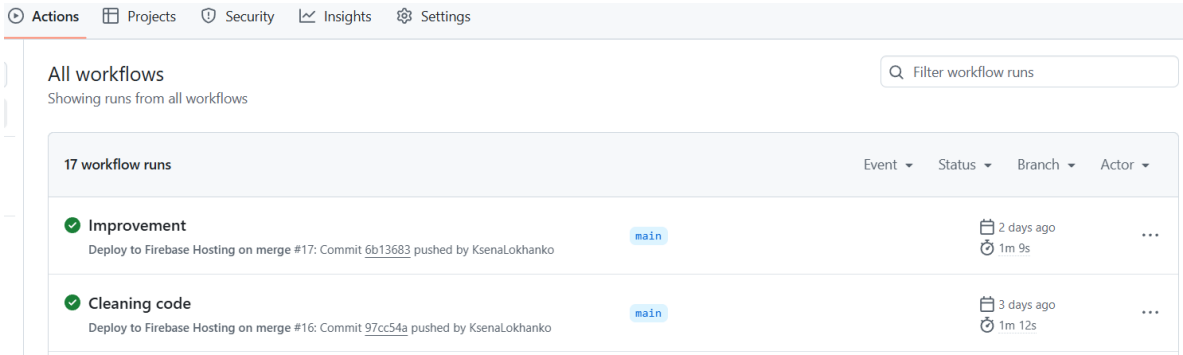


Рисунок 3.45 – Виконання workflow у GitHub Actions

Для безпечної аутентифікації під час автоматичного деплою використовується механізм GitHub Secrets (рис. 3.46). Секретні ключі, зокрема ключ FIREBASE_SERVICE_ACCOUNT_EDU_SAFETY_APP, зберігаються в розділі Settings → Secrets and variables → Actions. Це дозволяє уникнути витоку конфіденційної інформації, оскільки ключі не містяться у відкритому коді репозиторію. Завдяки цьому деплой може виконуватись безпечно з будь-

якого середовища, де запущено GitHub Actions, що відповідає сучасним вимогам безпеки.

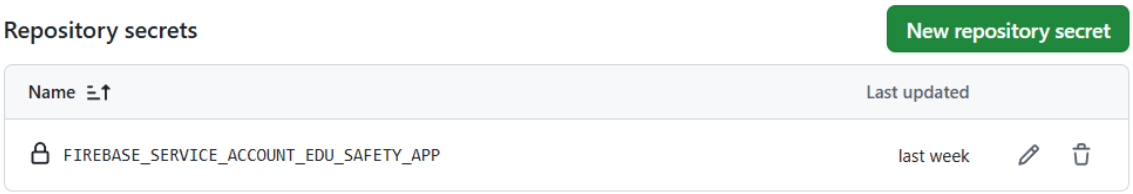


Рисунок 3.46 – GitHub Secrets

3.5 Безпека

З огляду на чутливість зібраної інформації та потребу в довірі користувачів, у процесі розробки освітнього веб-застосунку особливу увагу було приділено реалізації сучасних заходів інформаційної безпеки. Усі основні механізми захисту спрямовані на забезпечення конфіденційності, цілісності та доступності даних.

3.5.1 Захист даних користувача

Для автентифікації та керування сесіями користувачів застосовується Firebase Authentication. Цей сервіс забезпечує захищений облік користувачів з використанням email/пароля або облікових записів Google, а також зберігає токени доступу згідно з галузевими стандартами (OAuth 2.0, OpenID Connect).

Передача даних між клієнтом і сервером відбувається виключно через захищене з'єднання (HTTPS), що гарантує шифрування особистої інформації та недопущення її перехоплення.

Доступ до бази даних Firestore регулюється за допомогою Firestore Security Rules (рис. 3.47). Ці правила визначають, які користувачі мають право читати чи змінювати конкретні документи. Наприклад, звичайні користувачі не мають змоги змінювати чужі навчальні результати або вміст курсів, а викладачі можуть редагувати лише власні навчальні матеріали.

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

    // Отримання ролі з users/{userId}
    function isTeacher() {
      return get(/databases/$(database)/documents/users/$(request.auth.uid)).data.role == "викладач";
    }

    match /users/{userId} {
      allow read, write: if request.auth != null && request.auth.uid == userId;
    }

    match /courses/{courseId} {
      allow read: if true;
      allow create: if request.auth != null && isTeacher();
      allow update, delete: if request.auth != null
        && isTeacher()
        && resource.data.teacherId == request.auth.uid;
    }

    match /lessons/{lessonId} {
      allow read: if true;
      allow create: if request.auth != null && isTeacher();
      allow update, delete: if request.auth != null
        && isTeacher()
        && resource.data.teacherId == request.auth.uid;
    }

