МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ НАВЧАННЯМ ЗАСОБАМИ REACTJS ТА DJANGO»**

Виконав: студент 2 курсу, групи 8.1212-іпз-1
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
К.В. Митрошин
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, к.ф.-м.н. Кривохата А.Г.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.пед.н., Пшенична О.С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Митрошину Кирилу Владиславовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка системи управління навчанням засобами ReactJS та Django

керівник роботи Кривохата Анастасія Григорівна, к.ф.-м.н.
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » травня 2023 року № 642-с

2. Строк подання студентом роботи 27.11.2023 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка системи управління навчанням.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою докладу

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 05.05.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	10.05.2023	
2.	Збір вихідних даних.	07.06.2023	
3.	Обробка методичних та теоретичних	31.07.2023	
	джерел.		
4.	Розробка першого та другого розділу.	20.09.2023	
5.	Розробка третього розділу.	10.11.2023	
6.	Оформлення та нормоконтроль	21.11.2023	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	15.12.2023	

Студент _____ К.В. Митрошин
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.Г. Кривохата
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка системи управління навчанням засобами ReactJS та Django»: 62 с., 27 рис., 2 табл., 16 джерел, 2 додатки.

ВЕБДОДАТОК, НАВЧАННЯ, ПРОГРАМУВАННЯ, СИСТЕМА УПРАВЛІННЯ, DJANGO, DRF, JAVASCRIPT, PYTHON, REACTJS.

Об’єкт дослідження – процес розробки системи управління навчанням засобами ReactJS та Django.

Мета роботи: розробити систему управління навчанням засобами ReactJS та Django.

Методи дослідження – методи об’єктно-орієнтовного програмування, методи програмної інженерії.

У магістерській кваліфікаційній роботі досліджувалися системи управління навчанням, їх переваги та недоліки, способи реалізації таких систем. У результаті було виявлено найуживаніший функціонал та оптимальні технології розробки систем управління навчанням. Зокрема, були проаналізовані основні проблеми таких систем, такі як: складний інтерфейс, можливості підтримки та загального використання сервісу без технічних спеціалістів, відсутність безкоштовного функціоналу, тощо.

Кінцевий вебдодаток було реалізовано використовуючи як frontend, так і backend технології, зокрема мови програмування JavaScript та Python. У фреймворках використовувалося завантаження додаткових пакетів коду, для підвищення читабельності та функціональності отриманого застосунку.

Результати роботи можуть бути використані при організації онлайн навчання, зокрема у навчальних закладах різного рівня, позашкільних організаціях, та в системі освіти для дорослих.

SUMMARY

Master’s qualifying paper «Development of the Learning Management System using ReactJS and Django»: 62 pages, 27 figures, 2 tables, 16 references, 2 supplements.

DJANGO, DRF, JAVASCRIPT, LEARNING, MANAGEMENT SYSTEM, PROGRAMMING, PYTHON, REACTJS, WEB APPLICATION.

The object of the study is the process of developing a learning management system using ReactJS and Django.

The aim of the study is to develop a learning management system using ReactJS and Django.

The method of research are object-oriented programming methods, software engineering methods.

In the master’s qualification paper, the existing learning management systems, their advantages and disadvantages, and ways of implementing such systems were studied. As a result, the most used functionality and optimal technologies for developing learning management systems were identified. In particular, the main problems of such systems were analyzed, such as a complex interface, the ability to support and generally use the service without technical specialists, the lack of free functionality, etc.

The final web application was implemented using both frontend and backend technologies, in particular JavaScript and Python programming languages. The frameworks used the loading of additional code packages to increase the readability and functionality of the resulting application.

The results of the work can be used in the organization of online learning, in particular in educational institutions of various levels, out-of-school organizations, and in the adult education system.

ЗМІСТ

ВСТУП

В сьогоднішніх реаліях навчання в Україні перебуває в неодностайному положенні – дистанційне навчання за період карантину не встигло розвинутися, а під час військового стану навчання повністю розділилося на дистанційне, змішане та очне навчання, в залежності від можливості виконання умов для безпеки співробітників та учнів. Кожний з перерахованих способів навчання має свої переваги: очне навчання забезпечує візуальну оцінку розуміння матеріалу викладачем та спрощує сприйняття матеріалу учнями, змішане навчання дозволяє викладачам передавати лекційний матеріал наживо, а перевірку практичних робіт онлайн.

Дистанційний формат навчання, відповідно, переводить все навчання в інтернет, що у свою чергу значно збільшує комп'ютерну грамотність як здобувача освіти, так і викладача. Викладати матеріал онлайн можливо завдяки сервісам відеоконференцій: Zoom, Microsoft Teams, Big Blue Button, тощо.

Приймати домашні завдання можливо електронною поштою, у соціальних мережах та месенджерах, але тут вираховується основна проблема – відсутність контролю опублікованих здобувачем освіти робіт та чесності перевірки виконаних робіт викладачем, тому варіанти передачі робіт особисто можна розглядати як резервний спосіб здачі робіт. Саме по цій причині та для зручності масштабування існують платформи контролю навчання. Серед них Google Class, BlackBoard, Talent LMS, Nearpod та Moodle, який широко використовується у вищих навчальних закладах. Також, електроні системи навчання поширені у сфері навчальних курсів в IT компаніях.

Всі вищенаведені системи мають свою архітектуру, тим самим звужуючи круг користувачів, яким використання цього ПЗ буде доцільним.

Отже, мета дослідження наукової роботи – розробити систему управління навчанням засобами ReactJS та Django.

Для досягнення даної мети потрібно порівняти доступні платформи для

забезпечення здобуття якісної освіти та створити власну платформу, враховуючи переваги та недоліки вже існуючих платформ. Насамперед мова йде про здачу домашніх та практичних робіт, перевірка та оцінювання виконаних робіт викладачем. Система повинна мати високу здатність до масштабування та, насамперед, бути зручною як для професійного користувача ПК, так і для початківця.

Об’єктом дослідження є процес розробки системи управління навчанням.

