Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Освітня платформа для онлайн-навчання

(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІ-21-6
Олексій НІЛЬГА
(власне ім'я, прізвище)

Спеціальність
123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник: ас. Дар'я ТИМОШЕНКО
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Нільзі Олексію Миколайовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Освітня платформа для онлайн-навчання

затверджена наказом по університету від “ 26 ” травня 2025 р. № 424 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 17 червня 2025 р.

3. Вхідні дані до роботи

1) документація JavaScript
2) документація HTML і CSS;
3) документація MSSQL;
4) документація Express
5) середовище розробки JetBreins WEBSTORM

4. Перелік питань, що потрібно опрацювати у роботі

1) аналіз предметної області;
2) аналіз використовуваних технологій;
3) програмна реалізація;
4) інструкція користувача;

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Презентація – 14 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих рішень	27.05.25-30.05.25	
2	Вибір технології розробки та інструменталь-	31.05.25-03.06.25	
	них засобів		
3	Розробка алгоритмічного забезпечення	04.06.25-06.06.25	
4	Розробка та відлагодження програмного	07.06.25-09.06.25	
	забезпечення		
5	Оформлення матеріалів кваліфікаційної роботи	10.06.25-11.06.25	
6	Подання кваліфікаційної роботи керівникові та	12.06.25-13.06.25	
	її попередній захист		
7	Подання кваліфікаційної роботи на рецензу-	14.06.25-16.06.25	
	вання		

Дата видачі завдання " 26 " травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Дар'я ТИМОШЕНКО
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 62 с., 25 рис., 2 дод., 19 джерел.

ФУНКЦІОНАЛЬНЕ ПРОГРАМУВАННЯ, EXPRESS.JS, NODE.JS, REACT, JAVASCRIPT, HTML, CSS, SPA, MSSQL.

Метою кваліфікаційної роботи є створення веб-застосунку для надання можливостей навчання онлайн. Розроблений застосунок дозволяє створювати та класифікувати навчальні курси та їх наповнення.

У ході виконання кваліфікаційної роботи було розглянуто та досліджено переваги та недоліки схожих веб-сервісів. Сформовано список вимог до власного проєкту та побудовано власну архітектуру сервісу. Було розроблено застосунок з можливістю створення нових аккаунтів, авторизації, автентифікації. Було зроблено ієрахічну систему курсів і файлів для більш зручного пошуку їх на сайті.

Для реалізації поставленого завдання було використано сучасні інструменти для розробки JavaScript з ES15, бібліотека React, CSS, HTML5 Bootstrap React для клієнтської версії проєкту. В той час як для серверної частини були використані Express.JS, MSSQL, Node.JS.

ABSTRACT

Bachelor’s thesis: 62 pages, 25 figures, 2 appendices, 19 sources.

FUNCTIONAL PROGRAMMING, EXPRESS.JS, NODE.JS, REACT, JAVASCRIPT, HTML, CSS, SPA, MSSQL.

The major goal of this thesis is to create a web application for providing opportunities to study online. The developed application allows you to create and classify training courses and their content.

During the qualification work, the advantages and disadvantages of similar web services were reviewed and researched. A list of requirements for the project was formed and the service architecture was built. An application was developed with the ability to create new accounts, authorization, and authentication. A hierarchical system of courses and files was created for more convenient searching on the site.

To implement the task, modern tools for developing JavaScript with ES15, the React library, CSS, and HTML5 Bootstrap React were used for the client version of the project. Meanwhile, Express.JS, MSSQL, and Node.JS were used for the server part.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

API – інтерфейс прикладного програмування (англ., Application Programming Interface)

CRUD – створення, читання, оновлення, видалення (англ., Create / Read / Update / Delete)

CSS – каскадні таблиці стилів (англ., Cascading Style Sheets)

HTML – мова розмітки гіпертексту (англ., HyperText Markup Language)

IDE – інтегроване середовище розробки (англ., Integrated Development Environment)

JSON – об’єктна нотація JavaScript (англ., JavaScript Object Notation)

JSX – JavaScript + XML – розширення синтаксису для React (англ., JavaScript XML)

JWT – токен на основі JSON (англ., JSON Web Token)

SQL – мова структурованих запитів (англ., Structured Query Language)

UI – інтерфейс користувача (англ., User Interface)

ВСТУП

З кожним роком розвиток інформаційних технологій та активне впровадження онлайн навчання стрімко зростає. Це призводить до змін в організації навчального процесу. Традиційна форма навчання, яка передбачає фізичну присутність студента або учня у навчальних закладах, поступово доповнюється або навіть замінюється сучасними веб-застосунками. Питання онлайн-навчання стало дуже актуальним завдяки пандемії та вимушеному переселенню в інші регіони. Як наслідок, треба було переглянути класичні моделі навчання.

Навчання онлайн полегшує роботу як для викладачів, так і для студентів. Такий вид навчання забезпечує більш зручний графік та дозволяє отримувати знання незалежно від місця проживання. Незважаючи на велику кількість існуючих рішень, багато з них мають суттєві недоліки, це може бути складний інтерфейс, обмежений функціонал або навіть необхідність сплачувати підписку на сайт.

В Україні велика кількість людей вже навчається онлайн та використовує різні освітні платформи. Однак більшість з них розраховані на масштабне використання і часто не враховують специфічні потреби окремих навчальних закладів, невеликих приватних шкіл або університетів. Саме тому важливо створити освітню платформу, яка була б простою у використанні, легко налаштовувалась до конкретних потреб користувачів та не вимагала складної реєстрації чи оплати послуг.

Метою роботи є створення веб-застосунку для онлайн-навчання, який забезпечить зручне розміщення навчальних матеріалів та зворотній зв'язок між вчителем та учнем. Такий додаток дозволить ефективно організувати дистанційне роботу в межах навчального закладу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Сучасні освітні онлайн-платформи

Впродовж майже трьох років учасники освітнього процесу стикаються з новими викликами, що спричиняють трансформацію та необхідність адаптуватися до нових умов. Для забезпечення рівного доступу до освіти навчальний процес перейшов з очного формату до змішаного або дистанційного, що змусило вчителів та учнів вивчати різні цифрові інструменти.

Онлайн платформа – це спеціалізований інструмент, який дозволяє педагогам організовувати навчальні матеріали, оцінювати учнів, проводити дистанційні заняття, а також надавати доступ до методичних матеріалів. Такі платформи зазвичай мають функціонал для створення та збереження планів уроків, розробки інтерактивних завдань, а також підтримують спілкування між вчителями, учнями та батьками [1].

Завдяки онлайн платформам, навчання стало доступнішим, гнучкішим і більш інтерактивним. Педагоги можуть проводити заняття незалежно від географічного розташування та підтримувати тісний зв'язок з учнями у реальному часі. До того ж, такі платформи надають можливість зберігати результати оцінювання та відстежувати прогрес кожного учня.

З розвитком інформаційних технологій та зростанням попиту на гнучке навчання освітні онлайн-платформи стали невід’ємною частиною сучасного освітнього середовища. Їх використання особливо актуалізувалося в умовах дистанційного навчання, спричиненого пандемією. Онлайн-платформи активно впроваджуються як у вищих навчальних закладах, так і на ринку приватної освіти.

Переваги використання онлайн платформ:

- доступ до великої кількості матеріалів. Використовуючи онлайн

платформу, вчителі отримують доступ до безлічі освітніх ресурсів, таких як відеоуроки, презентації, інтерактивні вправи та багато іншого. Це дозволяє варіювати методики викладання, роблячи навчання цікавішим та різноманітнішим;

- економія часу на підготовку уроків. З платформою вчителі можуть створювати навчальні плани та використовувати вже готові матеріали, адаптуючи їх під потреби свого класу. Це значно полегшує процес підготовки до уроків і дозволяє більше часу приділяти індивідуальній роботі з учнями;

- зручна організація навчального процесу. Онлайн платформи дозволяють вчителям ефективно організовувати навчальний процес, створювати завдання, тести, збирати відповіді учнів і автоматично оцінювати їх. Це зменшує навантаження на педагога і допомагає вчасно реагувати на потреби учнів;

- гнучкість і доступність. Онлайн платформи забезпечують можливість дистанційного навчання, що дозволяє проводити заняття в будь-який час і з будь-якого місця. Це особливо актуально в умовах пандемії та інших непередбачуваних обставин, коли традиційне навчання стає неможливим;

- підвищення мотивації учнів. Завдяки інтерактивним елементам, таким як вікторини, тести та онлайн ігри, учні стають більш залученими у процес навчання. Онлайн платформи дозволяють створювати завдання, які викликають у дітей інтерес і допомагають засвоїти новий матеріал у цікавій формі.

Онлайн-платформи сприяють формуванню культури безперервного навчання (lifelong learning), розширюють можливості для самоосвіти та підвищення кваліфікації, а також сприяють цифровій трансформації традиційної освіти. З огляду на ці фактори, створення власної освітньої онлайн-платформи з урахуванням сучасних технологій є актуальним напрямом розвитку в сфері інформаційних систем та педагогіки.

1.2 Аналіз існуючих рішень

На сучасному ринку освітніх технологій існує велика кількість платформ, які надають можливість дистанційного або змішаного навчання. У цьому підрозділі розглянемо найбільш поширені рішення, що використовуються в Україні, а також проаналізуємо їх переваги та недоліки з погляду створення власної системи.

1.2.1 Освітній проект «Prosvita»

Prosvita – це сучасний освітній простір для ефективної організації навчального процесу. Ми використовуємо комплексний підхід, щоб забезпечити потреби учнів, батьків, педагогів та освітніх управлінців. Інноваційним рішенням стало впровадження системи «Вчись-Грай-Заробляй» [2].

На сайті prosvita.net є можливість отримати безкоштовний доступ на 3 місяці при реєстрації по номеру телефона і електронній пошті (рисунок 1.1).

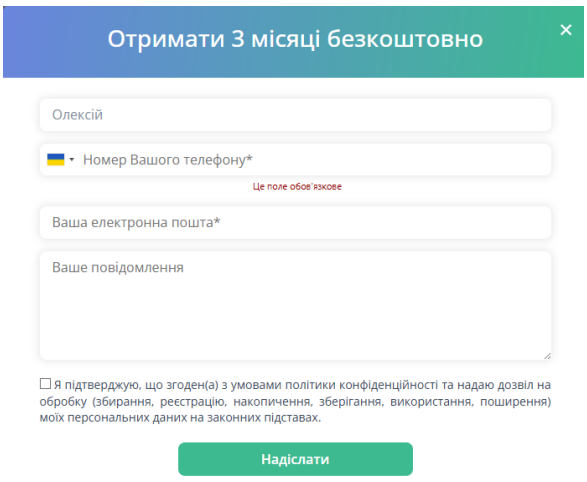

Рисунок 1.1 – Вікно Реєстрації на сервісі Prosvita

Сайт має такі можливості як налаштування розкладу дзвінків, планування уроків, синхронізація з щоденником учнів. Формування

класів/підгруп, заповнення особистої інформації про учнів, відрахування/зарахування, автоматичне переведення до наступного класу.

Одним із недоліків сайту є обов'язкова приналежність до державної, або приватної школи для того щоб створити повноцінний аккаунт (рисунок 1.2).