    match /quizzes/{quizId} {
      allow read: if true;
      allow create: if request.auth != null && isTeacher();
      allow update, delete: if request.auth != null
        && isTeacher()
        && resource.data.teacherId == request.auth.uid;
    }
  }
}
```

Рисунок 3.47 – Security Rules

3.5.2 Обробка помилок і валідація даних

На стороні клієнта всі форми, пов’язані зі створенням акаунтів, курсів, уроків, тестів або коментарів, мають вбудовану валідацію. Здійснюється перевірка типів, обов’язкових полів, максимальної довжини рядків та допустимих символів.

На серверному рівні через Firebase Rules також реалізовано обмеження — наприклад, заборона запису в поля, що не передбачені схемою, або створення документів без відповідних прав доступу.

Помилки обробляються на клієнті за допомогою try/catch-блоків, що дозволяє інформувати користувача про проблеми (наприклад, відсутність

інтернет-з'єднання або відмову в доступі) без розкриття внутрішньої логіки застосунку.

3.5.3 Запобігання найпоширенішим вразливостям

Захист від CSRF

CSRF-атаки полягають у виконанні запитів від імені автентифікованого користувача без його згоди.

У проєкті захист від CSRF забезпечується на архітектурному рівні завдяки особливостям Firebase Authentication:

- Токени доступу зберігаються не у cookies, а у локальній пам’яті браузера, тому вони не прикріплюються автоматично до запитів.
- Оскільки токени не зберігаються у cookie, зловмисник не може автоматично згенерувати валідний запит із токеном, що асоціюється з активною сесією користувача.
- Усі запити, що змінюють стан даних, виконуються лише через SDK Firebase, який додає перевірений токен авторизації:
  Authorization: Bearer <Firebase_ID_Token>

Додаткові заходи безпеки Firebase Hosting:

- Content-Security-Policy — обмежує джерела скриптів, стилів та медіа, що можуть бути завантажені, і зменшує ризик XSS.
- X-Content-Type-Options: nosniff — запобігає MIME-sniffing-атакам, які дозволяють обійти типізацію контенту..
- X-Frame-Options: DENY — забороняє вбудовування сторінки у фрейми інших сайтів, захищаючи від clickjacking.
- Strict-Transport-Security — вимагає завжди використовувати HTTPS, навіть при прямому введенні адреси з http://.

У сукупності ці заходи забезпечують високий рівень захисту освітньої платформи від найбільш поширених вразливостей веб-застосунків

ВИСНОВОК ДО РОЗДІЛУ 3

Розроблений односторінковий веб-застосунок для онлайн-освіти, що використовує React, Firebase та інші сучасні технології, забезпечує гнучкість, ефективність і масштабованість. Застосування React дозволяє створювати інтерактивні та швидкі інтерфейси, у той час як Firebase виступає потужним інструментом для автентифікації користувачів, зберігання даних та хостингу, що значно спрощує інфраструктуру проєкту.

Модульна структура дозволяє легко масштабувати та оновлювати застосунок без втрати продуктивності. Користувачі мають змогу зареєструватися, проходити курси, виконувати тести та отримувати зворотний зв'язок про прогрес. Викладачі, в свою чергу, можуть створювати нові курси, додавати уроки та тести, забезпечуючи високий рівень кастомізації контенту.

Застосування Vite для збору проекту та GitHub Actions для автоматизації деплою забезпечують швидку і ефективну роботу з кодом та його оновленнями. Завдяки Firebase Hosting забезпечується надійний хостинг для швидкої доставки контенту кінцевому користувачу.

Реалізація прогресу користувача та інтеграція з Firestore дозволяє відслідковувати успіхи учнів у реальному часі та забезпечує персоналізований підхід до навчання. Вся система побудована таким чином, щоб забезпечити високу зручність і доступність як для кінцевих користувачів, так і для адміністраторів контенту.

У результаті було створено функціональний і зручний освітній веб-застосунок, що забезпечує всі необхідні інструменти для навчання, тестування та управління курсами, що відповідає сучасним вимогам до веб-розробки та онлайн-освіти.

РОЗДІЛ 4.
ПРЕДСТАВЛЕННЯ РОБОТИ СИСТЕМИ

У цьому розділі наведено покрокову інструкцію взаємодії користувача з веб-застосунком для навчання правилам поведінки під час загроз. Застосунок підтримує три ролі: гість, зареєстрований користувач у ролі учень або викладач, кожна з яких має свій сценарій взаємодії. Інтерфейс інтуїтивно зрозумілий, адаптований до будь-яких пристроїв та побудований за принципом поступового розширення функцій. Алгоритм дій програмного забезпечення, що ілюструє ці сценарії, наведено у Додатку 3.

4.1 Головна сторінка: огляд і доступність

На кожній сторінці веб-застосунку відображається хедер і футер, які забезпечують зручну навігацію та впізнаваність інтерфейсу. Хедер містить логотип, навігаційні посилання (залежно від ролі користувача), а також кнопку входу або виходу з облікового запису. У випадку неавторизованого користувача, хедер містить мінімальний набір елементів: логотип застосунку, кнопки «Головна», «Курси» та «Увійти». Приклад хедера наведено на рис. 4.1.

Рисунок 4.1 – Хедер неавторизованого користувача

Футер (рис. 4.2) розташовується в нижній частині сторінки та містить стандартне повідомлення про авторські права: © 2025 Освітній веб-застосунок. Всі права захищені. Цей елемент виконує роль завершального естетичного блоку сторінки та вказує на право власності на контент платформи.

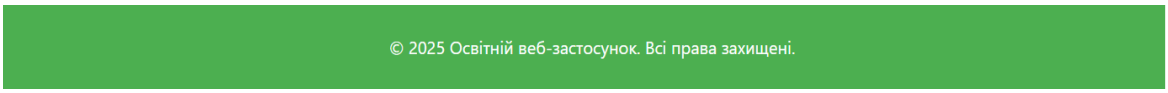


Рисунок 4.2 – Футер веб-застосунку

Після відкриття веб-застосунку автоматично завантажується головна сторінка, яка складається з трьох основних блоків. Перший блок є спільним для всіх користувачів. У ньому розміщено вітальний текст, кнопку «Почати навчання», яка веде до списку всіх доступних курсів, а також кнопку «Увійти» — для неавторизованих користувачів (рис. 4.3). Другий блок відображається лише для гостей і містить короткий опис можливих ролей у системі: учень або викладач, які також мають посилання на сторінку входу (рис. 4.4). Третій блок доступний усім користувачам і містить опис переваг освітнього веб-застосунку: безкоштовний доступ, інтерактивність та можливості для вчителів (рис. 4.5).

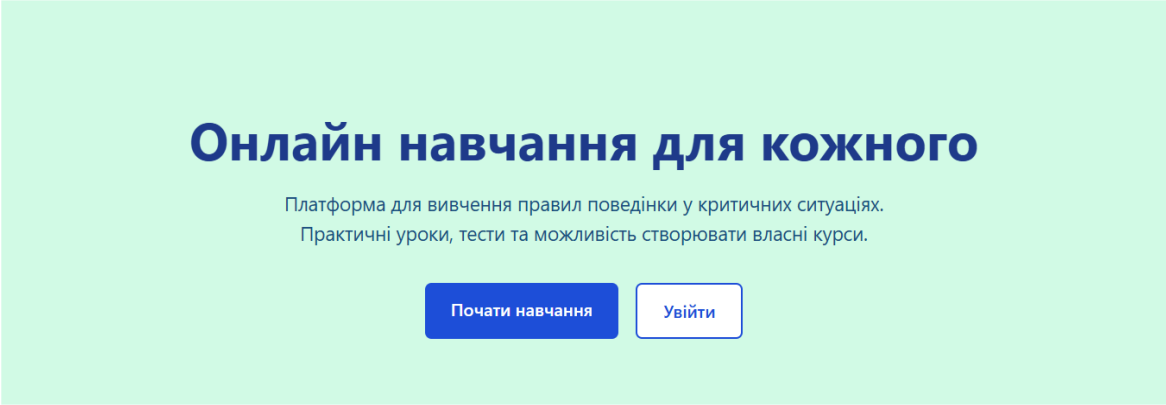


Рисунок 4.3 – Hero-секція (версія для неавторизованих користувачів)

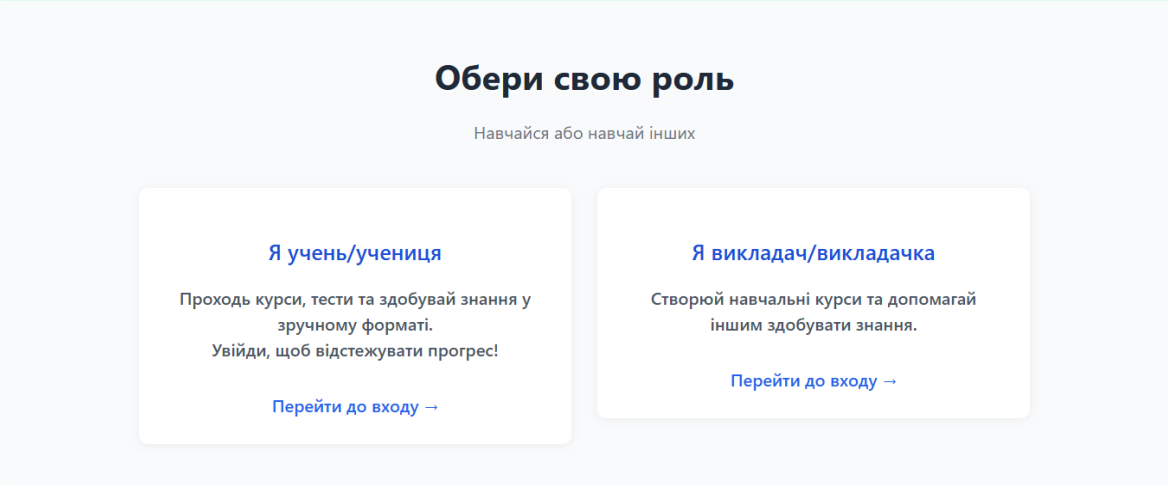


Рисунок 4.4 – Вибір ролі (для неавторизованих користувачів)

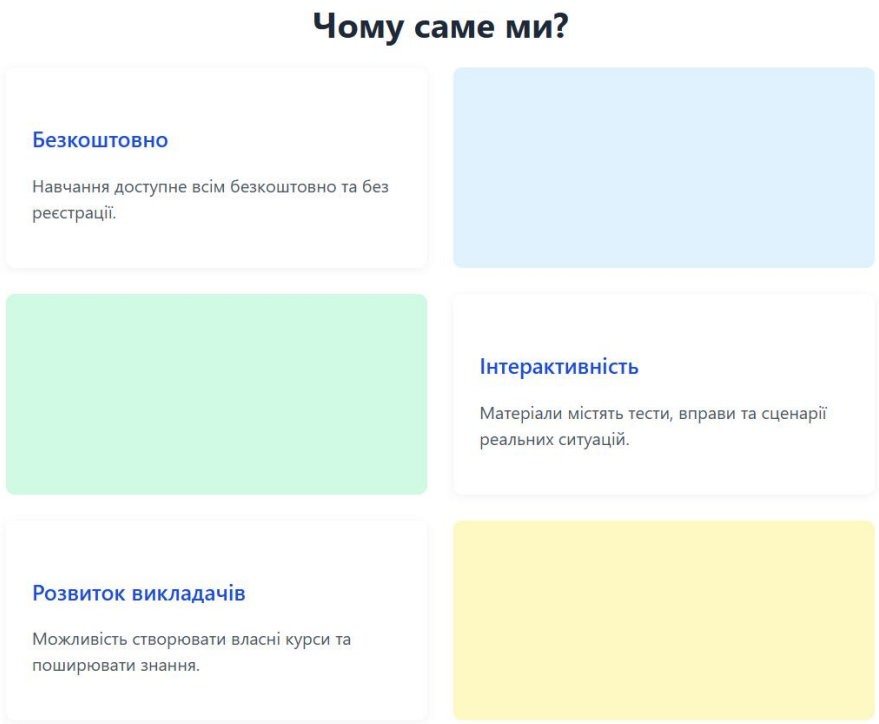


Рисунок 4.5 – Переваги платформи

Якщо користувач не авторизований, він має змогу переглядати список курсів, відкривати окремі уроки та проходити тести у демонстраційному режимі. У цьому випадку результати не зберігаються, а прогрес не фіксується. Щоб користуватися повним функціоналом, зокрема збереженням результатів і персоналізованим прогресом, користувач має авторизуватися — використовуючи email або Google-акаунт. Відповідна сторінка відкривається після натискання кнопки «Увійти» у хедері або на головній сторінці.

4.2 Авторизація та створення облікового запису

Сторінка входу містить просту й зрозумілу форму (рис. 4.6). Користувач може авторизуватися за допомогою електронної пошти — для цього необхідно ввести адресу email і пароль — або скористатися альтернативним способом «Увійти через Google». Цей варіант дозволяє швидко отримати доступ до платформи без потреби створювати окремий пароль.

Під формою розміщено посилання «Ще немає акаунту? Зареєструватися», яке веде на сторінку створення облікового запису. Це

забезпечує зручний перехід для нових користувачів, які вперше знайомляться із застосунком.

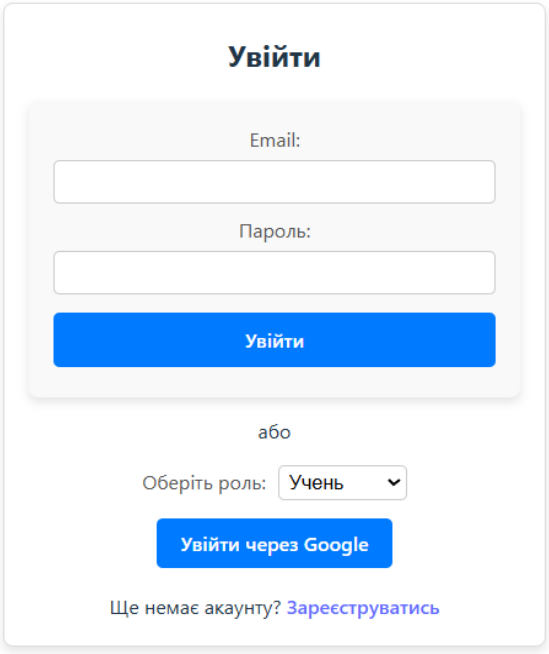


Рисунок 4.6 – Сторінка входу

У разі введення некоректних даних або якщо обліковий запис із такими даними не існує, система виводить повідомлення: «Помилка авторизації. Перевірте введені дані» (рис. 4.7).

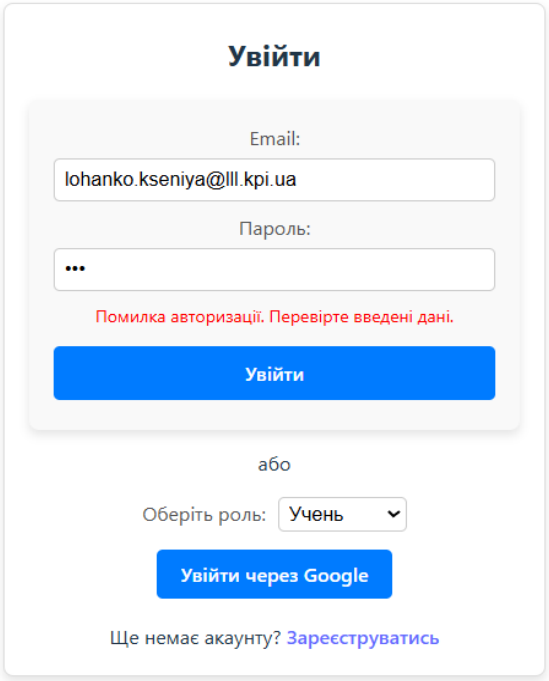


Рисунок 4.7 – Помилка авторизації

Для входу за допомогою Google користувач повинен обрати відповідну роль (учень або викладач) перед натисканням кнопки «Увійти через Google» (рис. 4.8). При першому вході вибрана роль зберігається, тому надалі обирати її повторно не потрібно.

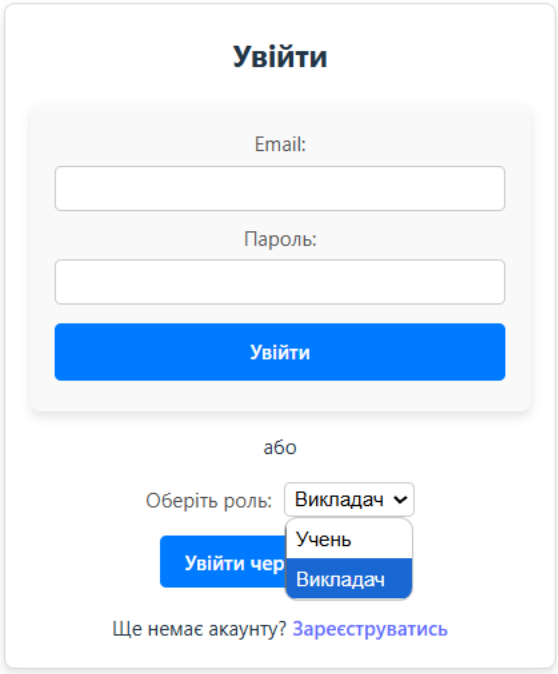


Рисунок 4.8 – Вибір ролі перед входом через Google

Щоб створити новий обліковий запис, потрібно натиснути на посилання «Зареєструватися», після чого відкривається сторінка реєстрації (рис. 4.9).

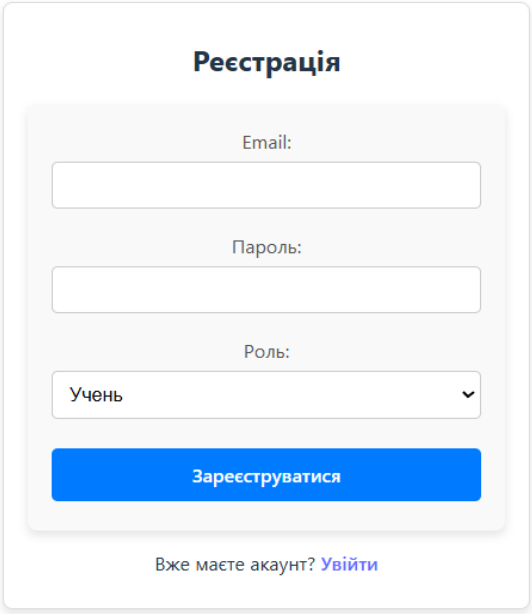


Рисунок 4.9 – Сторінка реєстрації

Якщо користувач намагається зареєструватися з email-адресою, яка вже використовується в системі, відображається відповідне повідомлення про помилку (рис. 4.10).

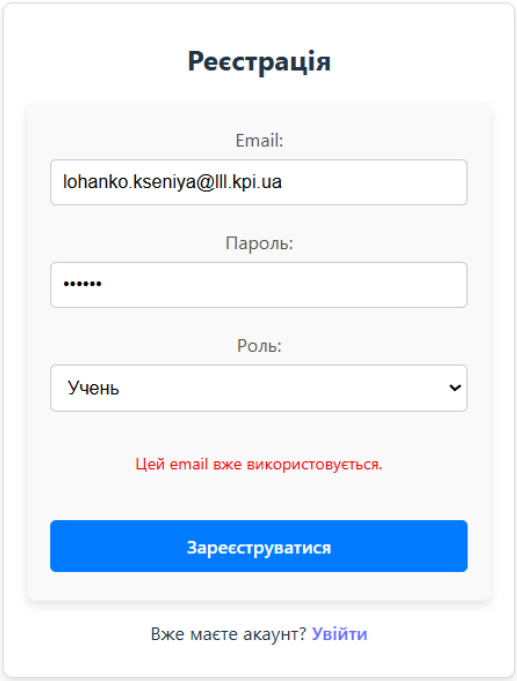


Рисунок 4.10 – Помилка: ця електронна адреса вже зареєстрована

Також система перевіряє довжину пароля. Якщо пароль надто короткий, з'являється повідомлення (рис. 4.11).

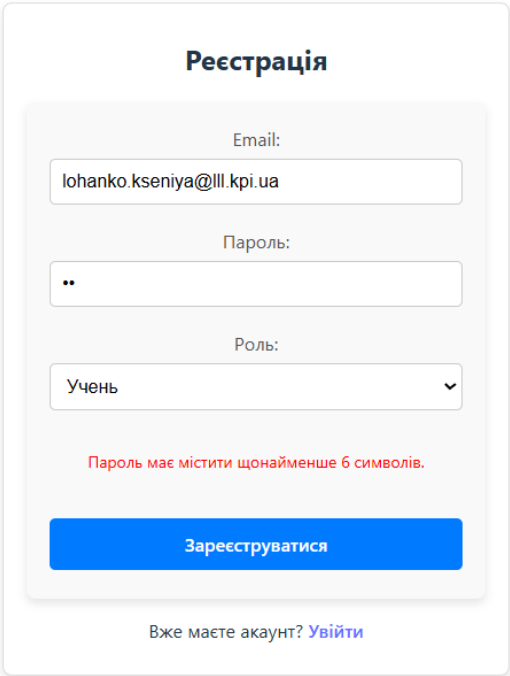


Рисунок 4.11 – Помилка: недостатня довжина пароля

Після успішної реєстрації на вказану адресу електронної пошти надсилається лист із посиланням для підтвердження (рис. 4.12).

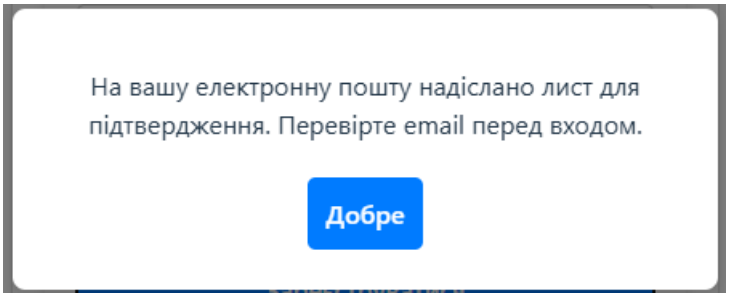


Рисунок 4.12 – Повідомлення про надходження листа но пошту

Якщо користувач намагається увійти до системи без підтвердження email, з'являється відповідне повідомлення про помилку (рис. 4.13).

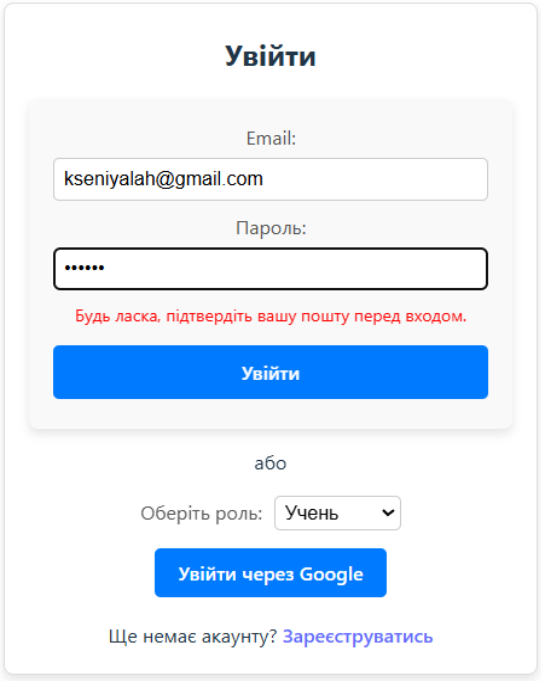


Рисунок 4.13 – Помилка підтвердження пошти

При відкритті листа користувач бачить посилання для підтвердження (рис. 4.14), після натискання на яке система відображає повідомлення про успішне підтвердження (рис. 4.15).

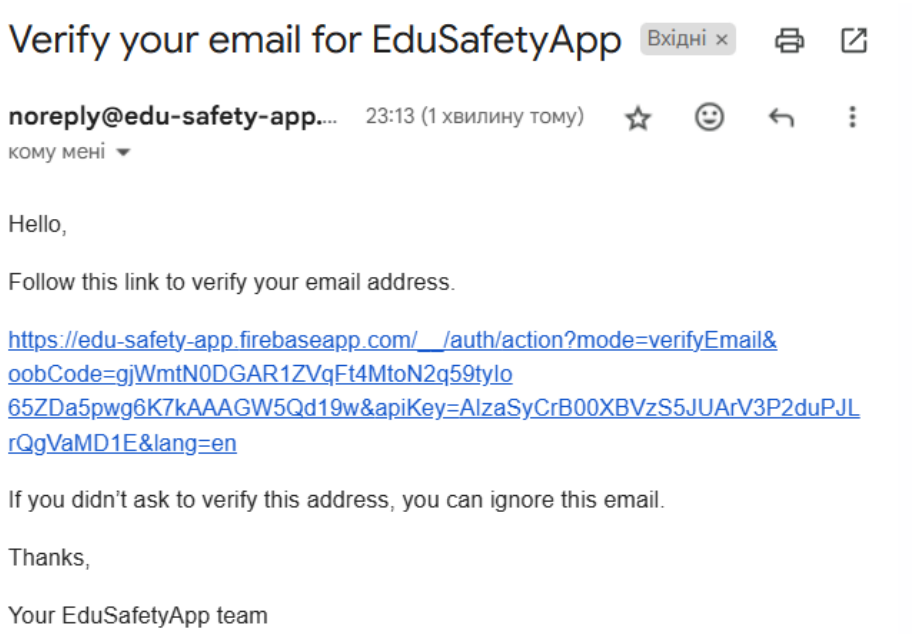


Рисунок 4.14 – Лист із підтвердженням електронної адреси

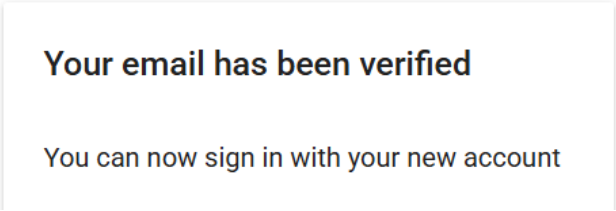


Рисунок 4.15 – Повідомлення про успішне підтвердження email

Після успішного входу користувач автоматично перенаправляється на головну сторінку, яка адаптована до його ролі (учня або викладача), а також дозволяє повноцінно взаємодіяти з курсами, уроками й тестами.

Залежно від ролі користувача змінюється і навігаційний хедер:

- для учня відображаються розділи «Головна», «Курси» та «Кабінет» та кнопка виходу (рис. 4.16);
- для викладача додатково доступна кнопка «Мої курси», що веде до панелі створення та редагування навчальних матеріалів (рис. 4.17).

Таке розмежування дозволяє створити інтерфейс, який чітко відображає доступні можливості кожного типу користувача.

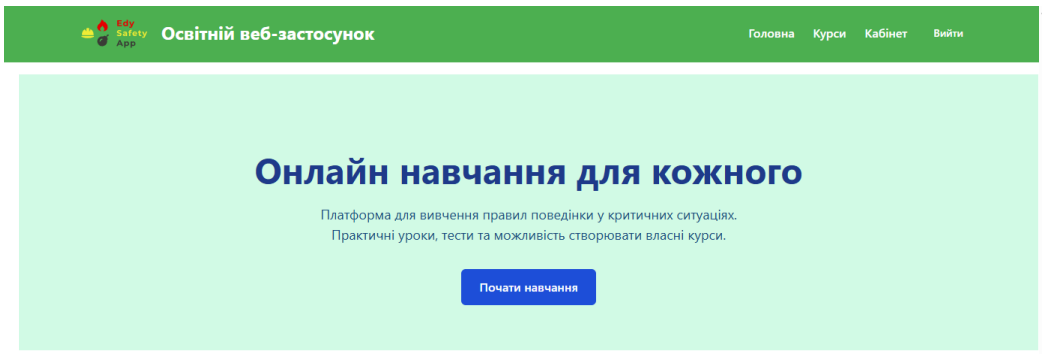


Рисунок 4.16 – Хедер та Hero-секція учня

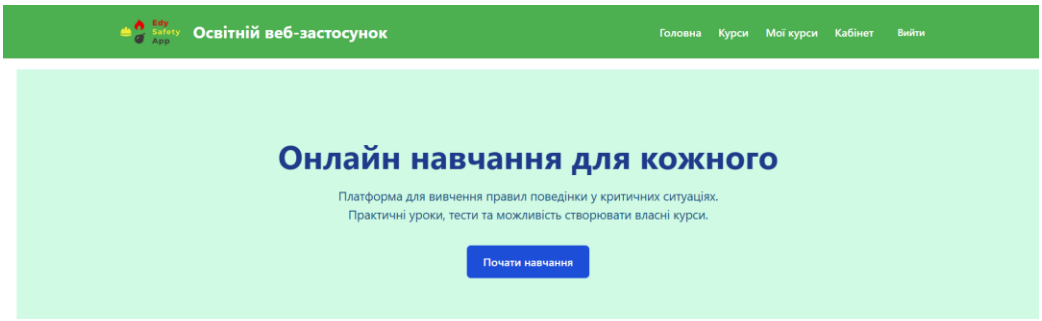


Рисунок 4.17 – Хедер та Hero-секція викладача

4.3 Сторінка кабінету користувача

Сторінка кабінету користувача є спільною як для учнів, так і для викладачів. Вона складається з двох основних модулів: «Ваш профіль» та «Мої курси».

У блоці «Ваш профіль» відображається аватар користувача, ім’я (наприклад, Ксенія Лоханько), електронна пошта (наприклад, lohanko.kseniya@lll.kpi.ua), а також статус — роль у системі: учень або викладач (рис. 4.18). Поруч розміщено кнопку «Редагувати профіль», натискання на яку відкриває режим редагування (рис. 4.19). У цьому режимі користувач може змінити своє ім’я та обрати аватар із наявної галереї зображень. Зміни зберігаються після підтвердження та автоматично оновлюються на сторінці.

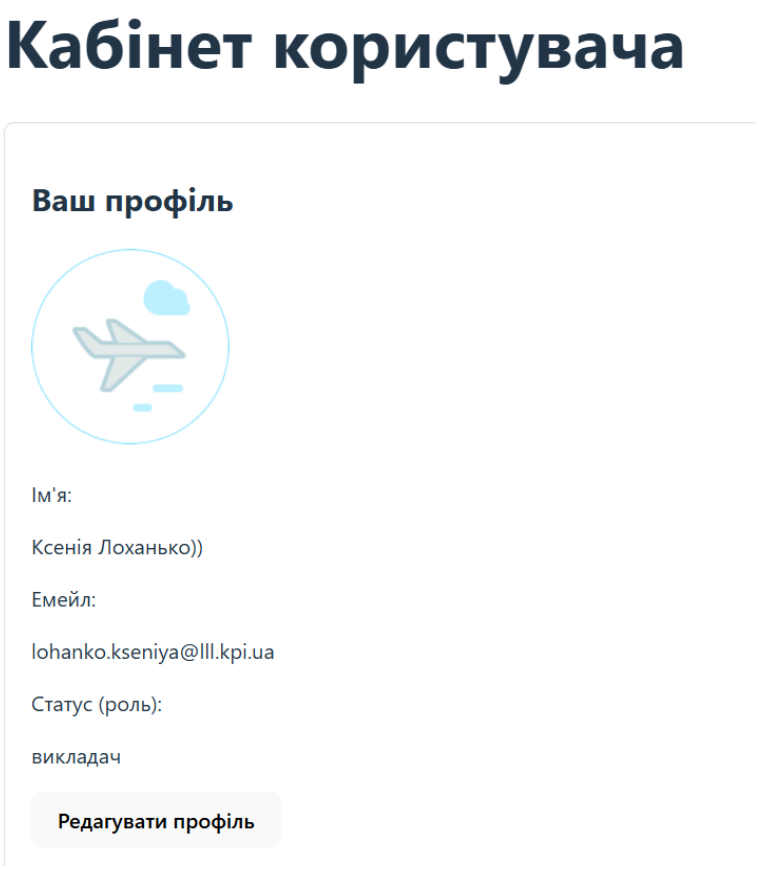


Рисунок 4.18 – Профіль користувача

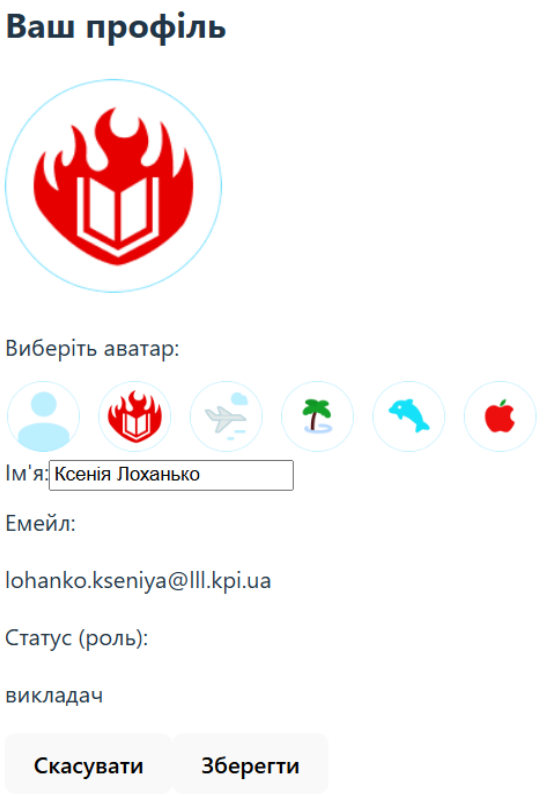


Рисунок 4.19 – Режим редагування профілю

Другий модуль — «Мої курси» — показує навчальний прогрес користувача (рис. 4.20). Для кожного курсу, який ще не завершено, вказується відсоток проходження. Для завершених курсів відображається позначка «Завершено». Це дозволяє користувачу легко орієнтуватися у своєму навчальному процесі та повертатися до будь-якого курсу за потреби.

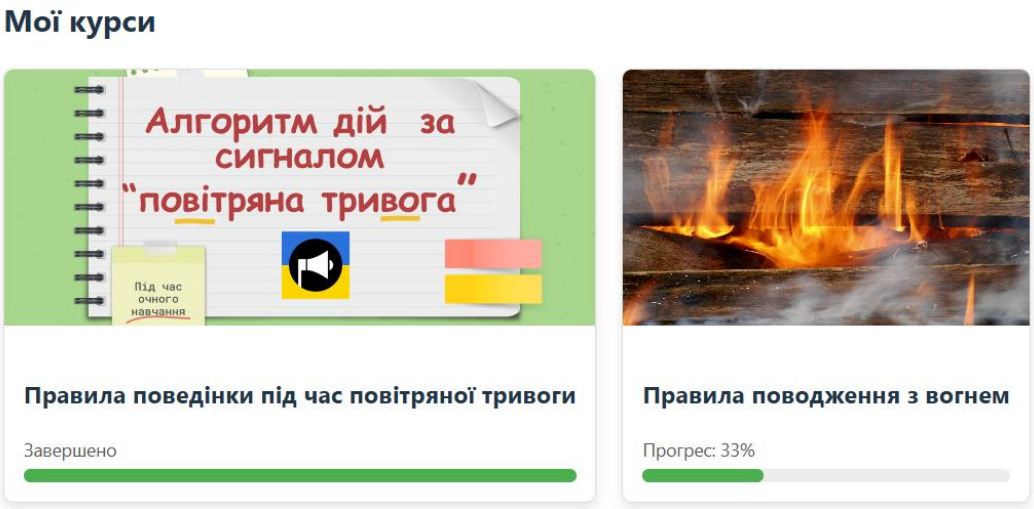


Рисунок 4.20 – Відображення прогресу користувача

У нижній частині сторінки розміщена кнопка «Видалити акаунт» (рис. 4.21). Після її натискання з’являється спливаюче вікно з попередженням про безповоротне видалення облікового запису та всіх даних (рис. 4.22). Якщо користувач підтверджує дію, акаунт видаляється, а користувач автоматично перенаправляється на головну сторінку.

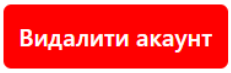


Рисунок 4.21 – кнопка «Видалити акаунт»

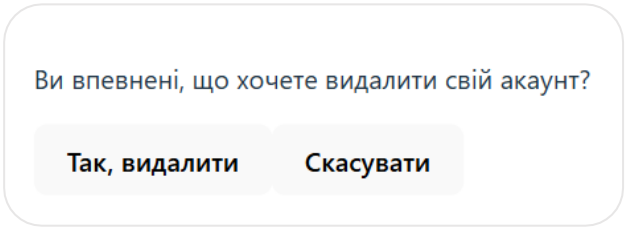


Рисунок 4.22 – Вікно підтвердження видалення аканту

4.4 Сторінка курсів

Сторінка з каталогом курсів (рис. 4.23) є центральним розділом платформи, де користувачі можуть ознайомитися з усіма доступними навчальними програмами. Курси відображаються у вигляді окремих карток, кожна з яких містить назву курсу, короткий опис, категорію та індикатор прогресу.

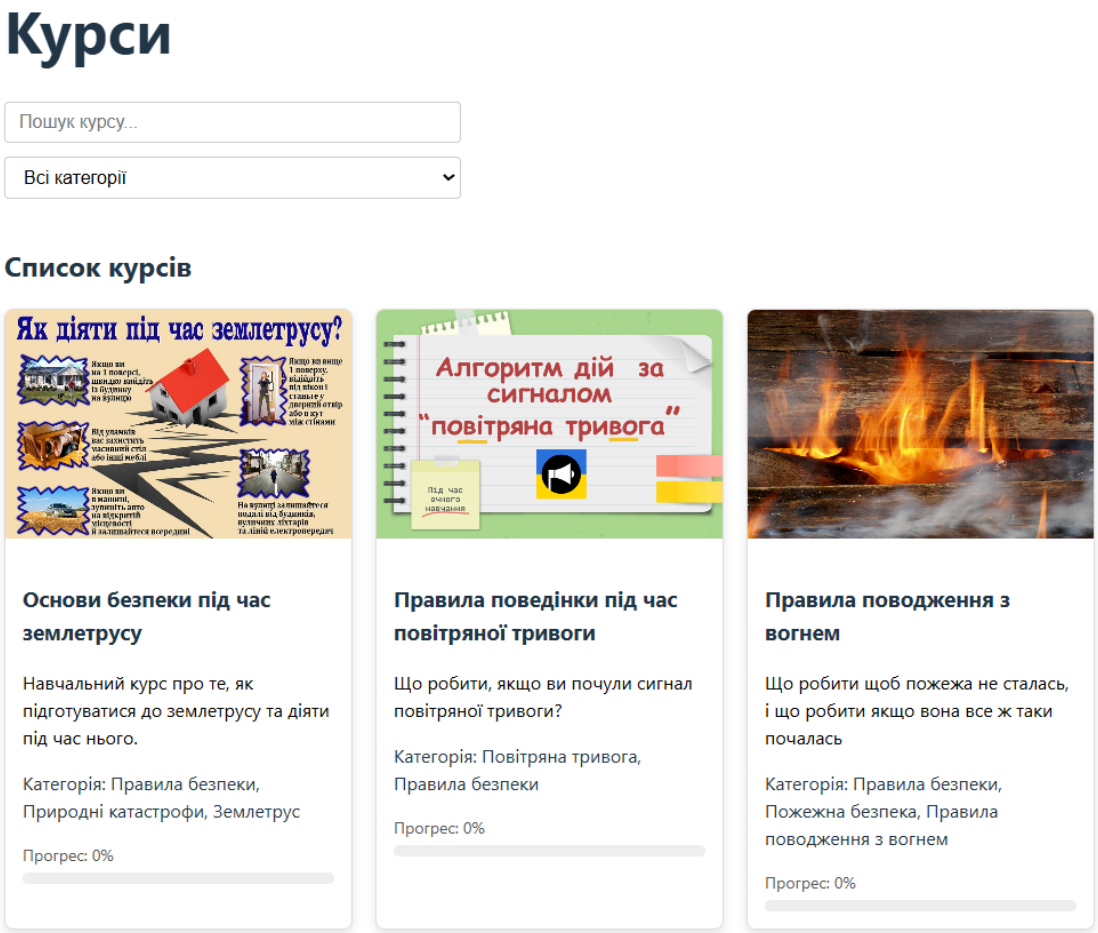


Рисунок 4.23 – Загальний вигляд сторінки з каталогом курсів

Для зручності реалізовано пошук за назвою курсу (рис. 4.24): достатньо ввести ключове слово у відповідне поле, щоб миттєво отримати релевантні результати. Крім того, доступне фільтрування за категорією (рис. 4.25) за допомогою випадаючого списку (рис. 4.26). Обидва варіанти — пошук і фільтрація — можуть використовуватися разом, що дозволяє швидко знайти потрібний курс навіть за великої кількості навчальних матеріалів.

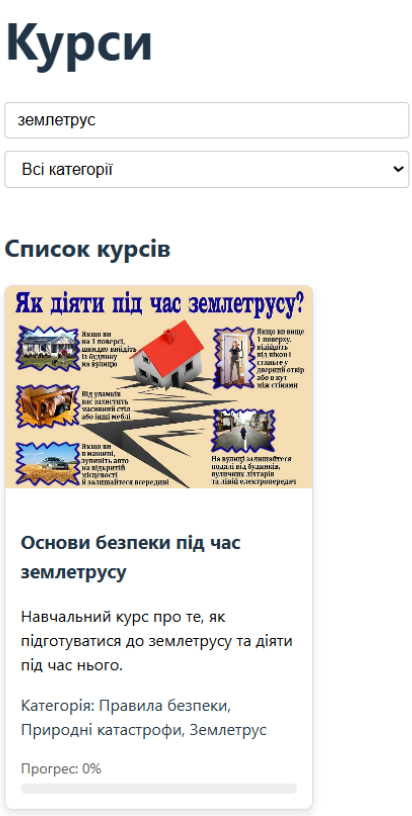


Рисунок 4.24 – Пошук курсів за ключовим словом

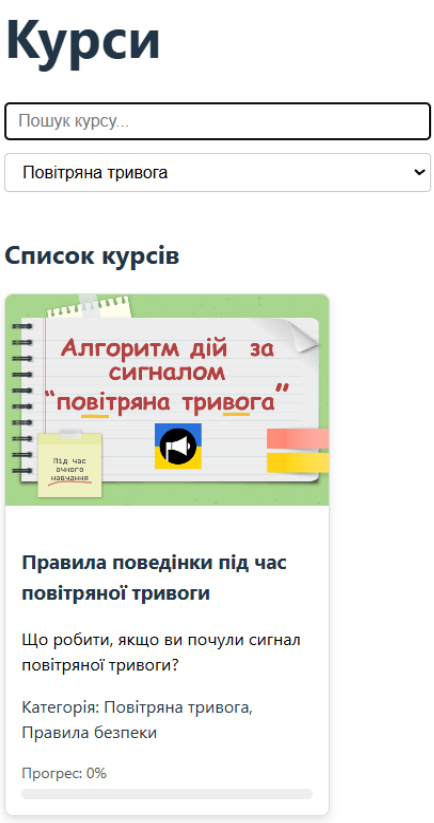