Предметом є методи розробки вебдодатків, з використанням API та технологіями frontend та backend розробки.

У ході вирішення поставлених завдань було отримано такі результати: проаналізовано ряд публікацій з теми розробки вебдодатків, спроєктовано, розроблено та протестована система управління навчанням з використанням ReactJS та Django, Django Rest Framework.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, переліку посилань (16 найменувань) та двох додатків.

1 ОГЛЯД МЕТОДІВ ТА ЗАСОБІВ РОЗРОБКИ СИСТЕМ УПРАВЛІННЯ НАВЧАННЯМ

1.1 Огляд систем управління навчанням

Система управління навчанням (LMS) – це програмне забезпечення, яке використовується для управління навчальними програмами та ресурсами. LMS можна використовувати для створення, управління та відстеження навчальних курсів, а також для надання доступу до навчальних матеріалів учням.

LMS може використовуватися в різних навчальних контекстах, включаючи школи, університети, підприємства та організації. Вони можуть використовуватися для навчання в класі, дистанційного навчання або комбінації обох.

Основні функції LMS включають:

– надання доступу до навчальних матеріалів;
– забезпечення можливістю оцінювання робіт;
– відстеження прогресу учнів;
– розсилання повідомлень учням.

Пояснення до функцій: LMS повинна дозволяє учням отримувати доступ до навчальних матеріалів, таких як відео, документи та тести та надавати можливість спілкування один з одним. Система має допомагати автоматизувати процес оцінювання завдань, тестів та інших форм вивчення матеріалів. Процес навчання кожного здобувача освіти повинна мати можливість відстежування і користувачі.

Системи можуть мати різні функції, залежно від їх призначення. Наприклад, LMS, призначені для використання в школах, можуть включати функції, такі як відстеження присутності та оцінка учнів. Можливе використання в бізнесі, включати функції, такі як управління сертифікатами та відстеження ефективності навчання.

LMS можуть бути корисним інструментом для підвищення ефективності навчання. Вони можуть допомогти викладачам створювати ефективні навчальні курси, а учням – краще навчатися. Додатковий аналіз системи управління навчанням та проблем дистанційної освіти було наведено у публікації [1].

Прикладами сучасних вебдодатків, а саме систем управління навчання існує безліч. Найпопулярніші серед них: Google Class, BlackBoard, Talent LMS, Nearpod та Moodle. Всі ці сервіси направлені на контроль навчання здобувачем освіти, але кожний робить це своїми методами та мають вкрай різну аудиторію.

Наприклад, система Moodle представляє з себе конструктор, завдяки якому технічні фахівці можуть видалити зайві та додати потрібні функції вже для використання викладачами та студентами [2]. Але не кожний заклад освіти має таких спеціалістів, тому етап підготовки платформи бути надскладним саме на цьому етапі.

Також, Moodle пропонує викладачам завчасно підготувати курс з системою накопичення балів. Отже, основний недолік цієї LMS – складність опанування студентами та викладачами, та пряма залежність від спеціалістів та програмістів, які програмуватимуть цю систему.

Google Classroom, у свою чергу, пропонує зручність з етапу реєстрації – викладачеві не потрібно звертатися до технічних спеціалістів [3]. Також, курси викладаються послідовно, але в системі відсутній журнал оцінок. Ні викладач, ні здобувач освіти не може контролювати свій середній бал за предмет, так як завдання ніяк не пов’язані один з одним. Це призводить до того, що Google Classroom автоматизує процес лише частково, повністю вести оцінювання учнів або студентів в ній неможливо.

Google Classroom використовує архітектуру мікросервісів, що означає, що він складається з набору невеликих, взаємодіючих програм. Серверна частина Google Classroom написана на Python і використовує фреймворк Django. Клієнтська частина Google Classroom написана на мові програмування JavaScript і використовує фреймворк React. Розмітка вебсторінки Google Classroom написана мовами HTML і CSS.

BlackBoard – це система управління навчанням, яка використовується навчальними закладами для створення та надання онлайн курсів [4]. Вона дозволяє викладачам створювати контент, публікувати завдання, оцінювати роботу студентів та надавати зворотний зв’язок. Студенти можуть використовувати BlackBoard для доступу до матеріалів курсу, виконання завдань, спілкування з викладачами та іншими студентами. Сильною стороною сервісу є взаємодія викладача та учня за допомогою відео та текстового чату, проте дана система є повністю платною. Дані у сервісі зберігаються за допомогою XML, серверна частина написана мовами програмування PHP та Java. Точна база даних невідома, але відомо, що вона SQL-подібна. Використовується понад 30 мільйонами студентів у понад 100 країнах світу.

Ще однією достатньо популярною системою управління навчанням є Talent LMS [5]. Вона пропонує широкий спектр функцій, які можуть допомогти організаціям створювати ефективні онлайн курси, а учасникам навчання – успішно навчатися. Talent LMS використовується компаніями різного розміру та галузей, включаючи освітні установи, підприємства, урядові організації та некомерційні організації. Система доступна у хмарі, що означає, що компанії можуть використовувати його без необхідності інвестувати у власну інфраструктуру. Основна перевага – розширений аналіз даних успішності учнів, що дає змогу покращувати процес навчання, для досягання найкращих результатів. Даний продукт має безкоштовну версію, проте дуже обмежену в функціоналі. Мови програмування аналогічні до Google Class: Python, JS. Розмітка написана мовами HTML та CSS.

Остання зі згаданих систем – Nearpod, є повноцінною системою навчання, з можливістю оцінювати учнів, студентів, тощо [6]. Найбільший недолік цієї LMS – інтерфейс надається тільки англійською, тому, наприклад, для учнів початкових класів та для викладачів, які не знають цієї мови, система буде дуже складною. Це у свою чергу може призвести до уникання використання системи.

Однак, Nearpod має сильну перевагу над своїми конкурентами завдяки реалізації асинхронного режиму навчання. Цьому сприяють можливості запису

занять, які потім можна переглянути, та можливість виконувати завдання без підключення до інтернету, якщо завантажити їх заздалегідь. В цілому система направлена на навчання людей за їх потребами, не обмежуючи рамками часу самих занять.