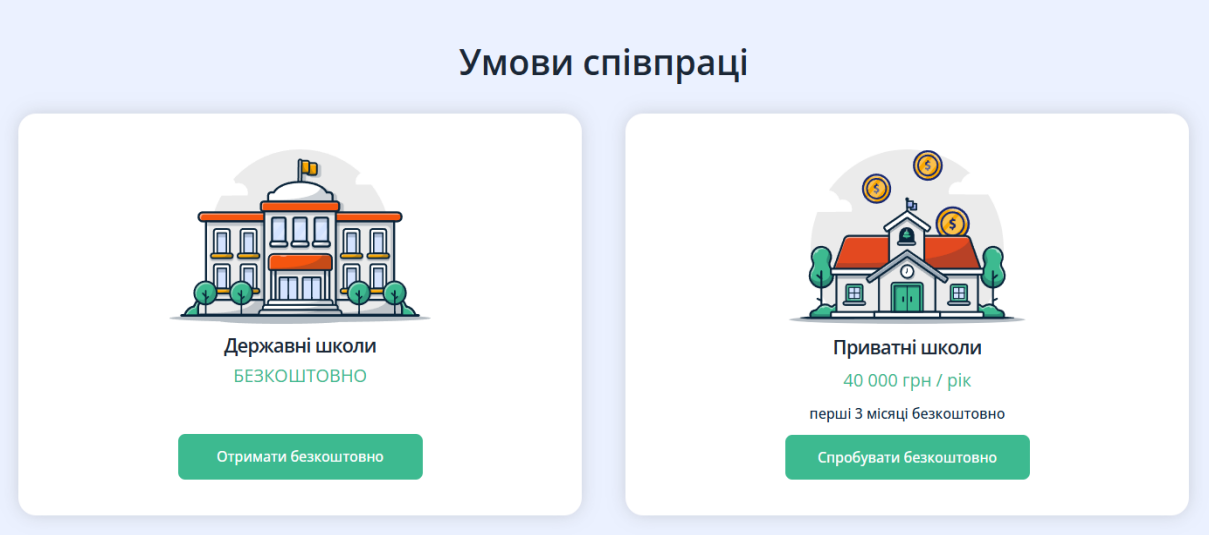

Рисунок 1.2 - Умови співпраці на сервісі Prosvita

З переваг можна відзначити:

- україномовний інтерфейс, що полегшує використання людям без знання англійської мови;

- доступ до навчальних матеріалів згідно з програмою МОН, включаючи типову навчальну програму, що дозволяє автоматизувати освітній процес;

- електронний журнал та щоденник, що забезпечує прозору систему оцінювання.

- Серед суттєвих недоліків веб-застосунку варто зазначити:

- орієнтованість виключно під шкільні навчальні заклади, неможливість використання вищими навчальними або іншими закладами;

- закритий вихідний код, що не дозволяє кастомізувати платформу під конкретні потреби навчальних закладів;

- основні функції доступні лише за платною підпискою, що не дає змоги малозабезпеченим групам людей отримувати якісний рівень освіти.

Загалом, платформа орієнтована на стандарти сучасної української школи та надає можливості для автоматизації та прозорості освітнього процесу. Водночас певні обмеження у доступності та гнучкості використання стримують потенціал платформи для ширшого кола освітніх установ та користувачів.

1.2.2 Освітній проект «На Урок»

Освітній проект «На Урок» – українська цифрова освітня екосистема для роботи та професійного зростання освітян України [3]. Платформа орієнтована на викладачів, школярів та їхніх батьків, забезпечує інструментами для організації дистанційного навчання. Платформа також дає змогу приймати участь у вебінарах, переглядати методичні матеріали, проходити тести (рисунок 1.3).

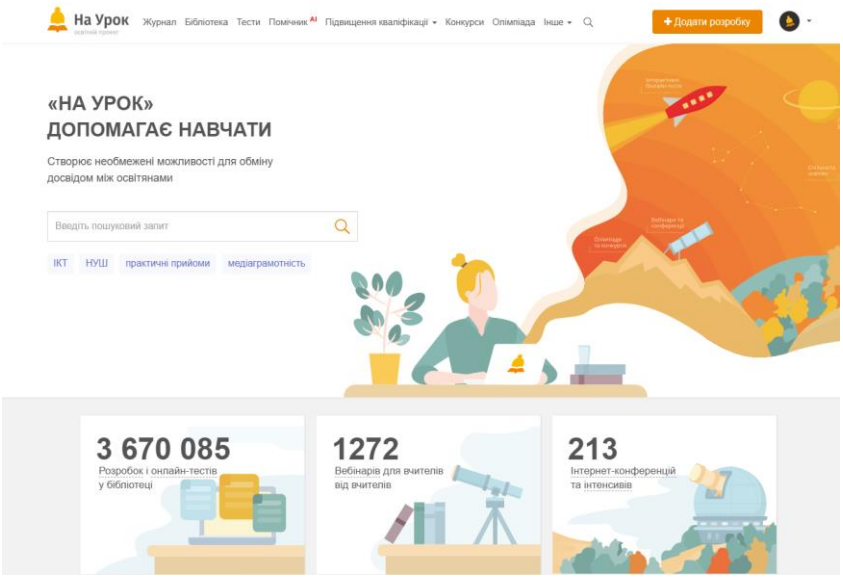

Рисунок 1.3 - Веб-застосунок На Урок

З переваг можна відзначити:

- простий інтерфейс, зручний для користувачів;

- велика база готових тестів. Користувачі можуть дивитися будь-які матеріали незалежно від навчального закладу до якого вони належать;
- можливість створення власних тестів і перевірки знань;
- безкоштовний доступ до основного функціоналу.
- Серед суттєвих недоліків веб-застосунку варто зазначити:
- фокус на шкільній освіті. Платформа здебільшого орієнтована на учнів 4 – 11 класів, тому вона не підійде учням вищих навчальних закладів або самостійним здобувачам освіти;
- немає повноцінної підтримки ролей, наприклад модераторів чи адміністраторів курсів. Вчителям буде складно організувати повноцінну інтеграцію навчального процесу;
- не підтримує складну структуру курсів або лекцій.

Основна мета проєкту – сприяти інтеграції сучасних цифрових технологій у навчальний процес, не вимагаючи від користувачів спеціальних знань або складної технічної підготовки. Однак фокус на шкільній освіті може завадити іншій освітнім групам отримувати якісну освіту.

1.2.3 Освітній проект «Google classroom»

Google Classroom – вебсервіс, створений Google для навчальних закладів з метою спрощення створення, поширення і класифікації завдань безпаперовим шляхом (рисунок 1.4). Основна мета сервісу – прискорити процес поширення файлів між педагогами та здобувачами освіти. Може використовуватися вчителями та учнями у школах, або у закладах вищої освіти викладачами та студентами.

Google Classroom поєднує в собі багато сервісів Google, такі як: Google Диск, Google Документи, Google Таблиці для спрощення та допомоги навчальним закладам перейти до онлайн навчання. Пізніше було інтегровано Google Calendar, для того, щоб допомогти з призначенням термінів виконання завдань, тестів та вебінарів.

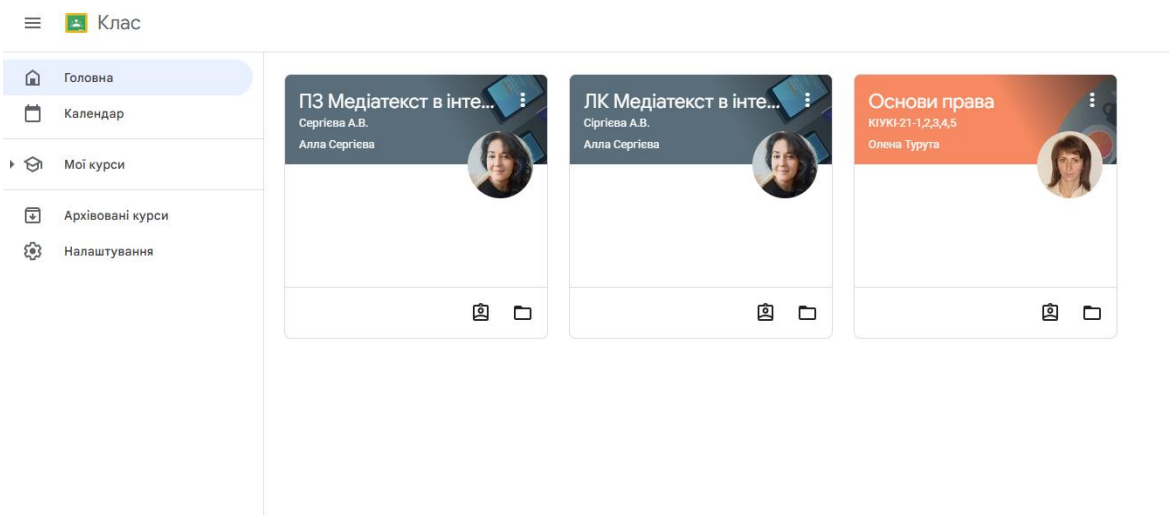

Рисунок 1.4 - Веб-застосунок Google classroom

Classroom базується на принципі, що освітні інструменти повинні бути простими та зручними у використанні, і розроблений для того, щоб дати вчителям більше часу для навчання, а учням – більше часу для навчання [4].

Однак, попри широкі можливості, платформа має і певні недоліки. Для використання Google Classroom необхідний обліковий запис Google та спеціальний аккаунт Google Workspace for Education (рисунок 1.5). Це може бути обмеженням для молодших учнів або шкіл, де не запроваджено централізовану систему акаунтів. Також застосунок не підтримує інтеграцію зі сторонніми сервісами поза екосистемою Google, що може ускладнити інтеграцію у великих або спеціалізованих освітніх проєктах.

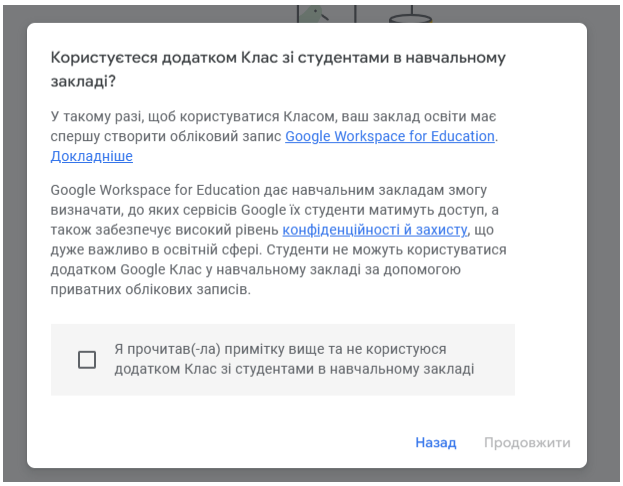

Рисунок 1.5 – Створення облікового запису Google Workspace for Education

Здобувачі освіти, неважливо студенти чи школярі, можуть бути запрошені в клас через базу навчального закладу, за допомогою приватного коду, який потім може бути доданий в користувацький інтерфейс здобувача, або автоматично імпортуватися зі шкільного сайту. Кожен клас, створений за допомогою Google Classroom, створює окрему папку на Google диску викладача, куди здобувач може подати роботу для оцінки.