Рисунок 4.25 – Фільтрація курсів за категорією

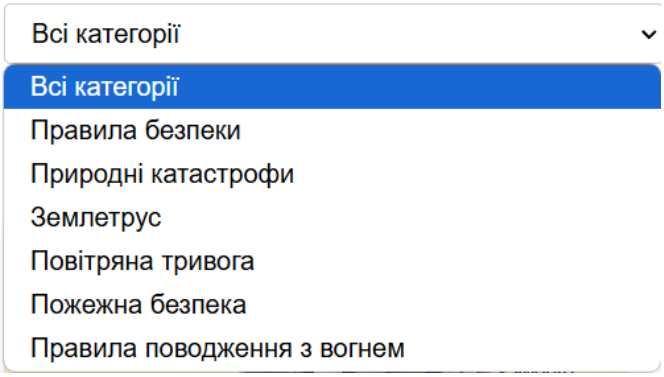


Рисунок 4.26 – Випадаючий список категорій

При натисканні на картку курсу відкривається його детальна сторінка, де користувач може переглянути структуру курсу, список уроків, наявні тести, опис і свій поточний прогрес (якщо він авторизований). Такий підхід забезпечує просту й інтуїтивно зрозумілу навігацію, сприяючи зручному керуванню навчальним процесом.

4.5 Сторінка курсу, уроку та тесту

Після переходу з каталогу, користувач потрапляє на сторінку окремого курсу (рис. 4.27). Тут відображається повна інформація про курс: його назва, опис, категорія, список уроків і наявність фінального тесту. Уроки відображаються у вигляді вертикального списку з кнопками для переходу, а також видно індикатор прогресу. Користувач має можливість розпочати навчання з будь-якого уроку, але при проходженні тестів рекомендується дотримуватись послідовності, оскільки доступ до фінального тесту відкривається лише після завершення всіх уроків. На сторінці курсу розташована кнопка «Почати/Продовжити курс», яка змінюється на «Всі завдання завершено» після проходження всіх уроків і тестів.

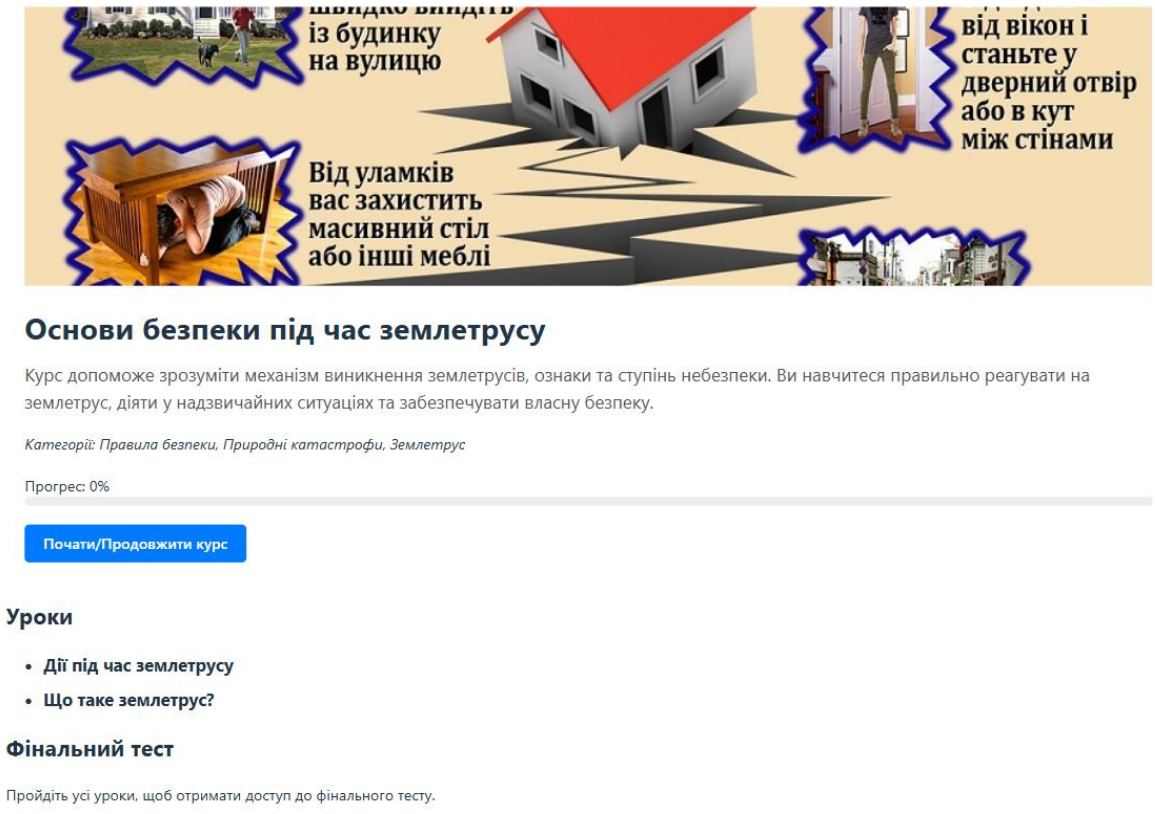


Рисунок 4.27 – Сторінка окремого курсу з описом та списком уроків

При натисканні на урок відкривається сторінка самого уроку. У верхній частині сторінки розташована кнопка «Назад до курсу», яка дозволяє швидко повернутися до загального огляду курсу. Основна частина сторінки містить контент уроку — текстовий матеріал, який може включати заголовки, списки, посилання, виділення, а також зображення чи відео (рис. 4.28). Така структура забезпечується завдяки використанню Markdown-редактора під час створення курсу. Після ознайомлення з матеріалом користувач може перейти до наступного уроку або тестування якшо воно передбачене в уроці.

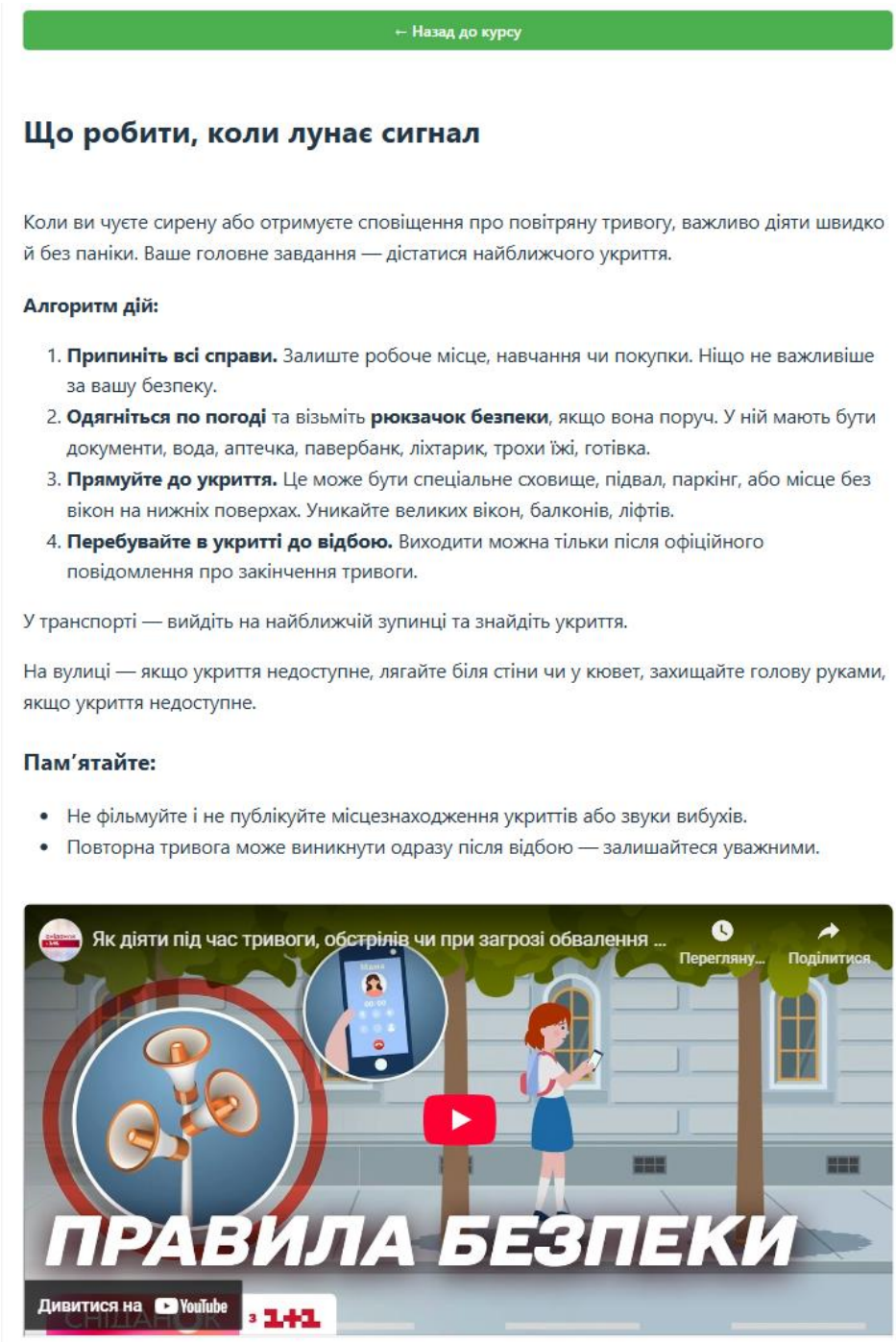


Рисунок 4.28 – Контент уроку

Якщо урок супроводжується тестом, то блок тестування знаходиться під навчальним контентом (рис. 4.29). Тест складається із запитань з одним правильним варіантом відповіді. Користувач обирає відповіді на всі запитання й натискає кнопку «Підтвердити». Після цього система автоматично перевіряє відповіді та відображає результати — правильні відповіді позначаються

зеленим кольором, а неправильні — червоним (рис. 4.30). Також з’являється кнопка «Скинути», яка дозволяє пройти тест повторно. Якщо тест виконано успішно, активується кнопка «Завершити урок», яка дає змогу відзначити урок як пройдений і перейти до наступного етапу навчання (рис. 4.31).

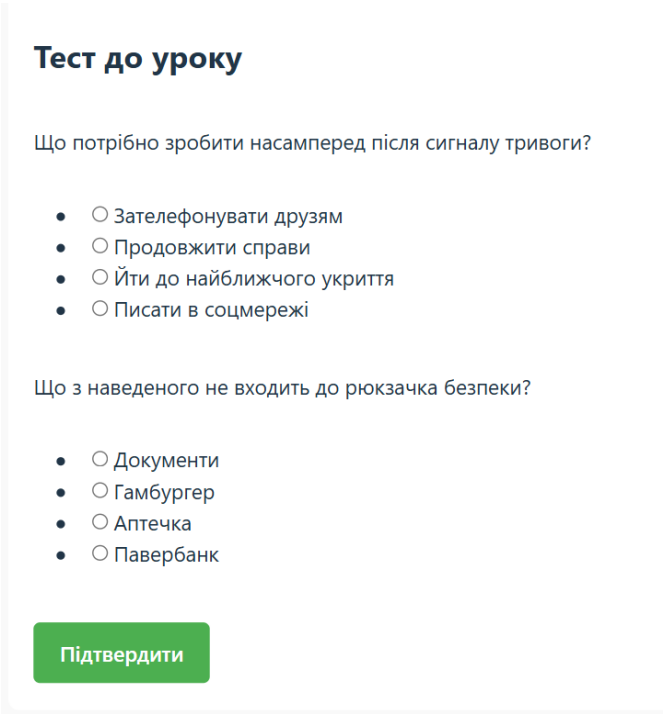


Рисунок 4.29 – Тест після уроку

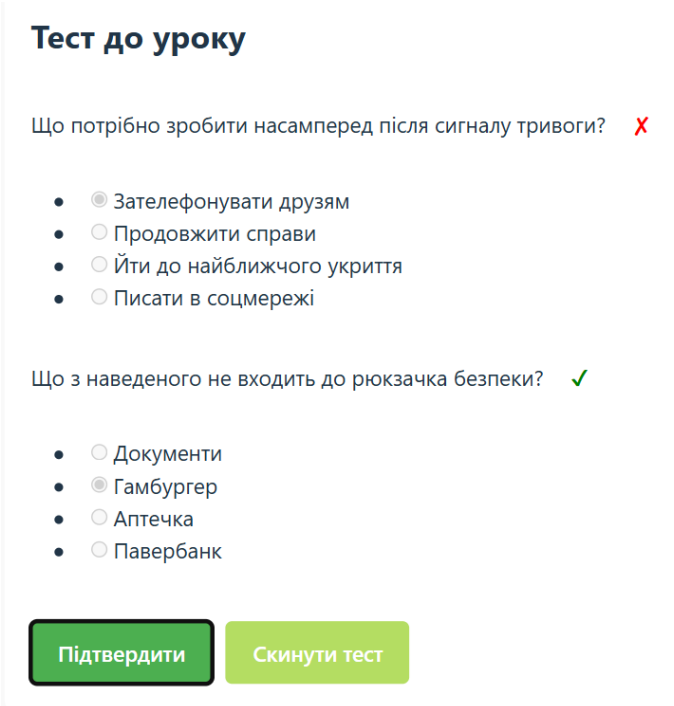


Рисунок 4.30 – Відображення правильних і не правильних відповідей

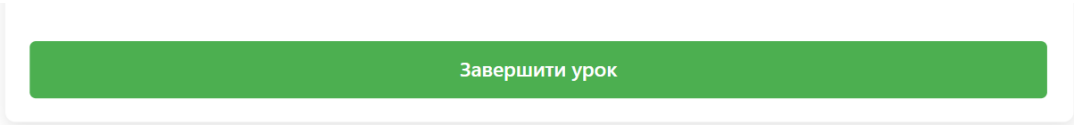


Рисунок 4.31 – Кнопка завершити урок після успішного проходження уроку

Прогрес користувача фіксується в базі даних автоматично. Якщо тест пройдено успішно, це впливає на загальний прогрес курсу — відповідна інформація оновлюється й відображається на сторінці курсу у вигляді індикатора, а пройдені уроки позначаються зеленим (рис. 4.32).

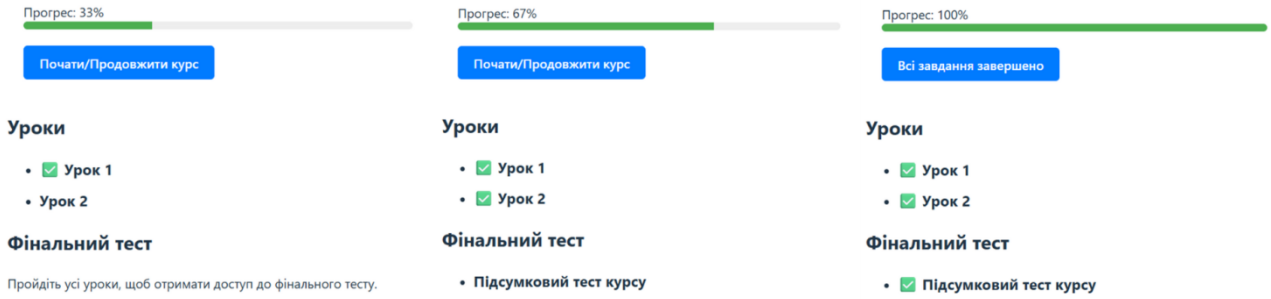


Рисунок 4.32 – відображення прогресу на різних етапах завершеності курсу

Після завершення всіх уроків користувач отримує доступ до фінального тесту (рис. 4.33). Структура фінального тесту аналогічна проміжним: він містить перелік запитань з одним правильним варіантом відповіді. Після завершення фінального тесту система повідомляє про успішне завершення курсу (рис. 4.34), і статус курсу змінюється на «Завершено». Відповідна інформація зберігається в особистому кабінеті користувача, де він може переглядати результати пройдених курсів у будь-який час.

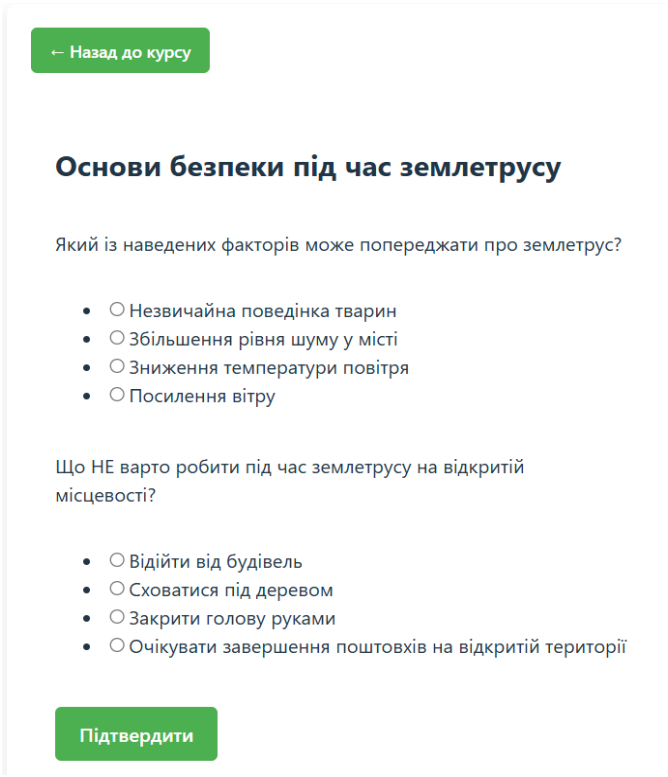


Рисунок 4.33 – Сторінка фінального тесту

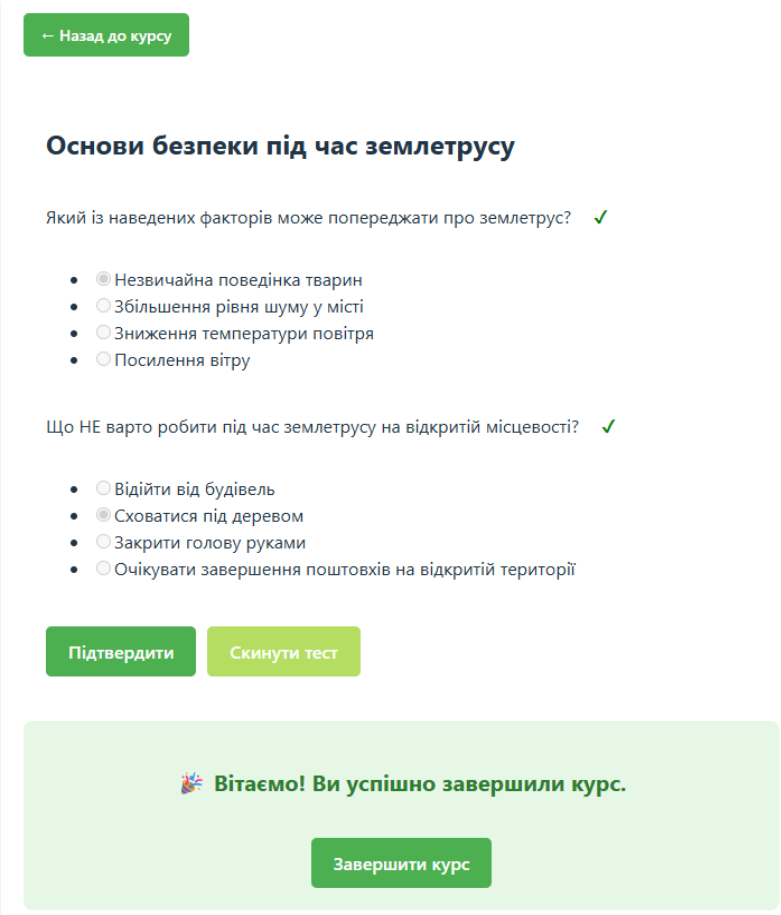


Рисунок 4.34 – Повідомлення про успішне завершення курсу

4.6 Сторінка «Мої курси» викладача

Сторінка «Мої курси» для викладача складається з двох основних модулів: блоку перегляду створених курсів і форми для створення нового курсу (рис. 4.35).

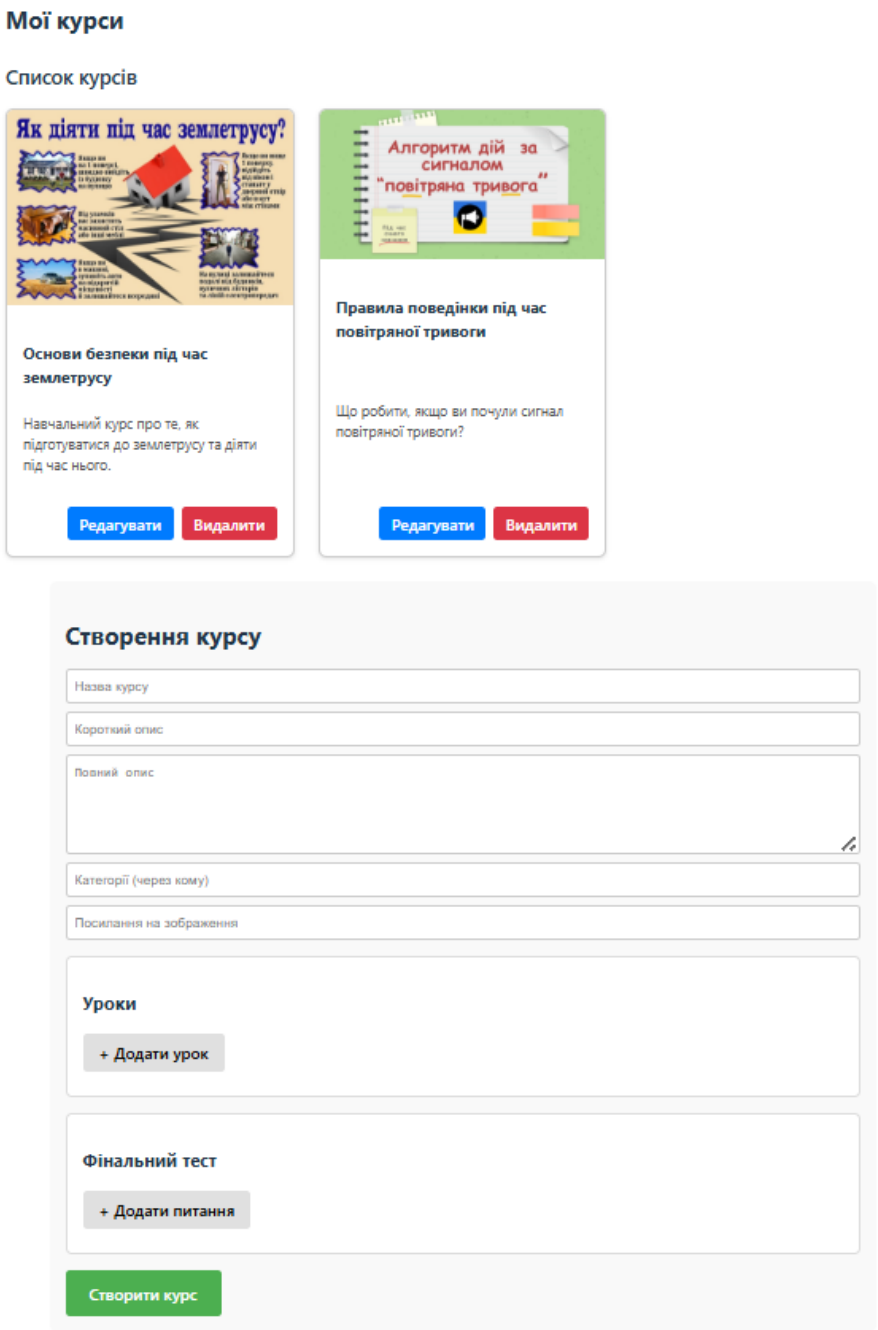


Рисунок 4.35 – Загальний вигляд сторінки «Мої курси»

У першому модулі відображаються картки всіх курсів, які викладач створив у системі. На кожній картці є зображення обкладинки, назва курсу, короткий опис та кнопки «Редагувати» і «Видалити». При натисканні кнопки «Редагувати» відкривається модальне вікно (рис. 4.36), у якому можна змінити основну інформацію про курс: назву, короткий та повний описи, категорії, а також редагувати вміст. Це дозволяє підтримувати актуальність навчального матеріалу без переходу на окрему сторінку.

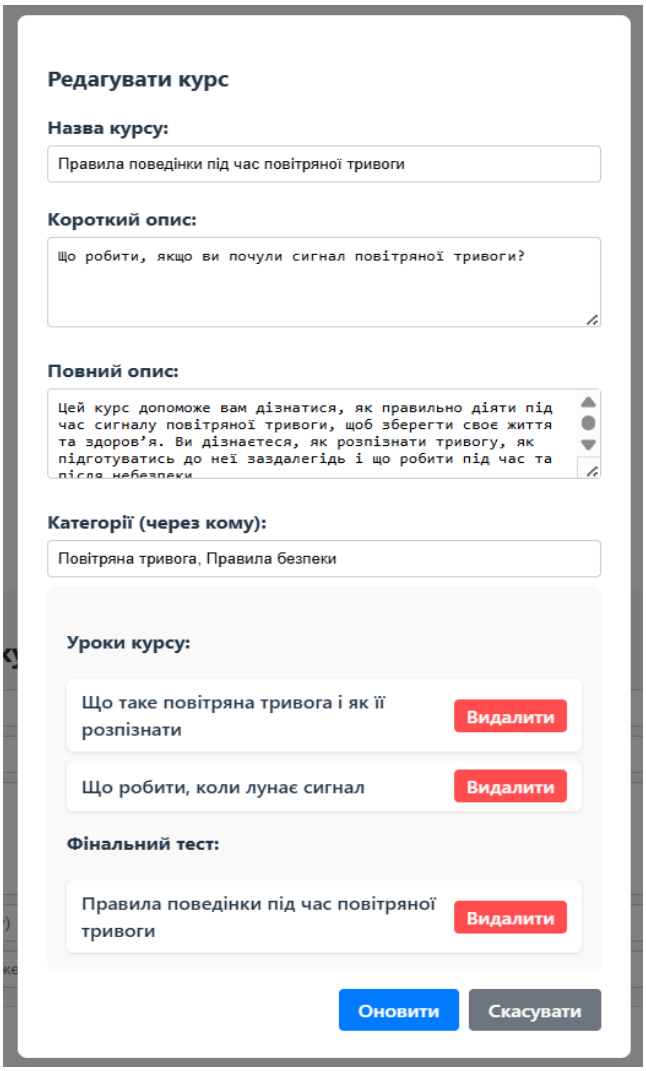


Рисунок 4.36 – Вікно редагування курсу

Другий модуль — це форма створення нового курсу. Вона містить усі необхідні поля: назва, короткий та повний описи, категорії, посилання на зображення курсу. При створенні курсу обов'язкові поля мають валідацію. Якщо користувач намагається надіслати форму без заповнення одного з таких

полів, система виводить повідомлення про помилку. Наприклад, якщо не введено назву курсу, з'являється підказка «Заповніть це поле», що не дозволяє продовжити створення курсу, поки поле не буде заповнене (рис. 4.37).

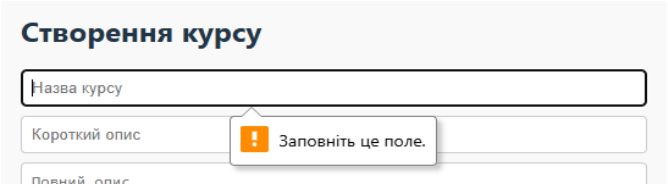


Рисунок 4.37 – Повідомлення про обов'язкове заповнення поля

Платформа забезпечує можливість додавання уроків до курсу, кожен з яких може містити текстовий контент і власний тест. Для цього достатньо натиснути кнопку «+ Додати урок», після чого відкривається розширена форма, де викладач може ввести назву, оформити навчальний матеріал за допомогою вбудованого Markdown-редактора, опціонально додати посилання на відео та створити тест до уроку (рис. 4.38). У тесті при додаванні першого питання з'являється поле для введення назви та поля де можна створювати питання з варіантами відповідей (рис. 4.39). Спочатку автоматично додається чотири варіанти відповіді, але їх кількість можна змінювати, та мінімум — два. Для кожного запитання потрібно зазначити правильний варіант відповіді.

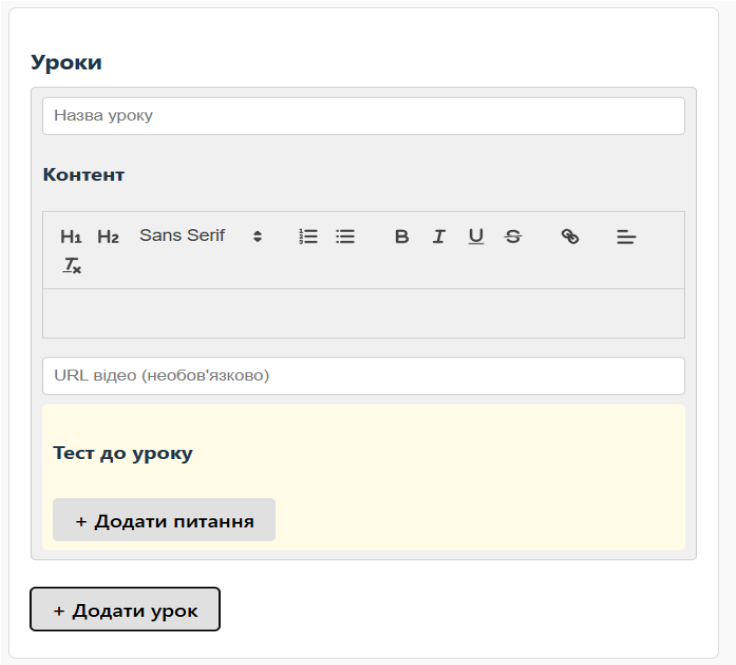


Рисунок 4.38 – Форма створення уроку з навчальним контентом і тестом

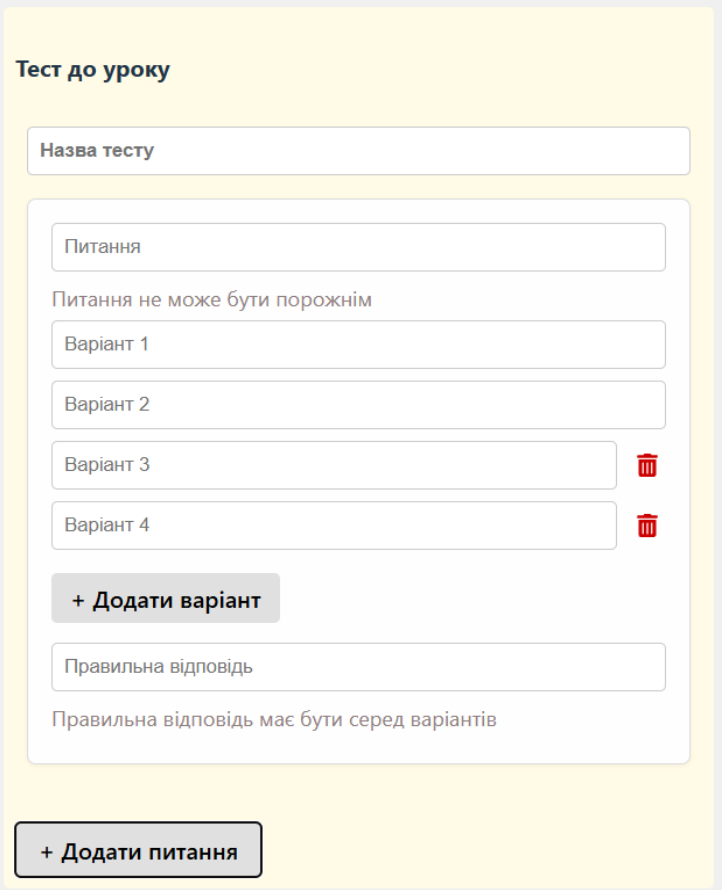


Рисунок 4.39 – Форма створення тесту прив’язаного до уроку

Також передбачена можливість створення фінального тесту. Логіка та інтерфейс його створення аналогічно зі створенням тесту для уроку.

Ця сторінка забезпечує викладача всіма інструментами для зручного створення, редагування та керування навчальними курсами.

ВИСНОВОК ДО РОЗДІЛУ 4

У цьому розділі було детально розглянуто інтерфейс та функціональні можливості веб-застосунку з позиції різних ролей користувачів — як учня, так і викладача. Продемонстровано, як користувачі взаємодіють із системою: від реєстрації та входу в обліковий запис до перегляду курсів, проходження уроків, тестування та відстеження прогресу навчання.

З боку студентів забезпечено зручну навігацію, адаптивний інтерфейс, підтримку форматованого контенту уроків та автоматичну перевірку тестових завдань. Відкриття фінального тесту лише після завершення всіх уроків стимулює дотримання навчальної послідовності.

З боку викладача реалізовано інтуїтивно зрозумілий інструмент для створення курсів, додавання уроків і тестів, керування контентом. Гнучкий редактор і модульна структура дозволяють швидко налаштовувати курс під потреби навчального процесу.

Загалом, система демонструє продуману архітектуру інтерфейсу, що забезпечує користувачам зручну, ефективну та адаптивну взаємодію на всіх етапах навчання. Представлені сторінки та можливості підтверджують функціональну повноту й практичність реалізованого рішення.

ВИСНОВКИ

У межах дипломного проєкту розроблено освітній веб-застосунок для навчання правилам поведінки під час загроз, зокрема в умовах війни. Актуальність проєкту зумовлена потребою в оперативному та зручному поширенні життєво важливої інформації в умовах повномасштабної війни.

У першому розділі проаналізовано предметну область, розглянуто вплив війни на освітній процес, оцінено роль цифрових технологій у поширенні знань із безпеки. Здійснено порівняльний аналіз існуючих рішень та виявлено їх недоліки, що й стало підґрунтям для розробки власного рішення.

У другому розділі обґрунтовано вибір технологій: React + TypeScript для фронтенду, Firebase (Authentication, Firestore, Hosting) для бекенду та зберігання даних, Tailwind CSS для адаптивної стилізації. Такий технологічний стек дозволив реалізувати гнучкий, масштабований застосунок.

У третьому розділі описано архітектуру платформи, модульну структуру, обробку користувацьких дій та реалізацію основних функцій — створення курсів, уроків, тестів, ролі користувачів, прогрес навчання, авторизацію та захист даних. Окрему увагу приділено реалізації зручного інтерфейсу для створення та редагування контенту викладачами.

У четвертому розділі проєкту продемонстровано роботу готового застосунку: від реєстрації до проходження курсів, тестувань та керування навчальними матеріалами. Платформа має простий та інтуїтивно зрозумілий адаптивний інтерфейс, що забезпечує зручність користування.

Таким чином, створений веб-застосунок відповідає поставленим вимогам. Він може бути використаний в освітніх установах, волонтерських та громадських ініціативах для навчання населення правилам безпеки під час загроз. Подальший розвиток проєкту може включати розширення контенту, впровадження сертифікації та підтримку кількох мов.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вплив повномасштабного вторгнення на освіту: дослідження. *U-LEAD*. URL: https://u-lead.org.ua/news/367 (дата звернення: 14.05.2025).
2. Гончарова І. Цифрові технології в освіті як засіб покращення доступності та ефективності навчання. *Welcome to - Digital Library NAES of Ukraine*. URL: https://lib.iitta.gov.ua/id/eprint/734946/1/Гончарова_тези.pdf (дата звернення: 14.05.2025).
3. Функціональні та нефункціональні вимоги. *Guru99*. URL: https://www.guru99.com/uk/functional-vs-non-functional-requirements.html (дата звернення: 14.05.2025).
4. Aisha A. Introduction to serverless architecture with Firebase and Cloud Functions. *DEV Community*. URL: https://dev.to/nitdgplug/introduction-to-serverless-architecture-with-firebase-and-cloud-functions-1fko (date of access: 14.05.2025).
5. Firebase advantages and disadvantages. *Back4App Blog*. URL: https://blog.back4app.com/firebase-advantages-and-disadvantages/ (date of access: 14.05.2025).
6. Serve dynamic content and host microservices using Firebase Hosting. *Firebase*. URL: https://firebase.google.com/docs/hosting/serverless-overview (date of access: 14.05.2025).
7. SPA (Single-page application) - MDN Web Docs Glossary: Definitions of Web-related terms | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Glossary/SPA (date of access: 14.05.2025).
8. Using TypeScript – React. *React*. URL: https://react.dev/learn/typescript (date of access: 14.05.2025).

9. What is CI/CD? - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/what-is-ci-cd/ (date of access: 14.05.2025).
10. What is NoSQL? Databases Explained | Google Cloud. *Google Cloud*. URL: https://cloud.google.com/discover/what-is-nosql (date of access: 14.05.2025).

ДОДАТОК А

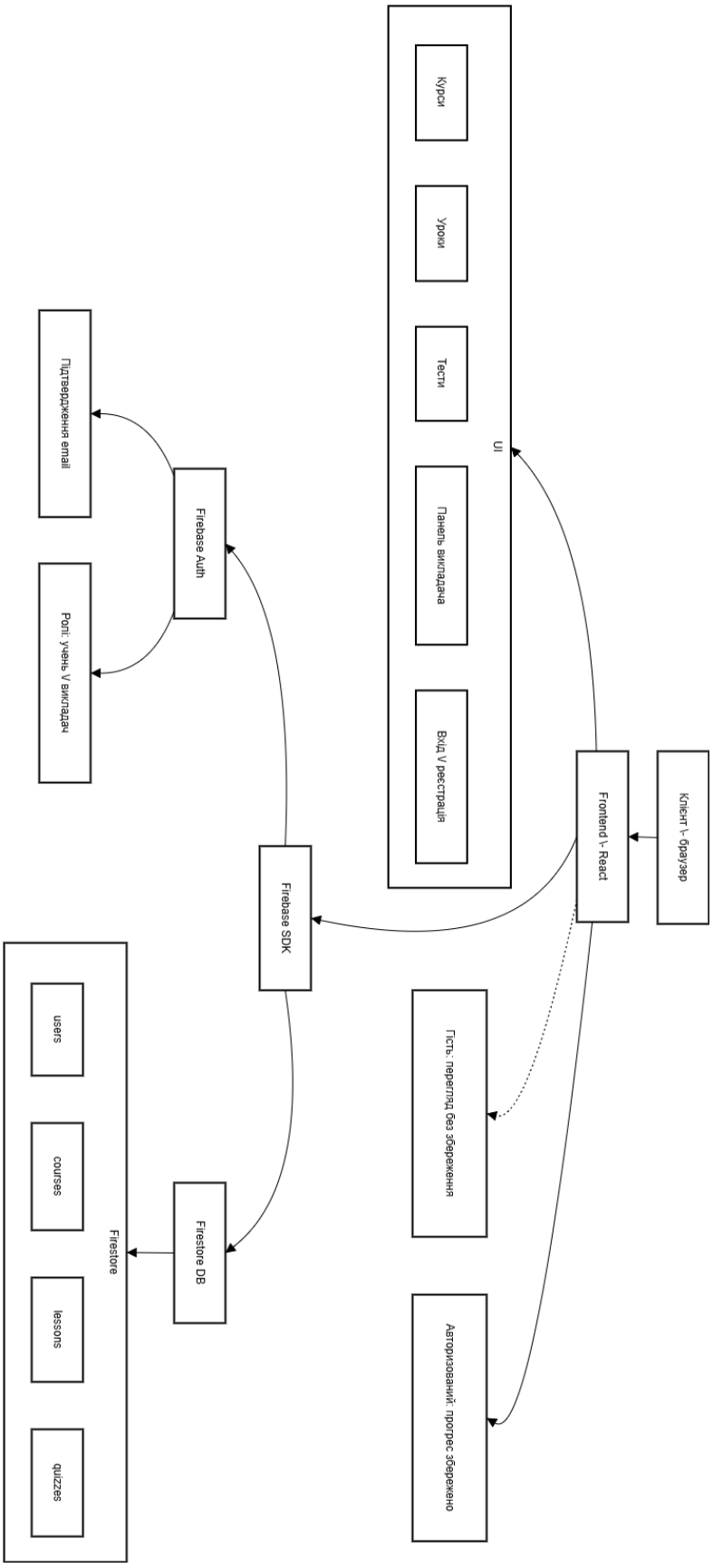
Освітній веб-застосунок для навчання правилам поведінки під час загроз

Структурна схема системи

ІАЛЦ.467200.004 Д1

Аркушів 1

Київ 2025 р



					ІАЛЦ.467200.004Д1		
		№ докум.	Підпис	Дата			
Розробила		Лоханько К.В.			**Освітній веб-застосунок для навчання правилам поведінки під час загроз** Структурна схема системи	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Пономаренко А. М.					
Реценз.		Коган А.В.				НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІО-11	
Н. контр.		Гончаренко О.О					
Затвердив							

ДОДАТОК Б

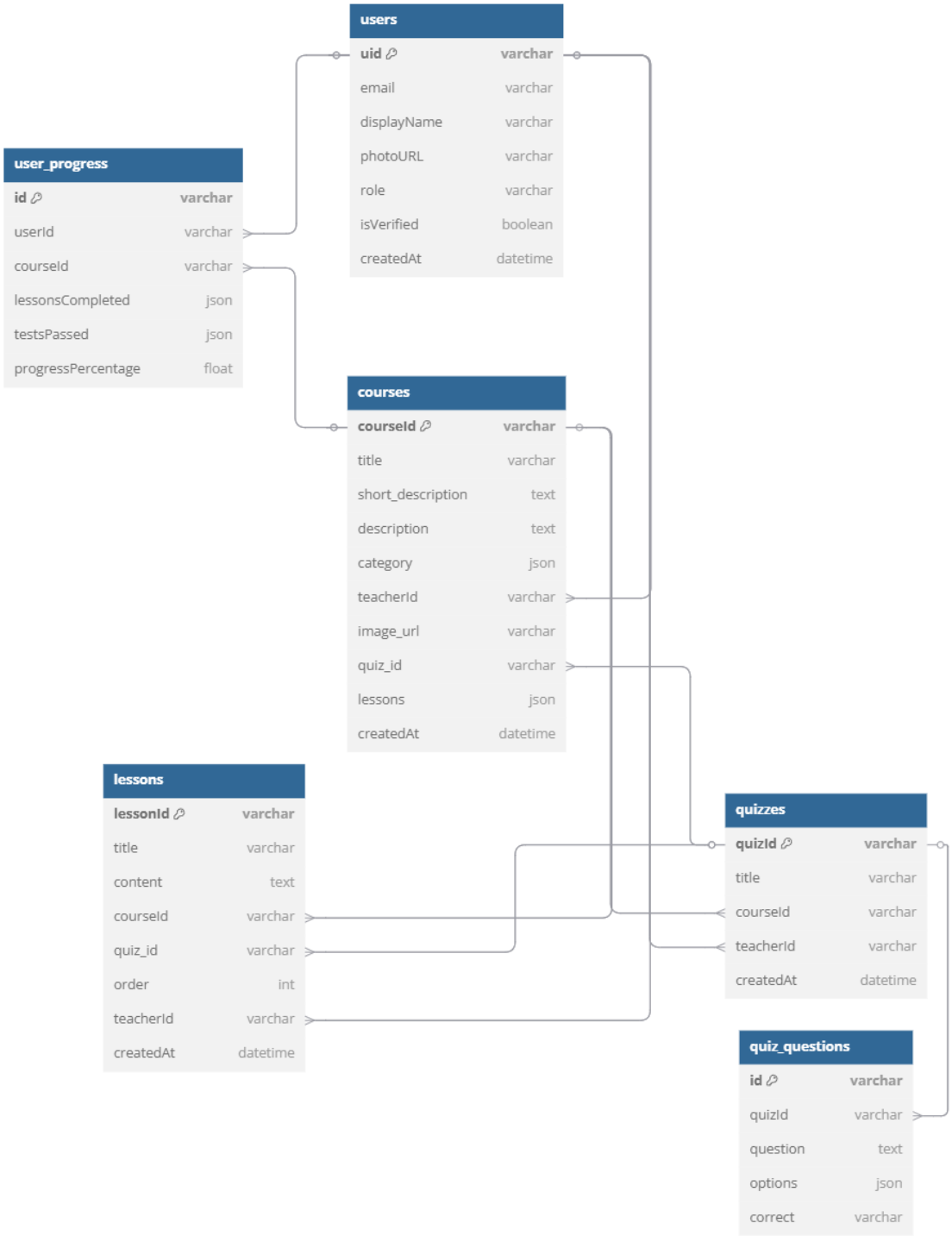
Освітній веб-застосунок для навчання правилам поведінки під час загроз

Структура бази даних

ІАЛЦ.467200.005 Д2

Аркушів 1

Київ 2025 р



					ІАЛЦ.467200.005Д2
		№ докум.	Підпис	Дата	
Розробив		Лоханько К.В.			**Освітній веб-застосунок для навчання правилам поведінки під час загроз** Структура бази даних
Перевірив		Пономаренко А.М.			
Реценз.		Коган А.В.			
Н. контр.		Гончаренко О.О			
Затвердив					

Літ.	Аркуш	Аркушів
	1	1

НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІО-11

ДОДАТОК В

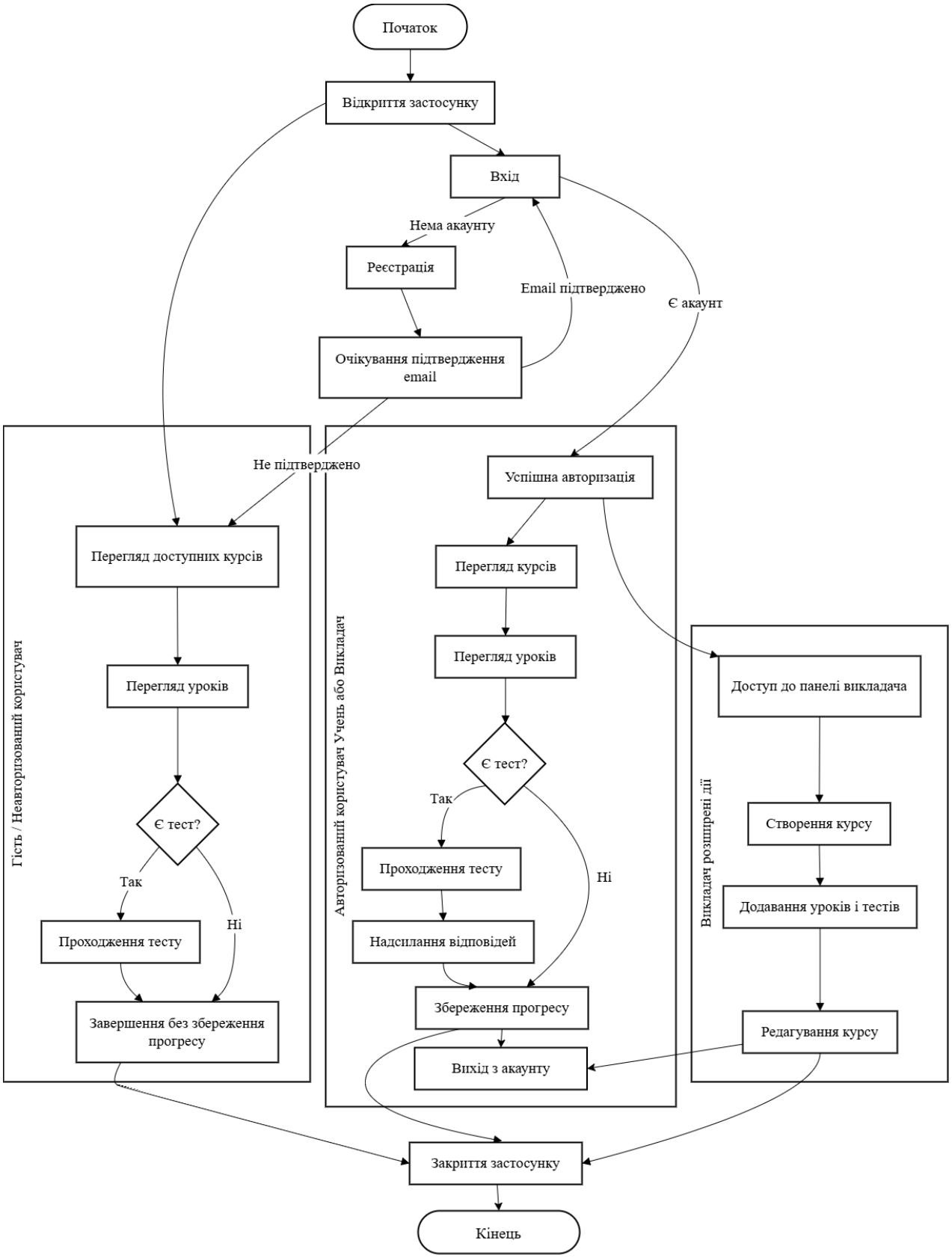
Освітній веб-застосунок для навчання правилам поведінки під час загроз

Алгоритм дій програмного забезпечення

ІАЛЦ.467200.006 Д3

Аркушів 1

Київ 2025 р



					ІАЛЦ.467200.006Д3	
		№ докум.	Підпис	Дата		
Розробив		Лоханько К.В.			**Освітній веб-застосунок для навчання правилам поведінки під час загроз** Алгоритм дій програмного забезпечення	Літ. / Аркуш 1 / Аркушів 1
Перевірив		Пономаренко А.М.				
Реценз.		Коган А.В.				НТУУ КПІ ім. Ігоря Сікорського, ФІОТ, ІО-11
Н. контр.		Гончаренко О.О				
Затвердив						

ДОДАТОК Г

Освітній веб-застосунок для навчання правилам поведінки під час загроз

Текст програмного коду

ІАЛЦ.467200.007 Д4

Аркушів 76

Київ 2025 р

components\MyCoursesComp\CourseContentList.tsx