1.2 Принцип роботи систем керування навчанням

Основна задача систем керування навчанням – оцінювання якості викладання, визначення рівня знань здобувачів освіти, контроль виконання плану навчання.

Для мінімального виконання цих умов, повинні існувати наступні групи користувачів:

- надання доступу до навчальних матеріалів;
- користувач-здобувач освіти (студент, учень, тощо);
- користувач-викладач;
- користувач-адміністратор;
- користувач з доступом перегляду оцінок;
- користувач-адміністратор групи (класний керівник, куратор);
- адміністратор системи.

Всі перераховані користувачі мають доступ до авторизації в системі. Користувач-здобувач освіти це основний користувач системи. Він має доступ до курсів, в які було запрошено його особисто, або було запрошено групу, до якої було попередньо додано користувача. Цей користувач має доступ до журналу власних оцінок зі всіх курсів, до яких має доступ, може проходити тести, залишати фотографії, відео або файли обмеженого розміру від завданням, яке було розміщено користувачем-викладачем. Також, здобувачі освіти можуть відправляти повідомлення викладачам.

Друга категорія користувачів – викладачі. Вони мають змогу редагувати вміст своїх курсів, створювати нові курси, оцінювати виконані роботи та

редагувати журнал оцінок відповідного курсу. Відповідно, в нього є доступ до списку групи або всіх, хто навчається на відповідному курсі. Також, ця роль може додавати або видаляти користувачів зі своїх курсів.

Користувач-адміністратор має доступ аналогічний до викладачів, але має доступ до відраховування студентів (системно – видалення акаунтів, або обмеження доступу). Ця група користувачів може дивитися статистику як особисто учня, так і оцінки категорії студентів в розрізі курсу. Також, адміністратор може передивлятися скарги від отримувачів освіті, як на курс, так і на викладача.

Користувач-адміністратор групи має додатковий до прав викладача доступ до перегляду журналу оцінок певного здобувача освіти, який навчається у групі, в якій користувач є адміністратором.

Група, яка має доступ тільки до перегляду оцінок може авторизуватися та переглядати оцінки відповідного користувача-здобувача освіти. Додаткових прав у цієї групи немає.

Адміністратор системи – остання група користувачів, яка має права до створення та видалення облікових записів як студентів, так і батьків. Також, адміністратор може переглядати системні дані, для створення технічного запиту до розробника, у випадку необхідності. Система має робити резервну копію щоденно, для безпечності зберігання даних.

Вебсайт виводить на екран тільки статичний контент, не надає можливості редагувати інформацію та не має можливості зберігати дані клієнта, які критично необхідні для даної задачі. Основна перевага вебдодатків у здатності взаємодіяти з кінцевим користувачем, також генерує динамічний контент, збільшуючи функціональність. Сама система є багатофункціональним вебдодатком, та, відповідно, не може буде реалізована вебсайтом. Підключення будь-яких видів баз даних є невід’ємною частиною вебдодатку.

Безпека є не менш важливою частиною використання баз даних. Зазвичай, у вебдодатках використовуються реляційні бази даних (SQLite3, MySQL, PostgreSQL, тощо), отримання та редагування запитів виконується за

допомогою мови запитів Structured Query Language. Дані зберігаються у вигляді таблиць, що інколи пришвидшує швидкість роботи з базою даних. Найбільша загроза використання таких типів баз даних – SQL Injection. Це код, завдяки якому зловмисник може нашкодити базі даних. Сучасні фреймворки використовують екранізацію для захисту від такого типу «атаки», але у деяких випадках є гостра необхідність використовувати екранізацію вручну.

Другий, не менш відомий, вид баз даних – нереляційні бази даних (надалі NoSQL). Дані, в основному, зберігаються у вигляді «ключ-значення», що вирішує питання масштабування та доступності даних, так як відсутня атомарність та узгодженість даних. Цей вид баз даних використовується переважно там, де реляційна база не буде оптимальним рішенням. Це можуть бути додатки з динамічним зростанням даних користувача. Наприклад, в кулінарних додатках краще зберігати дані саме в NoSQL, так як кількість інгредієнтів не має фіксованої кількості. В інших випадках використовувати NoSQL недоцільно, так як швидкість роботи з базою даних в SQL базах даних при запитах з невеликою кількістю зв’язків вище, аніж в NoSQL.

Повноцінний вебдодаток завжди складається з backend та frontend частини. Вони, відповідно, створюють логічну та візуальну складову вебдодатку.

1.3 Використання React JS, Django Rest Framework та PostgreSQL

Як зазначалося у розділі 1.2, вебдодаток має бути спроєктований та створений шляхом об’єднання логічної частини та інтерфейсу користувача.

Проєктування системи є важливим етапом у розробці будь-якого вебдодатку. Цей процес включає збір та аналіз вимог до продукту, розробку концепту готового продукту, створення діаграм класів, опис структури бази даних та інші важливі елементи.

Вебдодаток, про який йде мова в цьому тексті, має бути спроєктований та

створений шляхом об’єднання логічної частини та інтерфейсу користувача. Це означає, що вебдодаток повинен бути зручним для користувача, а також ефективним з точки зору обробки даних та виконання задач.

Для розробки вебзастосунків з API інтерфейсами можна використовувати різні технології. Одним із популярних варіантів є використання React JS, Django Rest Framework та PostgreSQL. React JS – це JavaScript бібліотека для побудови користувацьких інтерфейсів, Django Rest Framework – це потужний і гнучкий інструмент для побудови веб API, а PostgreSQL – це система керування реляційними базами даних, яка використовується для зберігання та відновлення даних.

Django Rest Framework (далі DRF) – це фреймворк для створення API інтерфейсів на Django. Django Rest Framework пропонує широкий спектр функцій, які полегшують і прискорюють розробку API інтерфейсів. DRF використовується для створення API інтерфейсів вебзастосунку з API інтерфейсом. Він дозволяє вебзастосунку отримувати доступ до даних у PostgreSQL.