З переваг можна відзначити:

- зручна інтеграція з сервісами Google. Навчальним закладам які звикли використовувати аккаунти Google буде легко інтегрувати Google Classroom у навчальний процес;
- підтримка мобільних пристроїв;
- хмарне зберігання даних;
- безкоштовний доступ до основного функціоналу.

Серед суттєвих недоліків веб-застосунку варто зазначити:

- інтерфейс не повністю локалізований українською. Хоча більшість функціоналу локалізовано українською мовою, окремі елементи та технічні деталі залишаються англійською мовою, що може створювати певні труднощі для користувачів що не володіють англійською мовою. Зокрема для молодших школярів;
- платформа активно використовує інфраструктуру Google для авторизації, обміну та збереження даних. Хоча для деяких студентів, вчителів та навчальних закладів це буде корисно, іноді це створює труднощі для тих користувачів, які не мають Google акаунта або свідомо не хочуть ним користуватись із міркувань конфіденційності чи цифрової безпеки;
- для створення курсів потрібен спеціальних обліковий запис Google Workspace for Education. Для його реєстарції навчальний заклад має бути перевірен та внесен в реєстр навчальних закладів Google, що може викликати додаткові труднощі для деяких закладів.

Платформа орієнтована на спрощення комунікації між учасниками освітнього процесу та створення цифрового середовища для ефективного

навчання. Однак залежність від інфраструктури Google, необхідність спеціалізованого облікового запису та часткова локалізація можуть обмежити її гнучкість і доступність для окремих категорій користувачів.

1.2.4 Освітній проект «Moodle»

Moodle – це безкоштовна онлайн-система управління навчанням, яка дозволяє викладачам створювати власні приватні веб-сайти з динамічними курсами, що розширюють можливості навчання, будь-коли та будь-де [5].

Moodle використовується для змішаного навчання та дистанційного навчання у школах, університетах, а також на робочих місцях (рисунок 1.6).

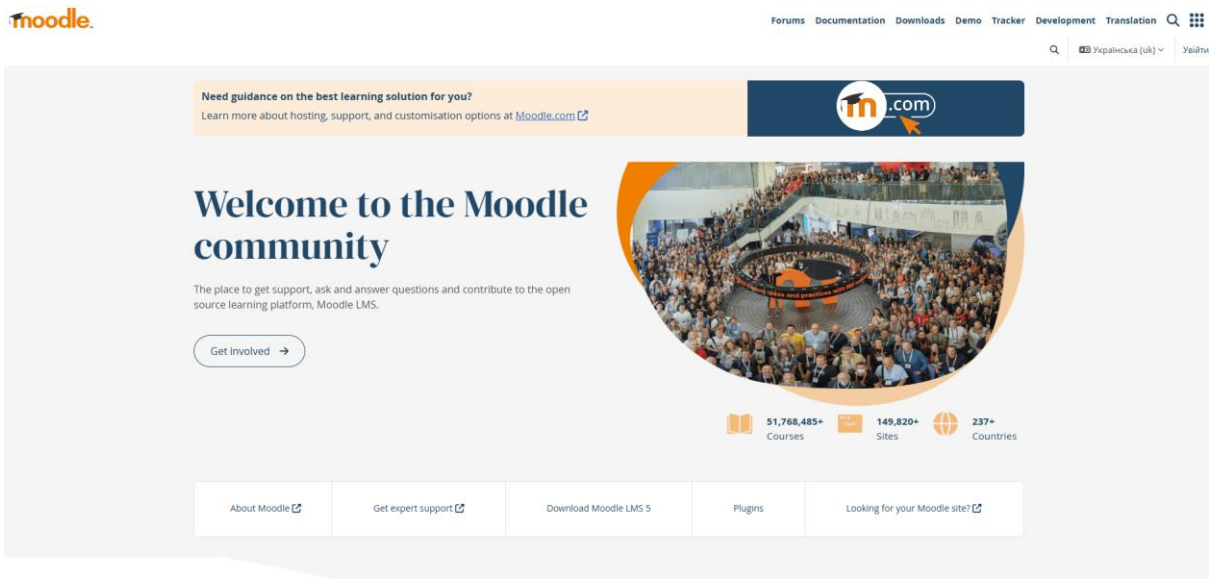

Рисунок 1.6 – Веб-застосунок Moodle

Платформа надає простір для спільної роботи вчителів та студентів. У Moodle є різні можливості для відстеження успішності учнів. Система має гнучкий інтерфейс із можливістю конфігурування макетів та дизайну окремих сторінок. Платформу можна інтегрувати з великою кількістю програмного забезпечення, включаючи інструменти для спілкування, спільної роботи, управління документами та інші програми для підвищення продуктивності.

З переваг можна відзначити:

- відкритий код і безкоштовне використання, що дозволяє налаштовувати систему під конкретні потреби;
- розвинена система ролей і прав доступу;
- підтримка великої кількості плагінів та інтеграцій.

Серед суттєвих недоліків веб-застосунку варто зазначити:

- інтерфейс не повністю локалізований українською. Хоча більшість функціоналу локалізовано українською мовою, окремі елементи та технічні деталі залишаються англійською мовою, що може створювати певні труднощі для користувачів що не володіють англійською мовою. Зокрема для молодших школярів;
- складний інтерфейс для початківців. Якщо викладач або студент не може швидко зорієнтуватися в системі, знайти потрібні функції чи зрозуміти, як завантажити або перевірити завдання, це знижує мотивацію до навчання та використання веб-застосунку Moodle.

Головною перевагою застосунку є адаптивність до потреб конкретного закладу завдяки підтримці розширень, налаштуванням дизайну та ролей користувачів. Утім, складність освоєння інтерфейсу для новачків і потреба в технічному обслуговуванні з боку закладу роблять цю систему менш придатною для організацій, які не мають належної IT-підтримки.

1.2.5 Підсумки аналізу існуючих рішень

У результаті аналізу зазначених освітніх онлайн-платформ було виявлено як сильні сторони, так і недоліки кожної з них. Вивчення можливостей різних платформ дозволило сформувати основні вимоги до власного освітнього веб-застосунку.

Проведений порівняльний аналіз таких поширених веб-сайтів:

- освітній проект «Prosvita»;
- освітній проект «На Урок»;

- освітній проект «Google classroom»;
- освітній проект «Moodle».

Результати порівняльного аналізу свідчать про необхідність створення універсального рішення з гнучкою структурою курсів, простим та зрозумілим дизайном, різними категоріями користувачів та локалізацією українською мовою. А головною умовою повинна бути відкрита та безкоштовна платформа.

1.3 Постановка задачі

При реалізації веб-застосунку першочерговим завданням є визначення ролей користувачів. У веб-застосунку буде роль адміністратора, користувача та гостя.

Гість матиме можливість зареєструватись або увійти в профіль. При доступі до функцій сайту йому буде відмовлено.

Адміністратор матиме можливість створювати, модифікувати, та видаляти курси та навчальні матеріали які знаходяться на сайті. Адміністратор має головну роль у застосунку, а тому повинен створювати нові аккаунти, редагувати та видаляти їх. Також у адміністратора має бути можливість створювати інші аккаунти адміністраторів для подальшого ведення курсів.

Зареєстрований користувач матиме особистий кабінет, в якому зможе переглядати особисті дані та доступні йому курси, переглядати інформацію пов’язану з цими курсами. А також мати можливість зворотнього зв'язку, залишати коментарі до доступних йому курсів і матеріалів та переглядати відгуки на них.

Сайт повинен мати навчальні матеріали до яких користувачі зможуть доступитись по ієрархічній системі курсів. Повинна бути система зворотного відгуку для користувачів, у вигляді коментарів під файлами.

2 АНАЛІЗ ВИКОРИСТОВУВАНИХ ТЕХНОЛОГІЙ

2.1 Огляд засобів розробки

Під час розробки веб-застосунку було використано низку інструментів, мов програмування, бібліотек та фреймворків, які забезпечують зручність розробки. Основною метою вибору цих засобів була потреба реалізувати повноцінний веб-застосунок із підтримкою реєстрації, автентифікації, збереження та обробки даних, а також зручним користувацьким інтерфейсом.

Під час розробки клієнтської частини було використано мову програмування JavaScript, для роботи з веб-інтерфейсами. Для побудови динамічного інтерфейсу було обрано бібліотеку React, яка забезпечує високу продуктивність та зручність розробки. Для оформлення зовнішнього вигляду використовувався фреймворк Bootstrap та Font Awesome для інтеграції іконок.

Серверна частина реалізована за допомогою Node.js з використанням фреймворку Express.js, що дозволило побудувати ефективний REST API. Для зберігання та обробки даних використано реляційну базу даних Microsoft SQL Server (MSSQL). Зв’язок між базою даних і сервером реалізовано через ORM Sequelize, що значно спрощує маніпуляції з базою даних і забезпечує підтримку міграцій. Для забезпечення безпеки автентифікації використано бібліотеки Bcrypt (для хешування паролів) та JsonWebToken (JWT) для генерації маркерів доступу. Всі етапи розробки здійснювались у середовищі розробки WebStorm, яке надає інструменти для ефективного написання, налагодження та тестування коду.

Кожен із перелічених засобів виконує чітко визначену роль у розробці та в сукупності дозволяє створити масштабований, безпечний та зручний у користуванні веб-застосунок.

2.2 Javascript

JavaScript – високорівнева, динамічна, інтерпретована мова програмування, яка добре підходить для об'єктно-орієнтованого та функціонального стилів програмування [6].

Функції JavaScript:

- додавати новий HTML-код на сторінку, змінювати наявний вміст, змінювати стилі;
- реагувати на дії користувача, опрацьовувати натискання миші, переміщення вказівника, натискання на клавіші клавіатури;
- запам’ятовувати дані на стороні клієнта, які будуть доступні в майбутніх сесіях на цьому веб-сайті;
- створювати інтерактивні елементи інтерфейсу: слайдери, модальні вікна, вкладки, спливаючі підказки, таймери, анімації тощо.

Сучасний JavaScript – це «безпечна» мова програмування. Вона не надає низькорівневого доступу до пам’яті чи процесора, оскільки була створена для браузерів, які цього не потребують. Можливості JavaScript значно залежать від середовища, у якому виконується скрипт. Вбудована в браузер JavaScript може робити все, що пов’язано з управлінням веб-сторінками, взаємодією з користувачем та веб-сервером [7].

2.3 ExpressJS

Express – це мінімальний та гнучкий фреймворк Node.js, який надає надійний набір функцій для веб та мобільних застосунків [8].

Функції Express:

- надійна маршрутизація;
- допомога в HTTP запитах(перенаправлення, кешування тощо);
- асинхронне програмування.

Express забезпечує мінімальну та гнучку структуру для створення веб-

серверів та API. Він дозволяє обробку маршрутів, інтеграцію проміжного програмного забезпечення та управління HTTP-запитами/відповідями, що робить його потужною основою для RESTful API-застосунків.

2.4 Sequelize

Sequelize – це сучасний ORM на TypeScript та Node.js для Oracle, Postgres, MySQL, MariaDB, SQLite та SQL Server, а також інших. Він має надійну підтримку транзакцій, зв'язків, активного та відкладеного завантаження, реплікації читання та багато іншого [9].