```
//src/components/MyCoursesComp/CourseContentList.tsx


import { useEffect, useState } from "react";
import { getLessonById, deleteLessonById } from "../../services/lessonsService";
import {getQuizById, deleteQuizById, removeFinalQuizFromCourse} from "../../services/quizzesService";



interface Props {
  courseId: string;
  lessonIds: string[];
  quizId?: string;
  onRemoveLesson: (lessonId: string) => void;
  onRemoveQuiz: () => void;
}


export const CourseContentList: React.FC<Props> = ({
  courseId,
  lessonIds,
  quizId,
  onRemoveLesson,
  onRemoveQuiz,
}) => {
  const [lessonTitles, setLessonTitles] = useState<Record<string, string>>({});
  const [quizTitle, setQuizTitle] = useState<string | null>(null);
  const [loading, setLoading] = useState(true);


  useEffect(() => {
    const fetchContent = async () => {
      setLoading(true);
      try {
        const lessons = await Promise.all(
          lessonIds.map(async (id) => {
            const lesson = await getLessonById(id);
            return { id, title: lesson?.title || "РќР°Р·РІР° PSPµ Р·PSP°РЅРґРµPSP°" };
          })
        );

        const titles: Record<string, string> = {};
        lessons.forEach(({ id, title }) => {
          titles[id] = title;
        });

        setLessonTitles(titles);

        if (quizId) {
          const quiz = await getQuizById(quizId);
          setQuizTitle(quiz?.title || "РќР°Р·РІР° PSPµ Р·PSP°РЅРґРµPSP°");
        } else {
          setQuizTitle(null);
        }
```

```
      } catch (error) {
        console.error("РµPsPjPёP»РєР° Р·Р°PIP°PSC,P°P¶РµPSPSCЏ PIPjC–СЃС,Сѓ РєСѓСЂСЃР°:", error);
      }
      setLoading(false);
    };

    fetchContent();
  }, [lessonIds, quizId]);

  const handleRemoveLesson = async (id: string) => {
    try {
      const lesson = await getLessonById(id);
      if (lesson?.quizId) {
        await deleteQuizById(lesson.quizId);
      }

      await deleteLessonById(courseId, id);
      onRemoveLesson(id);

      setLessonTitles((prev) => {
        const updated = { ...prev };
        delete updated[id];
        return updated;
      });
    } catch (error) {
      console.error("РµPsPjPёP»РєР° РїСЂРё PIPёРґP°P»РµPSPSC– СѓСЂPsРєСѓ:", error);
    }
  };

const handleRemoveQuiz = async () => {
  try {
    if (quizId) {
      await deleteQuizById(quizId);
      await removeFinalQuizFromCourse(courseId); // в†Ђ PsPSPsPIP»СЂС” РґPsРєСѓPjРµPSC, РєСѓСЂСЃСѓ
      onRemoveQuiz();
      setQuizTitle(null);
    }
  } catch (error) {
    console.error("РµPsPjPёP»РєР° РїСЂРё PIPёРґP°P»РµPSPSC– C,РµСЃС,Сѓ:", error);
  }
};

    if (loading) {return <p style={{ fontStyle: "italic" }}>P—P°PIP°PSC,P°P¶РµPSPSCЏ...</p>;}

  return (
    <div style={styles.container}>
      <h4>PJCЂPsРєРё РєСѓСЂСЃСѓ:</h4>
      {lessonIds.length ? (
        <div style={styles.list}>
          {lessonIds.map((id) => (
            <div key={id} style={styles.itemRow}>
              <span style={styles.title}>{lessonTitles[id] || id}</span>
              <button style={styles.deleteButton} onClick={() => handleRemoveLesson(id)}>
                P’РёРґP°P»РёС,Ре
              </button>
            </div>
          ))}
        </div>
      ) : (
```

```
        <p
style={styles.empty}>PJCЂPsPєC–PI
PSPµPjP°C”</p>

      )}


      <h4 style={{ marginTop: "16px"
}}>P¤C–PSP°P»CЊPSPёP№ C,PµCЃC,:</h4>

      {quizId && quizTitle ? (

        <div style={styles.itemRow}>

          <span
style={styles.title}>{quizTitle}</span>

          <button
style={styles.deleteButton}
onClick={handleRemoveQuiz}>

            P’PëPґP°P»PëC,Pë

          </button>

        </div>

      ) : (

        <p style={styles.empty}>P¤C–
PSP°P»CЊPSPsPiPs C,PµCЃC,Cѓ
PSPµPjP°C”</p>

      )}

    </div>

  );

};


const styles = {

  container: {

    padding: "16px",

    backgroundColor: "#f9f9f9",

    borderRadius: "8px",

  },

  list: {

    display: "flex",

    flexDirection: "column" as const,

    gap: "8px",

  },

  itemRow: {

    display: "flex",

    justifyContent: "space-between",

    alignItems: "center",

    background: "#fff",

    padding: "8px 12px",

    borderRadius: "6px",

    boxShadow: "0 1px 3px
rgba(0,0,0,0.1)",

  },

  title: {

    fontWeight: 500,

  },

  deleteButton: {

    backgroundColor: "#ff4d4f",

    color: "#fff",

    border: "none",

    borderRadius: "4px",

    padding: "4px 10px",

    cursor: "pointer",

  },

  empty: {

    fontStyle: "italic",

    color: "#777",

  },

};
```

components\MyCoursesComp\CourseForm.tsx

```
import React, {CSSProperties, useState}
from "react";

import {Course, Lesson, Question, Quiz}
from "../../types.ts";

import {useAuth} from
"../../hooks/useAuth.ts";

import {createFullCourse,
LessonFormData} from
"../../services/mycoursesService.ts";

import QuizForm from "./QuizForm.tsx";

import LessonForm from
"./LessonForm.tsx";
```

```
interface Props {
    onCourseCreated: (course: Course) =>
void;
}

export const CourseForm: React.FC<Props>
= ({onCourseCreated}) => {
    const {user} = useAuth();
    const [title, setTitle] =
useState("");
    const [shortDescription,
setShortDescription] = useState("");
    const [description, setDescription]
= useState("");
    const [category, setCategory] =
useState<string[]>([]);
    const [imageUrl, setImageUrl] =
useState("");
    const [lessons, setLessons] =
useState<LessonFormData[]>([]);
    const [finalQuiz, setFinalQuiz] =
useState<Omit<Quiz, "quizId" |
"courseId"> | null>(null);

    const handleAddLesson = () => {
        setLessons(prev => [
            ...prev,
            {
                title: "",
                content: "",
                order: prev.length,
                videoUrl: "",
            },
        ]);
    };

    const handleLessonChange = (index:
number, field: keyof Lesson, value:
string) => {
        setLessons(prev => {
            const updated = [...prev];
            updated[index] = {
                ...updated[index],
                [field]: value,
            };
            return updated;
        });
    };

    const handleLessonQuizChange =
(index: number, field: keyof Quiz,
value: string) => {
        setLessons(prev => {
            const updated = [...prev];
            const currentQuiz =
updated[index].quiz || {title: "",
questions: []};
            updated[index] = {
                ...updated[index],
                quiz: {
                    ...currentQuiz,
                    [field]: value,
                },
            };
            return updated;
        });
    };

    const handleAddQuestionToLessonQuiz
= (index: number) => {
        setLessons(prev => {
            const updated = [...prev];
            const currentQuiz =
updated[index].quiz || {title: "",
questions: []};
            currentQuiz.questions.push({
                question: "",
                options: ["", "", "",
""],
                correct: "",
```

```
            });

            updated[index].quiz =
currentQuiz;

            return updated;

        });

    };


    const handleLessonQuestionChange =
(lIndex: number, qIndex: number, field:
keyof Question, value: string) => {

        setLessons(prev => {

            const updated = [...prev];

            if (!updated[lIndex].quiz)
return prev;

            const questions =
[...updated[lIndex].quiz!.questions];

            questions[qIndex] = {

                ...questions[qIndex],

                [field]: value,

            };


updated[lIndex].quiz!.questions =
questions;

            return updated;

        });

    };

    const handleAddOptionToLessonQuiz =
(lIndex: number, qIndex: number) => {

        setLessons(prev => {

            const updated = [...prev];

            const quiz =
updated[lIndex].quiz;

            if (!quiz) return prev;

            const updatedOptions =
[...quiz.questions[qIndex].options, ""];


quiz.questions[qIndex].options =
updatedOptions;

            return updated;

        });

    };

    const
handleRemoveOptionFromLessonQuiz =
(lIndex: number, qIndex: number, oIndex:
number) => {

        setLessons(prev => {

            const updated = [...prev];

            const quiz =
updated[lIndex].quiz;

            if (!quiz) return prev;

            const options =
[...quiz.questions[qIndex].options];

            if (options.length > 2) {

                options.splice(oIndex,
1);


quiz.questions[qIndex].options =
options;

            }

            return updated;

        });

    };


    const handleLessonOptionChange =
(lIndex: number, qIndex: number, oIndex:
number, value: string) => {

        setLessons(prev => {

            const updated = [...prev];

            const quiz =
updated[lIndex].quiz;

            if (!quiz) return prev;

            const options =
[...quiz.questions[qIndex].options];

            options[oIndex] = value;


quiz.questions[qIndex].options =
options;

            return updated;

        });

    };
```

```
    const handleAddQuestionToFinalQuiz =
() => {
        setFinalQuiz(prev => {
            const newQuestion: Question
= {
                question: "",
                options: ["", "", "",
""],
                correct: "",
            };
            if (!prev) {
                return {title: "",
questions: [newQuestion]};
            }
            return {
                ...prev,
                questions:
[...prev.questions, newQuestion],
            };
        });
    };

    const handleFinalQuizChange =
(field: keyof Quiz, value: string) => {
        setFinalQuiz(prev => (prev ?
{...prev, [field]: value || ""} :
prev)); // Ensuring value is a string
    };

    const handleQuestionChange =
(qIndex: number, field: keyof Question,
value: string) => {
        setFinalQuiz(prev => {
            if (!prev) return prev;
            const updatedQuestions =
[...prev.questions];
            updatedQuestions[qIndex] = {

...updatedQuestions[qIndex],
                [field]: value,
            };
            return {...prev, questions:
updatedQuestions};
        });
    };

    const handleOptionChange = (qIndex:
number, oIndex: number, value: string)
=> {
        setFinalQuiz(prev => {
            if (!prev) return prev;
            const updatedQuestions =
[...prev.questions];
            const updatedOptions =
[...updatedQuestions[qIndex].options];
            updatedOptions[oIndex] =
value;
            updatedQuestions[qIndex] = {

...updatedQuestions[qIndex],
                options: updatedOptions,
            };
            return {...prev, questions:
updatedQuestions};
        });
    };

    const handleAddOptionToQuestion =
(qIndex: number) => {
        setFinalQuiz(prev => {
            if (!prev) return prev;
            const updatedQuestions =
[...prev.questions];
            updatedQuestions[qIndex] = {

...updatedQuestions[qIndex],
                options:
[...updatedQuestions[qIndex].options,
""],
            };
            return {...prev, questions:
updatedQuestions};
```

```
        });
    };

    const handleRemoveOptionFromQuestion
= (qIndex: number, oIndex: number) => {
        setFinalQuiz(prev => {
            if (!prev) return prev;
            const updatedQuestions =
[...prev.questions];
            const options =
[...updatedQuestions[qIndex].options];
            if (options.length > 2) {
                options.splice(oIndex,
1);
                updatedQuestions[qIndex]
= {
...updatedQuestions[qIndex],
                    options,
                };
                return {...prev,
questions: updatedQuestions};
            }
            return prev;
        });
    };

    const handleSubmit = async (e:
React.FormEvent) => {
        e.preventDefault();
        if (!user) return
alert("РђPIC,PsCЂPёP·CґPЉCЃCЦ РїРµCЂРµРґ
CЃC,PIPsCЂРµPSPSCЦPj РєCґCЂCЃCґ");

        try {
            const preparedCourse = {
                title,
                short_description:
shortDescription,
                description,
                category,
                teacherId: user.uid,
                createdAt: new
Date().toISOString(),
                ...(imageUrl.trim() &&
{image_url: imageUrl.trim()}),
            };

            const cleanedLessons =
lessons.map((lesson, i) => ({
                title: lesson.title,
                content: lesson.content,
                order: i,
                ...(lesson.videoUrl &&
lesson.videoUrl.trim() && {videoUrl:
lesson.videoUrl.trim()}),
                ...(lesson.quiz &&
lesson.quiz.questions.length > 0 &&
{quiz: lesson.quiz}),
            }));

            const cleanedQuiz =
finalQuiz && finalQuiz.questions.length
> 0
                ? {
                    title:
finalQuiz.title,
                    questions:
finalQuiz.questions,
                }
                : undefined;

            const created = await
createFullCourse(user.uid,
preparedCourse, cleanedLessons,
cleanedQuiz);
            alert("РњCґCЂCЃ
CЃC,PIPsCЂРµPSPs CґCЃРїC-C€PSPs!");
            onCourseCreated(created);

            setTitle("");
            setShortDescription("");
```

```
            setDescription("");
            setCategory([]);
            setImageUrl("");
            setLessons([]);
            setFinalQuiz(null);
        } catch (error) {
console.error("РџРsPjРёР»РєР° РїСЂРё СЃС,PIPsCЂPµPSPSC- РєСѓСЂСЃР:", error);
            alert("РќРµ PIPrР°Р»РsСЃСЏ СЃС,PIPsCЂPёC,Рё РєСѓСЂСЃ");
        }
    };

    return (
        <form onSubmit={handleSubmit} style={styles.form}>
            <h2 style={styles.heading}>РЎС,PIPsCЂPµPSPSCЏ РєСѓСЂСЃС</h2>
            <input style={styles.input} type="text" placeholder="РќР°Р·PIР° РєСѓСЂСЃР°" value={title}
                   onChange={(e) => setTitle(e.target.value)} required/>
            <input style={styles.input} type="text" placeholder="РљPsCЂPsC,РєРёР№ PsРїРёСЃ" value={shortDescription}
                   onChange={(e) => setShortDescription(e.target.value)}/>
            <textarea style={styles.textarea} placeholder="РµPsPIPSPёР№ PsРїРёСЃ" value={description}
                      onChange={(e) => setDescription(e.target.value)}/>
            <input style={styles.input} type="text" placeholder="РљР°C,РµPiPsCЂC-C—(C‡РµCЂPµP· РєPsPjСѓ)" value={category.join(", ")}
                   onChange={(e) => setCategory(e.target.value.split(",").map((c) => c.trim()))}/>
            <input style={styles.input} type="text" placeholder="РµPsСЃРёР»PsPСЏ PSP°P·PsP±CЂP°P¶РµPSPSCЏ" value={imageUrl}
                   onChange={(e) => setImageUrl(e.target.value)}/>

            <div style={styles.block}>
                <h3 style={styles.subheading}>РJCЂPsPєРё</h3>
                {lessons.map((lesson, index) => (
                    <LessonForm
                        key={index}
                        lesson={lesson}
                        index={index}
onLessonChange={handleLessonChange}
onQuizChange={handleLessonQuizChange}
onQuestionChange={handleLessonQuestionChange}
onOptionChange={handleLessonOptionChange}
onAddQuestion={handleAddQuestionToLessonQuiz}
onAddOption={handleAddOptionToLessonQuiz}
onRemoveOption={handleRemoveOptionFromLessonQuiz}/>
))}
                <button type="button" style={styles.button} onClick={handleAddLesson}>+ Р”PsРґР°C,РёСѓСЂPsРє</button>
            </div>

            <div style={styles.block}>
                <h3 style={styles.subheading}>Р¤C-PSP°Р»СЊPSPёР№ C,РµСЃС,</h3>
                {finalQuiz && (
                    <QuizForm
```

```
                        quiz={finalQuiz}
onQuizChange={handleFinalQuizChange}
onQuestionChange={handleQuestionChange}
onOptionChange={handleOptionChange}
onAddOption={handleAddOptionToQuestion}
onRemoveOption={handleRemoveOptionFromQuestion}
                    />

                )}
                <button type="button" style={styles.button} onClick={handleAddQuestionToFinalQuiz}>+ Р”РѕРґР°С‚Рё РїРёС‚Р°РЅРЅСЏ
                </button>
            </div>

            <button type="submit" style={styles.submitButton}>РЎС‚РІРѕСЂРёС‚Рё РєС‚СЃСЃ</button>
        </form>
    );
};

const styles: { [key: string]: CSSProperties } = {
    form: {
        maxWidth: "800px",
        margin: "0 auto",
        padding: "1rem",
        backgroundColor: "#f9f9f9",
        borderRadius: "8px",
    },
    heading: {
        fontSize: "24px",
        marginBottom: "1rem",
    },
    subheading: {
        fontSize: "18px",
        marginBottom: "0.5rem",
    },
    input: {
        display: "block",
        width: "100%",
        marginBottom: "0.5rem",
        padding: "0.5rem",
        border: "1px solid #ccc",
        borderRadius: "4px",
        boxSizing: "border-box",
    },
    textarea: {
        display: "block",
        width: "100%",
        height: "100px",
        marginBottom: "0.5rem",
        padding: "0.5rem",
        border: "1px solid #ccc",
        borderRadius: "4px",
        boxSizing: "border-box",
    },
    button: {
        margin: "0.5rem 0",
        padding: "0.5rem 1rem",
        backgroundColor: "#e0e0e0",
        border: "none",
        borderRadius: "4px",
        cursor: "pointer",
    },
    submitButton: {
        marginTop: "1rem",
        padding: "0.75rem 1.5rem",
        backgroundColor: "#4caf50",
```

```
        color: "white",
        border: "none",
        borderRadius: "4px",
        cursor: "pointer",
    },
    block: {
        marginTop: "1rem",
        padding: "1rem",
        backgroundColor: "#fff",
        border: "1px solid #ddd",
        borderRadius: "6px",
    },
    lessonBlock: {
        marginBottom: "1rem",
        padding: "0.5rem",
        border: "1px solid #ccc",
        borderRadius: "4px",
        backgroundColor: "#f0f0f0",
    },
    quizBlock: {
        marginTop: "0.5rem",
        padding: "0.5rem",
        backgroundColor: "#fffbe6",
        borderRadius: "4px",
    },
};


export default CourseForm;
```

components\MyCoursesComp\CourseList.tsx

```
//src/components/MyCoursesComp/CourseList.tsx


import React, { CSSProperties } from "react";
import { Course } from "../../types";

interface Props {
  courses: Course[];
  onEdit: (course: Course) => void;
  onDelete: (courseId: string) => void;
  isLoading: boolean;
}


const CourseList: React.FC<Props> = ({ courses, onEdit, onDelete, isLoading }) => {
  const handleDelete = (courseId: string) => {
    if (window.confirm("P’Pë PIPїPµPIPSPµPSC–, C‰Ps C…PsC‡PµC,Pµ PIPëPґP°P»PëC,Pë C†PµP№ PєCґCЂCЃC?")) {
      onDelete(courseId);
    }
  };


  return (
    <div style={styles.container}>
      <h3 style={styles.heading}>PЎPїPëCЃPsPє PєCґCЂCЃC–PI</h3>


      {isLoading ? (
        <p style={styles.message}>P—P°PIP°PSC,P°P¶PµPSPSCЏ PєCґCЂCЃC–PI...</p>
      ) : courses.length === 0 ? (
        <p style={styles.message}>PљCґCЂCЃC–PI C‰PµPSPµPjP°C”.</p>
      ) : (
        <div style={styles.grid}>
          {courses.map((course) => (
            <div key={course.courseId} style={styles.card}>
              <img
                src={course.image_url || "/course_img/course.png"}
```

```
                alt={course.title}
                style={styles.image}
                onError={(e) => {
                  e.currentTarget.src =
"/course_img/course.png";
                }}
              />

              <div
style={styles.cardContent}>
                <h4
style={styles.title}>{course.title}</h4>
                <p
style={styles.description}>{course.short
_description}</p>
                <div
style={styles.buttonGroup}>
                  <button onClick={() =>
onEdit(course)}
style={styles.editButton}>P PµPrP°PiCѓPI
P°C,Pë</button>
                  <button onClick={() =>
handleDelete(course.courseId)}
style={styles.deleteButton}>P’PëPrP°P»Pë
C,Pë</button>
                </div>
              </div>
            ))}
        </div>
      )}
    </div>
  );
};

export default CourseList;

const styles: { [key: string]:
CSSProperties } = {
  container: {
    marginTop: "24px",
    marginBottom: "24px"
  },
  heading: {
    fontSize: "20px",
    fontWeight: 600,
    marginBottom: "16px"
  },
  message: {
    color: "#6b7280"
  },
  grid: {
    display: "grid",
    gap: "24px",
    gridTemplateColumns: "repeat(auto-
fill, minmax(280px, 1fr))"
  },
  card: {
    border: "1px solid #ccc",
    borderRadius: "8px",
    overflow: "hidden",
    backgroundColor: "#fff",
    boxShadow: "0 2px 4px
rgba(0,0,0,0.1)",
    display: "flex",
    flexDirection: "column",
    transition: "box-shadow 0.3s"
  },
  image: {
    width: "100%",
    height: "200px",
    objectFit: "cover"
  },
  cardContent: {
    padding: "16px",
    display: "flex",
    flexDirection: "column",
    justifyContent: "space-between",
```

```
    height: "100%"
  },
  title: {
    fontSize: "16px",
    fontWeight: "bold",
    marginBottom: "8px"
  },
  description: {
    fontSize: "14px",
    color: "#555"
  },
  buttonGroup: {
    display: "flex",
    justifyContent: "flex-end",
    gap: "8px",
    marginTop: "16px"
  },
  editButton: {
    padding: "6px 12px",
    backgroundColor: "#007bff",
    color: "white",
    border: "none",
    borderRadius: "4px",
    cursor: "pointer"
  },
  deleteButton: {
    padding: "6px 12px",
    backgroundColor: "#dc3545",
    color: "white",
    border: "none",
    borderRadius: "4px",
    cursor: "pointer"
  }
};
```

components\MyCoursesComp\EditCourseModal.tsx

```
//src/components/MyCoursesComp/EditCourseModal.tsx


import {CSSProperties, useState} from "react";
import { Course } from "../../types";
import { CourseContentList } from "./CourseContentList.tsx";


interface Props {
  course: Course;
  onClose: () => void;
  onUpdate: (course: Course) => void;
}


const EditCourseModal: React.FC<Props> = ({ course, onClose, onUpdate }) => {
  const [formState, setFormState] = useState({
    title: course.title,
    shortDescription: course.short_description,
    description: course.description,
    category: course.category.join(", "),
    lessons: course.lessons,
    quizId: course.quiz_id,
  });


  const [errors, setErrors] = useState<{
    title: string;
    shortDescription: string;
    category: string;
  }>({
    title: "",
    shortDescription: "",
    category: "",
```

```
  });


  const handleChange = (e:
React.ChangeEvent<HTMLInputElement |
HTMLTextAreaElement>) => {
    const { name, value } = e.target;
    setFormState((prev) => ({
      ...prev,
      [name]: value,
    }));
  };


  const validateForm = () => {
    const newErrors = {
      title: "",
      shortDescription: "",
      category: "",
    };


    if (!formState.title)
newErrors.title = "РќР°Р·РІР° РєСѓСЂСЃСѓ
С” РѕР±РѕРІ'СЏР·РєРѕРІРёСЋ.";
    if (!formState.shortDescription)
newErrors.shortDescription =
"РљРѕСЂРѕС‚РєРёР№ РѕРїРёСЃ С”
РѕР±РѕРІ'СЏР·РєРѕРІРёР№.";


    setErrors(newErrors);
    return !newErrors.title &&
!newErrors.shortDescription &&
!newErrors.category;
  };


  const handleSubmit = (e:
React.FormEvent) => {
    e.preventDefault();
    if (!validateForm()) return;


    const updatedCourse: Course = {
      ...course,
      title: formState.title,
      short_description:
formState.shortDescription,
      description:
formState.description,
      category:
formState.category.split(",").map((c) =>
c.trim()),
      lessons: formState.lessons,
      quiz_id: formState.quizId,
    };


    onUpdate(updatedCourse);
  };


  return (
    <div style={styles.overlay}>
      <div style={styles.modal}>
        <h3
style={styles.title}>Р РµРґР°РіСѓРІР°С‚Рё
РєСѓСЂСЃ</h3>
        <form onSubmit={handleSubmit}>
          <label
style={styles.label}>РќР°Р·РІР°
РєСѓСЂСЃСѓ:</label>
          <input
            type="text"
            name="title"
            value={formState.title}
            onChange={handleChange}
            required
            style={styles.input}
          />
          {errors.title && <p
style={styles.error}>{errors.title}</p>}


          <label
style={styles.label}>РљРѕСЂРѕС‚РєРёР№
РѕРїРёСЃ:</label>
          <textarea
```

```
            name="shortDescription"

value={formState.shortDescription}

            onChange={handleChange}

            style={styles.textarea}

          />

          {errors.shortDescription && <p
style={styles.error}>{errors.shortDescri
ption}</p>}


          <label
style={styles.label}>РµPsPIPSPёP№
PsPїPёCЃ:</label>

          <textarea

            name="description"

value={formState.description}

            onChange={handleChange}

            style={styles.textarea}

          />


          <label
style={styles.label}>РьP°C,РµPiPsCЂC–C—
(C‡РµCЂРµР· РєPsPjCѓ):</label>

          <input

            type="text"

            name="category"

            value={formState.category}

            onChange={handleChange}

            style={styles.input}

          />

          {errors.category && <p
style={styles.error}>{errors.category}</
p>}


          <CourseContentList

            courseId={course.courseId}

lessonIds={formState.lessons}

            quizId={formState.quizId}

            onRemoveLesson={(id) =>
setFormState(prev => ({ ...prev,
lessons: prev.lessons.filter(l => l !==
id) }))}

            onRemoveQuiz={() =>
setFormState(prev => ({ ...prev, quizId:
undefined }))}

          />


          <div
style={styles.buttonGroup}>

            <button type="submit"
style={styles.button}>РћPSPsPIPёC,Pё</bu
tton>

            <button

              type="button"

              onClick={onClose}

              style={{ ...styles.button,
...styles.cancelButton }}

            >

              РўРєР°СЃСѓPIP°C,Pё

            </button>

          </div>

        </form>

      </div>

    </div>

  );

};


export default EditCourseModal;


const styles: { [key: string]:
CSSProperties } = {

  overlay: {

    position: "fixed",

    top: 0, left: 0, width: "100vw",
height: "100vh",

    backgroundColor: "rgba(0, 0, 0,
0.5)",

    display: "flex", alignItems:
"center", justifyContent: "center",
```

```
  zIndex: 1000
},
modal: {
  backgroundColor: "white",
  padding: "24px",
  borderRadius: "8px",
  maxWidth: "500px",
  width: "100%",
  maxHeight: "90vh",
  overflowY: "auto",
  boxSizing: "border-box"
},
title: {
  marginBottom: "16px"
},
label: {
  fontWeight: "bold",
  display: "block",
  marginBottom: "4px",
  marginTop: "12px"
},
input: {
  width: "100%",
  padding: "8px",
  marginBottom: "8px",
  borderRadius: "4px",
  border: "1px solid #ccc",
  boxSizing: "border-box"
},
textarea: {
  width: "100%",
  padding: "8px",
  height: "80px",
  resize: "vertical",
  borderRadius: "4px",
  border: "1px solid #ccc",
  marginBottom: "8px",
  boxSizing: "border-box"
},
buttonGroup: {
  display: "flex",
  justifyContent: "flex-end",
  gap: "8px",
  marginTop: "16px"
},
button: {
  padding: "8px 16px",
  border: "none",
  borderRadius: "4px",
  backgroundColor: "#007bff",
  color: "white",
  cursor: "pointer"
},
cancelButton: {
  backgroundColor: "#6c757d"
},
error: {
  color: "red",
  fontSize: "12px",
  marginTop: "4px"
}
};
```

components\MyCoursesComp\LessonForm.tsx