Django виступає у якості контролера та виконує перевірку URL запиту користувача. На один і той же URL запит користувач може отримати різні шаблони (наприклад головна сторінка сайту для авторизованого та не авторизованого користувача). Якщо URL запит коректний – Django повертає користувачеві відповідь у вигляді шаблону. Звичайно, для використання такого патерну обов’язково потребується база даних – зазвичай це SQLite3 або PostgreSQL.

React JS це фреймворк, який виконує рендер даних, отриманих на backend стороні [7]. Він створений на основі мови програмування JavaScript у вигляді компонентів, завдяки яким вже створюється багатофункціональний інтерфейс. На відміну від багатьох інших мов програмування та фреймворків, для використання React не потрібно мати конкретну платформу, його можна запустити як на сервері за допомогою Node.js, так і на мобільній платформі. На відміну від використання HTML, для використання такого фреймворку,

доведеться відмовитися від засобів стандартної генерації шаблонів Django, та використовувати його як REST API (за допомогою GET та POST запитів), для отримання даних на стороні React.

Останній аргумент на користь обраних фреймворків – достатньо обширна документація та безліч прикладів використання того чи іншого механізму. Також, для нових користувачів фреймворку є текстові матеріали для створення першого проєкту з нуля. Це робить ці фреймворки доступними для великої кількості розробників, незалежно від їхнього досвіду та знань.

2 ПРОЄКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ НАВЧАННЯМ

Етап проєктування системи є важливим елементом розробки будь-якого програмного продукту. Він допомагає розробникам краще розуміти вимоги до продукту, структуру продукту та його функціональність. Це, в свою чергу, допомагає їм створити продукт, який відповідає потребам

Процес проєктування складається з 4 етапів: аналізу вимог до програмного забезпечення, концептуального проєктування, логічного проєктування та фізичного проєктування.

2.1 Аналіз вимог до системи управління навчанням

Першим кроком є збір та аналіз вимог до продукту. Це включає в себе визначення потреб користувачів, вимог до функціональності та інших важливих аспектів продукту. Цей процес допомагає забезпечити, що продукт відповідає очікуванням користувачів і відповідає їх потребам.

Далі розробники створюють діаграми класів. Ці діаграми допомагають візуалізувати структуру продукту, включаючи його класи, об’єкти та взаємозв’язки між ними. Це допомагає розробникам краще розуміти структуру продукту і спрощує процес його розробки.

Метою системи управління навчанням (LMS) є забезпечення ефективного та комфортного процесу навчання для вчителів, учнів та адміністраторів. LMS повинна дозволяти:

- створювати та керувати навчальними курсами;
- гадавати доступ до навчального матеріалу;
- спілкуватися з іншими учасниками навчального процесу;
- оцінювати результати навчання;
- вести відомості по успішності здобувачів освіти.

Система повинна давати можливість створювати навчальні курси з різними типами навчального матеріалу, такими як текст, аудіо, відео, презентації тощо. Також, потрібна можливість керувати доступом до навчальних курсів та їх налаштування. Також, система повинна надавати доступ до навчального матеріалу як з мобільного пристрою, так і з комп’ютера, планшета, тощо. Користувач повинен мати змогу завантажувати та публікувати необхідний для навчання матеріал. Повинна забезпечуватись можливість спілкування між вчителями, учнями, адміністраторами у вигляді коментарів, текстового чату. Система повинна надавати можливість викладачам ставити оцінки в електронному журналі. Всі оцінки повинні зберігатися в журналі, до якого матимуть доступ персони з адміністраторськими правами та здобувачі освіти, для актуалізації інформації.

Для виконання робіт нефункціональними вимогами є:

– використання бази даних;
– використання фреймворку;
– швидкість запитів не більше 2 секунд;
– використання системи сеансів;
– імплементація API для додаткових аналізів;
– підтримка більшості відомих браузерів.

Розглянемо вимоги детальніше. Для виконання робіт було обрано базу даних PostgreSQL, так як вона найбільш сумісна з Django. Фреймворк, у свою чергу, надає власний ORM, який спрощує та пришвидшує процес розробки. Було обрано фреймворк React JS, за рахунок його гнучкості та можливостей додаткових пакетів, збільшується швидкість розробки, без суттєвих втрат продуктивності. Django ORM і використання SQL-подібних збільшує в рази швидкість запитів та, за необхідності, індексує таблиці, що, у свою чергу, незначно зменшує час запиту.

Сеанси повинні мати час дії, відповідно, було обрано алгоритми JsonWebToken. Вони виконують додаткове шифрування даних сеансу, зберігають в собі час дії сеансу, та достатньо просто розшифровуються за

наявності секретного ключа. Також, за допомогою токенів можна зберігати дані користувача для ідентифікації в запитах бази даних.

Використання Django Rest Framework надає користувачам готові рішення API як для рендеру вебсторінки, так і для зовнішніх запитів, за наявності даних авторизації та відповідних прав на необхідні запити.

React JS 18-ої версії підтримується всіма браузерами, які мають підтримку JavaScript та ES6 модулів. Edge, Firefox, Chrome, Safari останніх версій і браузери, які використовують драйвера цих браузерів також підтримують React, тому проблем з сумісністю виникати не повинно.

2.2 Концептуальне проєктування системи управління навчанням

На даному етапі розробки системи управління навчанням особлива увага приділяється визначенню ключових концепцій та стратегій, які будуть лежати в основі проєкту. Концептуальне проєктування є критичним етапом, оскільки від нього залежить подальший успіх у втіленні системи.

Основною метою системи управління навчанням є автоматизація процесу здачі та оцінювання домашніх завдань. Для досягнення цієї мети система повинна виконувати такі функції:

– реєстрація та авторизація користувачів;
– створення та редагування домашніх завдань;
– здача домашніх завдань учнями;
– оцінювання домашніх завдань вчителями;
– відстеження виконання домашніх завдань учнями.

Відобразимо діаграму прецедентів для системи управління навчанням на рисунку 2.1.