Sequelize – це об'єктно-реляційний маппер (ORM) для баз даних SQL. Він спрощує взаємодію з базами даних, дозволяючи розробникам визначати моделі в JavaScript, які представляють таблиці. Він підтримує перевірку даних, зв'язки (асоціації) та запити без написання необробленого SQL.

Sequelize також підтримує міграції для створення та зміни бази даних, а також так звані «сідери» для початкового заповнення даними. Sequelize розроблений як незалежна від бази даних, тобто він може працювати з різними механізмами баз даних, такими як My SQL, MS SQL та багатьма іншими.

2.5 Bcrypt

Bcrypt використовується для безпечного хешування паролів користувачів для їх зберігання в базі даних [10]. Він також надає методи порівняння паролів у звичайному тексті з хешованими версіями під час входу. Це критично важливий компонент для забезпечення безпечної автентифікації та захисту облікових даних користувачів від витоків або порушень. Навіть якщо база даних скомпрометована, зловмисник не зможе відновити оригінальні паролі.

2.6 Jsonwebtoken

Jsonwebtoken – бібліотека, яка реалізує функціональність JSON Web Token (JWT). Токени JWT використовуються для автентифікації користувачів після входу в систему. Сервер видає токен, який клієнти включають у наступні запити, що дозволяє безпечну обробку сеансів без збереження стану та зберігання сеансів користувачів на сервері [11].

2.7 React

React – популярна бібліотека, яка використовується для створення інтерфейсів користувача. Вона була створена у Facebook з метою вирішити проблеми, пов'язані з великомасштабними сайтами, керованими даними [12].

React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. Він обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC).

React підтримує віртуальний DOM, а не покладається виключно на DOM браузера [13]. Це дозволяє бібліотеці визначити, які частини DOM змінилися, порівняно зі збереженою версією віртуального DOM, і таким чином визначити, як найефективніше оновити DOM браузера. Таким чином програміст працює зі сторінкою, вважаючи що вона оновлюється вся, але бібліотека самостійно вирішує які компоненти сторінки треба оновити.

2.8 Середовище для розробки WebStorm

JetBrains WebStorm – інтегроване середовище розробки JavaScript, CSS та HTML від компанії JetBrains, розроблене на основі платформи IntelliJ IDEA. WebStorm добре розуміє структуру проектів та допомагає з будь-

якими аспектами написання коду. Має автодоповнення коду, безпечний рефакторинг, постійний пошук потенційних проблем та підказки щодо їх виправлення. Не потрібно встановлювати та налаштовувати безліч плагінів – можна відразу починати писати код.

У WebStorm із коробки є все необхідне для розробки на JavaScript та TypeScript. Для індивідуального налаштування IDE можна використовувати різноманітні плагіни та додаткові параметри [14].

Основний перелік можливостей:

- інтеграція з системами управління версіями Git, GitHub підтримуються за замовченням з можливістю побудови списку змін і відкладених змін;
- інтеграція з системами відстеження помилок;
- модифікація файлів .css, .html, .js з одночасним переглядом результатів;
- підтримує такі інструменти розробки як: Angular, React, Vue.js, Node.js.

Усі необхідні інструменти зібрані разом. IDE можна використовувати для налагодження та тестування клієнтського коду та програм на Node.js, а також для роботи з Git та GitHub. Крім того, доступні менеджери залежностей, інструменти збірки, Prettier, а також термінал [15].

2.9 БД Microsoft SQL Server

Microsoft SQL Server (MSSQL) – це ефективна система управління реляційними базами даних. Виступаючи як сервер баз даних, MSSQL пропонує безпечну масштабовану платформу для зберігання, пошуку та маніпулювання даними відповідно до потреб різноманітних програмних додатків [16].

Такі додатки можуть працювати як локально на тому ж самому комп'ютері, так і віддалено через мережу, включаючи Інтернет. Ця

універсальність розширює можливості бізнесу для ефективного керування великими наборами і складними структурами даних у бізнес середовищі.

2.10 Bootstrap

Bootstrap – це безкоштовний фреймворк CSS з відкритим кодом, який допомагає розробникам ефективніше створювати адаптивні веб-сайти та веб-додатки, орієнтовані на мобільні пристрої [17]. Він надає колекцію попередньо розроблених компонентів HTML, CSS та JavaScript, таких як сітки, навігація, кнопки та форми, які можна легко інтегрувати в проект. Це дозволяє розробникам швидко створювати візуально привабливі та узгоджені веб-інтерфейси без необхідності писати весь CSS та JavaScript з нуля [18].

2.11 Font Awesome

Font Awesome – це популярна бібліотека іконок з відкритим вихідним кодом, яка широко використовується у веб-розробці для додавання масштабованих векторних іконок на сайти та у додатки [19]. Бібліотека забезпечує величезний набір сучасних, стильних іконок, які можна легко стилізувати за допомогою CSS і використовувати як шрифти або SVG.

Це потужна та популярна бібліотека для веб-розробників, що дозволяє легко і красиво оформлювати інтерфейси, економить час на створення власних іконок, а також гарантує сумісність і якість на всіх пристроях.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Структура даних MSSQL

У розробці програмного забезпечення для зберігання та обробки даних було обрано Microsoft SQL Server (MSSQL) – сучасну, масштабовану та продуктивну систему керування реляційними базами даних, розроблену компанією Microsoft. MSSQL широко використовується як у великих корпоративних, так і у середніх або малих IT системах завдяки надійності, потужному функціоналу, засобів безпеки та гнучкості масштабування.

У розробленому додатку структура бази даних MSSQL складається з взаємопов’язаних таблиць, що відповідають основним сутностям предметної області.

3.1.1 Міграції

У фреймворку Sequelize для створення та оновлення структури бази даних використовуються міграції. Їх виконання здійснюється через відповідні команди CLI, зокрема npx sequelize-cli db:migrate. Міграції містять два основні методи – up() для створення або модифікації таблиць, і down() для скасування змін (листинг 3.1).

У даній міграції створюється таблиця User, яка містить поля для зберігання імені користувача, паролю, ролі та особистих даних (ім’я та прізвище). Також використовується обмеження унікальності для username, що забезпечує унікальність логінів у системі.

Скасування міграції можливе за допомогою команди npx sequelize-cli db:migrate:undo, яка викликає метод down() і видаляє таблицю User.

Усі файли міграцій зберігаються у директорії migrations, що відповідає структурі проєкту Sequelize.

Лістинг 3.1 – Функції up та down для створення та видалення таблиці користувачів

```
await queryInterface.createTable('User', {
  id: {
    type: Sequelize.INTEGER,
    primaryKey: true,
    autoIncrement: true,
  },
  username: {
    type: Sequelize.STRING(255),
    allowNull: false,
    unique: true
  },
  password: {
    type: Sequelize.STRING(255),
    allowNull: false,
  },
  role: {
    type: Sequelize.STRING(10),
    allowNull: false
  },
  firstName: {
    type: Sequelize.STRING(100),
    allowNull: false
  },
  lastName: {
    type: Sequelize.STRING(100),
    allowNull: false
  },
});
```

3.1.2 Моделі

У Sequelize для зберігання інформації про користувачів, категорії, вміст і коментарі використовується низка таблиць.

Таблиця User містить інформацію про користувачів. Їхні ідентифікатори, імена, паролі та ішні необхідні дані системи.

Основні поля таблиці:

- id – унікальний ідентифікатор користувача (первинний ключ);
- username – унікальне ім’я користувача, яке є дійсною email-адресою;
- password – хеш пароля. Система використовує алгоритм Blowfish із

додаванням випадкової «солі» для кожного пароля, що унеможливлює розпізнавання навіть однакових паролів;

- role – роль користувача, може мати значення “Admin” або “User”;
- firstName – ім’я користувача;
- lastName – прізвище користувача.

Таблиця Category містить категорії, які використовуються для структуризації навчального контенту. Основні поля:

- id – унікальний ідентифікатор категорії;
- name – унікальна назва категорії;
- description – докладний опис категорії;
- parentId – необов’язкове посилання на батьківську категорію, що дозволяє створювати ієрархію;
- userId – необов’язкове посилання на користувача, який створив категорію (встановлюється в NULL у разі видалення користувача);
- createdAt – дата і час створення категорії.

Таблиця Content містить метадані файлів навчального контенту. Видаляється разом із пов’язаною категорією. Основні поля:

- id – унікальний ідентифікатор вмісту (збігається з іменем файлу на файловому сховищі);
- path – шлях до файлу у файловій системі (встановлюється автоматично сервером);
- name – оригінальна назва файлу;
- description – розширений опис вмісту;
- mimeType – MIME-тип файлу, що дозволяє визначити застосунок для відкриття;
- size – розмір файлу у байтах;
- userId – ID користувача, що створив файл (встановлюється в NULL при видаленні користувача);
- categoryId – ID категорії, до якої належить файл;
- createdAt – дата та час створення вмісту.

Таблиця ContentComment містить коментарі користувачів до контенту. Видаляється разом із пов’язаним вмістом. Основні поля:

- id – унікальний ідентифікатор коментаря;
- text – текст коментаря;
- contentId – ID контенту, до якого належить коментар;
- userId – ID користувача, який створив коментар (встановлюється в NULL у разі видалення користувача);
- createdAt – дата та час створення коментаря.

Кожна з цих таблиць має чітко визначену структуру з унікальними ідентифікаторами, типами даних, зв’язками між таблицями та датами створення записів (рисунок 3.1).

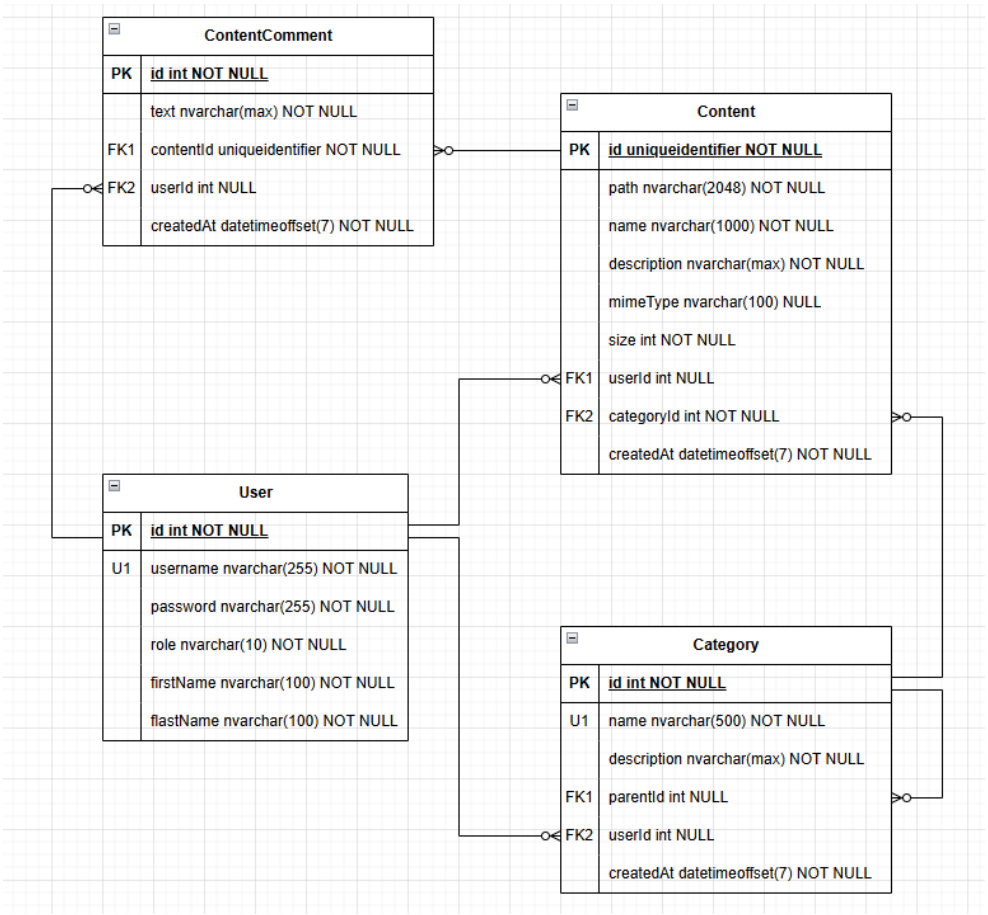

Рисунок 3.1 – Схема бази даних

Таке моделювання забезпечує надійне збереження даних та логічну організацію контенту у застосунку.