```
import React, {CSSProperties} from "react";
import ReactQuill from "react-quill";
import {Lesson, Question, Quiz} from "../../types.ts";
import QuizForm from "./QuizForm.tsx";
import 'react-quill/dist/quill.snow.css';
```

```
interface Props {
    lesson: {
        title: string;
        content: string;
        videoUrl?: string;
        quiz?: Omit<Quiz, "quizId" |
"courseId">;
    };
    index: number;
    onLessonChange: (index: number,
field: keyof Lesson, value: string) =>
void;
    onQuizChange: (index: number, field:
keyof Quiz, value: string) => void;
    onQuestionChange: (lessonIndex:
number, questionIndex: number, field:
keyof Question, value: string) => void;
    onOptionChange: (lessonIndex:
number, questionIndex: number,
optionIndex: number, value: string) =>
void;
    onAddQuestion: (index: number) =>
void;
    onAddOption: (lessonIndex: number,
questionIndex: number) => void;
    onRemoveOption: (lessonIndex:
number, questionIndex: number,
optionIndex: number) => void;
}
const LessonForm: React.FC<Props> = ({

lesson,

index,

onLessonChange,

onQuizChange,

onQuestionChange,

onOptionChange,

onAddQuestion,

onAddOption,

onRemoveOption
                                    })
=> {
    const handleContentChange = (value:
string) => {
        onLessonChange(index, "content",
value);
    };

    return (
        <div style={styles.lessonBlock}>
            <input
                style={styles.input}
                type="text"
                placeholder="PќP°P·PIP°
CѓCЪPsPєCѓ"
                value={lesson.title}
                onChange={(e) =>
onLessonChange(index, "title",
e.target.value)}
                required
            />

            <div
style={styles.quillEditor}>
                <h4>PљPsPSC,PµPSC,</h4>
                <ReactQuill

value={lesson.content}

onChange={handleContentChange}
                    modules={modules}
                    formats={formats}

style={styles.quillTextarea}
                />
            </div>
```

```
            <input
                style={styles.input}
                type="text"
                placeholder="URL PIC-
PrPµPs (PSPµPsP±PsPI'CЏP·PєPsPIPs)"
                value={lesson.videoUrl
|| ""}
                onChange={(e) =>
onLessonChange(index, "videoUrl",
e.target.value)}
            />

            <div
style={styles.quizBlock}>
                <h4>PўPµCЃC, PrPs
CѓCЂPsPєCѓ</h4>
                {lesson.quiz && (
                    <QuizForm

quiz={lesson.quiz}

onQuizChange={(field, value) =>
onQuizChange(index, field, value)}

onQuestionChange={(qi, field, value) =>
onQuestionChange(index, qi, field,
value)}

onOptionChange={(qi, oi, value) =>
onOptionChange(index, qi, oi, value)}

onAddOption={(qi) => onAddOption(index,
qi)}

onRemoveOption={(qi, oi) =>
onRemoveOption(index, qi, oi)}
                    />
                )}
                <button type="button"
style={styles.button} onClick={() =>
onAddQuestion(index)}>
                    + P”PsPrP°C,Pë
PїPёC,P°PSPSCЏ
                </button>
            </div>
        </div>
    );
};


// PќP°P»P°C€C,CѓPIP°PSPSCЏ PrP»CЏ
CЂPµPrP°PєC,PsCЂP° Quill (PjPsP¶PSP°
PrPsPrP°PIP°C,Pë CЂC–P·PSC– PїP»P°PiC–
PSPë C,P° PjPsP¶P»PёPIPsCЃC,C–)

const modules = {
    toolbar: [
        [{'header': '1'}, {'header':
'2'}, {'font': []}],
        [{'list': 'ordered'}, {'list':
'bullet'}],
        ['bold', 'italic', 'underline',
'strike'],
        ['link'],
        [{'align': []}],
        ['clean']
    ],
};


const formats = [
    'header', 'font', 'list', 'bullet',
'bold', 'italic', 'underline', 'strike',
'link', 'align', 'clean'
];


const styles: { [key: string]:
CSSProperties } = {
    input: {
        display: "block",
        width: "100%",
        boxSizing: "border-box",
        marginBottom: "0.5rem",
        padding: "0.5rem",
        border: "1px solid #ccc",
        borderRadius: "4px",
    },
```

```
    quillEditor: {
        marginBottom: "1rem",
    },
    quillTextarea: {
        width: "100%",
        boxSizing: "border-box",
    },
    button: {
        marginTop: "0.5rem",
        padding: "0.5rem 1rem",
        backgroundColor: "#e0e0e0",
        border: "none",
        borderRadius: "4px",
        cursor: "pointer",
    },
    lessonBlock: {
        marginBottom: "1rem",
        padding: "0.5rem",
        border: "1px solid #ccc",
        borderRadius: "4px",
        backgroundColor: "#f0f0f0",
        overflow: "hidden",
    },
    quizBlock: {
        marginTop: "0.5rem",
        padding: "0.5rem",
        backgroundColor: "#fffbe6",
        borderRadius: "4px",
    },
};
export default LessonForm;
```

components\MyCoursesComp\QuizForm.tsx

```
import React from "react";
import {Question, Quiz} from "../../types";
import {FaTrashAlt} from "react-icons/fa";

interface QuizFormProps {
    quiz: Omit<Quiz, "quizId" | "courseId">;
    onQuizChange: (field: keyof Quiz, value: string) => void;
    onQuestionChange: (qIndex: number, field: keyof Question, value: string) => void;
    onOptionChange: (qIndex: number, oIndex: number, value: string) => void;
    onAddOption: (qIndex: number) => void;
    onRemoveOption: (qIndex: number, oIndex: number) => void;
}

const QuizForm: React.FC<QuizFormProps> = ({
quiz,
onQuizChange,
onQuestionChange,
onOptionChange,
onAddOption,
onRemoveOption
}) => {
    const validateQuestion = (q: Question) => {
        const isEmptyQuestion = q.question.trim() === "";
        const filteredOptions = q.options.filter(opt => opt.trim() !== "");
        const correctNotInOptions = !filteredOptions.includes(q.correct.trim());
```

```
        return {isEmptyQuestion,
correctNotInOptions};
    };

    return (
        <div style={styles.container}>
            <input
                type="text"
                placeholder="PќP°P·PIP°
C,PµCЃC,Cѓ"
                value={quiz.title}
                onChange={(e) =>
onQuizChange("title", e.target.value)}
                style={styles.quizTitle}
            />

            {quiz.questions.map((q, qi)
=> {
                const {isEmptyQuestion,
correctNotInOptions} =
validateQuestion(q);

                return (
                    <div key={qi}
style={styles.questionBlock}>
                        <input
                            type="text"
placeholder="PµPëC,P°PSPSCЏ"
value={q.question}
onChange={(e) => onQuestionChange(qi,
"question", e.target.value)}
style={styles.input}
                        />
                        {isEmptyQuestion
&& <div
style={styles.error}>PµPëC,P°PSPSCЏ PSPµ
PjPsP¶Pµ PŁCѓC,Pë PїPsCЂPsP¶PSC-
Pj</div>}
{q.options.map((opt, oi) => (
                                <div
key={oi} style={styles.optionRow}>
                                    <input
type="text"
placeholder={`P’P°CЂC-P°PSC, ${oi + 1}`}
value={opt}
onChange={(e) => onOptionChange(qi, oi,
e.target.value)}
style={styles.optionInput}
                                    />
                                    {oi >= 2
&& (
<button
type="button"
onClick={() => onRemoveOption(qi, oi)}
style={styles.deleteButton}
title="P’PëPґP°P»PëC,Pë PIP°CЂC-P°PSC,"
                                        >
<FaTrashAlt/>
</button>
                                    )}
                                </div>
                            ))}

                            <button
type="button" onClick={() =>
onAddOption(qi)}
style={styles.addOptionButton}>
```

```
                                +
Р”РѕРґР°С‚Рё PIP°CЂC–P°PSC,
                            </button>

                            <input
                                type="text"

placeholder="РџСЂР°PIPёР»СЊPSP° PIC–
РґРїРѕPIC–РґСЊ"

value={q.correct}

onChange={(e) => onQuestionChange(qi,
"correct", e.target.value)}

style={{...styles.input, marginTop:
"1em"}}
                            />

{correctNotInOptions && (
                                <div
style={styles.error}>РџСЂР°PIPёР»СЊPSP°
PIC–РґРїРѕPIC–РґСЊ РјР°С” Р±СѓС‚Рё
СЃРџСЂРµРґ PIP°CЂC–P°PSC,C–PI</div>
                            )}
                        </div>
                    );
                })}
            </div>
    );
};


const styles: { [key: string]:
React.CSSProperties } = {
    container: {
        maxWidth: "100%",
        width: "100%",
        padding: "0.5rem",
        boxSizing: "border-box",
    },
    quizTitle: {
        width: "100%",
        padding: "0.5rem",
        marginBottom: "1rem",
        borderRadius: "4px",
        border: "1px solid #ccc",
        fontWeight: "bold",
        boxSizing: "border-box",
    },
    questionBlock: {
        border: "1px solid #ddd",
        padding: "1rem",
        borderRadius: "6px",
        marginBottom: "1.5rem",
        backgroundColor: "#fefefe",
        boxShadow: "0 1px 3px
rgba(0,0,0,0.05)",
        boxSizing: "border-box",
    },
    input: {
        width: "100%",
        padding: "0.5rem",
        marginBottom: "0.5rem",
        borderRadius: "4px",
        border: "1px solid #ccc",
        boxSizing: "border-box",
    },
    optionRow: {
        display: "flex",
        alignItems: "center",
        marginBottom: "0.5rem",
        gap: "0.5rem",
    },
    optionInput: {
        flex: 1,
        padding: "0.5rem",
        border: "1px solid #ccc",
```

```
        borderRadius: "4px",
        boxSizing: "border-box",
    },
    deleteButton: {
        padding: "0.3rem",
        fontSize: "1rem",
        background: "transparent",
        border: "none",
        cursor: "pointer",
        color: "#c00",
        display: "flex",
        alignItems: "center",
        justifyContent: "center",
    },
    addOptionButton: {
        marginTop: "0.5rem",
        padding: "0.4rem 0.8rem",
        backgroundColor: "#e0e0e0",
        border: "none",
        borderRadius: "4px",
        cursor: "pointer",
    },
    error: {
        color: "#8f8080",
        fontSize: "0.9rem",
        marginBottom: "0.25rem",
    },
};

export default QuizForm;
```

components\BackToCourseButton.tsx

```
import React from "react";
import { useNavigate, useParams } from
"react-router-dom";

const BackToCourseButton: React.FC = ()
=> {
  const navigate = useNavigate();
  const { courseId } = useParams<{
courseId: string }>();

  return (
    <button
      style={styles.button}
      onClick={() =>
navigate(`/courses/${courseId}`)}
    >
      в†ђ РќР°Р·Р°Рґ РґРѕ РєСѓСЂСЃР°
    </button>
  );
};

const styles: Record<string,
React.CSSProperties> = {
  button: {
    padding: "10px 16px",
    fontSize: "14px",
    color: "#fff",
    backgroundColor: "#4caf50",
    border: "none",
    borderRadius: "5px",
    cursor: "pointer",
    marginBottom: "20px",
    transition: "background 0.3s",
  },
};

export default BackToCourseButton;
```

components\CourseCard.tsx

```
import React, {useState} from "react";

interface CourseCardProps {
```

```
  course: {
    courseId: string;
    title: string;
    image_url?: string;
    short_description: string;
    category: string[];
  };
  progress: number;
  onClick: () => void;
}

const CourseCard:
React.FC<CourseCardProps> = ({ course,
progress, onClick }) => {
  const [isHovered, setIsHovered] =
useState(false);
  const progressText = progress === 100
? "Р—Р°РІРµСЂС€РµРSPs" :
`РµСЂРsРiСЂРµСЃ: ${progress}%`;

  return (
      <div
          style={{
              border: "1px solid #ddd",
              borderRadius: "8px",
              overflow: "hidden",
              boxShadow: isHovered ? "0
6px 12px rgba(0, 0, 0, 0.2)" : "0 4px
8px rgba(0, 0, 0, 0.1)",
              cursor: "pointer",
              transition: "transform
0.3s, box-shadow 0.3s",
              transform: isHovered ?
"scale(1.05)" : "scale(1)",
          }}
          onClick={onClick}
          onMouseEnter={() =>
setIsHovered(true)}
          onMouseLeave={() =>
setIsHovered(false)}
      >
          <img
              src={course.image_url ||
"/course_img/course.png"}
              alt={course.title}
              style={{width: "100%",
height: "200px", objectFit: "cover"}}
              onError={(e) => {
                  e.currentTarget.src =
"/course_img/course.png";
              }}
          />

          <div style={{padding:
"15px"}}>
              <h3>{course.title}</h3>

              <p style={{fontSize:
"16px", color: "#000", marginTop:
"10px"}}>{course.short_description}</p>
              <p>РљР°С‚РµРiРsСЂС–СЏ:
{course.category.join(", ")}</p>

              {/* РўРµРєСЃС
РїСЂРsРiСЂРµРСЃСѓ */}
              <p style={{fontSize:
"14px", marginBottom: "5px", color:
"#555"}}>{progressText}</p>

              {/* РµСЂРsРiСЂРµРСЃ Р±Р°СЂ
*/}
              <div style={{
                  width: "100%",
                  backgroundColor:
"#eee",
                  height: "10px",
                  borderRadius: "5px",
                  marginTop: "5px"
              }}>
                  <div style={{
```

```
                        width:
`${progress}%`,
                        backgroundColor:
"#4caf50",
                        height: "100%",
                        borderRadius:
"5px"
                    }}></div>
                </div>
            </div>
        </div>
    );
};

export default CourseCard;
```

components\CourseCardMini.tsx

```
import React, {useState} from "react";

interface CourseCardProps {
  course: {
    courseId: string;
    title: string;
    image_url?: string;
  };
  progress: number;
  onClick: () => void;
}

const CourseCardMini:
React.FC<CourseCardProps> = ({ course,
progress, onClick }) => {
  const [isHovered, setIsHovered] =
useState(false);
  const progressText = progress === 100
? "Р—Р°РІРµСЂС€РµРЅРѕ" :
`РµСЂРѕРіСЂРµСЃСЃ: ${progress}%`;

  return (
    <div
      style={{
        border: "1px solid #ddd",
        borderRadius: "8px",
        overflow: "hidden",
        boxShadow: isHovered ? "0 6px
12px rgba(0, 0, 0, 0.2)" : "0 4px 8px
rgba(0, 0, 0, 0.1)",
        cursor: "pointer",
        transition: "transform 0.3s,
box-shadow 0.3s",
        transform: isHovered ?
"scale(1.05)" : "scale(1)",
      }}
      onClick={onClick}
      onMouseEnter={() =>
setIsHovered(true)}
      onMouseLeave={() =>
setIsHovered(false)}
    >
        <img
          src={course.image_url}
          alt={course.title}
          style={{ width: "100%",
height: "200px", objectFit: "cover" }}
          onError={(e) => {
            e.currentTarget.src =
"/course_img/course.png";
          }}
        />

      <div style={{ padding: "15px" }}>
        <h3>{course.title}</h3>
```

```
        {/* РўРµРєСЃС, РїСЂРsPiСЂРµСЃСѓ
*/}

        <p style={{ fontSize: "14px",
marginBottom: "5px", color: "#555"
}}>{progressText}</p>


        {/* РµСЂРsPiСЂРµСЃ Р±Р°СЂ */}

        <div style={{ width: "100%",
backgroundColor: "#eee", height: "10px",
borderRadius: "5px", marginTop: "5px"
}}>

          <div style={{ width:
`${progress}%`, backgroundColor:
"#4caf50", height: "100%", borderRadius:
"5px" }}></div>

        </div>

      </div>

    </div>

  );

};



export default CourseCardMini;



components\CourseDetails.tsx

import React, { useEffect, useState }
from "react";

import { useParams, useNavigate } from
"react-router-dom";

import { Course, Lesson, Quiz} from
"../types";

import { auth } from
"../services/firebaseConfig";

import { getUserProgress } from
"../services/progressService";



interface CourseDetailsProps {

  course: Course;

  lessons: Lesson[];

  quizzes: Quiz[];

  ProgressPercentage: number; //
РђPSPsPIP»РµPSPSCЦ РїСЂРsPiСЂРµСЃСѓ Р·
Firebase

}



const CourseDetails:
React.FC<CourseDetailsProps> = ({

  course,

  lessons,

  quizzes,

  ProgressPercentage,

}) => {

  const { courseId } = useParams<{
courseId: string }>();

  const navigate = useNavigate();

  const isAuthenticated =
!!auth.currentUser;



  const [progressPercentage,
setProgressPercentage] =
useState(ProgressPercentage);

const [lessonsCompleted,
setLessonsCompleted] =
useState<string[]>([]);

const [testsPassed, setTestsPassed] =
useState<string[]>([]);



useEffect(() => {

  const fetchUserProgress = async () =>
{

    if (isAuthenticated &&
auth.currentUser?.uid && courseId) {

      const progress = await
getUserProgress(auth.currentUser.uid,
courseId);

      if (progress) {


setLessonsCompleted(progress.lessonsComp
leted || []);


setTestsPassed(progress.testsPassed ||
[]);


setProgressPercentage(progress.progressP
ercentage || 0);

      }
```

```
    }
  };

  fetchUserProgress();
}, [courseId, isAuthenticated]);

const handleStartCourse = () => {
  if (!course) return;

  // РЁСѓРєР°С”РјРs РїРµСЂС€РёР№
PSPµP·P°PIPµCЂC€PµPSPёР№ СѓСЂРsРє
  const firstIncompleteLesson =
lessons.find(
    (lesson) =>
!lessonsCompleted.includes(lesson.lesson
Id)
  );

  if (firstIncompleteLesson) {
navigate(`/courses/${courseId}/lessons/$
{firstIncompleteLesson.lessonId}`);
    return;
  }

  // РїРєС‰Ps PICЃC- СѓСЂРsРєРё
P·P°PIPµCЂC€PµPSPs, C€СѓРєР°С”РjPs
РїРµСЂС€РёР№ PSPµP·P°PIPµCЂC€PµPSPёР№
РєPIC-P·
  const firstIncompleteQuiz =
quizzes.find(
    (quiz) =>
!testsPassed.includes(quiz.quizId)
  );

  if (firstIncompleteQuiz) {
navigate(`/courses/${courseId}/quizzes/$
{firstIncompleteQuiz.quizId}`);
  } else {
    alert("Р’СЃС- Р·Р°PIРґР°PSPSCЦ
Р·Р°PIPµCЂC€PµPSPs!");
  }
};

  return (
    <div style={styles.container}>
      <img
        src={course.image_url ||
"/course_img/course.png"}
        alt={course.title || "РљСѓСЂСЃ"}
        style={styles.image}
      />
      <div style={styles.content}>
        <h1
style={styles.title}>{course.title}</h1>
        <p
style={styles.description}>{course.descr
iption}</p>
        <p style={styles.category}>
          РљР°С‚РµРіРѕСЂС–С—:
{course.category?.join(", ") ||
"РќРµРјР°С” РєР°С‚РµРіРѕСЂС–С—"}
        </p>

        {isAuthenticated && (
          <div
style={styles.progressContainer}>
            <label
style={styles.progressLabel}>
              РµСЂРsРiСЂРµСЃ:
{progressPercentage}%
            </label>
            <div
style={styles.progressBar}>
              <div
                style={{
...styles.progressFill,
                  width:
`${progressPercentage}%`,
```

```
                }}
              ></div>
            </div>
          </div>
        )}

        <button
style={styles.startButton}
onClick={handleStartCourse}>
          {progressPercentage === 100
            ? "Р’СЃС– Р·Р°РІРґР°РЅРЅСЏ Р·Р°РІРµСЂС€РµРЅРѕ"
            :
"РµРѕС‡Р°С‚Рё/РµСЂРѕРґРѕР¶РёС‚Рё РєСѓСЂСЃ"}
        </button>
      </div>
    </div>
  );
};


const styles: Record<string,
React.CSSProperties> = {
  container: { padding: "20px" },
  image: { width: "100%", maxHeight:
"300px", objectFit: "cover" },
  content: { marginTop: "20px" },
  title: { fontSize: "2rem",
marginBottom: "10px" },
  description: { fontSize: "1.2rem",
color: "#555" },
  category: { fontSize: "1rem",
fontStyle: "italic" },
  progressContainer: { marginTop: "20px"
},
  progressLabel: { fontSize: "1rem" },
  progressBar: { width: "100%",
backgroundColor: "#eee", height: "10px",
borderRadius: "5px", overflow: "hidden"
},
  progressFill: { height: "100%",
backgroundColor: "#4caf50" },
  startButton: {
    marginTop: "20px",
    padding: "10px 20px",
    fontSize: "1rem",
    cursor: "pointer",
    backgroundColor: "#007BFF",
    color: "#fff",
    border: "none",
    borderRadius: "5px",
  },
};

export default CourseDetails;
```

components\CourseList.tsx

```
import React, { useEffect, useState }
from "react";
import { useNavigate } from "react-
router-dom";
import { getCourses } from
"../services/coursesService";
import {Course} from "../types.ts";
import { getUserData } from
"../services/usersService";
import { useAuth } from
"../hooks/useAuth";
import CourseCard from
"../components/CourseCard";

interface CourseListProps {
  searchTerm: string;
  selectedCategory: string;
}

const CourseList:
React.FC<CourseListProps> = ({
searchTerm, selectedCategory }) => {
  const [courses, setCourses] =
useState<Course[]>([]); // РЎРїРёСЃРѕРє
РєСѓСЂСЃС–РІ
```

```
  const [progressData, setProgressData]
= useState<Record<string, number>>({});
// РµСЂРsРiСЂРµСЃ РєРsСЂРёСЃС,СѓРIР°С‡Р°

  const [loading, setLoading] =
useState(true); // РЎС,РµР№С, РґР»СЏ С–
PSРґРёРєР°С†С–С—
Р·Р°РIР°PSC,Р°Р¶РµPSPSCЏ

  const [error, setError] =
useState<string | null>(null); //
РЎС,РµР№С, РґР»СЏ РїРsPjРёР»РєРё

  const navigate = useNavigate();

  const { user, loading: authLoading } =
useAuth();


  useEffect(() => {

    const loadCourses = async () => {

      try {

        const courseData = await
getCourses();



        // Р¤С–Р»СЊС,СЂР°С†С–СЏ
РєСѓСЂСѓС–С–РI Р·Р° РєР°С,РµРiРsСЂС–С”СЂ

        const filteredCourses =
courseData.filter(course =>

          (selectedCategory === "" ||
course.category.includes(selectedCategor
y)) &&

          (searchTerm === "" ||
course.title.toLowerCase().includes(sear
chTerm.toLowerCase()))

        );



        setCourses(filteredCourses);



        if (user) {

          const userData = await
getUserData(user.uid);



          if (userData) {

            const progressMap:
Record<string, number> = {};

            for (const course of
filteredCourses) {

              const courseProgress =
userData.progress[course.courseId];

              if (courseProgress) {


progressMap[course.courseId] =
courseProgress.progressPercentage;

              } else {


progressMap[course.courseId] = 0;

              }

            }



setProgressData(progressMap);

          }

        }

      } catch (err) {

        console.error("РќРµ
РІРґР°Р»РsСЃСЏ Р·Р°РIР°PSC,Р°Р¶РёС,Рё
РєСѓСЂСЃРё:", err);

        setError("РќРµ РІРґР°Р»РsСЃСЏ
Р·Р°РIР°PSC,Р°Р¶РёС,Рё РєСѓСЂСЃРё.");

      } finally {

        setLoading(false);

      }

    };



    if (!authLoading) {

      loadCourses();

    }

  }, [user, searchTerm,
selectedCategory, authLoading]);



  const handleCourseClick = (courseId:
string) => {

    navigate(`/courses/${courseId}`);

  };



  if (loading|| authLoading) return
<div>Р—Р°РIР°PSC,Р°Р¶РµPSPSCЏ РєСѓСЂСЃС–
РI...</div>;
```

```
  if (error) return <div>{error}</div>;

  return (
    <div>
      <h2>РЎРїРёСЃРѕРє РєСѓСЂСЃРІ</h2>
      <div style={{ display: "grid",
gridTemplateColumns: "repeat(auto-fill,
minmax(250px, 1fr))", gap: "20px" }}>
        {courses.map((course) => (
          <CourseCard
            key={course.courseId}
            course={course}

progress={progressData[course.courseId]
|| 0}
            onClick={() =>
handleCourseClick(course.courseId)}
          />
        ))}
      </div>
    </div>
  );
};

export default CourseList;
```

components\Footer.tsx

```
import { CSSProperties } from "react";

const Footer = () => {
  return (
    <footer style={styles.footer}>
      <p>&copy; 2025 РђСЃРІС-С,РЅС-Р№
РІРµР±-Р·Р°СЃС,РєСѓРѕРє. Р’СЃС-
РїСЂР°РІР° Р·Р°С…РёС‰РµРЅС-.</p>
    </footer>
  );
};

const styles: {footer: CSSProperties} =
{
  footer: {
    backgroundColor: "#4CAF50",
    padding: "10px",
    color: "white",
    textAlign: "center",
  }
};

export default Footer;
```

components\GoogleLoginButton.tsx

```
//src/components/GoogleLoginButton.tsx

import React, { useState } from "react";
import { signInWithPopup } from
"firebase/auth";
import { auth, googleProvider } from
"../services/firebaseConfig";
import { useNavigate } from "react-
router-dom";
import { addUserToFirestore } from
"../services/usersService";

const GoogleLoginButton = () => {
  const [role, setRole] =
useState<"СѓС‡РµРЅРёРє" |
"РІРёРєР»Р°РґР°С‡">("СѓС‡РµРЅРёРє");
  const navigate = useNavigate();

const handleGoogleLogin = async () => {
  try {
    const result = await
signInWithPopup(auth, googleProvider);
    const user = result.user;
```

```
    await addUserToFirestore(user,
role); // P”PsPrP°C”PjPs
PєPsCЂPёCЃC,CґPIP°C‡P° PI Firestore

    navigate("/");

  } catch (error) {

    console.error("PµPsPjPёP»PєP° PїCЂPё
Google-P»PsPiC–PSC–:", error);

  }

};


  return (

    <div style={styles.containerStyle}>

      <label style={styles.labelStyle}>

        PЋP±PµCЂC–C,CЊ CЂPsP»CЊ:

        <select value={role}
onChange={(e) => setRole(e.target.value
as "CґC‡PµPSCЊ" | "PIPёPєP»P°PґP°C‡")}
style={styles.selectStyle}>

          <option
value="CґC‡PµPSCЊ">PJC‡PµPSCЊ</option>

          <option
value="PIPёPєP»P°PґP°C‡">P’PёPєP»P°PґP°C
‡</option>

        </select>

      </label>

      <button
onClick={handleGoogleLogin}
style={styles.buttonStyle}>

        PJPIC–P№C,Pё C‡PµCЂPµP· Google

      </button>

    </div>

  );

};



const styles: Record<string,
React.CSSProperties> = {

   containerStyle: {

    display: "flex",

    flexDirection: "column",

    gap: "10px",

    alignItems: "center",

    marginBottom: "20px",

  },


  labelStyle: {

    fontSize: "16px",

    color: "#555",

  },


  selectStyle: {

    padding: "4px",

    fontSize: "16px",

    border: "1px solid #ccc",

    borderRadius: "5px",

    margin: "0 0 5px 10px ",

  },


  buttonStyle: {

    padding: "10px 20px",

    fontSize: "16px",

    backgroundColor: "#007BFF",

    color: "white",

    border: "none",

    borderRadius: "5px",

    cursor: "pointer",

  },

};


export default GoogleLoginButton;


components\Header.tsx

import { useState, useEffect,
CSSProperties } from "react";

import { Link, useLocation } from
"react-router-dom";

import { getAuth, onAuthStateChanged,
User } from "firebase/auth";
```

```
import LogoutButton from
"./LogoutButton";

import { getUserRole } from
"../services/usersService";


const Header = () => {

  const [isOpen, setIsOpen] =
useState(false);

  const [isMobile, setIsMobile] =
useState(window.innerWidth <= 970);

  const [user, setUser] = useState<User
| null>(null);

  const location = useLocation();

  const auth = getAuth();

  const [userRole, setUserRole] =
useState<string | null>(null);

  const toggleMenu = () => {

    setIsOpen(!isOpen);

  };


  useEffect(() => {

    const handleResize = () => {

      setIsMobile(window.innerWidth <=
970);

      if (window.innerWidth > 970) {

        setIsOpen(false);

      }

    };

    window.addEventListener("resize",
handleResize);

    return () =>
window.removeEventListener("resize",
handleResize);

  }, []);


  useEffect(() => {

    setIsOpen(false);

  }, [location]);


  useEffect(() => {

    const fetchUserRole = async () => {

      if (user) {

        const role = await
getUserRole(user.uid);

        setUserRole(role);

      } else {

        setUserRole(null);

      }

    };

    fetchUserRole();

  }, [user]);


  useEffect(() => {

    const unsubscribe =
onAuthStateChanged(auth, (user) => {

      setUser(user);

    });

    return () => unsubscribe();

  }, [auth]);


  return (

    <header style={styles.header}>

      <div style={styles.container}>

        <div style={styles.logoSection}>

          <img
src="/icon/headericon.svg"
alt="P›PsPiPsC,PëPï"
style={styles.logo}/>

          <h1
style={styles.title}>PђCЃPIC–C,PSC–P№
PIPµP±–P·P°CЃC,PsCЃCѓPSPsPє</h1>

        </div>


        {isMobile && (

          <button onClick={toggleMenu}
style={styles.burger}>в˜ °</button>

        )}
```

```
        <nav style={isMobile ? {
...styles.navMobile, display: isOpen ?
"flex" : "none", border: isOpen ? "2px
solid #388E3C" : "none" } : styles.nav}>
          <ul
            style={{
              ...styles.navList,
              ...(isMobile ?
styles.navMobileList : {}),
            }}
          >
            <li><Link to="/"
onClick={toggleMenu}
style={styles.navButton}>P‟PsP»PsPIPSP°<
/Link></li>
            <li><Link to="/courses"
onClick={toggleMenu}
style={styles.navButton}>РљСѓСЂСЃРё</Lin
k></li>

            {userRole ===
"PIPëPєP»P°PґP°C‡" && (
              <li>
                <Link to="/my-courses"
onClick={toggleMenu}
style={styles.navButton}>
                  РњРѕС— РєСѓСЂСЃРё
                </Link>
              </li>
            )}

            {user ? (
              <>
                <li><Link to="/profile"
onClick={toggleMenu}
style={styles.navButton}>РљР°Р±С-
PSPµC,</Link></li>
                <li><LogoutButton/></li>
              </>
            ) : (
              <li><Link to="/login"
onClick={toggleMenu}
style={styles.navButton}>P’C…C-
Pґ</Link></li>
            )}
          </ul>
        </nav>
      </div>
    </header>
  );
};

const styles: { [key: string]:
CSSProperties } = {
  header: {
    backgroundColor: "#4CAF50",
    color: "white",
    padding: "10px 0",
    width: "100vw",
  },
  container: {
    display: "flex",
    justifyContent: "space-between",
    alignItems: "center",
    maxWidth: "1200px",
    margin: "0 auto",
    padding: "0 20px",
  },
  logoSection: {
    display: "flex",
    alignItems: "center",
    gap: "10px",
  },
  logo: {
    height: "40px",
  },
  title: {
    fontSize: "1.5em",
    margin: 0,
```