На поданій діаграмі прецедентів, відображено три основні ролі: адміністратор, викладач та студент. Ці ролі відіграють важливу роль у функціонуванні системи, оскільки вони визначають різні рівні доступу до функцій системи.

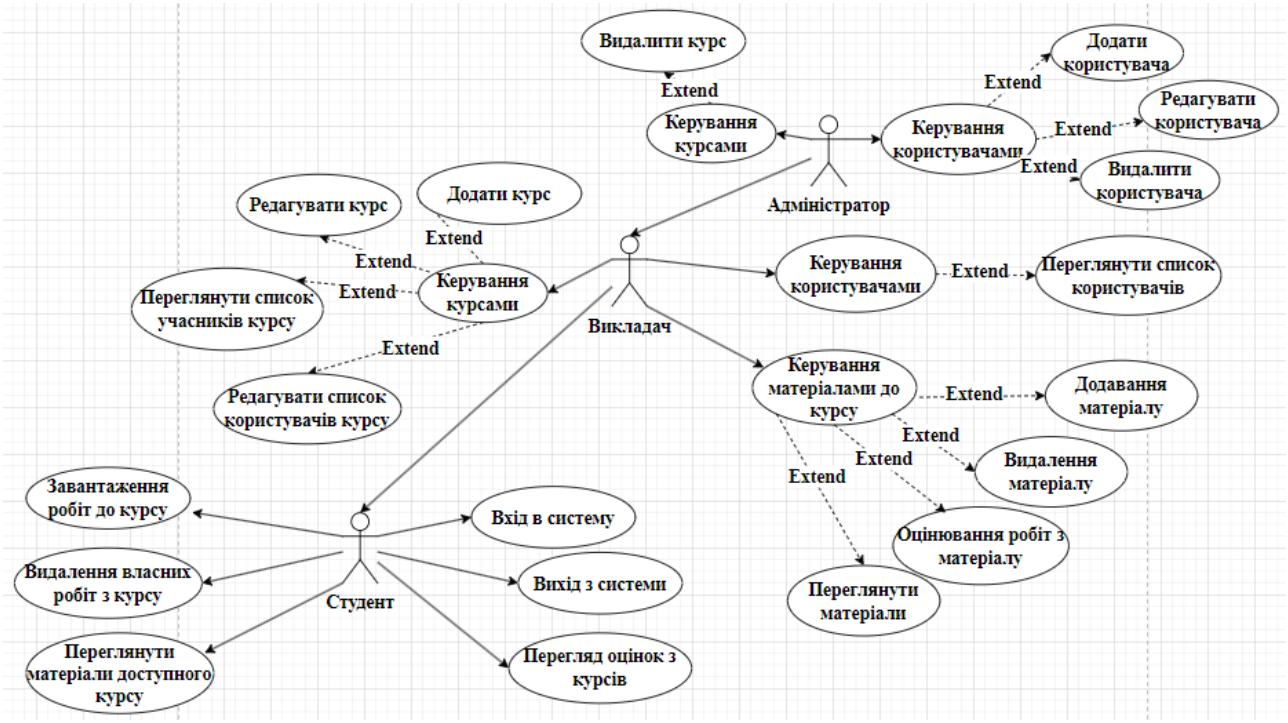

Рисунок 2.1 – Діаграма прецедентів системи управління навчанням

Здобувач освіти є основним користувачем системи. Він має можливість входити та виходити з системи (авторизація), переглядати свої оцінки з курсів, завантажувати та виконувати домашні роботи. Він також може переглядати матеріали курсу, які були завантажені користувачем з правами викладача. Це дозволяє студентам активно взаємодіяти з системою і отримувати максимальну користь від її використання.

Викладач, з іншого боку, має більш широкий спектр можливостей. Він технічно має всі права студента, але також може редагувати власні курси, додавати нові, редагувати список учасників курсів, оцінювати їх роботи. Він також може додавати нові матеріали, оцінювати роботи по матеріалу, переглядати їх і видаляти за потреби. Це робить роль викладача надзвичайно важливою для функціонування системи.

Адміністратор системи має найвищий рівень доступу. Він може створювати, видаляти та редагувати облікові записи користувачів. Це дає йому повний контроль над системою і дозволяє йому управляти всіма аспектами її роботи. Він також має право видаляти безпосередньо курс, що дає йому можливість управляти контентом, який доступний для користувачів.

Ці ролі ієрархічно пов’язані, що означає, що кожна наступна роль має всі права попередньої, а також додаткові. Це створює чітку структуру управління, яка допомагає забезпечити ефективне функціонування системи.

Ця система авторизації є важливою складовою додатка, оскільки вона забезпечує безпеку даних користувачів і дозволяє адміністратору контролювати доступ до додатка. Завдяки цій системі, користувачі можуть використовувати додаток, знаючи, що їхні дані захищені. Загалом, ця система управління навчанням є важливим кроком вперед у сфері освіти, який допомагає підвищити якість навчання та забезпечити кращі результати для студентів.

В додатку, який ми розглядаємо, існує система авторизації користувачів. Це означає, що користувачі повинні ввести свої облікові дані, щоб отримати доступ до функцій додатка. Процес реєстрації, з іншого боку, здійснюється адміністратором через адміністративну панель. Це дозволяє адміністратору контролювати, хто має доступ до додатка, і забезпечує додатковий рівень безпеки.

На рисунку 2.2 наведено діаграму послідовності для реалізації авторизації шляхом використання JsonWebToken (надалі JWT).

JWT – це відкритий стандарт (RFC 7519), який визначає спосіб безпечного обміну даними між двома сторонами за допомогою JSON-об’єкта. Цей JSON-об’єкт буде зашифровано, і може бути перевірено та дешифровано лише стороною, яка має секретний ключ.

Основні об’єкти взаємодії – backend сервіс та база даних. Backend сервіс відповідає за обробку запитів від користувачів, включаючи запити на авторизацію, тоді як база даних зберігає всю інформацію про користувачів, включаючи їх облікові дані.