3.2 Контролери

У процесі реалізації серверної логіки для категорій у проєкті було створено окремий контролери , які відповідають за обробку HTTP-запитів, пов’язаних із створенням, редагуванням, видаленням та переглядом.

3.2.1 Контролер Category

Функції, реалізовані у контролері:

- createCategory – створює нову категорію. Метод перевіряє авторизацію користувача та створює новий запис у базі даних на основі отриманих з тіла запиту параметрів: name, description та parentId.
- getCategory – дозволяє отримати категорію за її id. У запиті також підключається пов’язаний об'єкт User, який створив категорію (через alias creator), з полями id та username.
- listCategories – повертає список категорій з урахуванням фільтрації за parentId та userId, якщо вони вказані у запиті. Також при вибірці включаються зв’язки з автором (creator) та батьківською категорією (parent), щоб забезпечити повну структуру для побудови ієрархічного дерева категорій.
- updateCategory – дозволяє оновити дані категорії. Перед оновленням перевіряється, чи є поточний користувач автором категорії або адміністратором. Також додається перевірка, що категорія не може бути батьківською самій собі.
- deleteCategory – видаляє категорію за її id, попередньо перевіривши права доступу. Видалення здійснюється за допомогою відповідного методу сервісу.

Контролер відповідає за керування категоріями у системі, обробку запитів, взаємодію з бізнес-логікою та базою даних, а також за формування відповідей клієнту.

3.2.2 Контролер Content

Функції, реалізовані у контролері:

- createContent – створення нового контенту. Перевіряється авторизація користувача і наявність завантаженого файлу. Для кожного файлу генерується унікальний ідентифікатор uuidv4(), створюється директорія за категорією, у якій зберігається файл з унікальним ім’ям. У базі даних зберігаються метадані: ім’я файлу, опис, користувач-власник, категорія, шлях до файлу, MIME-тип і розмір.

- getContent – отримання метаданих контенту за ідентифікатором, разом з інформацією про користувача, який його створив (creator).

- getContentFile – віддача фізичного файлу клієнту. Перевіряється існування файлу на диску, встановлюються коректні HTTP-заголовки (Content-Type та Content-Disposition) для коректного відображення/завантаження файлу.

- listContents – отримання списку контенту з можливістю фільтрації за categoryId та userId. При вибірці також підключаються зв’язки з творцем (creator) і категорією (category) для зручного формування інформації на клієнтській частині.

- updateContent – оновлення метаданих контенту (ім’я, опис, категорія). Перевіряється, чи користувач має права власника або адміністратора.

- deleteContent – видалення контенту. Перевіряються права користувача. Фізичне видалення файлу і запису виконується у сервісі contentService.

- createContentComment – створення нового коментаря до контенту. Перед створенням перевіряється існування контенту.

Контролер відповідає за управління контентом (файлами та їх метаданими) у системі.

3.2.3 Контролер User

Контролер користувачів відповідає за обробку запитів, пов’язаних із реєстрацією, авторизацією, отриманням та оновленням профілю, а також видаленням користувачів у веб-застосунку.

Функції, реалізовані у контролері:

- registerUser – реалізує логіку реєстрації нового користувача. Проводить валідацію вхідних даних, хешує пароль та зберігає нового користувача у базі даних.

- loginUser(req, res) – відповідає за аутентифікацію користувача. Перевіряє відповідність email і пароля, а при успішній перевірці видає JWT токен для подальшої авторизації у системі.

- getUserProfile(req, res) – повертає інформацію про поточного аутентифікованого користувача, використовуючи дані, що зберігаються у токені або сесії.

- updateUser(req, res) – забезпечує оновлення інформації користувача, такої як ім’я, email або пароль. Передбачає валідацію і захист від некоректних або небезпечних змін.

- deleteUser(req, res) – реалізує логіку видалення акаунту користувача із системи.

Ці функції забезпечують взаємодію між моделлю користувача та відповідними уявленнями або API-відповідями.

3.3 Клієнтська логіка

У сучасних веб-застосунках, клієнтська логіка відіграє ключову роль у забезпеченні інтерактивності та зручності користувача. Вона відповідає за управління станом застосунку, обробку автентифікації, взаємодію з сервером через API, а також за синхронізацію даних між різними компонентами інтерфейсу в режимі реального часу.

3.3.1 Клієнтський UserAPI

UserAPI реалізує асинхронну функцію loginUser(), яка відповідає за процес авторизації користувача на сервері (лістинг 3.2).

Лістінг коду 3.2 – Функція loginUser

```
export async function loginUser(formData) {

    try {
        const response = await axios.post(
            `${process.env.REACT_APP_API_BASE_URL}/user/login`,
            formData,
            {headers: {'Content-Type': 'application/json'}}
        );
        return {
            status: response.status,
            success: true,
            message: "Success",
            token: response.data.token
        }
    } catch (error) {
        return {
            status: error.response.status,
            success: false,
            message: error.response.data.error
        }
    }
}
```

Функція приймає об'єкт formData, що містить дані для входу, наприклад, електронну адресу та пароль, і відправляє їх у вигляді POST-запиту на кінцеву точку /user/login. Запит відправляється з заголовком Content-Type: application/json, що вказує серверу на формат передаваних даних. Якщо запит проходить успішно, функція повертає об'єкт, у якому міститься HTTP-статус відповіді, флагом успішності, повідомлення про успішне виконання та отриманий від сервера токен аутентифікації. У випадку виникнення помилки, наприклад, через неправильні облікові дані або проблеми на сервері, функція опрацьовує виключення і повертає об'єкт із кодом помилки, флагом помилки та текстом повідомлення про помилку,

отриманим від сервера. Таким чином, ця функція забезпечує стандартизований спосіб комунікації між клієнтом і сервером під час процесу входу користувача, спрощуючи обробку результатів аутентифікації в React-застосунку.

3.3.2 Клієнтський ContentAPI

Цей файл містить набір асинхронних функцій для роботи з контентом клієнта.

Функція getContent приймає ідентифікатор категорії categoryId і відправляє GET-запит на сервер для отримання списку контенту, пов’язаного з цією категорією. В запиті додається авторизаційний токен, що зберігається у localStorage, у заголовках. При успішній відповіді функція повертає статус, прапорець успіху, повідомлення та отримані дані. У разі помилки повертається відповідний статус та повідомлення про помилку.

Функція downloadFile відповідає за завантаження файлу за його ідентифікатором fileId (лістинг 3.3).

Лістинг 3.3 – Функція downloadFile

```
export async function downloadFile(fileId) {
    try {
        const res = await axios.get(buildFileUrl(fileId), {
            headers: getAuthHeaders(),
            responseType: 'blob'
        });

        const fileName =
getFileNameFromHeader(res.headers['content-disposition'], `file-
${fileId}`);
        saveBlobAsFile(new Blob([res.data]), fileName);

        return { status: res.status, success: true, message:
"File downloaded successfully", data: null };
    } catch (e) {
        return handleDownloadError(e);
    }
}
```

Функція надсилає GET-запит із заголовком авторизації і встановлює responseType у blob для отримання двійкових даних файлу. Після отримання відповіді визначає ім’я файлу із заголовка content-disposition і створює тимчасове посилання для завантаження файлу у браузері. Файл автоматично завантажується користувачу. Якщо операція завершується помилкою, функція повертає інформацію про неуспіх.

Функція deleteFile видаляє файл з серверу за його fileId, надсилаючи DELETE-запит із авторизаційним токеном у заголовках. Після успішного видалення повертається повідомлення про успіх, а у випадку помилки – повідомлення з описом проблеми.

Функція uploadFile дозволяє завантажити файл на сервер, пов’язаний із певною категорією та описом. Для цього створюється об’єкт FormData, куди додаються файл, ідентифікатор категорії та опис. Потім виконується POST-запит з відповідними заголовками, включаючи multipart/form-data для коректної передачі файлів. Після успішного завантаження функція повертає статус, повідомлення і отримані дані від сервера, а при помилці – відповідний опис проблеми.

Усі функції інтегровані з механізмом автентифікації через токен, що гарантує доступ лише авторизованим користувачам. Вони забезпечують повний цикл роботи з контентом на платформі: перегляд, завантаження, видалення та додавання файлів.

3.3.3 Компонент AuthContext

Компонент AuthContext реалізує систему управління автентифікацією користувача в React-застосунку за допомогою контексту (лістинг 3.4). У ньому створюється контекст з початковими значеннями, що відображають стан користувача: чи він увійшов у систему, яка у нього роль, а також містяться функції для входу, виходу та отримання ідентифікатора користувача.

Лістинг 3.4 – Контекст AuthContext

```
export const AuthContext = createContext({
    isLoggedIn: false,
    role: String,
    loginContext: () => {},
    logoutContext: () => {},
    getId: () => {}
});
```

Для роботи з токеном аутентифікації, який зберігається у localStorage, визначені допоміжні функції: getToken перевіряє наявність і валідність токена за терміном дії, getId бере ідентифікатор користувача з токена, а getRole отримує роль користувача або повертає "Guest", якщо токен відсутній чи недійсний. Компонент AuthProvider підтримує у собі стани isLoggedIn і role, які оновлюються за допомогою функції refreshAuthState. Ця функція викликається після зміни токена, що дозволяє синхронізувати стан автентифікації з даними у localStorage. За допомогою хука useEffect додано обробник події "storage", щоб реагувати на зміни токена, якщо вони відбуваються в інших вкладках браузера. Функції loginContext і logoutContext керують записом і видаленням токена у сховище, відповідно оновлюючи стан автентифікації (лістинг 3.5).

Лістинг 3.5 – Функції loginContext і logoutContext

```
const loginContext = useCallback((newToken) => {
        localStorage.setItem("token", newToken);
        refreshAuthState();
    }, [refreshAuthState]);
const logoutContext = useCallback(() => {
        localStorage.removeItem("token");
        refreshAuthState();
    }, [refreshAuthState]);
```

За допомогою useMemo створюється об'єкт значення контексту, що включає актуальні дані та методи, і передається в AuthContext.Provider.

4 ІНСТРУКЦІЯ КОРИСТУВАЧА

4.1 Звичайний користувач

Коли користувач переходить до застосунку, відображається головна сторінка з інструкцією для користувача (рисунок 4.1).