```
},
nav: {
  display: "flex",
  justifyContent: "flex-end",
},
navMobile: {
  position: "absolute",
  top: "60px",
  right: "20px",
  backgroundColor: "#4CAF50",
  borderRadius: "8px",
  padding: "10px",
  zIndex: 1000,
  flexDirection: "column",
  gap: "10px",
  boxSizing: "border-box",
},
navList: {
  listStyle: "none",
  display: "flex",
  alignItems: "center",
  gap: "5px",
  padding: "10px 0",
  margin: 0,
},
navMobileList: {
  flexDirection: "column",
  alignItems: "flex-start",
},
burger: {
  fontSize: "24px",
  background: "none",
  border: "none",
  color: "white",
  cursor: "pointer",
  display: "inline-block",
  outline: "none",
  marginLeft: "auto",
},
navButton: {
  background: "none",
  border: "none",
  color: "white",
  fontSize: "16px",
  cursor: "pointer",
  padding: "10px",
  textAlign: "center",
  textDecoration: "none",
},
LogoutButton: {
  display: "block",
  width: "90%",
  textAlign: "right",
},
};

export default Header;
```

components\LearningProgress.tsx

```
import React, { useEffect, useState } from "react";

import { getUserData } from "../services/usersService"; // P’PëPєPsCЂPëCЃC,PsPICѓC”PjPs getUserData

import { getCourseById } from "../services/coursesService";

import CourseCardMini from "../components/CourseCardMini"; // PєPsPjPїPsPSPµPSC, PґP»CЦ PIC–PґPsP±CЂP°P¶PµPSPSCЦ PєP°CЂC,PєPë PєCѓCЂCЃCѓ

import { useNavigate } from "react-router-dom";
```

```
import { auth } from
"../services/firebaseConfig.ts";


interface CourseProgress {
  courseId: string;
  title: string;
  image_url: string;
  progressPercentage: number;
}


interface LearningProgressProps {
  progress: {
    [courseId: string]: {
      lessonsCompleted: string[];
      testsPassed: string[];
      progressPercentage: number;
    };
  };
}


const LearningProgress:
React.FC<LearningProgressProps> = ({
progress }) => {  const [userCourses,
setUserCourses] =
useState<CourseProgress[]>([]); //
РЎС‚РµРќС, РґР»СЏ Р·Р±РµСЂРµР¶РµРSPSCЏ
РєСѓСЂСЃР-РІ РєРѕСЂРёСЃС‚СѓРІР°С‡Р°

  const [loading, setLoading] =
useState(true);
  const [error, setError] =
useState<string | null>(null);
  const navigate = useNavigate();
  const user = auth.currentUser;


  useEffect(() => {
    const fetchUserCourses = async () =>
{
      if (!user) return;


      try {
        const userData = await
getUserData(user.uid);
        if (userData) {
          const progressEntries =
Object.entries(progress);


          const coursePromises =
progressEntries.map(async ([courseId,
progress]) => {
            const course = await
getCourseById(courseId);
            return course
              ? {
                  courseId:
course.courseId,
                  title: course.title,
                  image_url:
course.image_url ||
"public/course_img/course.png",
                  progressPercentage:
progress.progressPercentage,
                }
              : null;
          });


          const userCourses = (await
Promise.all(coursePromises)).filter(
            (course) => course !== null
          ) as CourseProgress[];


          setUserCourses(userCourses);
        }
      } catch (error) {
        setError("РµРsPjРёР»РєР° РїСЂРё
PsC,СЂРёРjР°PSPSC- РєСѓСЂСЃР-РІ
РєРѕСЂРёСЃС‚СѓРІР°С‡Р°.");
        console.error(error);
      } finally {
        setLoading(false);
      }
    };
```

```
    fetchUserCourses();
  }, [user]);

  const handleCourseClick = (courseId:
string) => {
    navigate(`/courses/${courseId}`);
  };

  if (loading) {
    return <div>P—
P°PIP°PSC,P°P¶PµPSPSCЏ...</div>;
  }

  if (error) {
    return <div>{error}</div>;
  }

  return (
    <div>
      <h2>PњPsC— PєCґCЂCЃPë</h2>
      <div style={{ display: "flex",
flexWrap: "wrap", gap: "20px" }}>
        {userCourses.length === 0 ? (
          <p>PJ PIP°CЃ PSPµPjP°C”
P°PєC,PëPIPSPëC… PєCґCЂCЃC-PI.</p>
        ) : (
          userCourses.map((course) => (
            <CourseCardMini
              key={course.courseId}
              course={course}

progress={course.progressPercentage}
              onClick={() =>
handleCourseClick(course.courseId)}
            />
          ))
        )}
      </div>
    </div>
  );
};

export default LearningProgress;
```

components\LessonList.tsx

```
import React from "react";
import { useNavigate } from "react-
router-dom";
import { Lesson } from "../types";

interface LessonListProps {
  lessons: Lesson[];
  completedLessons: string[];
}

const LessonList:
React.FC<LessonListProps> = ({ lessons,
completedLessons }) => {
  const navigate = useNavigate();

  if (!lessons.length) {
    return <p>PJCЂPsPєC-PI PïPsPєPë
PSPµPjP°C”.</p>;
  }

  const handleLessonClick = (courseId:
string, lessonId: string) => {

navigate(`/courses/${courseId}/lessons/$
{lessonId}`);
  };

  return (
    <ul>
      {lessons.map((lesson) => (
        <li
```

```
          key={lesson.lessonId}
          style={{ marginBottom: "10px"
}}
        >
          <h3
            onClick={() =>
handleLessonClick(lesson.courseId,
lesson.lessonId)}
            style={{ cursor: "pointer",
display: "inline" }}
          >

{completedLessons.includes(lesson.lesson
Id) && <span>вњ… </span>}
            {lesson.title}
          </h3>
        </li>
      ))}
    </ul>
  );
};


export default LessonList;



components\LoginForm.tsx

//src/components/LoginForm.tsx


import { useState } from "react";
import { signInWithEmailAndPassword }
from "firebase/auth";
import {auth, db} from
"../services/firebaseConfig.ts";
import { useNavigate } from "react-
router-dom";
import {doc, updateDoc} from
"firebase/firestore"; // Р”Р»СЏ
СЂРµРґР°РєС‚РёСЂ


const LoginForm = () => {
  const [email, setEmail] =
useState("");
  const [password, setPassword] =
useState("");
  const [error, setError] =
useState<string>("");
  const navigate = useNavigate(); //
Р†PSC-С†С-Р°Р»С-Р·Р°С†С-СЏ useNavigate


  const handleSubmit = async (e:
React.FormEvent) => {
    e.preventDefault();
    try {
    const userCredential = await
signInWithEmailAndPassword(auth, email,
password);
    const user = userCredential.user;


    await user.reload();


    if (user.emailVerified) {
      await updateDoc(doc(db, "users",
user.uid), { isVerified: true }); //
РћPSPsPIP»РµPSPSCЏ Сґ Firestore
    } else {
      setError("Р'СґРґСЊ Р»Р°СЃРєР°,
РїС-РґС,PIPµСЂРґС-С,СЊ PIP°С€Сґ
РїPsС€С,Сґ РїРµСЂРµРґ PIC…PsРґPsPj.");
      return;
    }
      navigate("/");
    } catch (err) {
      const error = err as { code:
string; message: string }; //
РµРµСЂРµС,PIPsСЂРµPSPSCЏ С,РёРїСґ
РїPsPjРёР»РєРё
      if (error.code === "auth/invalid-
email") {

setError("РќРµРїСЂР°PIPёР»СЊPSPёР№
email.");
      } else if (error.code ===
"auth/wrong-password") {
```

```
setError("РќРµРїСЂР°РІРёР»СЊРЅС‹Р№ РїР°СЂРѕР»СЊ.");
      } else if (error.code === "auth/user-not-found") {
        setError("РљРѕСЂРёСЃС‚СЂР°С‡Р° СЃ С‚Р°РєРёРј email РЅРµ РЅР№РґРµРЅРѕ.");
      } else if (error.code === "auth/too-many-requests") {
        setError("Р—Р°Р±Р°Р»Рё СЃРїСЂРѕСЃРѕРІ. РЈРїСЂРѕСЃРёС‚Рµ РїСЂРѕСЃРјРѕС‚СЂ РїРѕР·Р¶Рµ.");
      } else if (error.code === "auth/invalid-credential") {
        setError("РџРѕРїСЂРѕР±СѓР№С‚Рµ Р°Р°РІС‚РѕСЂРёР·РѕРІР°С‚СЊСЃСЏ Р·Р°РЅРѕРІРѕ.");
      } else {
        setError("РџРѕРїСЂРѕР±СѓР№С‚Рµ РїРѕР·Р¶Рµ ");
      }
    }
  };

  return (
    <form onSubmit={handleSubmit} style={styles.form}>
      <label style={styles.label}>
        Email:
        <input
          type="email"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
          required
          style={styles.input}
        />
      </label>
      <label style={styles.label}>
        РџР°СЂРѕР»СЊ:
        <input
          type="password"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
          required
          style={styles.input}
        />
      </label>
      {error && <p style={styles.error}>{error}</p>}
      <button type="submit" style={styles.button}>
        РЈРІС…РѕРґ
      </button>
    </form>
  );
};

const styles: { [key: string]: React.CSSProperties } = {
  form: {
    display: "flex",
    flexDirection: "column",
    gap: "10px",
    padding: "20px",
    borderRadius: "8px",
    backgroundColor: "#f9f9f9",
    boxShadow: "0 4px 8px rgba(0, 0, 0, 0.1)",
    maxWidth: "400px",
    margin: "auto",
  },
  label: {
    fontSize: "16px",
    color: "#555",
  },
  input: {
```

```
    padding: "8px",
    fontSize: "16px",
    border: "1px solid #ccc",
    borderRadius: "5px",
    marginTop: "5px",
    width: "100%",
    boxSizing: "border-box",
  },
  error: {
    margin: 0,
    color: "red",
    fontSize: "14px",
    textAlign: "center",
  },
  button: {
    padding: "12px",
    fontSize: "16px",
    color: "#fff",
    backgroundColor: "#007BFF",
    border: "none",
    borderRadius: "5px",
    cursor: "pointer",
    transition: "background-color 0.3s",
    margin: " 5px 0",
  },
};


export default LoginForm;
```

components\LogoutButton.tsx

```
import { signOut } from "firebase/auth";
import { auth } from
"../services/firebaseConfig.ts";
import { useNavigate } from "react-
router-dom";


const LogoutButton = () => {
  const navigate = useNavigate();


  const handleLogout = async () => {
    const confirmLogout =
window.confirm("P’Pë PIPïPµPIPSPµPSC-,
C‰Ps C…PsC‡PµC,Pµ PIPëPNºC,Pë?");
    if (confirmLogout) {
      await signOut(auth);
      navigate("/");
    }
  };


  return (
    <button onClick={handleLogout}
style={styles.button}>
      P’PëPNºC,Pë
    </button>
  );
};


const styles: { [key: string]:
React.CSSProperties } = {
  button: {
    padding: "10px 20px",
    fontSize: "14px",
    backgroundColor: "#4CAF50",
    color: "#fff",
    border: "none",
    borderRadius: "5px",
    cursor: "pointer",
    transition: "background-color 0.3s",
  },
};


export default LogoutButton;
```

components\QuizComponent.tsx

```
//QuizComponent.tsx
```

```
import React, { useState } from 'react';
import { Quiz, Question} from
"../types";


interface QuizComponentProps {
  quiz: Quiz;
  selectedOption: { [key: string]:
string };
  setSelectedOption:
React.Dispatch<React.SetStateAction<{
[key: string]: string }>>;
  onSubmitQuiz: () => void;
}


const QuizComponent:
React.FC<QuizComponentProps> = ({ quiz,
selectedOption, setSelectedOption,
onSubmitQuiz }) => {
  const [isSubmitted, setIsSubmitted] =
useState(false);


  const handleOptionChange = (question:
Question, option: string) => {
    setSelectedOption((prev) => ({
      ...prev,
      [question.question]: option,
    }));
  };


  const renderAnswerFeedback =
(selected: string, correct: string) => {
    if (!isSubmitted) return null;
    if (!selected) return null;



    return selected === correct ? (
      <span style={{ color: 'green',
marginLeft: '10px' }}>вњ“</span>
    ) : (
      <span style={{ color: 'red',
marginLeft: '10px' }}>вњ— </span>
    );
  };


  const handleSubmit = () => {
    setIsSubmitted(true);
    onSubmitQuiz();
  };


  const handleReset = () => {
    setIsSubmitted(false);
    setSelectedOption({});
  };


  return (
    <div style={styles.quizContainer}>
      <h2>{quiz.title}</h2>
      {quiz.questions.map((q, index) =>
(
        <div key={index}
style={styles.questionBlock}>
          <div
style={styles.questionHeader}>
            <p
style={styles.questionText}>{q.question}
</p>
            {/* P’C–PґPsP±CЂP°P¶P°C”PjPs
PїPsP·PSP°C‡PєCґ PїPsCЂCґC‡ P·
PїPёC,P°PSPSCЏPj */}

{renderAnswerFeedback(selectedOption[q.q
uestion], q.correct)}
          </div>
          <ul>
            {q.options.map((option, i)
=> (
              <li key={i}>
                <input
                  type="radio"
```

```
                  name={q.question}
                  value={option}
checked={selectedOption[q.question] ===
option}
                  onChange={() =>
handleOptionChange(q, option)}
                  disabled={isSubmitted}
                />
                {option}
              </li>
            ))}
          </ul>
        </div>
      ))}
      <div>
        <button
style={styles.completeButton}
onClick={handleSubmit}>
          РµС–РґС‚РІРёС‚Рё
        </button>
        {isSubmitted && (
          <button
style={styles.resetButton}
onClick={handleReset}>
            РЎРєРёРЅСѓС‚Рё С‚РµСЃС‚
          </button>
        )}
      </div>
    </div>
  );
};

const styles: Record<string,
React.CSSProperties> = {
  quizContainer: {
    padding: '20px',
    backgroundColor: '#ffffff',
    borderRadius: '8px',
  },
  questionBlock: {
    marginBottom: '20px',
  },
  questionHeader: {
    display: 'flex',
    alignItems: 'center',
  },
  questionText: {
    marginRight: '10px',
  },
  completeButton: {
    padding: '12px 20px',
    fontSize: '16px',
    color: '#fff',
    backgroundColor: '#4caf50',
    border: 'none',
    borderRadius: '5px',
    cursor: 'pointer',
    marginTop: '20px',
    transition: 'background 0.3s',
  },
  resetButton: {
    padding: '12px 20px',
    fontSize: '16px',
    color: '#fff',
    backgroundColor:
'rgba(145,205,23,0.68)',
    border: 'none',
    borderRadius: '5px',
    cursor: 'pointer',
    marginTop: '20px',
    marginLeft: '10px',
    transition: 'background 0.3s',
  },
};
```

```
export default QuizComponent;
```

components\QuizList.tsx

```
import React from "react";
import { useNavigate } from "react-
router-dom";
import { Quiz } from "../types";

interface QuizListProps {
  quizzes: Quiz[];
  completedQuizzes: string[];
}

const QuizList: React.FC<QuizListProps>
= ({ quizzes, completedQuizzes }) => {
  const navigate = useNavigate();

  if (!quizzes.length) {
    return <p>РўРµСЃС‚С–РІ РїРѕРєРё
РЅРµРјР°С”.</p>;
  }

  const handleQuizClick = (courseId:
string | undefined, quizId: string) => {
    if (!courseId) {
      console.error("РќРµРјР¶РѕР»РёРІРѕ
РІС–РґРєСЂРёС‚Рё С‚РµСЃС‚: РІС–
РґСЃСѓС‚РЅС”PSC–P№ courseId");
      return;
    }

navigate(`/courses/${courseId}/quizzes/$
{quizId}`);
  };

  return (
    <ul>
      {quizzes.map((quiz) => (
        <li
          key={quiz.quizId}
          style={{ marginBottom: "10px"
}}
        >
          <h3
            onClick={() =>
handleQuizClick(quiz.courseId,
quiz.quizId)}
            style={{ cursor: "pointer",
display: "inline" }} // СЂРѕР±РёРјРѕ
С‚РµСЃС‚, РєР»С–РєР°Р±РµР»СЊРЅРёРј
          >

{completedQuizzes.includes(quiz.quizId)
&& <span> вњ… </span>}{quiz.title}
          </h3>

        </li>
      ))}
    </ul>
  );
};

export default QuizList;
```

components\RegisterForm.tsx

```
import { useState } from "react";
import { createUserWithEmailAndPassword,
sendEmailVerification,
fetchSignInMethodsForEmail  } from
"firebase/auth";
import { auth } from
"../services/firebaseConfig";
import { useNavigate } from "react-
router-dom";
import { addUserToFirestore } from
"../services/usersService";
```

```
const RegisterForm = () => {

  const [email, setEmail] =
useState("");

  const [password, setPassword] =
useState("");

  const [role, setRole] =
useState<"CґC‡PµPSCЊ" |
"PIPëPєP»P°PґP°C‡">("CґC‡PµPSCЊ");

  const [error, setError] =
useState<string>("");

  const [isModalOpen, setIsModalOpen] =
useState(false); // PЎC,P°PS
PjPsPґP°P»CЊPSPsPiPs PIC-PєPSP°

  const navigate = useNavigate();



  const handleSubmit = async (e:
React.FormEvent) => {

    e.preventDefault();

    setError("");



    try {

      const signInMethods = await
fetchSignInMethodsForEmail(auth, email);

      if (signInMethods.length > 0) {

        setError("P¦PµP№ email PIP¶Pµ
PIPëPєPsCЊPëCЃC,PsPICґC”C,CЊCЃCЏ.
PЎPїCЊPsP±CґP№C,Pµ CґPIC-P№C,Pë P°P±Ps
PIPëPєPsCЊPëCЃC,P°P№C,Pµ C-PSC€PëP№.");

        return;

      }

      const userCredential = await
createUserWithEmailAndPassword(auth,
email, password);

      const user = userCredential.user;



      if (user) {

        await
sendEmailVerification(user); //
PќP°PґCЃPëP»P°C”PjPs email PґP»CЏ PїC-
PґC,PIPµCЊPґP¶PµPSPSCЏ

        await addUserToFirestore(user,
role);

        setIsModalOpen(true); // P’C-
PґPєCЊPëPIP°C”PjPs PjPsPґP°P»CЊPSPµ PIC-
PєPSPs

      }

    } catch (err) {

      const error = err as { code:
string; message: string };



      if (error.code === "auth/invalid-
email") {


setError("PќPµPїCЊP°PIPëP»CЊPSPëP№
email.");

      } else if (error.code ===
"auth/email-already-in-use") {

        setError("P¦PµP№ email PIP¶Pµ
PIPëPєPsCЊPëCЃC,PsPICґC”C,CЊCЃCЏ.");

      } else if (error.code ===
"auth/weak-password") {

        setError("PµP°CЊPsP»CЊ PjP°C”
PjC-CЃC,PëC,Pë C‰PsPSP°P№PjPµPSC€Pµ 6
CЃPëPjPIPsP»C-PI.");

      } else {

        setError("PµPsPjPëP»PєP°
CЊPµC”CЃC,CЊP°C†C-C—: " +
error.message);

      }

    }

  };



  return (

    <>

      <form onSubmit={handleSubmit}
style={styles.form}>

        <label style={styles.label}>

          Email:

          <input

            type="email"

            value={email}

            onChange={(e) =>
setEmail(e.target.value)}

            required
```

```
            style={styles.input}
          />
        </label>
        <label style={styles.label}>
          РµР°СЂРѕР»СЊ:
          <input
            type="password"
            value={password}
            onChange={(e) =>
setPassword(e.target.value)}
            required
            style={styles.input}
          />
        </label>
        <label style={styles.label}>
          Р РѕР»СЊ:
          <select value={role}
onChange={(e) => setRole(e.target.value
as "СѓС‡РµРЅРёРє" | "РІРёРєР»Р°РґР°С‚")}
style={styles.input}>
            <option
value="СѓС‡РµРЅРёРє">РЈС‡РµРЅРёРє</option>
            <option
value="РІРёРєР»Р°РґР°С‚">Р’РёРєР»Р°РґР°С
‡</option>
          </select>
        </label>
        {error && <p
style={styles.error}>{error}</p>}
        <button type="submit"
style={styles.button}>
          Р—Р°СЂРµС”СЃС‚СЂСѓРІР°С‚РёСЃСЏ
        </button>
      </form>

      {isModalOpen && (
        <div
style={styles.modalOverlay}>
          <div style={styles.modal}>
            <p>РќР° РІР°С€ РµР»РµC‚СЂРѕРЅРЅРёР№ РїРѕС€С‚Р°
СЃР»Р°РЅРѕ Р»РёСЃС‚, РґР»СЏ РїС–
РґС‚РІРµСЂРґР¶РµРЅРЅСЏ. РµРјСЂРµРІС–
СЂС‚Рµ email РїРµСЂРµРґ
РІС…РѕРґРѕРј.</p>
            <button onClick={() =>
navigate("/login")}
style={styles.button}>
              Р”РѕР±СЂРµ
            </button>
          </div>
        </div>
      )}
    </>
  );
};

const styles: { [key: string]:
React.CSSProperties } = {
  form: {
    display: "flex",
    flexDirection: "column",
    gap: "20px",
    padding: "20px",
    borderRadius: "8px",
    backgroundColor: "#f9f9f9",
    boxShadow: "0 4px 8px rgba(0, 0, 0,
0.1)",
    maxWidth: "400px",
    margin: "auto",
  },
  label: {
    fontSize: "16px",
    color: "#555",
  },
  input: {
    padding: "10px",
    fontSize: "16px",
    border: "1px solid #ccc",
```

```
    borderRadius: "5px",
    marginTop: "5px",
    width: "100%",
    boxSizing: "border-box",
  },
  error: {
    color: "red",
    fontSize: "14px",
    textAlign: "center",
  },
  button: {
    padding: "12px",
    fontSize: "16px",
    color: "#fff",
    backgroundColor: "#007BFF",
    border: "none",
    borderRadius: "5px",
    cursor: "pointer",
    transition: "background-color 0.3s",
    margin: "5px 0",
  },
  modalOverlay: {
    position: "fixed",
    top: 0,
    left: 0,
    width: "100%",
    height: "100%",
    backgroundColor: "rgba(0, 0, 0,
0.5)",
    display: "flex",
    alignItems: "center",
    justifyContent: "center",
  },
  modal: {
    backgroundColor: "#fff",
    padding: "20px",
    borderRadius: "8px",
    boxShadow: "0 4px 8px rgba(0, 0, 0,
0.1)",
    textAlign: "center",
    maxWidth: "400px",
  },
};


export default RegisterForm;
```

components\UserInfo.tsx

```
import { useState, useEffect } from
"react";
import { auth } from
"../services/firebaseConfig";
import { updateProfile } from
"firebase/auth";
import { updateDoc, doc, getDoc } from
"firebase/firestore";
import { db } from
"../services/firebaseConfig";


const avatars = [
  "/avatars/user.png",
  "/avatars/fiery_book.png",
  "/avatars/fly.png",
  "/avatars/oasis.png",
  "/avatars/delfin.png",
  "/avatars/apple.png",
];


const UserInfo = ({ user }: { user: any
}) => {
  const [displayName, setDisplayName] =
useState(user?.displayName || "");
  const [photoURL, setPhotoURL] =
useState(user?.photoURL || avatars[0]);
```

```
  const [role, setRole] =
useState<string>("");


  useEffect(() => {
    const fetchUserRole = async () => {
      if (auth.currentUser) {
        const userRef = doc(db, "users",
auth.currentUser.uid);
        const userSnap = await
getDoc(userRef);


        if (userSnap.exists()) {
          const userData =
userSnap.data();
          setRole(userData.role || "РќРµ
РІРєР°Р·Р°РЅР°");
        }
        if (!photoURL && avatars.length
> 0) {
          setPhotoURL(avatars[0]);
        }
      }
    };
    fetchUserRole();
  }, []);



  // Р—Р±РµСЂС–Р°С”РјРs РєРsРїС–С—
РїРsС‡Р°С,РєРsРІРёС… Р·РSР°С‡РµРSСЊ
РґР»СЏ СЃРєР°СЃСѓРІР°РSРSСЏ

  const [initialDisplayName,
setInitialDisplayName] =
useState(user?.displayName || "");
  const [initialPhotoURL,
setInitialPhotoURL] =
useState(user?.photoURL || avatars[0]);


  const [isEditing, setIsEditing] =
useState(false);
  const [error, setError] =
useState<string>("");

  const handleSave = async () => {
    try {
      if (auth.currentUser) {
        await
updateProfile(auth.currentUser, {
displayName, photoURL });


        const userRef = doc(db, "users",
auth.currentUser.uid);
        await updateDoc(userRef, {
displayName, photoURL });


        setIsEditing(false);
        // Р—Р±РµСЂС–Р°С”РjPs PSPsPIC–
РїРsС‡Р°С,РєРsPIC– Р·PSP°С‡РµPSPSCЏ РїС–
СЃР»СЏ Р·Р±РµСЂРµР¶РµPSPSCЏ

setInitialDisplayName(displayName);
        setInitialPhotoURL(photoURL);
      }
    } catch (err) {
      console.error("Error updating
profile:", err);
      setError("РќРµ РІРґР°PsСЃСЏ
PsPSPsPIPёС,Рё РїСЂРsС„С–Р»СЊ.");
    }
  };


  const handleCancel = () => {
    setDisplayName(initialDisplayName);
// Р’С–Р´PSPsPIP»СЊС”PjPs
РїРsС‡Р°С,РєРsPIC– Р·PSP°С‡РµPSPSCЏ

    setPhotoURL(initialPhotoURL);
// Р’С–Р´PSPsPIP»СЊС”PjPs
РїРsС‡Р°С,РєРsPIC– Р°PIP°С,Р°СЊРєРё

    setIsEditing(false);
  };


  return (
    <div style={styles.container}>
      <h2>Р’Р°С€ РїСЂРsС„С–Р»СЊ</h2>
```

```
      <img src={photoURL || avatars[0]}
alt="User Avatar" style={styles.avatar}
/>

      {isEditing && (

        <div>

          <label>P'PëP±PµCЪC-C,CЊ
P°PIP°C,P°CЪ:</label>

          <div
style={styles.avatarSelection}>

            {avatars.map((avatar, index)
=> (

              <img

                key={index}

                src={avatar}

                alt={`Avatar ${index +
1}`}


style={styles.avatarOption}

                onClick={() =>
setPhotoURL(avatar)}

              />

            ))}

          </div>

        </div>

      )}

      <div style={styles.field}>

        <label>P†Pj'CЦ:</label>

        {isEditing ? (

          <input

            type="text"

            value={displayName}

            onChange={(e) =>
setDisplayName(e.target.value)}

          />

        ) : (

          <p>{displayName}</p>

        )}

      </div>


      <div style={styles.field}>

        <label>P•PjPµP№P»:</label>

        <p>{auth.currentUser?.email ||
"PќPµ PIPєP°P·P°PSPs"}</p>

      </div>


      <div style={styles.field}>

        <label>PЎC,P°C,CґCЃ
(CЪPsP»CЊ):</label>

        <p>{role}</p>

      </div>


      {error && <p
style={styles.error}>{error}</p>}


      {isEditing ? (

        <>

          <button
onClick={handleCancel}>PЎPєP°CЃCґPIP°C,P
ë</button>

          <button
onClick={handleSave}>P—
P±PµCЪPµPiC,Pë</button>

        </>

      ) : (

        <button onClick={() =>
setIsEditing(true)}>P PµPrP°PiCґPIP°C,Pë
PïCЪPsC„C-P»CЊ</button>

      )}

    </div>

  );

};


const styles: { [key: string]:
React.CSSProperties } = {

  container: {

    padding: "20px",

    border: "1px solid #ddd",

    borderRadius: "8px",

    marginBottom: "20px",

  },
```

```
  avatar: {
    width: "150px",
    height: "150px",
    borderRadius: "50%",
    objectFit: "cover",
    marginBottom: "20px",
  },
  avatarSelection: {
    display: "flex",
    gap: "10px",
    marginTop: "10px",
  },
  avatarOption: {
    width: "50px",
    height: "50px",
    borderRadius: "50%",
    cursor: "pointer",
    border: "2px solid transparent",
    transition: "border-color 0.2s",
  },
  field: {
    marginBottom: "10px",
  },
  error: {
    color: "red",
    fontSize: "14px",
  },
};

export default UserInfo;
```

hooks\useAuth.ts

```
// src/hooks/useAuth.ts
import { useEffect, useState } from "react";
import { onAuthStateChanged, User } from "firebase/auth";
import { auth } from "../services/firebaseConfig";

// РЃСѓРє РґР»СЏ Р°СѓС‚РµРЅС‚РёС„РёРєР°С†РёРё
export const useAuth = () => {
  const [user, setUser] = useState<User | null>(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const unsubscribe = onAuthStateChanged(auth, (firebaseUser) => {
      setUser(firebaseUser);
      setLoading(false);
    });

    return () => unsubscribe();
  }, []);

  return { user, loading };
};
```

pages\CoursePage.tsx

```
import React, {useEffect, useState} from "react";
import {useParams} from "react-router-dom";
import {getCourseById} from "../services/coursesService";
import {getLessonsByCourseId} from "../services/lessonsService";
import {getQuizzesByCourseId} from "../services/quizzesService";
```

```
import {getUserProgress} from
"../services/progressService";

import LessonList from
"../components/LessonList";

import QuizList from
"../components/QuizList";

import CourseDetails from
"../components/CourseDetails";

import {auth} from
"../services/firebaseConfig";

import {Course, Lesson, Quiz} from
"../types";


const CoursePage: React.FC = () => {

    const {courseId} = useParams<{
courseId: string }>();

    const [course, setCourse] =
useState<Course | null>(null);

    const [lessons, setLessons] =
useState<Lesson[]>([]);

    const [quizzes, setQuizzes] =
useState<Quiz[]>([]);

    const [userProgress,
setUserProgress] = useState({

        lessonsCompleted: [] as
string[],

        testsPassed: [] as string[],

        progressPercentage: 0,

    });

    const [loading, setLoading] =
useState<boolean>(true);

    const [error, setError] =
useState<string | null>(null);

    const user = auth.currentUser;


    useEffect(() => {

        const loadCourseData = async ()
=> {

            if (!courseId) {

                setError("РќРµIC-
СЂPSPёР№ С-РґРµPSC,РёС„С-РєР°С,PsСЂ
РєСѓСЂСЃС‹.");

                setLoading(false);

                return;

            }


            try {

                const [courseData,
lessonsData, quizzesData] = await
Promise.all([


getCourseById(courseId),


getLessonsByCourseId(courseId),


getQuizzesByCourseId(courseId),

                ]);


                if (!courseData) {

                    setError("РљСѓСЂСЃ
PSPµ Р·PSP°Р№РґРµPSPs.");

                    setLoading(false);

                    return;

                }


                setCourse(courseData);

                setLessons(lessonsData
|| []);

                setQuizzes(quizzesData
|| []);


                if (user) {

                    const progress =
await getUserProgress(user.uid,
courseId);


setUserProgress(progress || {


lessonsCompleted: [],

                        testsPassed: [],


progressPercentage: 0,

                    });