Оскільки база даних не зберігає паролі у явному вигляді, їх потрібно шифрувати за допомогою секретного набору символів, використовуючи криптографічні алгоритми. У нашому випадку це SHA-256. SHA-256 – це криптографічний алгоритм хешування, який генерує унікальний хешкод з вхідних даних. Цей хешкод потім може бути використаний для перевірки автентичності даних без необхідності зберігати оригінальні дані.

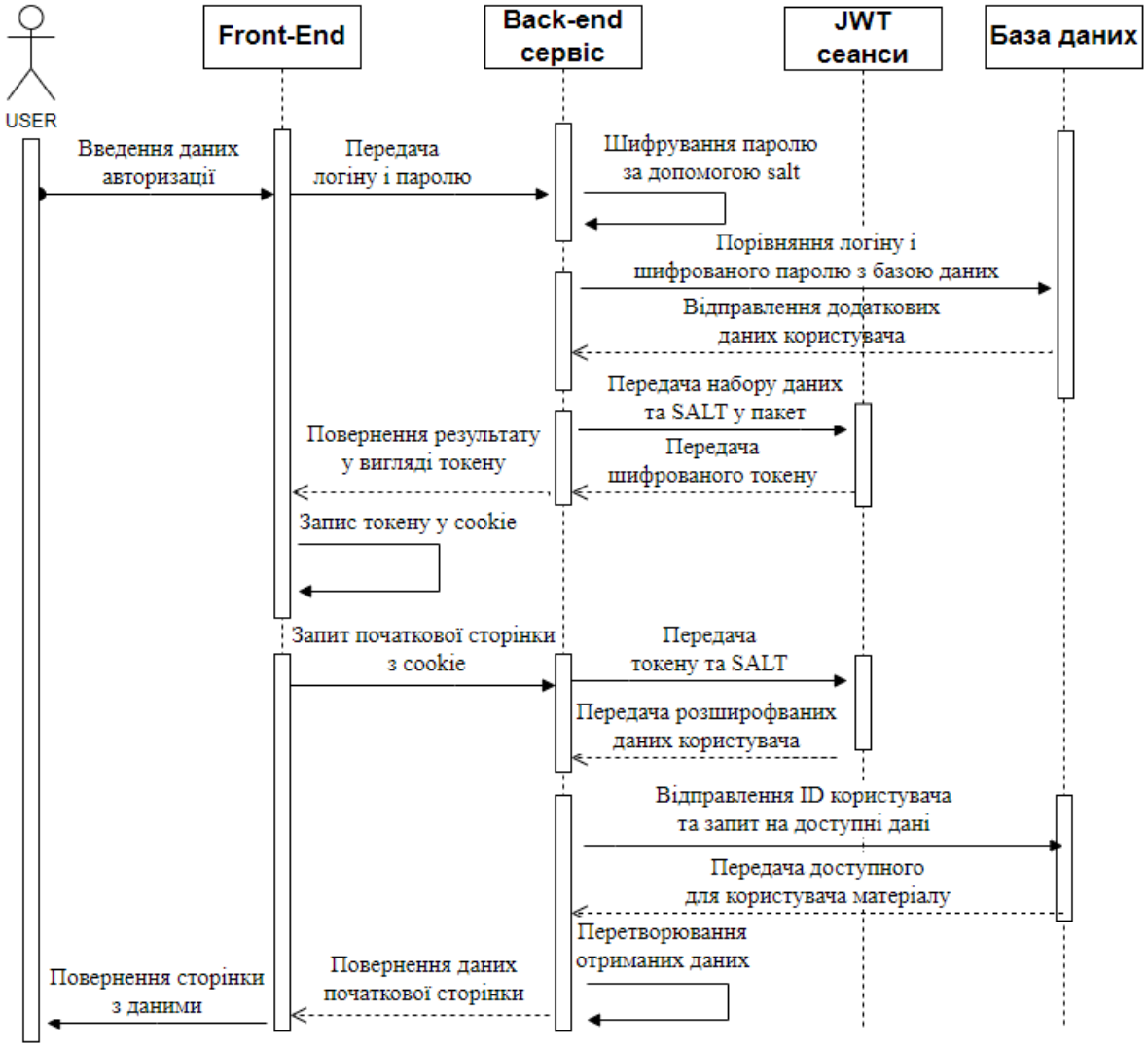

Рисунок 2.2 – Діаграма послідовності JWT

Пароль шифрується при реєстрації та авторизації однаково, відповідно при порівнюванні паролю авторизації та реєстрації потрібно або розшифрувати, або порівняти у зашифрованому вигляді. Це означає, що система може перевірити, чи відповідає введений користувачем пароль збереженому в базі даних паролю, не розкриваючи реальний пароль.

Якщо логін і паролі співпали – система повертає дані про користувача з бази даних, тим самим підтверджуючи факт авторизації. Це означає, що користувач успішно пройшов процес авторизації і тепер має доступ до свого облікового запису.

Ця система авторизації є важливою складовою додатка, оскільки вона забезпечує безпеку даних користувачів і дозволяє адміністратору контролювати

доступ до додатка. Завдяки цій системі, користувачі можуть використовувати додаток, знаючи, що їхні дані захищені.

JWT (JSON Web Token) не є способом реєстрації, але, аналогічно до SHA-256, шифрує певні дані. Отримавши дані з бази даних, їх потрібно конвертувати у зручний вигляд та разом з SALT (секретним «словом») перетворити на шифрований ключ, який передається на клієнтську частину [8]. Також, в параметрах JWT можна вказати термін дії ключу, відповідно, алгоритм розрахує кінцевий момент дії токену.

При запиті до серверу з цим токеном можна звіряти кінцеву дату дії токену та відхилити доступ, або оновити токен, подовживши доступ.

У якості відповіді «ОК» від серверу потрібно завантажити сторінку з вмістом, на який користувач має доступ. Для цього frontend частина, отримавши токен від серверу, записує його у cookies і знову робить запит на сервер, але тепер не на авторизацію, а на отримання певної сторінки (у нашому випадку – початкової) з використанням токену з файлу cookie.

На backend складовій починається процес, зворотній до JWT-шифрування – розшифрування токену. Для цього використовується безпосередньо токен, та SALT, який зберігається безпосередньо в програмі серверу.