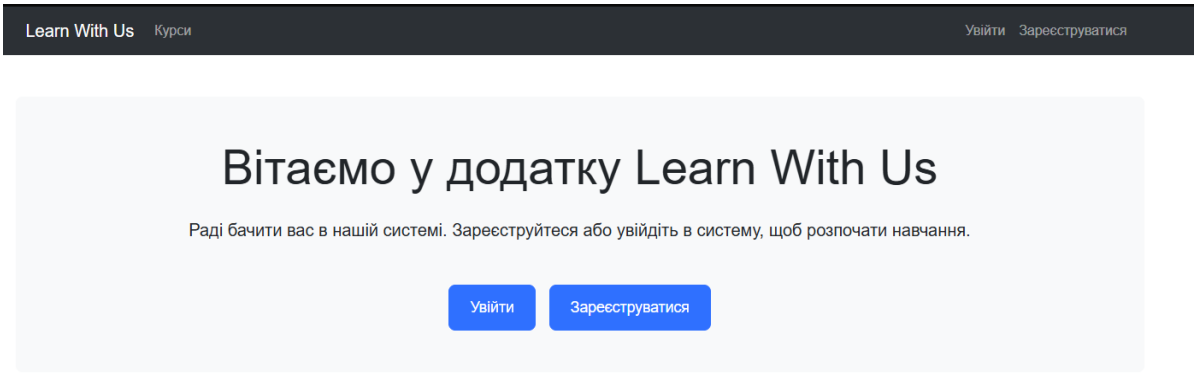

Рисунок 4.1 – Головна сторінка веб-застосунку

При натисканні на кнопку «Зареєструватися» користувач переходить у вікно реєстрації (рисунок 4.2). Після цього йому треба ввести обов’язкові дані. Це вікно має перевірку на коректність введеної інформації.

Введені дані:

- ім’я має бути не менш ніж 3 символи та не містити в собі знаків откріім букв кирилиці або латиниці.
- прізвище має бути не менш ніж 3 символи та не містити в собі знаків откріім букв кирилиці або латиниці.
- email адреса є обов’язковою та має бути у правильному форматі (наприклад: name@domain.com)
- пароль повинен бути не менш ніж 3 символи

Після введення даних, якщо вони пройшли перевірку, користувач зможе натиснути кнопку «Підтвердити».

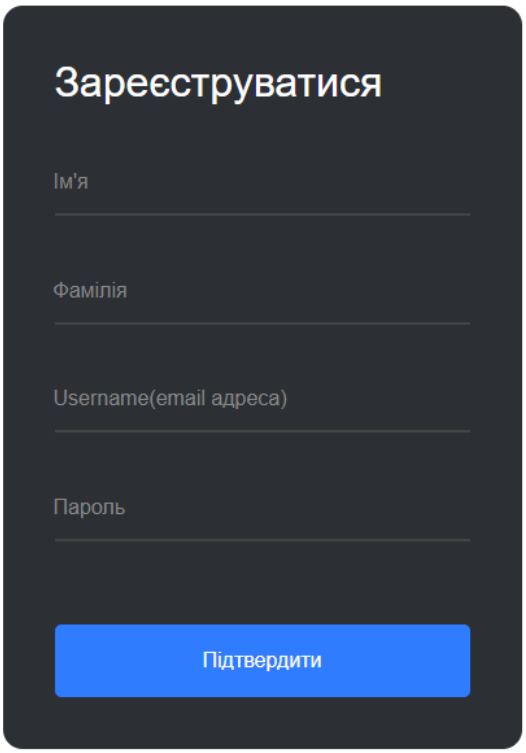

Рисунок 4.2 – Вікно реєстрації

Якщо користувач вже має створений профіль то йому треба перейти на сторінку аутентифікації (рисунок 4.3).

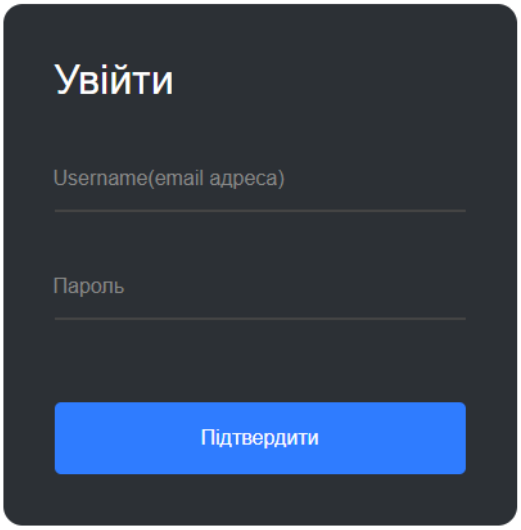

Рисунок 4.3 – Вікно аутентифікації

Користувачу треба ввести email адресу та пароль від існуючого акаунта, після цього натиснути кнопку «Підтвердити».

4.2 Зареєстрований користувач

При переході на головну сторінку сайту, користувач може переглянути нещодавно відкриті курси або завантажені файли (рисунок 4.4).

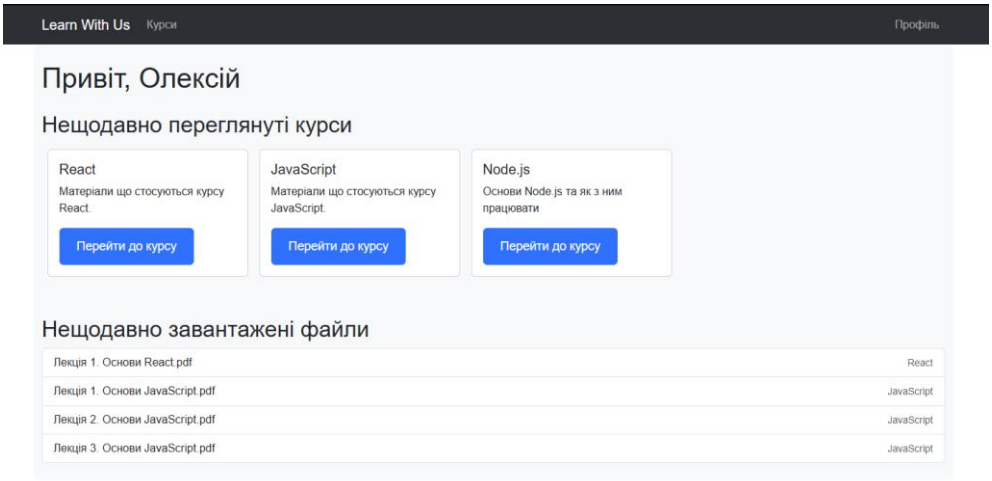

Рисунок 4.4 – Головна сторінка веб-застосунку

Якщо користувач хоче переглянути доступні йому курси, він може натиснути відповідну кнопку «Курси» у панелі навігації. Після цього він попадає на сторінку де знаходиться перелік курсів (рисунок 4.5).

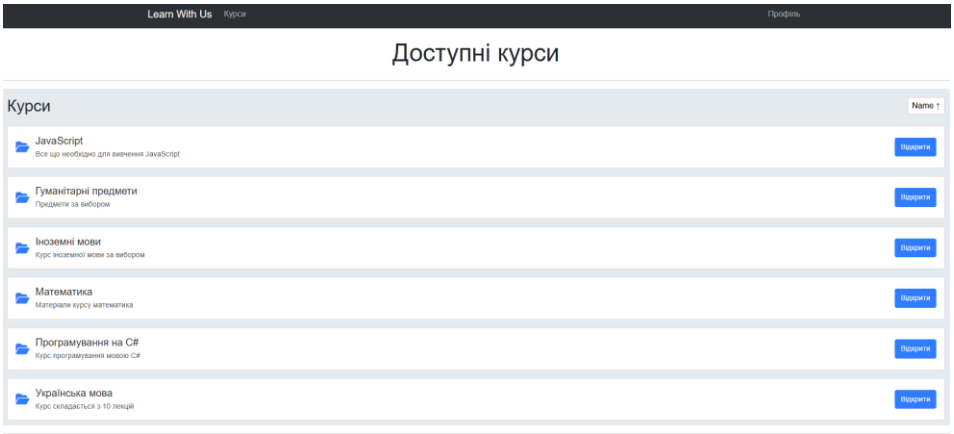

Рисунок 4.5 – Перелік доступних курсів

Користувач може переглянути назву кожного розділу та його опис. Кожен курс може мати в собі інші курси, або файли які відносяться до

обраного розділу. Також курси можна відсортувати за алфавітним порядком та в зворотному алфавітному порядку (рисунок 4.6).

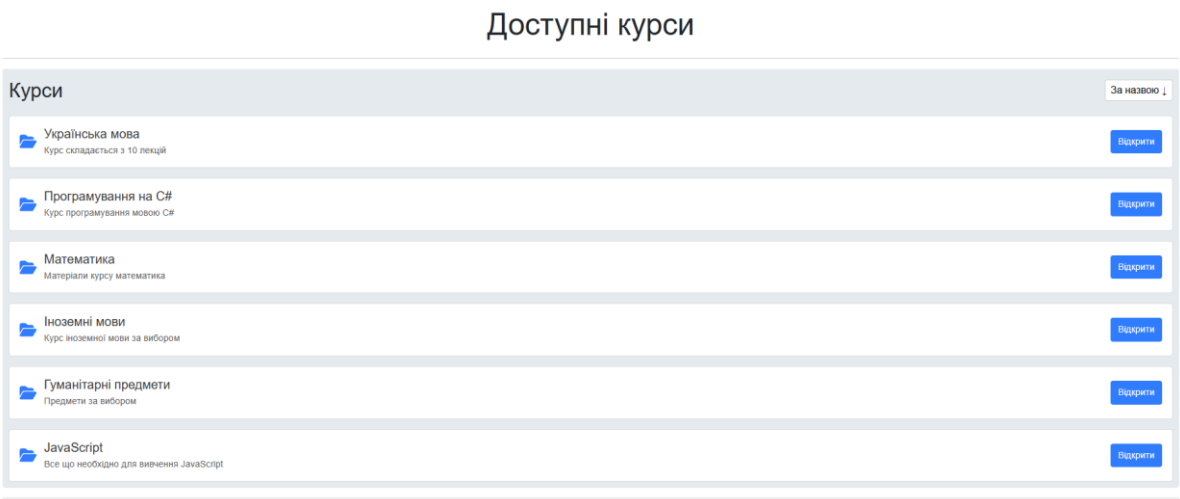

Рисунок 4.6 – Список курсів відсортований в зворотному порядку

При переході в наступну категорію відкриваються вкладені курси, якщо вони там є. Зверху написана поточна категорія в якій знаходиться користувач. Знизу під курсами відображається список файлів та інформація про файл (рисунок 4.7).