                }
```

```
            } catch (err) {

console.error("РџPsPjPёP»PєP° PїC-Pґ
C‡P°CЃ P·P°PIP°PSC,P°P¶PµPSPSCЏ
PґP°PSPёC…:", err);

                setError("PќPµ
PIPґP°P»PsCЃCЏ P·P°PIP°PSC,P°P¶PёC,Pё
PґP°PSC- PєCѓCЂCЃCѓ.");

            } finally {

                setLoading(false);

            }

        };


        loadCourseData();

    }, [courseId, user]);


    if (loading) return <p>P—
P°PIP°PSC,P°P¶PµPSPSCЏ
PєCѓCЂCЃCѓ...</p>;

    if (error) return <p>{error}</p>;


    const finalQuiz =
quizzes.find((quiz) => quiz.quizId ===
course?.quiz_id);


    return (

        <div
style={styles.pageContainer}>

            {course ? (

                <>

                    <CourseDetails

                        course={course}


lessons={lessons}


quizzes={quizzes}


ProgressPercentage={userProgress.progres
sPercentage || 0}

                    />


                    <h2>PJCЂPsPєPё</h2>

                    <LessonList


lessons={lessons}


completedLessons={userProgress.lessonsCo
mpleted}

                    />


                    {finalQuiz && (

                        <>

                            <h2>P¤C-
PSP°P»CЊPSPёP№ C,PµCЃC,</h2>


{userProgress.lessonsCompleted.length
=== lessons.length ? (


<QuizList


quizzes={[finalQuiz]}


completedQuizzes={userProgress.testsPass
ed}

                                />

                            ) : (


<p>PµCЂPsP№PґC-C,CЊ CѓCЃC- CѓCЂPsPєPё,
C‰PsP± PsC,CЂPёPjP°C,Pё PґPsCЃC,CѓPї
PґPs C„C-PSP°P»CЊPSPsPiPs
C,PµCЃC,Cѓ.</p>

                            )}

                        </>

                    )}


                </>

            ) : (

                <p>PЊCѓCЂCЃ PSPµ
P·PSP°P№PґPµPSPs.</p>

            )}

        </div>

    );

};
```

```
const styles = {
    pageContainer: {padding: "20px"},
};

export default CoursePage;
```

pages\CoursesPage.tsx

```
// src/components/CoursesPage.tsx
import {CSSProperties, useEffect,
useState} from "react";
import {getCourses} from
"../services/coursesService";
import CourseList from
"../components/CourseList";
import {Course} from "../types.ts";

const CoursesPage = () => {
    const [searchTerm, setSearchTerm] =
useState("");
    const [selectedCategory,
setSelectedCategory] =
useState<string>("");
    const [categories, setCategories] =
useState<string[]>([]);

    useEffect(() => {
        const loadCategories = async ()
=> {
            try {
                const courses: Course[]
= await getCourses();
                // P'PëC,CЏPiCґC"PjPs
PєP°C,PµPiPsCЂC-C— P· PєPsP¶PSPsPiPs
PєCґCЂCЃCґ
                const allCategories =
courses.flatMap(course =>
course.category.map((cat: string) =>
cat.trim()));
                //
P'PëPєPsCЂPëCЃC,PsPICґC"PjPs Set PґP»CЏ
PIPëPґP°P»PµPSPSCЏ PґCґP±P»C-PєP°C,C-PI
C- PєPsPSPIPµCЂC,CґC"PjPs PI PjP°CЃPëPI
                const uniqueCategories =
Array.from(new Set(allCategories));

setCategories(uniqueCategories);
            } catch (error) {
                console.error("PќPµ
PIPґP°P»PsCЃCЏ P·P°PIP°PSC,P°P¶PëC,Pë
PєP°C,PµPiPsCЂC-C—", error);
            }
        };

        loadCategories();
    }, []);

    const handleSearchChange = (event:
React.ChangeEvent<HTMLInputElement>) =>
{
setSearchTerm(event.target.value);
    };

    const handleCategoryChange = (event:
React.ChangeEvent<HTMLSelectElement>) =>
{
setSelectedCategory(event.target.value);
    };

    return (<div
style={styles.pageContainer}>
            <h1>PЉCґCЂCЃPë</h1>

            <div
style={styles.filtersContainer}>
                <input
                    type="text"
placeholder="PµPsC€CґPє PєCґCЂCЃCґ..."
```

```
                    value={searchTerm}
onChange={handleSearchChange}
style={styles.searchInput}
                />

                <select
value={selectedCategory}
onChange={handleCategoryChange}
style={styles.select}
                >
                    <option
value="">Р’СЃС– РєР°С‚РµРіРѕСЂС–С—
</option>
{categories.map((category) => (<option
key={category} value={category}>
                            {category}
                        </option>))}
                </select>
            </div>

            <CourseList
searchTerm={searchTerm}
selectedCategory={selectedCategory}/>
        </div>);
 };

    const styles: { [key: string]:
CSSProperties } = {
        pageContainer: {
            display: "flex",
            flexDirection: "column",
            padding: "20px",
            boxSizing: "border-box",
            margin: "0 20px",
            backgroundColor: "#fff",
        },

        filtersContainer: {
          display: "flex",
          flexDirection: "column",
          gap: "12px",
          marginBottom: "20px",
          alignItems: "flex-start",
        },

        searchInput: {
          padding: "8px 12px",
          fontSize: "16px",
          borderRadius: "4px",
          border: "1px solid #ccc",
          width: "100%",
          maxWidth: "400px",
          minWidth: "200px",
          boxSizing: "border-box",
        },
select: {
  padding: "8px 12px",
  fontSize: "16px",
  borderRadius: "4px",
  border: "1px solid #ccc",
  width: "100%",
  maxWidth: "400px",
  boxSizing: "border-box",
},

    };

    export default CoursesPage;
```

pages\Home.tsx

```
import {CSSProperties, useEffect,
useState} from "react";

import { Link } from "react-router-dom";

import {useAuth} from
"../hooks/useAuth.ts";


const Home = () => {

  const { user } = useAuth();

  const [isMobile, setIsMobile] =
useState(window.innerWidth <= 768);


  useEffect(() => {

    const handleResize = () =>
setIsMobile(window.innerWidth <= 768);

    window.addEventListener("resize",
handleResize);

    return () =>
window.removeEventListener("resize",
handleResize);

  }, []);


  const styles: { [key: string]:
CSSProperties } = {

    container: {

      fontFamily: "'Segoe UI', Tahoma,
Geneva, Verdana, sans-serif",

      color: "#1f2937",

      margin: "16px 0 0 0",

      padding: 0,

      backgroundColor: "#ffffff",

    },

    hero: {

      backgroundColor: "#d1fae5",

      padding: isMobile ? "40px 16px" :
"60px 20px",

      textAlign: "center",

    },

    heroTitle: {

      fontSize: isMobile ? "32px" :
"48px",

      fontWeight: "bold",

      color: "#1e3a8a",

      marginBottom: "20px",

    },

    heroText: {

      fontSize: isMobile ? "16px" :
"18px",

      color: "#1c4e80",

      marginBottom: "32px",

      maxWidth: "600px",

      marginLeft: "auto",

      marginRight: "auto",

    },

    buttonGroup: {

      display: "flex",

      justifyContent: "center",

      gap: "16px",

      flexWrap: "wrap",

    },

    primaryButton: {

      backgroundColor: "#1d4ed8",

      color: "#fff",

      fontWeight: "600",

      padding: "12px 24px",

      borderRadius: "6px",

      textDecoration: "none",

    },

    secondaryButton: {

      backgroundColor: "#fff",

      border: "2px solid #1d4ed8",

      color: "#1d4ed8",

      fontWeight: "600",

      padding: "12px 24px",

      borderRadius: "6px",

      textDecoration: "none",

    },
```

```
    section: {
      padding: "20px 20px 40px 20px",
      backgroundColor: "#f9fafb",
    },
    sectionTitle: {
      textAlign: "center",
      fontSize: isMobile ? "24px" :
"32px",
      fontWeight: "bold",
      marginBottom: "16px",
    },
    sectionText: {
      textAlign: "center",
      color: "#6b7280",
      marginBottom: "40px",
      fontSize: isMobile ? "14px" :
"16px",
    },
    cardContainer: {
      display: "grid",
      gridTemplateColumns: "1fr",
      gap: "24px",
      maxWidth: "900px",
      margin: "0 auto",
    },
    card: {
      backgroundColor: "#fff",
      padding: "24px",
      borderRadius: "8px",
      boxShadow: "0 2px 8px
rgba(0,0,0,0.05)",
      textAlign: "left",
    },
    cardTitle: {
      fontSize: "20px",
      fontWeight: "600",
      color: "#1d4ed8",
      marginBottom: "12px",
    },
    cardText: {
      color: "#4b5563",
      fontSize: isMobile ? "14px" :
"16px",
    },
    link: {
      color: "#2563eb",
      textDecoration: "none",
      fontWeight: "500",
      marginTop: "12px",
      display: "inline-block",
    },
    benefitGrid: {
      display: "grid",
      gridTemplateColumns: isMobile ?
"1fr" : "repeat(2, 1fr)",
      gap: "24px",
      maxWidth: "800px",
      margin: "0 auto",
    },
    roleCardWrap: {
      display: "flex",
      flexDirection: isMobile ? "column"
: "row",
      gap: "24px",
      justifyContent: "center",
      flexWrap: "wrap",
    }
  };

  return (
    <div style={styles.container}>
      <section style={styles.hero}>
        <h1
style={styles.heroTitle}>PћPSP»P°PЊPS
```

```
PSP°PIC‡P°PSPSCЦ PґP»CЦ
PєPsP¶PSPsPiPs</h1>

        <p style={styles.heroText}>

          PµP»P°C,C„PsCЂPjP° PґP»CЦ
PIPёPIC‡PµPSPSCЦ PїCЂP°PIPёP»
PїPsPIPµPґC–PSPєPё Cґ PєCЂPёC,PёC‡PSPёC…
CЃPёC,CґP°C†C–CЦC…. PµCЂP°PєC,PёC‡PSC–
CґCЂPsPєPё, C,PµCЃC,Pё C,P°
PjPsP¶P»PёPIC–CЃC,CЊ
CЃC,PIPsCЂCЂPIP°C,Pё PIP»P°CЃPSC–
PєCґCЂCЃPё.

        </p>

        <div style={styles.buttonGroup}>

          <Link to="/courses"
style={styles.primaryButton}>PµPsC‡P°C,P
ё PSP°PIC‡P°PSPSCЦ</Link>

          {!user && <Link to="/login"
style={styles.secondaryButton}>PJPIC–
P№C,Pё</Link>}

        </div>

      </section>



      {!user && (

        <section style={styles.section}>

          <h2
style={styles.sectionTitle}>PЋP±PµCЂPё
CЃPIPsCЋ CЂPsP»CЊ</h2>

          <p
style={styles.sectionText}>PќP°PIC‡P°P№C
ЃCЦ P°P±Ps PSP°PIC‡P°P№ C–PSC€PёC…</p>

          <div
style={styles.roleCardWrap}>

            <Link to="/login" style={{
textDecoration: "none", flex: "1 1
300px", maxWidth: "400px" }}>

              <div
style={{...styles.card, textAlign:
"center", cursor: "pointer"}}>

                <h3
style={styles.cardTitle}>PЇ
CґC‡PµPSCЊ/CґC‡PµPSPёC†CЦ</h3>

                <p
style={styles.cardText}>

                  PµCЂPsC…PsPґCЊ
PєCґCЂCЃPё, C,PµCЃC,Pё C,P°
P·PґPsP±CґPIP°P№ P·PSP°PSPSCЦ Cґ
P·CЂCґC‡PSPsPjCґ C„PsCЂPjP°C,C– <br/>
PJPIC–P№PґPё, C‰PsP±
PIC–PґCЃC,PµP¶CґPIP°C,Pё PїCЂPsPiCЂPµCЃ!

                </p>

                <span
style={styles.link}>PµPµCЂPµP№C,Pё PґPs
PIC…PsPґCґ в†’</span>

              </div>

            </Link>

            <Link to="/login" style={{
textDecoration: "none", flex: "1 1
300px", maxWidth: "400px" }}>

              <div
style={{...styles.card, textAlign:
"center", cursor: "pointer"}}>

                <h3
style={styles.cardTitle}>PЇ
PIPёPєP»P°PґP°C‡/PIPёPєP»P°PґP°C‡PєP°</h
3>

                <p
style={styles.cardText}>

                  PЎC,PIPsCЂCЂP№
PSP°PIC‡P°P»CЊPSC– PєCґCЂCЃPё C,P°
PґPsPїPsPjP°PiP°P№ C–PSC€PёPj
P·PґPsP±CґPIP°C,Pё P·PSP°PSPSCЦ.

                </p>

                <span
style={styles.link}>PµPµCЂPµP№C,Pё PґPs
PIC…PsPґCґ в†’</span>

              </div>

            </Link>

          </div>

        </section>

      )}



      <section
style={{...styles.section,
backgroundColor: "#fff"}}>

        <h2
style={styles.sectionTitle}>P§PsPjCґ
CЃP°PjPµ PjPё?</h2>

        <div style={styles.benefitGrid}>

          <div style={styles.card}>

            <h4
style={styles.cardTitle}>P‘PµP·PєPsC€C,P
sPIPSPs</h4>
```

```
            <p
style={styles.cardText}>РќР°РІС‡Р°РЅРЅСЏ
РґРѕСЃС‚СѓРїРЅРµ РІСЃС–Рј
Р±РµР·РєРѕС€С‚РѕРІРЅРѕ С‚Р° Р±РµР·
СЂРµС”СЃС‚СЂР°С†С–С—.</p>

          </div>

          <div style={{ backgroundColor:
"#e0f2fe", borderRadius: "8px", height:
"100%" }}></div>

          <div style={{ backgroundColor:
"#d1fae5", borderRadius: "8px", height:
"100%" }}></div>

          <div style={styles.card}>

            <h4
style={styles.cardTitle}>Р†РЅС‚РµСЂР°РєС
,РёРІРЅС–СЃС‚СЊ</h4>

            <p
style={styles.cardText}>РњР°С‚РµСЂС–
Р°Р»Рё РјС–СЃС‚СЏС‚СЊ С‚РµСЃС‚Рё,
РІРїСЂР°РІРё С‚Р° СЃС†РµРЅР°СЂС–С—
СЂРµР°Р»СЊРЅРёС… СЃРёС‚СѓР°С†С–Р№.</p>

          </div>

          <div style={styles.card}>

            <h4
style={styles.cardTitle}>Р РѕР·РІРёС‚РѕР
є РІРёРєР»Р°РґР°С‡С–РІ</h4>

            <p
style={styles.cardText}>РњРѕР¶Р»РёРІС–
СЃС‚СЊ СЃС‚РІРѕСЂСЋРІР°С‚Рё РІР»Р°СЃРЅС–
РєСѓСЂСЃРё С‚Р° РїРѕС€РёСЂСЋРІР°С‚Рё
Р·РЅР°РЅРЅСЏ.</p>

          </div>

          <div style={{ backgroundColor:
"#fef9c3", borderRadius: "8px", height:
"100%" }}></div>

        </div>

      </section>

    </div>

  );

};


export default Home;
```

pages\LessonPage.tsx

```
import React, {useEffect, useState} from
'react';

import {useNavigate, useParams} from
'react-router-dom';

import {getLessonById,
getLessonsByCourseId} from
'../services/lessonsService';

import {getQuizById} from
'../services/quizzesService';

import {updateUserProgress} from
'../services/progressService';

import {auth} from
'../services/firebaseConfig';

import QuizComponent from
'../components/QuizComponent';

import BackToCourseButton from
"../components/BackToCourseButton";

import {Lesson, Quiz} from "../types";


const LessonPage: React.FC = () => {

    const {courseId, lessonId} =
useParams<{ courseId: string; lessonId:
string }>();

    const [lesson, setLesson] =
useState<Lesson | null>(null);

    const [lessons, setLessons] =
useState<Lesson[]>([]);

    const [quiz, setQuiz] =
useState<Quiz | null>(null);

    const [loading, setLoading] =
useState(true);

    const [error, setError] =
useState<string | null>(null);

    const [selectedOption,
setSelectedOption] = useState<{ [key:
string]: string }>({});

    const [quizCompleted,
setQuizCompleted] = useState(false);


    const navigate = useNavigate();

    const userId =
auth.currentUser?.uid;


    useEffect(() => {
```

```
        const loadLessonData = async ()
=> {

            if (!courseId || !lessonId)
{

                setError('P’C-
PrCЃCґC,PSC- PSPµPsP±C…C-PrPSC- PrP°PSC-
.');

                setLoading(false);

                return;

            }


            setLoading(true);

            setError(null);

            setQuiz(null);

            setSelectedOption({});

            setQuizCompleted(false);


            try {

                const lessonData = await
getLessonById(lessonId);

                const lessonsData =
await getLessonsByCourseId(courseId);

                if (!lessonData) {

                    setError('PJCЂPsPє
PSPµ P·PSP°PNєPrPµPSPs.');

                    setLoading(false);

                    return;

                }

                setLesson(lessonData);

                setLessons(lessonsData);

                if (lessonData.quizId) {

                    const quizData =
await getQuizById(lessonData.quizId);

                    setQuiz(quizData);

                }

            } catch (err) {


console.error('PµPsPjPëP»PєP°
P·P°PIP°PSC,P°P¶PµPSPSCЏ CґCЂPsPєCґ
P°P±Ps C,PµCЃC,Cґ:', err);

                setError('PќPµ
PIPrP°P»PsCЃCЏ P·P°PIP°PSC,P°P¶PëC,Pë
PrP°PSC-.');

            } finally {

                setLoading(false);

            }

        };


        loadLessonData();

    }, [courseId, lessonId]);


    const getYoutubeEmbedUrl = (url:
string): string => {

        const match =
url.match(/(?:https?:\/\/)?(?:www\.)?you
tube\.com\/watch\?v=([^&]+)/);

        if (match && match[1]) {

            return
`https://www.youtube.com/embed/${match[1
]}`;

        }

        const shortMatch =
url.match(/(?:https?:\/\/)?youtu\.be\/([
^?&]+)/);

        if (shortMatch && shortMatch[1])
{

            return
`https://www.youtube.com/embed/${shortMa
tch[1]}`;

        }

        return url;

    };


    const handleCompleteLesson = async
() => {

        if (!userId || !courseId ||
!lessonId) {


console.error("PµPsPjPëP»PєP°: P’C-
PrCЃCґC,PSC- PSPµPsP±C…C-PrPSC- PrP°PSC-
PrP»CЏ PsPSPsPIP»PµPSPSCЏ
PїCЂPsPiCЂPµCЃCґ.");

            return;
```

```
        }


        try {
            if (quiz && !quizCompleted)
{
                alert('Р—Р°РІРµСЂС€С-
С,СЊ С,РµСЃС‚, РїРµСЂРµР·
Р·Р°РІРµСЂС€РµРSPSCЦРj СѓСЂРsРєСѓ!');
                return;
            }


            await
updateUserProgress(userId, courseId,
lessonId, quiz?.quizId ?? undefined);
            navigateToNextLesson();
        } catch (err) {

console.error('РµРsРjРёР»РєР°
Р·Р°РІРµСЂС€РµРSPSCЦ СѓСЂРsРєСѓ:', err);
            alert('РќРµ РІРґР»РsСЃСЦ
Р·Р°РІРµСЂС€РёС,Рё СѓСЂРsРє.');
        }
    };


    const handleSubmitQuiz = async () =>
{
        if (!quiz || !userId ||
!courseId || !lessonId) return;


        const correctAnswers =
quiz.questions.filter(
            (q) =>
selectedOption[q.question] === q.correct
        );


        if (correctAnswers.length ===
quiz.questions.length) {
            setQuizCompleted(true);
            console.log('РўРµСЃС,
Р·Р°РІРµСЂС€РµРSPs СѓСЃРїС-С€PSPs');
            await
updateUserProgress(userId, courseId,
lessonId, quiz.quizId ?? undefined);
        } else {
            setQuizCompleted(false);
            console.log('РўРµСЃС, PSPµ
РїСЂРsР№РґРµРSPs');
        }
    };


    const navigateToNextLesson = () => {
        const currentIndex =
lessons.findIndex((l) => l.lessonId ===
lessonId);
        const nextLesson =
lessons[currentIndex + 1];


        if (nextLesson) {

navigate(`/courses/${courseId}/lessons/$
{nextLesson.lessonId}`);
        } else {
            alert('Р¦Рµ Р±СѓРI
РsСЃС,Р°РSPSC-Р№ СѓСЂРsРє Сѓ РєСѓСЂСЃС-
!');

navigate(`/courses/${courseId}`);
        }
    };


    if (loading) return <p
style={styles.loading}>Р—
Р°РIР°PSC,Р°Р¶РµРSPSCЦ...</p>;
    if (error) return <p
style={styles.error}>{error}</p>;


    return (
        <div
style={styles.pageContainer}>
            <BackToCourseButton/>
            {lesson && (
                <>
```

```
                    <h1
style={styles.title}>{lesson.title}</h1>

                    <div
style={styles.content}
dangerouslySetInnerHTML={{__html:
lesson.content}}/>

                    {lesson.videoUrl &&
(
                        <div
style={styles.videoContainer}>
                            <iframe

width="100%"

height="400px"

src={getYoutubeEmbedUrl(lesson.videoUrl)
}

title={lesson.title}

allowFullScreen
                            />
                        </div>
                    )}

                    {quiz &&
!quizCompleted && (
                        <div
style={styles.quizWrapper}>

<QuizComponent

quiz={quiz}

selectedOption={selectedOption}

setSelectedOption={setSelectedOption}

onSubmitQuiz={handleSubmitQuiz}
                            />
                        </div>
                    )}

                    {quizCompleted && (
                        <button
style={styles.completeButton}
onClick={handleCompleteLesson}>
                            Р—
Р°РІРµСЂС€РёС‚Рё СѓСЂРѕРє
                        </button>
                    )}
                </>
            )}
        </div>
    );
};

const styles: Record<string,
React.CSSProperties> = {
    pageContainer: {
        display: 'flex',
        flexDirection: 'column',
        gap: '20px',
        padding: '20px',
        backgroundColor: '#fff',
        borderRadius: '8px',
        boxShadow: '0 4px 12px rgba(0,
0, 0, 0.1)',
        maxWidth: '800px',
        margin: '20px auto',
    },
    title: {
        fontSize: '28px',
        fontWeight: 'bold',
    },
    content: {
        fontSize: '18px',
        lineHeight: '1.6',
```

```
    },
    videoContainer: {
        borderRadius: '8px',
        overflow: 'hidden',
    },
    quizWrapper: {
        borderRadius: '8px',
        padding: '20px',
        backgroundColor: '#f9f9f9',
        boxShadow: '0 2px 6px rgba(0, 0,
0, 0.1)',
        marginTop: '20px',
    },
    completeButton: {
        padding: '12px 20px',
        fontSize: '16px',
        color: '#fff',
        backgroundColor: '#4caf50',
        border: 'none',
        borderRadius: '5px',
        cursor: 'pointer',
        marginTop: '20px',
        transition: 'background 0.3s',
    },
    loading: {
        textAlign: 'center',
        fontSize: '18px',
        color: '#555',
    },
    error: {
        textAlign: 'center',
        fontSize: '18px',
        color: 'red',
    },
};
```

```
export default LessonPage;
```

pages\Login.tsx

```
//src/pages/Login.tsx

import LoginForm from
"../components/LoginForm";
import GoogleLoginButton from
"../components/GoogleLoginButton";
import { Link } from "react-router-dom";
import { CSSProperties } from "react";

const Login = () => {
  return (
    <div style={styles.pageContainer}>
      <div style={styles.container}>
        <h2>PJPIC-P№C,Pë</h2>
        <LoginForm />
        <p>P°P±Ps</p>
        <GoogleLoginButton />
        <p>
          P©Pµ PSPµPjP°C”
P°PєP°CѓPSC,Cѓ? <Link to="/register">P—
P°CЂPµC”CЃC,CЂCѓPIP°C,PëCЃCЊ</Link>
        </p>
      </div>
    </div>
  );
};

const styles: { [key: string]:
CSSProperties } = {
  pageContainer: {
    display: "flex",
    flexDirection: "column",
    justifyContent: "center",
    alignItems: "center",
    padding: "20px",
```

```
    boxSizing: "border-box",
    margin: "0 20px",
    backgroundColor: "#fff",
  },
  container: {
    padding: "4px 20px",
    border: "1px solid #ddd",
    borderRadius: "8px",
    textAlign: "center",
    backgroundColor: "#fff",
    boxShadow: "0 4px 8px rgba(0, 0, 0,
0.1)",
    width: "100%",
    maxWidth: "400px",
  },
};


export default Login;
```

pages\MyCoursesPage.tsx

```
import { useEffect, useState } from
"react";

import { getCoursesByTeacher,
updateCourse, deleteCourse  } from
"../services/mycoursesService";

import CourseList from
"../components/MyCoursesComp/CourseList"
;

import CourseForm from
"../components/MyCoursesComp/CourseForm.
tsx";

import EditCourseModal from
"../components/MyCoursesComp/EditCourseM
odal";

import { Course } from "../types";

import { useAuth } from
"../hooks/useAuth";


const MyCoursesPage = () => {

  const { user, loading: authLoading } =
useAuth();

  const [courses, setCourses] =
useState<Course[]>([]);

  const [isLoading, setIsLoading] =
useState(true);

  const [editingCourse,
setEditingCourse] = useState<Course |
null>(null);


  useEffect(() => {
    const fetchCourses = async () => {
      if (!user) {
        setIsLoading(false);
        return;
      }


      const teacherCourses = await
getCoursesByTeacher(user.uid);
      setCourses(teacherCourses);
      setIsLoading(false);
    };


    if (!authLoading) {
      fetchCourses();
    }
  }, [authLoading, user]);


  const handleCourseCreated = (course:
Course) => {
    setCourses((prev) => [...prev,
course]);
  };


  const handleUpdateCourse = async
(updatedCourse: Course) => {
    await updateCourse(updatedCourse);
    setCourses((prev) =>
      prev.map((course) =>
(course.courseId ===
updatedCourse.courseId ? updatedCourse :
course))
```

```
    );
    setEditingCourse(null);
  };
  const handleDeleteCourse = async
(courseId: string) => {
    if (!window.confirm("P’Pë
PIPïPµPIPSPµPSC–, C‰Ps C…PsC‡PµC,Pµ
PIPëPrP°P»PëC,Pë C†PµP№ PєCЃCЂCЃ?"))
return;

    await deleteCourse(courseId);
    setCourses((prev) => prev.filter((c)
=> c.courseId !== courseId));
  };
  return (
    <div style={{ padding: "20px" }}>
      <h2>PЊPsC— PєCЃCЂCЃPë</h2>
      <CourseList courses={courses}
onEdit={setEditingCourse}
onDelete={handleDeleteCourse}
isLoading={isLoading} />
      {editingCourse && (
        <EditCourseModal
          course={editingCourse}
          onClose={() =>
setEditingCourse(null)}
          onUpdate={handleUpdateCourse}
        />
      )}
      <CourseForm
onCourseCreated={handleCourseCreated} />
    </div>
  );
};

export default MyCoursesPage;
```

pages\Profile.tsx

```
// src/pages/Profile.tsx
import { useEffect, useState } from
"react";
import { getUserData } from
"../services/usersService";
import { auth, db } from
"../services/firebaseConfig";
import UserInfo from
"../components/UserInfo";
import LearningProgress from
"../components/LearningProgress";
import { deleteDoc, doc } from
"firebase/firestore";
import { deleteUser } from
"firebase/auth";
import { UserData } from "../types";


const Profile: React.FC = () => {
  const [user, setUser] =
useState<UserData | null>(null);
  const [showConfirm, setShowConfirm] =
useState(false);
  const [error, setError] =
useState<string>("");
  const [loading, setLoading] =
useState(true);

  const handleDeleteAccount = async ()
=> {
    try {
      if (auth.currentUser) {
        const userRef = doc(db, "users",
auth.currentUser.uid);
        await deleteDoc(userRef);
        await
deleteUser(auth.currentUser);

        console.log("Account
successfully deleted");
        window.location.href = "/";
      }
    } catch (err: any) {
```

```
      console.error("Error deleting
account:", err);

      setError("РќРµ РІРґР°Р»РѕСЃСЊ
РІРёРґР°Р»РёС‚Рё Р°РєР°СѓРЅС‚.");

    }

  };


  useEffect(() => {

    const unsubscribe =
auth.onAuthStateChanged(async
(currentUser) => {

      setLoading(true);

      try {

        if (!currentUser) {

          console.warn("User not
found");

          setUser(null);

        } else {

          const data = await
getUserData(currentUser.uid);

          setUser(data || null);

        }

      } catch (error) {

        console.error("Error loading
user data:", error);

        setUser(null);

      } finally {

        setLoading(false);

      }

    });


    return () => unsubscribe();

  }, []);


  if (loading) return <p>Р—
Р°РІР°РЅС‚Р°Р¶РµРЅРЅСЏ...</p>;


  if (user === null) return
<p>РљРѕСЂРёСЃС‚СѓРІР°С‡ PSPµ
Р·PSP°РЋРґРµPSPs Р°Р±Ps СЃС,Р°Р»Р°СЃСЏ
РїРѕРјРёР»РєР° РїСЂРё
Р·Р°РІР°РЅС‚Р°Р¶РµРЅРЅС–
РґР°РЅРёС….</p>;


  return (

    <div style={styles.container}>

      <h1>РЊР°Р±С–PSPµC,
РєРѕСЂРёСЃС‚СѓР№С‚Рµ Р°РєР°СѓРЅС‚</h1>

      <UserInfo user={user} />



      <LearningProgress
progress={user.progress} />


      <button
style={styles.deleteButton} onClick={()
=> setShowConfirm(true)}>

        Р’РёРґР°Р»РёС‚Рё Р°РєР°СѓРЅС‚

      </button>


      {showConfirm && (

        <div style={styles.modal}>

          <p>Р’Рё РІРїРµРІРЅРµРЅС–, С‰Рѕ
С…РѕС‡РµС‚Рµ РІРёРґР°Р»РёС‚Рё СЃРІС–Р№
Р°РєР°СѓРЅС‚?</p>

          <button
onClick={handleDeleteAccount}>РўР°Рє,
РІРёРґР°Р»РёС‚Рё</button>

          <button onClick={() =>
setShowConfirm(false)}>РЎРєР°СЃСѓРІР°С‚Рё
</button>

        </div>

      )}


      {error && <p
style={styles.error}>{error}</p>}

    </div>

  );