Кінцевий результат розшифрування – об’єкт з набором даних користувача, включаючи його ID. ID передається до бази даних, для отримання ролі і даних, які потрібні для рендеру початкової сторінки та відправляються на backend.

Дані отримують фінальну обробку, за необхідності перетворюються на більш складну структуру. Далі дані повертаються на frontend, де, відповідно до шаблонів, перетворюються на вебсторінку.

Кожна дія користувача (оновлення сторінки, відправлення даних на сервер) супроводжується відправленням JWT токену. Якщо токен застарів, або взагалі відсутній – користувачу відмовляється на надані доступу, та повертає на сторінку авторизації або реєстрації.

2.3 Логічне проєктування системи управління навчанням

Третій етап розробки інформаційної системи – це логічне проєктування. На цьому етапі розробляється модель системи, яка відповідає вимогам, визначеним на етапі вимог.

Діаграма класів є одним з основних елементів логічного проєктування. Вона описує об’єкти системи, їхні властивості та взаємодії між ними. Опис структури бази даних також є важливим елементом логічного проєктування. Він описує структуру даних, які будуть зберігатися в базі даних, а також зв'язки між цими даними.

Отже, на етапі логічного проєктування необхідно розробити діаграму класів і описати структуру бази даних. Для розроблюваної системи управління навчанням діаграма класів буде виглядати наступним чином (див. рис. 2.3).

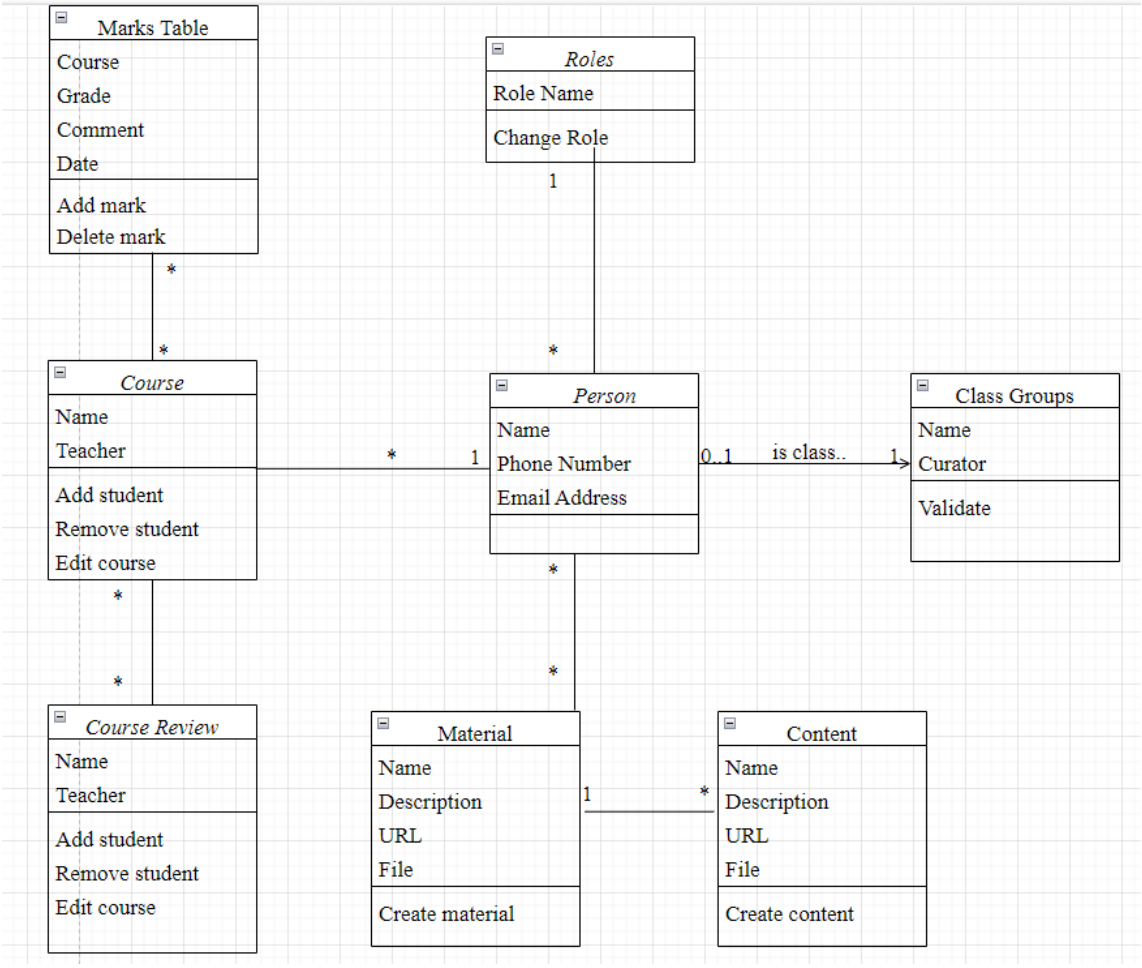

Рисунок 2.3 – Діаграма класів системи управління навчанням

На поданій діаграмі класів наведено основний клас – Person. Технічно всі процеси залежать від цього об’єкта, він має основні дані для зв’язку та валідації. Кожний користувач може мати тільки одну роль: студент, викладач, адміністратор.

Кожний користувач може знаходитися тільки в одній групі. У групи існує користувач-викладач, або користувач-адміністратор.

Користувачі-викладачі можуть викладати матеріал: посилання на вебсторінку, файл, або текстовий вміст. Матеріал має мати назву.

Користувачі, які мають доступ до матеріалу, можуть завантажувати контент відносно цього матеріалу. Аналогічно, це може бути посилання на вебресурс, файл або звичайний текст.

Кожний користувач може зареєстрований в курсі, який має свого викладача. Кожний курс має таблицю оцінок, куди викладач може виставляти оцінку відносно користувача, залишати коментар та дату виставлення оцінки.

Кожний користувач курсу може залишити відгук, та поставити оцінку в діапазоні від 1 до 5.