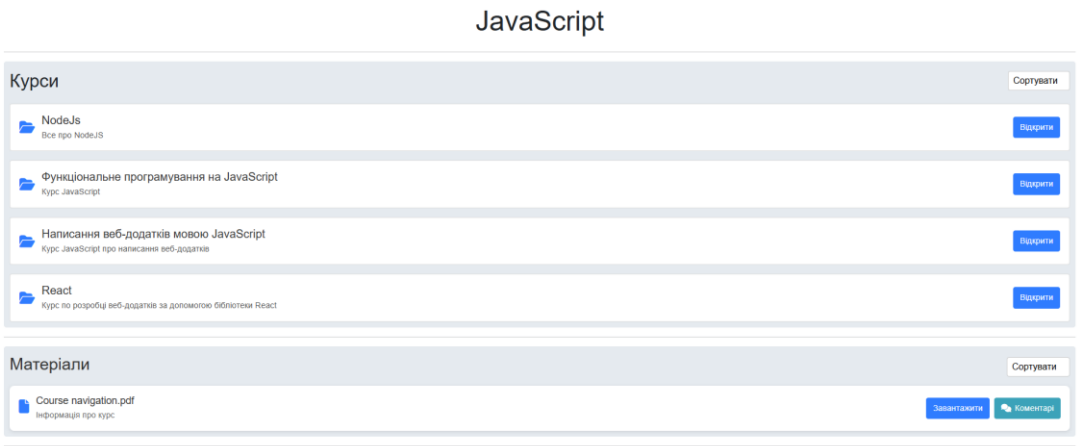

Рисунок 4.7 – Вкладена категорія

При переході в курс у якого нема вкладених курсів, є тільки список наявних файлів (рисунок 4.8).

Рисунок 4.8 – Матеріали курсу

Користувач може відсортувати ці курси по алфавітному, зворотному алфавітному порядку, а також по типу файлу та зворотному типу файлу. Для цього треба натиснути кнопку сортувати і обрати потрібний користувачу варіант (рисунок 4.9).

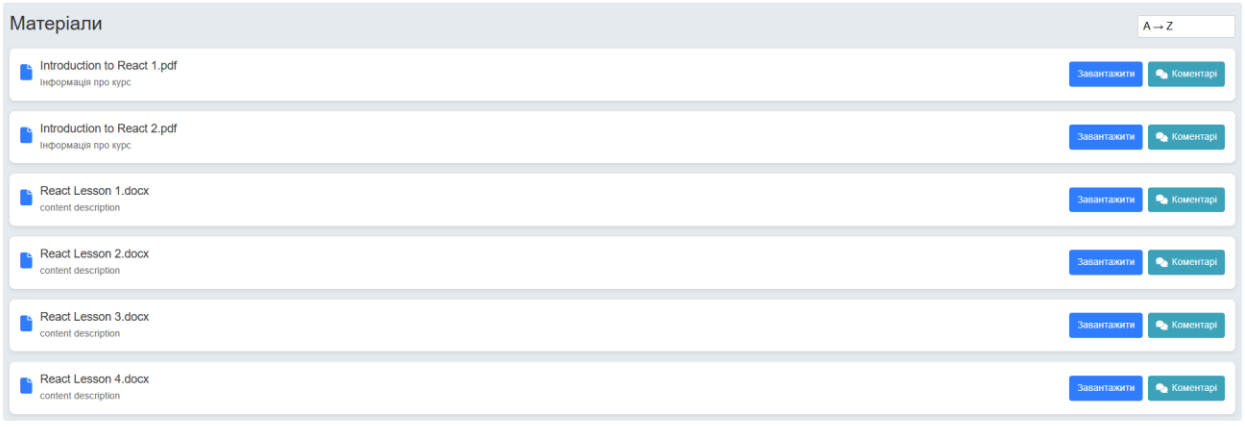

Рисунок 4.9 – Відсортовані файли в алфавітному порядку

Також є можливість завантажити файл, для цього треба натиснути відповідну кнопку напроти файлу. При цьому файл завантажується на пристрій і його можна переглянути (рисунок 4.10).

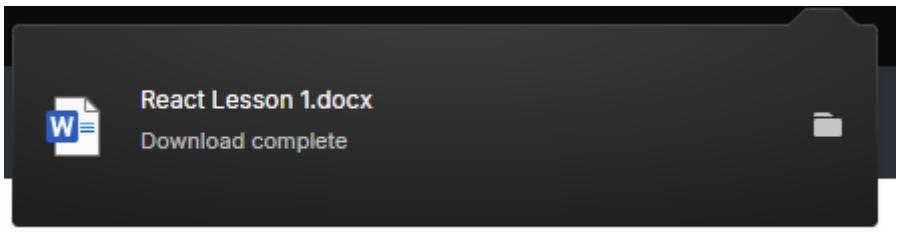

Рисунок 4.10 – Завантажений файл

При натисканні на кнопку «Коментарі» відкриється форма з відгуками інрших студентів на цей файл. Кожен з відгуків має Ім’я та Фамілію студента та відповідний текст коментаря. За бажанням користувач може сам залишити коментар, для цього треба ввести його в поле для відгуку та натиснути «Додати коментар» (рисунок 4.11).

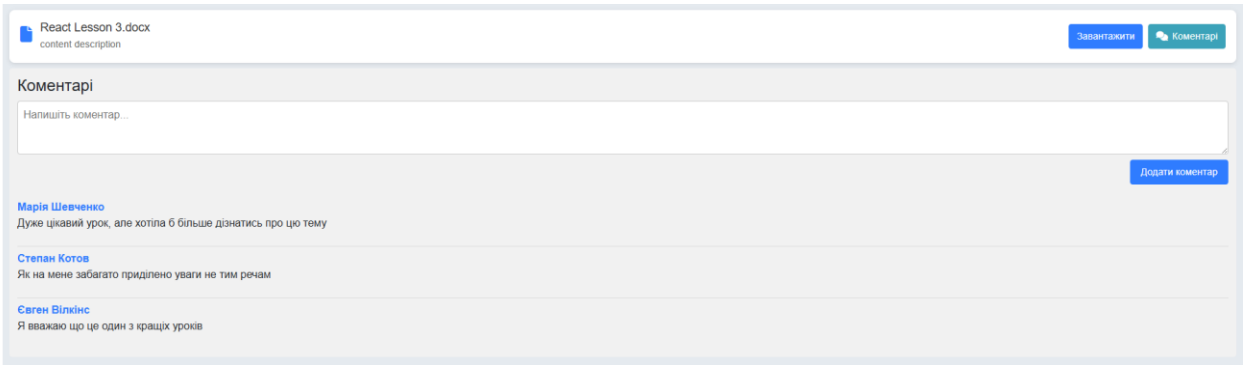

Рисунок 4.11 – Розділ коментарів

Для перегляду інформації про свій профіль, користувачу необхідно натиснути кнопку «Профіль», розташовану у верхній навігаційній панелі сайту. Після цього система автоматично перенаправить користувача на спеціальну сторінку, де відображатиметься детальна інформація про його обліковий запис (рисунок 4.12). Буде вказано його ім’я, прізвище, пошта та роль. При натисканні кнопки «Вийти» користувач вийде зі свого профілю і втратить доступ до функцій сайту, допоки знову не авторизується в системі. Це забезпечує безпеку та контроль доступу до особистої інформації і функціоналу платформи.

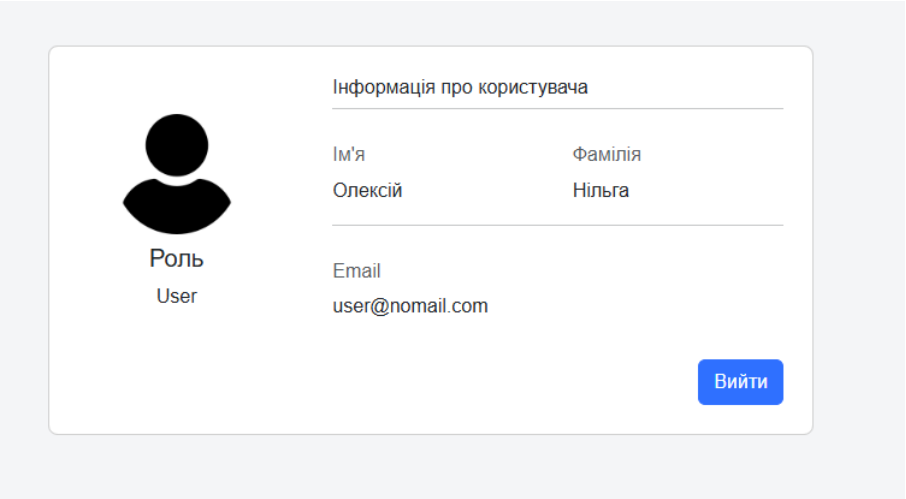

Рисунок 4.12 – Профіль користувача

4.3 Адміністратор застосунку

Адміністратор застосунку має головну роль, отже у нього в навігаційній панелі з’являється вкладка «Користувачі». При переході на цю сторінку, адміністратор може переглядати наявних користувачів на сайті (рисунок 4.13).

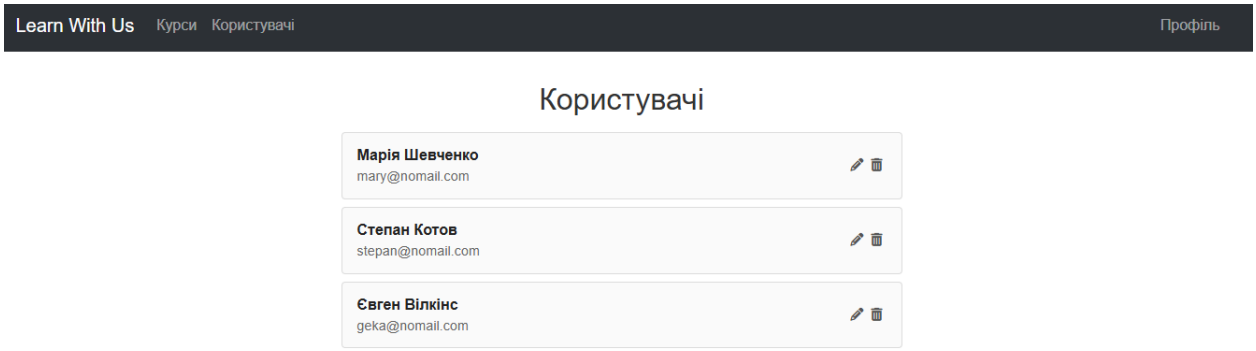

Рисунок 4.13 – Сторінка з користувачами

Також адміністратор може змінювати дані акаунтів інших користувачів та видаляти їх облікові записи (рисунок 4.14).

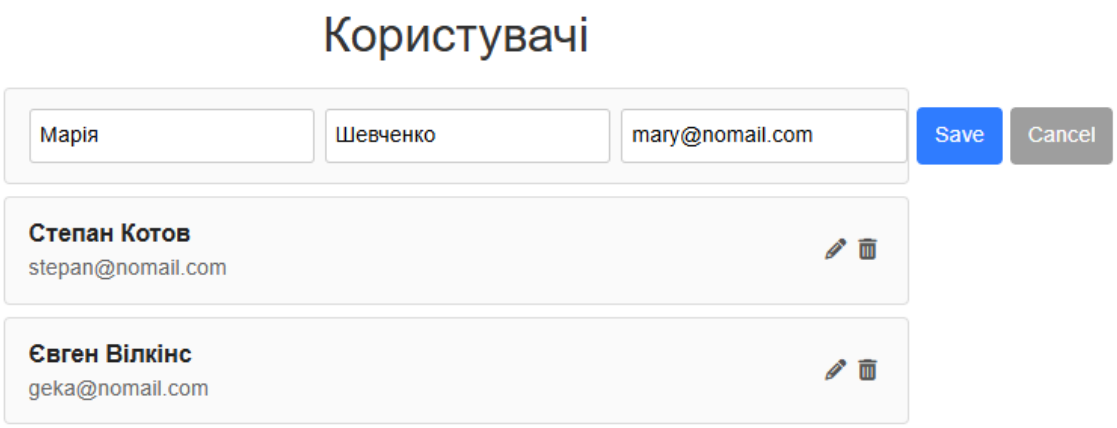

Рисунок 4.14 – Змінення даних користувача

На сторінці з курсами у адміністратора з’являється функція додавання нової категорії (рисунок 4.15).