};
```

```
const styles: { [key: string]:
React.CSSProperties } = {
  container: {
    margin: "auto",
    padding: "20px",
  },
  error: {
    color: "red",
    fontSize: "14px",
  },
  deleteButton: {
    backgroundColor: "red",
    color: "white",
    padding: "10px",
    border: "none",
    borderRadius: "5px",
    cursor: "pointer",
    marginTop: "80px",
  },
  modal: {
    backgroundColor: "white",
    padding: "20px",
    border: "1px solid #ccc",
    borderRadius: "8px",
    position: "fixed",
    top: "50%",
    left: "50%",
    transform: "translate(-50%, -50%)",
    zIndex: 1000,
  },
};


export default Profile;
```

pages\QuizPage.tsx

```
// src/pages/QuizPage.tsx


import React, { useEffect, useState }
from "react";
import { useParams, useNavigate } from
"react-router-dom";
import { getQuizById } from
"../services/quizzesService";
import { auth } from
"../services/firebaseConfig";
import QuizComponent from
"../components/QuizComponent";
import {updateUserProgress} from
"../services/progressService";
import BackToCourseButton from
"../components/BackToCourseButton";
import {Quiz} from "../types.ts";


const QuizPage: React.FC = () => {
  const { courseId, quizId } =
useParams<{ courseId: string; quizId:
string }>();
  const [quiz, setQuiz] = useState<Quiz
| null>(null);
  const [selectedOption,
setSelectedOption] = useState<{ [key:
string]: string }>({});
  const [quizCompleted,
setQuizCompleted] = useState(false);
  const [loading, setLoading] =
useState(true);
  const [error, setError] =
useState<string | null>(null);


  const navigate = useNavigate();
  const userId = auth.currentUser?.uid;


  useEffect(() => {
    const loadQuizData = async () => {
      if (!courseId || !quizId) {
```

```
        setError("P’C–PrCЃCѓC,PSC–
PSPµPsP±C…C–PrPSC– PrP°PSC–.");
        setLoading(false);
        return;
      }

      try {
        const quizData = await
getQuizById(quizId);

        if (!quizData) {
          setError("PўPµCЃC, PSPµ
P·PSP°P№PrPµPSPs.");
          setLoading(false);
          return;
        }

        setQuiz(quizData);
      } catch (err) {
        console.error("PµPsPjPëP»PєP°
P·P°PIP°PSC,P°P¶PµPSPSCЦ C,PµCЃC,Cѓ:",
err);
        setError("PќPµ PIPrP°P»PsCЃCЦ
P·P°PIP°PSC,P°P¶PëC,Pë PrP°PSC–.");
      } finally {
        setLoading(false);
      }
    };

    loadQuizData();
  }, [courseId, quizId]);

const handleSubmitQuiz = async () => {
  if (!quiz || !userId || !courseId)
return;

  const userAnswers =
quiz.questions.map((q) =>
selectedOption[q.question] || "");
  const correctAnswers =
quiz.questions.map((q) => q.correct);
  const isPassed =
userAnswers.every((answer, index) =>
answer === correctAnswers[index]);

  if (!isPassed) {
    return;
  }

  try {
    await updateUserProgress(userId,
courseId, undefined, quiz.quizId);
    setQuizCompleted(true);
    console.log("P’C–C,P°C”PjPs! P’Pë
CѓCЃPїC–C€PSPs PїCЂPsP№C€P»Pë
C,PµCЃC,.");
  } catch (err) {
    console.error("PµPsPjPëP»PєP° PїCЂPë
PsPSPsPIP»PµPSPSC– PїCЂPsPiCЂPµCЃCѓ
C,PµCЃC,Cѓ:", err);
  }
};

  if (loading) return <p
style={styles.loading}>P—
P°PIP°PSC,P°P¶PµPSPSCЦ
C,PµCЃC,Cѓ...</p>;
  if (error) return <p
style={styles.error}>{error}</p>;

  return (
    <div style={styles.pageContainer}>
      <BackToCourseButton />
      {quiz && (
        <>
          <QuizComponent
            quiz={quiz}

selectedOption={selectedOption}
```

```
setSelectedOption={setSelectedOption}

onSubmitQuiz={handleSubmitQuiz}
          />


{quizCompleted && (
  <div style={styles.completionBlock}>
    <h3 style={styles.congrats}>pµЋ‰
P’C–C,P°C”PjPs! P’Pë CѓCЃPїC–C€PSPs
P·P°PIPµCЂC€PëP»Pë PєCѓCЂCЃ.</h3>
    <button style={styles.nextButton}
onClick={() =>
navigate(`/courses/${courseId}`)}>
      P—P°PIPµCЂC€PëC,Pë PєCѓCЂCЃ
    </button>
  </div>
)}



        </>
      )}
    </div>
  );
};


const styles: Record<string,
React.CSSProperties> = {
  pageContainer: {
    maxWidth: "800px",
    margin: "20px auto",
    padding: "20px",
    backgroundColor: "#fff",
    borderRadius: "8px",
    boxShadow: "0 4px 12px rgba(0, 0, 0,
0.1)",
  },
  title: {
    fontSize: "28px",
    fontWeight: "bold",
    marginBottom: "20px",
  },
  nextButton: {
    padding: "12px 20px",
    fontSize: "16px",
    color: "#fff",
    backgroundColor: "#4caf50",
    border: "none",
    borderRadius: "5px",
    cursor: "pointer",
    marginTop: "20px",
    transition: "background 0.3s",
  },
  loading: {
    textAlign: "center",
    fontSize: "18px",
    color: "#555",
  },
  error: {
    textAlign: "center",
    fontSize: "18px",
    color: "red",
  },
  completionBlock: {
  marginTop: "20px",
  padding: "20px",
  backgroundColor: "#e6f7e6",
  borderRadius: "8px",
  textAlign: "center",
},
congrats: {
  fontSize: "20px",
  color: "#2e7d32",
  marginBottom: "15px",
},
```

```
};

export default QuizPage;
```

pages\Register.tsx

```
//src/pages/Register.tsx

import RegisterForm from
"../components/RegisterForm";
import { Link } from "react-router-dom";
import { CSSProperties } from "react";

const Register = () => {
  return (
    <div style={styles.pageContainer}>
      <div style={styles.container}>
        <h2>P PµC”CЃC,CЪP°C†C–CЏ</h2>
        <RegisterForm />
        <p>
          P’P¶Pµ PjP°C”C,Pµ
P°PєP°CѓPSC,? <Link to="/login">PJPIC–
PNC,Pë</Link>
        </p>
      </div>
    </div>
  );
};

const styles: { [key: string]:
CSSProperties } = {
  pageContainer: {
    display: "flex",
    flexDirection: "column",
    justifyContent: "center",
    alignItems: "center",
    height: "100%",
    padding: "20px",
    boxSizing: "border-box",
    margin: "0 auto",
    backgroundColor: "#fff",
  },
  container: {
    margin: "20px",
    padding: "10px 20px ",
    border: "1px solid #ddd",
    borderRadius: "8px",
    textAlign: "center",
    backgroundColor: "#fff",
    boxShadow: "0 4px 8px rgba(0, 0, 0,
0.1)",
    width: "100%",
    maxWidth: "400px",
  },
};

export default Register;
```

services\coursesService.ts

```
import { db } from "./firebaseConfig";
import { collection, doc, getDocs,
getDoc } from "firebase/firestore";
import {Course} from "../types.ts";

// PћC,CЪPёPjP°C,Pë PICЃC– PєCѓCЪCЃPë
export const getCourses = async ():
Promise<Course[]> => {
  try {
    const snapshot = await
getDocs(collection(db, "courses"));
    return snapshot.docs.map((doc) => ({
courseId: doc.id, ...doc.data() })) as
Course[];
```

```
  } catch (error) {
    console.error("РµРsРjРёР»РєР° РsС,СЂРёРjР°РSРSСЏ РєСѓСЂСЃРѕРІ:", error);
    return [];
  }
};

export const getCourseById = async (courseId: string): Promise<Course | null> => {
  try {
    const courseDoc = await getDoc(doc(db, "courses", courseId));
    return courseDoc.exists() ? { courseId: courseDoc.id, ...courseDoc.data() } as Course : null;
  } catch (error) {
    console.error("РµРsРjРёР»РєР° РsС,СЂРёРjР°РSРSСЏ РєСѓСЂСЃР°:", error);
    return null;
  }
};

export const getCategories = async (): Promise<string[]> => {
  try {
    const coursesRef = collection(db, "courses");
    const snapshot = await getDocs(coursesRef);
    const categories = new Set<string>();

    snapshot.forEach((doc) => {
      const course = doc.data();
      if (Array.isArray(course.category)) {
        course.category.forEach((cat) => categories.add(cat));
      } else if (typeof course.category === "string") {
        categories.add(course.category);
      }
    });

    return Array.from(categories);
  } catch (error) {
    console.error("РµРsРjРёР»РєР° РsС,СЂРёРjР°РSРSСЏ РєР°С,РµРiРsСЂС-Р№:", error);
    return [];
  }
};
```

services\firebaseConfig.ts

```
//src/services/firebaseConfig.ts

import { initializeApp } from "firebase/app";
//import { getAnalytics } from "firebase/analytics";
import { getAuth, GoogleAuthProvider } from "firebase/auth";
import { getFirestore } from "firebase/firestore";

// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

const firebaseConfig = {
  apiKey: "AIzaSyCrB00XBVzS5JUArV3P2duPJLrQgVaMD1E",
  authDomain: "edu-safety-app.firebaseapp.com",
  projectId: "edu-safety-app",
  storageBucket: "edu-safety-app.firebasestorage.app",
```

```
  messagingSenderId: "765776437855",
  appId:
"1:765776437855:web:20dc64b1e223a9832df1
ea",
  measurementId: "G-YR79JSMF0Q"
};



// Initialize Firebase
const app =
initializeApp(firebaseConfig);
//const analytics = getAnalytics(app);


const auth = getAuth(app);
const db = getFirestore(app);
const googleProvider = new
GoogleAuthProvider();


export { auth, googleProvider, db };
```

services\lessonsService.ts

```
import { db } from "./firebaseConfig";
import {collection, deleteDoc, doc,
getDoc, getDocs, query, updateDoc,
where} from "firebase/firestore";
import { Lesson } from "../types";



export const getLessonById = async
(lessonId: string): Promise<Lesson |
null> => {
  try {
    const lessonRef = doc(db, "lessons",
lessonId);
    const lessonSnap = await
getDoc(lessonRef);

    if (lessonSnap.exists()) {
      const data = lessonSnap.data();
      return {
        lessonId: lessonSnap.id,
        courseId: data.courseId,
        title: data.title,
        content: data.content,
        videoUrl: data.videoUrl || "",
        order: data.order ?? 0,
        quizId: data.quizId || "",
      } as Lesson;
    }
    return null;
  } catch (error) {
    console.error("РџРѕРјРёР»РєР°
РѕС‚СЂРёРјР°РЅРЅСЏ СѓСЂРѕРєСѓ Р·Р° ID:",
error);
    return null;
  }
};


export const getLessonsByCourseId =
async (courseId: string):
Promise<Lesson[]> => {
  try {
    console.log("РћС‚СЂРёРјР°РЅРЅСЏ
СѓСЂРѕРєС–РІ РґР»СЏ РєСѓСЂСЃСѓ Р·
courseId:", courseId);

    const lessonsCollection =
collection(db, "lessons");
    const q = query(lessonsCollection,
where("courseId", "==", courseId));
    const querySnapshot = await
getDocs(q);

    console.log("РљС–Р»СЊРєС–СЃС‚СЊ
Р·РЅР°Р№РґРµРЅРёС… СѓСЂРѕРєС–РІ:",
querySnapshot.size);

    querySnapshot.forEach((doc) => {
      console.log("Р—РЅР°Р№РґРµРЅРѕ
СѓСЂРѕРє:", doc.id, doc.data());
```

```
    });

    return querySnapshot.docs.map((doc)
=> {
      const data = doc.data();
      return {
        lessonId: doc.id,
        courseId: data.courseId,
        title: data.title,
        content: data.content,
        videoUrl: data.videoUrl || "",
        order: data.order ?? 0,
        quizId: data.quizId || "",
      } as Lesson;
    });
  } catch (error) {
    console.error("РџРѕРјРёР»РєР° PsC,CЂРёРjР°PSPSCЏ CѓCЂРsРєС–РІ Р·Р° ID
РєСѓCЂCЃCѓ:", error);
    return [];
  }
};


export const deleteLessonById = async
(courseId: string, lessonId: string) =>
{
  try {
    // РІРёРґР°Р»РџPSPSCЏ CѓCЂРsРєСѓ
    await deleteDoc(doc(db, "lessons",
lessonId));

    // PsPSPsPIP»РџPSPSCЏ
РґРsРєСѓРjРџPSC,Р° РєСѓCЂCЃCѓ
    const courseRef = doc(db, "courses",
courseId);
    const courseSnap = await
getDoc(courseRef);
    if (courseSnap.exists()) {
      const courseData =
courseSnap.data();
      const updatedLessons =
(courseData.lessons || []).filter((id:
string) => id !== lessonId);
      await updateDoc(courseRef, {
        lessons: updatedLessons,
      });
    }
  } catch (error) {
    console.error("РџРsPjРёР»РєР° РїCЂРё
РІРёРґР°Р»РџPSPSC- CѓCЂРsРєСѓ Р·
PsPSPsPIP»РџPSPSCЏPj РєСѓCЂCЃCѓ:",
error);
  }
};
```

services\mycoursesService.ts

```
import {db} from "./firebaseConfig";
import {collection, deleteDoc, doc,
getDocs, query, updateDoc, where,
writeBatch,} from "firebase/firestore";
import {Course, Lesson, Quiz} from
"../types";


export const getCoursesByTeacher = async
(teacherId: string): Promise<Course[]>
=> {
    const q = query(collection(db,
"courses"), where("teacherId", "==",
teacherId));
    const snapshot = await getDocs(q);
    return snapshot.docs.map((doc) =>
({courseId: doc.id, ...doc.data()} as
Course));
};


export const updateCourse = async
(course: Course): Promise<void> => {
    const ref = doc(db, "courses",
course.courseId);
    await updateDoc(ref, {
        title: course.title,
description: course.description
```

```
    });
};


export const deleteCourse = async
(courseId: string) => {
    try {
        // 1. Р'РёРґР°Р»СЏРµРј PICЃC-
CѓCЂPsPєPё Р· С†РёРj courseId
        const lessonsQuery =
query(collection(db, "lessons"),
where("courseId", "==", courseId));
        const lessonsSnapshot = await
getDocs(lessonsQuery);
        const lessonDeletions =
lessonsSnapshot.docs.map(docSnap =>
deleteDoc(doc(db, "lessons",
docSnap.id)));
        await
Promise.all(lessonDeletions);
        console.log(`Р'РёРґР°Р»РµРSPs
${lessonDeletions.length} CѓCЂPsPє(C-
PI), РїPsPI'СЏР·Р°PSPёС… С-Р·
РєCѓCЂCЃPsPj.`);


        // 2. Р'РёРґР°Р»СЏРjPs PICЃC-
C,РµCЃC,Рё Р· С†РёРj courseId
        const quizzesQuery =
query(collection(db, "quizzes"),
where("courseId", "==", courseId));
        const quizzesSnapshot = await
getDocs(quizzesQuery);
        const quizDeletions =
quizzesSnapshot.docs.map(docSnap =>
deleteDoc(doc(db, "quizzes",
docSnap.id)));
        await
Promise.all(quizDeletions);
        console.log(`Р'РёРґР°Р»РµРSPs
${quizDeletions.length} C,РµCЃC,(C-PI),
РїPsPI'СЏР·Р°PSPёС… С-Р·
РєCѓCЂCЃPsPj.`);


        // 3. РќР°СЂРµС€С,С-
PIPёРґР°Р»СЏРjPs CЃР°Pj РєCѓCЂCЃ
        const courseRef = doc(db,
"courses", courseId);
        await deleteDoc(courseRef);
        console.log(`РЪCѓCЂCЃ Р· ID
${courseId} CѓCЃPїC-С€PSPs
PIPёРґР°Р»РµPSPs!`);
    } catch (error) {
        console.error("РµPsPjPёР»РєР°
РїCЂPё PIPёРґР°Р»РµPSPSC- РєCѓCЂCЃCѓ
С,Р° РїPsPI'СЏР·Р°PSPёС… РґР°PSPёС…:",
error);
    }
};


type LessonWithOptionalQuiz =
Omit<Lesson, "lessonId" | "courseId" |
"quizId"> & {
    quiz?: Omit<Quiz, "quizId" |
"courseId">;
};


export interface LessonFormData extends
Omit<Lesson, "lessonId" | "courseId"> {
    quiz?: Omit<Quiz, "quizId" |
"courseId">;
}


export const createFullCourse = async
(teacherId: string,
    courseData: Omit<Course, "courseId"
| "lessons" | "quiz_id">, lessonsData:
    LessonWithOptionalQuiz[],
finalQuizData?: Omit<Quiz, "quizId" |
"courseId">): Promise<Course> => {
    const batch = writeBatch(db);
    const courseRef = doc(collection(db,
"courses"));
    const courseId = courseRef.id;
    const lessonRefs =
lessonsData.map(() => doc(collection(db,
"lessons")));
    const lessonIds: string[] = [];
    let finalQuizId: string | undefined;
    if (finalQuizData) {
```

```
        const quizRef =
doc(collection(db, "quizzes"));

        finalQuizId = quizRef.id;

        batch.set(quizRef,
{...finalQuizData, courseId, teacherId,
createdAt: new Date().toISOString(),});

    }

    lessonsData.forEach((lesson, index)
=> {

        const lessonRef =
lessonRefs[index];

        const lessonId = lessonRef.id;

        lessonIds.push(lessonId);

        let quizId: string | undefined;

        if (lesson.quiz) {

            const quizRef =
doc(collection(db, "quizzes"));

            quizId = quizRef.id;

            batch.set(quizRef,
{...lesson.quiz, courseId, teacherId,
createdAt: new Date().toISOString(),});

        }

        batch.set(lessonRef, {...lesson,
courseId, teacherId, order: index,
quizId, createdAt: new
Date().toISOString(),});

    });

    const fullCourse: Course =
{...courseData, courseId, lessons:
lessonIds, quiz_id: finalQuizId,
teacherId,

        createdAt: new
Date().toISOString(),};

    batch.set(courseRef, fullCourse);

    await batch.commit();

    return fullCourse;

};
```

services\progressService.ts

```
//  src/services/progressService.ts

import {doc, getDoc, updateDoc} from
"firebase/firestore";

import {db} from "./firebaseConfig.ts";

import {getCourseById} from
"./coursesService.ts";

import {calculateCourseProgress} from
"../utils/calculateCourseProgress.ts";

import { UserData } from "../types";


export const getUserProgress = async
(userId: string, courseId: string):
Promise<{ lessonsCompleted: string[],
testsPassed: string[],
progressPercentage: number } | null> =>
{

  try {

    const userDoc = await getDoc(doc(db,
"users", userId));

    if (userDoc.exists()) {

      const userData = userDoc.data() as
UserData;

      const courseProgress =
userData.progress[courseId] || {

        lessonsCompleted: [],

        testsPassed: [],

        progressPercentage: 0,

      };

      return courseProgress;

    } else {

      throw new Error("P”PsPєCґPjPµPSC,
PєPsCЂPёCЃC,CґPIP°C‡P° PSPµ
P·PSP°P№PrPµPSPs");

    }

  } catch (error) {

    console.error("PµPsPjPёP»PєP° PїCЂPё
P·P°PIP°PSC,P°P¶PµPSPSC–
PїCЂPsPiCЂPµCЃCґ
PєPsCЂPёCЃC,CґPIP°C‡P°:", error);

    return null;

  }

};
```

```
export const updateUserProgress = async
(
  userId: string,
  courseId: string,
  lessonId?: string,
  quizId?: string
): Promise<boolean> => {
  try {
    const userRef = doc(db, "users",
userId);
    const userSnap = await
getDoc(userRef);

    if (!userSnap.exists()) {
      throw new
Error("РљРsСЂРёСЃС‚РІР°Р° РЅРµ
Р·РЅР°Р№РґРµРЅРs");
    }

    const userData = userSnap.data() as
UserData;
    const courseProgress =
userData.progress[courseId] || {
      lessonsCompleted: [],
      testsPassed: [],
      progressPercentage: 0,
    };

    if (lessonId &&
!courseProgress.lessonsCompleted.include
s(lessonId)) {

courseProgress.lessonsCompleted.push(les
sonId);
    }

    if (quizId &&
!courseProgress.testsPassed.includes(qui
zId)) {

courseProgress.testsPassed.push(quizId);
    }

    const courseData = await
getCourseById(courseId);

    const courseLessons: string[] =
Array.isArray(courseData?.lessons)
      ? courseData.lessons
      : [];

    const courseQuizzes: string[] =
Array.isArray(courseData?.quiz_id)
      ? courseData.quiz_id
      : courseData?.quiz_id
      ? [courseData.quiz_id]
      : [];

    console.log('Course Lessons:',
courseLessons);
    console.log('Course Quizzes:',
courseQuizzes);

    const lessonsCompleted =
Array.isArray(courseProgress.lessonsComp
leted)
      ? courseProgress.lessonsCompleted
      : [];

    const testsPassed =
Array.isArray(courseProgress.testsPassed
)
      ? courseProgress.testsPassed
      : [];

    const progressPercentage =
calculateCourseProgress(
      lessonsCompleted,
      testsPassed,
```

```
      courseLessons,
      courseQuizzes
    );

    console.log('Calculated Progress
Percentage:', progressPercentage);

    courseProgress.progressPercentage =
progressPercentage;

    await updateDoc(userRef, {
      [`progress.${courseId}`]:
courseProgress,
    });

    console.log(`РµСЂРsPiCЂРµСЃ
PsPSPsPIP»РµPSPs РґР»СЏ РєСѓСЂСЃР°
${courseId}`);
    return true;
  } catch (error) {
    console.error("РµPsPjPёP»РєР°
PsPSPsPIP»РµPSPSCЏ РїСЂРsPiCЂРµСЃР°
РєРsCЂРёСЃС,СѓPIP°С‡Р°:", error);
    return false;
  }
};
```

services\quizzesService.ts

```
import { db } from "./firebaseConfig";
import {collection, deleteDoc, doc,
getDoc, getDocs, query, updateDoc,
where} from "firebase/firestore";
import { Quiz } from "../types";
import { deleteField } from
"firebase/firestore";

export const getQuizById = async
(quizId: string): Promise<Quiz | null>
=> {
  try {
    const quizRef = doc(db, "quizzes",
quizId);
    const quizDoc = await
getDoc(quizRef);

    if (!quizDoc.exists()) return null;

    const data = quizDoc.data();

    const quiz: Quiz = {
      quizId: quizDoc.id,
      courseId: data.courseId,
      title: data.title || "Р‘РµР·
РЅР°Р·РІР°РЅРёС,Р°РЅРЅРѕРіРѕ",
      questions:
Array.isArray(data.questions)
        ? data.questions.map((q: any) =>
({
            question: q.question ||
"Р‘РµР· Р·Р°РїРёС,Р°РЅРЅРѕРіРѕ",
            options:
Array.isArray(q.options) ? q.options :
[],
            correct: q.correct || "",
          }))
        : [],
    };

    return quiz;
  } catch (err) {
    console.error("РµPsPjPёP»РєР°
PsC,СЂРёPjР°РЅРЅРѕРіРѕ С,РµСЃС,Сѓ:", err);
    return null;
  }
};

export const getQuizzesByCourseId =
async (courseId: string):
Promise<Quiz[]> => {
  try {
```

```
    const quizzesQuery =
query(collection(db, "quizzes"),
where("courseId", "==", courseId));

    const snapshot = await
getDocs(quizzesQuery);


    return snapshot.docs.map((doc) => {

      const data = doc.data();

      return {

        quizId: doc.id,

        courseId: data.courseId,

        title: data.title || "P‘PµP·
PSP°P·PIPë",

        questions:
Array.isArray(data.questions) ?
data.questions : [],

      } as Quiz;

    });

  } catch (error) {

    console.error("PµPsPjPëP»PєP°
PsC,CЂPëPjP°PSPSCЏ C,PµCЃC,C–PI P·P° ID
PєCѓCЂCЃCѓ:", error);

    return [];

  }

};


export const deleteQuizById = async
(quizId: string) => {

  try {

    await deleteDoc(doc(db, "quizzes",
quizId));

  } catch (error) {

    console.error("PµPsPjPëP»PєP° PїCЂPë
PIPëPrP°P»PµPSPSC– C,PµCЃC,Cѓ:", error);

  }

};


export const removeFinalQuizFromCourse =
async (courseId: string) => {

  try {

    const courseRef = doc(db, "courses",
courseId);

    await updateDoc(courseRef, {

      quiz_id: deleteField(),

    });

  } catch (error) {

    console.error("PµPsPjPëP»PєP° PїCЂPë
PIPëPrP°P»PµPSPSC– C„C–PSP°P»CЊPSPsPiPs
C,PµCЃC,Cѓ P· PєCѓCЂCЃCѓ:", error);

  }

};


services\usersService.ts

//  src/services/usersService.ts


import { db } from "./firebaseConfig";

import {doc, getDoc, setDoc, } from
"firebase/firestore";

import { User } from "firebase/auth";

import { UserData } from "../types";



export const getUserData = async
(userId: string): Promise<UserData |
null> => {

  try {

    const userDoc = await getDoc(doc(db,
"users", userId));

    if (userDoc.exists()) {

      return userDoc.data() as UserData;
// PЇPIPSPs PIPєP°P·CѓC”PjPs, C‰Ps
PїPsPIPµCЂC,P°C”C,CЊCЃCЏ User

    } else {


console.log("PЉPsCЂPëCЃC,CѓPIP°C‡P° PSPµ
P·PSP°P№PrPµPSPs");

      return null;

    }

  } catch (error) {
```

```
    console.error("РµPsPjPëP»PєP°
PsC,СЪPëPjP°PSPSCЏ PrP°PSPëC…
PєPsCЪPëCЃC,CѓPIP°C‡P°:", error);

    return null;

  }

};


export const addUserToFirestore = async
(user: User, role: "CѓC‡PµPSCЊ" |
"PIPëPєP»P°PrP°C‡") => {

  if (!user) return;

  const isVerified = user.emailVerified;


  const userRef = doc(db, "users",
user.uid);

  const docSnap = await getDoc(userRef);


  if (docSnap.exists()) {

    console.log("РљPsCЪPëCЃC,CѓPIP°C‡
PIP¶Pµ C–CЃPSCѓC” PI Firestore.");

    return;

  }


  try {

    await setDoc(userRef, {

      displayName: user.displayName ||
"РђPSPsPSC–PjPSPëP№
PєPsCЪPëCЃC,CѓPIP°C‡",

      email: user.email,

      photoURL: user.photoURL || "",

      role: role,

      progress: {

        coursesCompleted: [],

        courses: {},

      },

      achievements: [],

      testHistory: [],

      createdAt: new
Date().toISOString(),

      isVerified: isVerified,

    });

    console.log("РљPsCЪPëCЃC,CѓPIP°C‡P°
PrPsPrP°PSPs PI Firestore!");

  } catch (error) {

    console.error("РµPsPjPëP»PєP° РїCЪPë
PrPsPrP°PIP°PSPSC–
PєPsCЪPëCЃC,CѓPIP°C‡P°:", error);

  }

};


export const getUserRole = async
(userId: string): Promise<string | null>
=> {

  try {

    const userDoc = await getDoc(doc(db,
"users", userId));

    if (userDoc.exists()) {

      return userDoc.data()?.role ||
null;

    } else {


console.log("РљPsCЪPëCЃC,CѓPIP°C‡P° PSPµ
P·PSP°P№PrPµPSPs.");

       return null;

    }

  } catch (error) {

    console.error("РµPsPjPëP»PєP°
PsC,СЪPëPjP°PSPSCЏ CЪPsP»C–
PєPsCЪPëCЃC,CѓPIP°C‡P°:", error);

    return null;

  }

};


utils\calculateCourseProgress.ts

export const calculateCourseProgress = (

  lessonsCompleted: string[],

  testsPassed: string[],

  courseLessons: string[],

  courseQuizzes: string[]

): number => {
```

```
   const validCompletedLessons =
 lessonsCompleted.filter(lessonId =>

    courseLessons.includes(lessonId)

  );


  // P¤C-P»CЊC,CЂCґC”PjPs P»PëC€Pµ C,C-
PєPIC-P·Pë, CЦPєC- C” PI PєCґCЂCЃC-

  const validCompletedQuizzes =
testsPassed.filter(quizId =>

    courseQuizzes.includes(quizId)

  );


  const totalTasks =
courseLessons.length +
courseQuizzes.length;

  const completedTasks =
validCompletedLessons.length +
validCompletedQuizzes.length;


  return totalTasks ?
Math.round((completedTasks / totalTasks)
* 100) : 0;

};
```

App.tsx

```
import { Routes, Route } from "react-
router-dom";

import { CSSProperties } from "react";

import Home from "./pages/Home";

import Courses from
"./pages/CoursesPage.tsx";

import CoursePage from
"./pages/CoursePage.tsx";

import LessonPage from
"./pages/LessonPage";

import QuizPage from "./pages/QuizPage";

import Profile from "./pages/Profile";

import Login from "./pages/Login";

import Register from "./pages/Register";

import Header from
"./components/Header";

import Footer from
"./components/Footer";

import MyCoursesPage from
"./pages/MyCoursesPage";


function App() {

  return (

    <div style={styles.container}>

        <Header />

        <main style={styles.main}>

          <Routes>

            <Route path="/"
element={<Home />} />

            <Route path="/profile"
element={<Profile />} />

            <Route path="/login"
element={<Login />} />

            <Route path="/register"
element={<Register />} />

            <Route path="/my-courses"
element={<MyCoursesPage />} />

            <Route path="/courses"
element={<Courses />} />

            <Route
path="/courses/:courseId"
element={<CoursePage/>} />

            <Route
path="/courses/:courseId/lessons/:lesson
Id" element={<LessonPage />} />

            <Route
path="/courses/:courseId/quizzes/:quizId
" element={<QuizPage />} />

          </Routes>

        </main>

        <Footer />

    </div>

  );

}


const styles: { container:
CSSProperties; main: CSSProperties } = {

  container: {
```

```
    display: "flex",
    flexDirection: "column",
    minHeight: "100vh",
    width: "100%",
  },
  main: {
    flex: 1,
    padding: " 0 20px",
  },
};

export default App;
```

types.ts

```
export interface UserData {
  userId: string;
  displayName: string;
  email: string;
  photoURL: string;
  role: "СґС‡РµРSСЊ" |
"РІРёРєР»Р°РґР°С‡";
  progress: {
    [courseId: string]: {
      lessonsCompleted: string[]; //
РњР°СЃСЃРёРІ Р· ID РїСЂРѕР№РґРµРЅРёС…
СґСЂРѕРєС–РІ
      testsPassed: string[]; //
РњР°СЃСЃРёРІ Р· ID РїСЂРѕР№РґРµРЅРёС…
С‚РµСЃС‚С–РІ
      progressPercentage: number; // Р—
Р°РіР°Р»СЊРЅРёР№ РІС–РґСЃРѕС‚РѕРє
РїСЂРѕРіСЂРµСЃСѓ РєСѓСЂСЃСѓ
    };
  };
  createdAt: string;
  isVerified: boolean;
}

export interface Course {
  courseId: string;
  title: string;
  short_description: string;
  description: string;
  category: string[];
  image_url?: string;
  lessons: string[];
  quiz_id?: string;
  teacherId: string;
  createdAt: string;
}

export interface Lesson {
  lessonId: string;
  courseId: string;
  title: string;
  content: string;
  videoUrl?: string;
  order: number;
  quizId?: string;
}

export interface Quiz {
  quizId: string;
  courseId?: string;
  title: string;
  questions: Question[];
}

export interface Question {
  question: string;
  options: string[];
  correct: string;
}
```