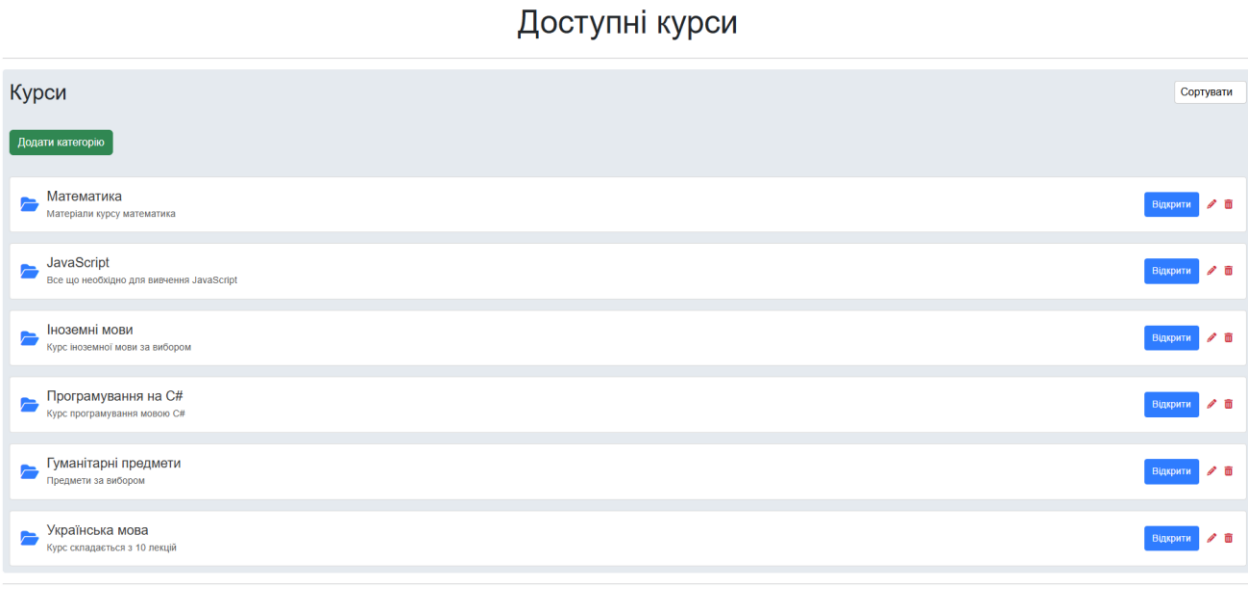

Рисунок 4.15 – Сторінка з курсами для адміністратора

При додаванні нової категорії, з’являється новий елемент, у якого є поля для назви категорії та опису цієї категорії. Адміністратору треба заповнити ці поля і натиснути на кнопку «Додати категорію» або у разі потреби натиснути «Відмінити» для того щоб закрити цей елемент (рисунок 4.16).

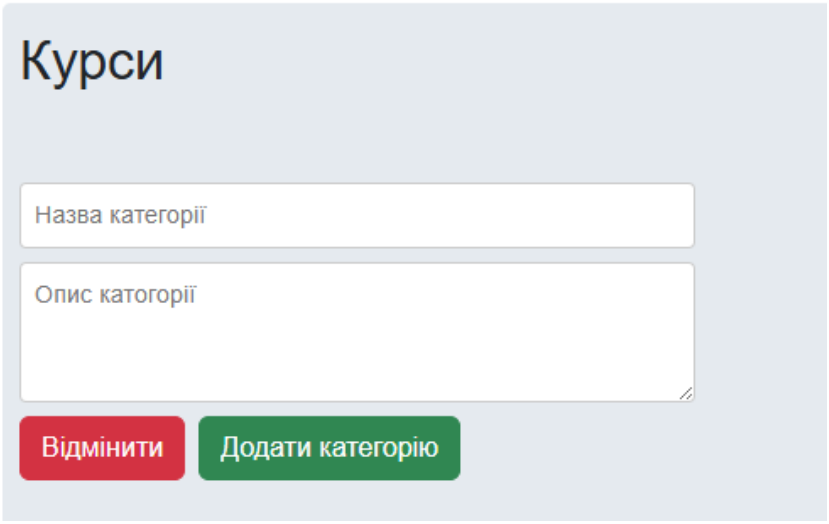

Рисунок 4.16 – Додавання нової категорії

При необхідності змінити вже створену категорію, або для її видалення, адміністратор може натиснути на іконку редагування або видалення і категорію буде змінено (рисунок 4.17).

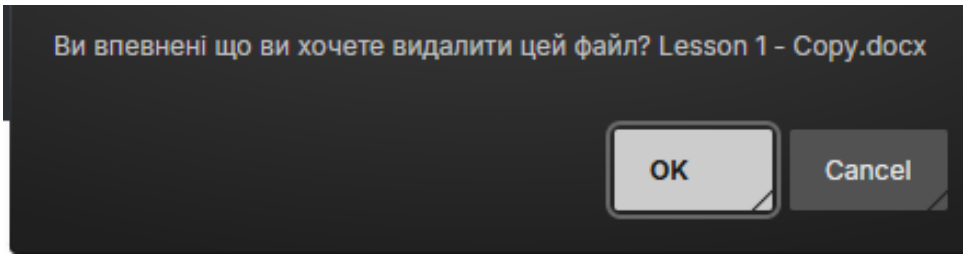

Рисунок 4.17 – Підтвердження про видалення файлу

Для додавання нового файлу треба натиснути кнопку «Додати файл». Після цього з’явиться поле де треба вписати опис файлу, вибрати файл на натиснути на кнопку «Загрузити» (рисунок 4.18).

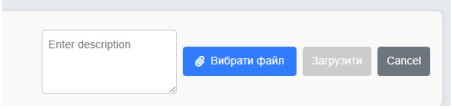

Рисунок 4.18 – Додавання нового файлу

Після цього файл з’явиться у категорії.

ВИСНОВКИ

Кваліфікаційна робота присвячена розробці веб-сервісу для онлайн навчання. Цей додаток має мету покращити або вдосконалити якість навчання для людей різного віку та матеріального статусу.

У процесі дослідження були вивчені можливості популярних освітніх платформ, таких як Prosvita, На Урок, Google Classroom та Moodle, визначено їхні переваги й недоліки. Це дозволило сформувати вимоги до власного застосунку, з урахуванням потреб користувачів, які не повністю задовольняються існуючими рішеннями.

Було спроєктовано архітектуру веб-застосунку, реалізовано систему користувачів з розмежуванням ролей, створено інтерфейс для управління навчальними курсами, завданнями, тестами та оцінюванням. Система також підтримує зворотний зв'язок, що сприяє ефективній взаємодії між учасниками навчального процесу.

Застосунок демонструє приклад сучасного React додатку. Чітку архітектуру, використання безпечної автентифікації та захисту облікових даних користувачів, грамотну маршрутизацію та інтеграцію з популярними UI-бібліотеками. Основний функціонал включає в себе реєстрацію, вхід, збереження токена, навігацію по категоріях та перегляд курсів.

Проєкт є повністю робочим та виконує всі поставлені завдання. У подальшому можливим напрямом розвитку проєкту є розширення функціоналу, а саме: інтеграція з відеозв’язком, впровадження мобільної версії застосунку, підтримка багатомовності, реалізація рішення для відстеження прогресу студентів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Платформи для дистанційного навчання: як обрати найкращу? [Електронний ресурс] – Режим доступу: www/ URL: https://osvita.ua/school/method/technol/93225/ – 16.05.2025 р. – Загол. з екрану.

2. Prosvita [Електронний ресурс] – Режим доступу: www/ URL: https://prosvita.net – 16.05.2025 – Загол. з екрану.

3. На Урок [Електронний ресурс] – Режим доступу: www/ URL: https://naurok.com.ua – 16.05.2025 – Загол. з екрану.

4. Google Classroom Helps Teachers Create, Organize Assignments [Електронний ресурс] – Режим доступу: www/ URL: https://www.pcmag.com/news/google-classroom-helps-teachers-create-organize-assignments – 16.05.2025 – Загол. з екрану.

5. Moodle [Електронний ресурс] – Режим доступу: www/ URL: https://docs.moodle.org/500/en/Features – 16.05.2025 – Загол. з екрану.

6. Flanagan, D. JavaScript: The Definitive Guide [Текст] / D. Flanagan. // O'Reilly Media. – 2006. – 1093 с.

7. JavaScript Documentation [Електронний ресурс] – Режим доступу : https://developer.mozilla.org/en-US/docs/Web/JavaScript – 05.06.2025 р. – Загол. з екрану.

8. Express [Електронний ресурс] – Режим доступу: www/ URL: https://expressjs.com – 18.05.2025 – Загол. з екрану.

9. Sequelize [Електронний ресурс] – Режим доступу: www/ URL: https://sequelize.org – 18.05.2025 – Загол. з екрану.

10. Bcrypt [Електронний ресурс] – Режим доступу : www/ URL: https://www.npmjs.com/package/bcrypt – 10.06.2022 р. – Загол. з екрану.

11. Jsonwebtoken [Електронний ресурс] – Режим доступу : www/ URL: https://www.npmjs.com/package/jsonwebtoken – 10.06.2022 р. – Загол. з екрану.

12. Banks, A. Learning React: Functional Web Development with React and

Redux [Текст] / A. Banks, E. Porcello. // O'Reilly Media. – 2017. – P. 350

13. DOM [Електронний ресурс] – Режим доступу : www/ URL: https://learn.javascript.ru/dom-nodes – 05.06.2025 р. – Загол. з екрану.

14. JetBrains WebStorm [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/ru-ru/webstorm/ – 18.05.2025 – Загол. з екрану.

15. JetBrains [Електронний ресурс] – Режим доступу : www/ URL: https://www.jetbrains.com – 10.06.2025 р. – Загол. з екрану.

16. MSSQL Documentation [Електронний ресурс] – Режим доступу : www/ URL: https://learn.microsoft.com/ru-ru/sql/sql-server/what-is-sql-server?view=sql-server-ver16 – 10.06.2025 р. – Загол. з екрану.

17. Bootstrap Documentation [Електронний ресурс] – Режим доступу : www/ URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/ – 10.06.2025 р. – Загол. з екрану.

18. React Bootstrap Documentation [Електронний ресурс] – Режим доступу : www/ URL: https://react-bootstrap.netlify.app/docs/getting-started/introduction – 10.06.2022 р. – Загол. з екрану.

19. Font Awesome Documentation [Електронний ресурс] – Режим доступу : https://docs.fontawesome.com – 10.06.2025 р. – Загол. з екрану