2.4 Фізичне проєктування системи управління навчанням

Етап фізичного проєктування – це важливий етап у процесі розробки системи управління навчанням. Він дозволяє розробити детальний план системи, який відповідає вимогам організації та може бути реалізований за допомогою доступних технологій. На цьому етапі необхідно розробити план інфраструктури LMS, який визначає обладнання та мережі, які будуть використовуватися для запуску системи. На рисунку 2.4 продемонстровано діаграму компонентів інфраструктури системи.

Користувач, взаємодіючи з веббраузером на своєму персональному комп’ютері, завжди має справу з важливим елементом – JavaScript, який використовується для рендерингу вебсторінок. В процесі рендерингу, браузер

вступає в взаємодію з вебсервером за допомогою API та заздалегідь визначених кінцевих точок (endpoints).

Коли вебсервер отримує запити по API, він обробляє отримані дані, а у випадку наявності паролю, дані можуть бути зашифровані для безпечності передачі. Після цього відбувається важлива взаємодія з базою даних PostgreSQL. Вебсервер використовує вебсокети або TCP/IP з'єднання для ефективної комунікації з базою даних, використовуючи SQL-запити.

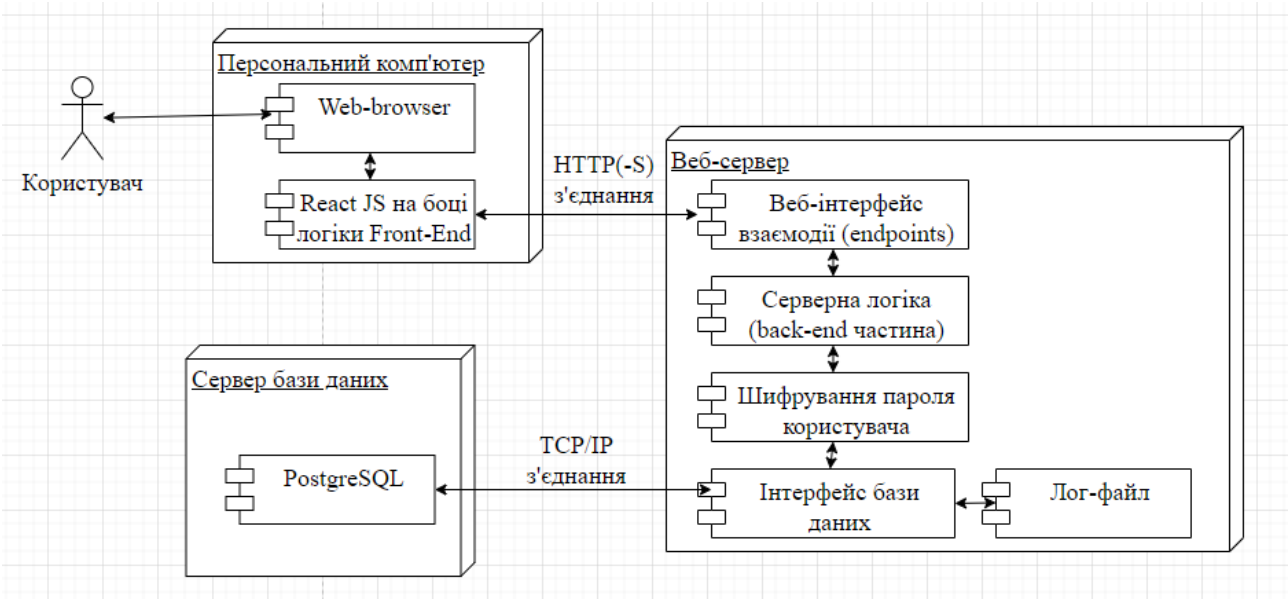

Рисунок 2.4 – Діаграма компонентів системи управління навчанням

База даних виконує обробку отриманих запитів, зокрема, перевірку паролю, якщо це необхідно. Результати операцій логуються для подальшого аналізу. Після успішного взаємодії з базою даних, оброблені дані передаються назад на вебсервер.

Нарешті, дані передаються клієнтській частині через HTTP/HTTPS з'єднання, де вони піддаються необхідній обробці. Завдяки цьому обробці та отриманим даним, вебсторінка може бути ефективно відображена в браузері користувача, завершуючи весь цей цикл взаємодії між клієнтом, вебсервером та базою даних.

3 РЕАЛІЗАЦІЯ СИСТЕМИ УПРАВЛІННЯ НАВЧАННЯМ

3.1 Backend частина системи управління навчанням

Розробка backend складової є невід'ємною частиною циклу створення будь-якого вебдодатку. Використовуючи фреймворк Django та Django Rest Framework, було налаштуємо механізми взаємодії між клієнтом і сервером [9, 10]. Використання Python для програмної реалізації дозволив створити ефективні та легко розширювані API, забезпечуючи зручний доступ до функціоналу системи.

Всі налаштування бази даних виконуються автоматично – шляхом використання міграцій та моделей. В кожній моделі (відповідно – таблиці бази даних) вказуються поля, їх тип, обмеження та додаткові параметри. Django має готові налаштування для завантаження файлів, зображень та відеоматеріалу. На рисунку 3.1 наведено приклад моделі.

```
12
13  class Material(models.Model):
14      first_name = models.CharField(max_length=30)
15      last_name = models.CharField(max_length=30)
16      photo = models.ImageField(upload_to="cars")
17      specs = models.FileField(upload_to="specs")
18
```

Рисунок 3.1 – Приклад моделі в Django

Особлива увага буде приділена забезпеченню безпеки за допомогою механізму авторизації на основі «Json Web Token». В Django вже існують безліч реалізацій даного алгоритму, але найпопулярніший – «Simple JWT» [11].

Згідно алгоритму, використовуючи endpoints, Django сервер отримує дані авторизації з клієнту, перевіряє актуальність даних з базою даних, та шифрує потрібні дані авторизації за допомогою JWT. В шифрованих даних JWT записуються ключові дані: відкритий ключ, безпосередньо дані та час дії токену (див. рис. 3.2).

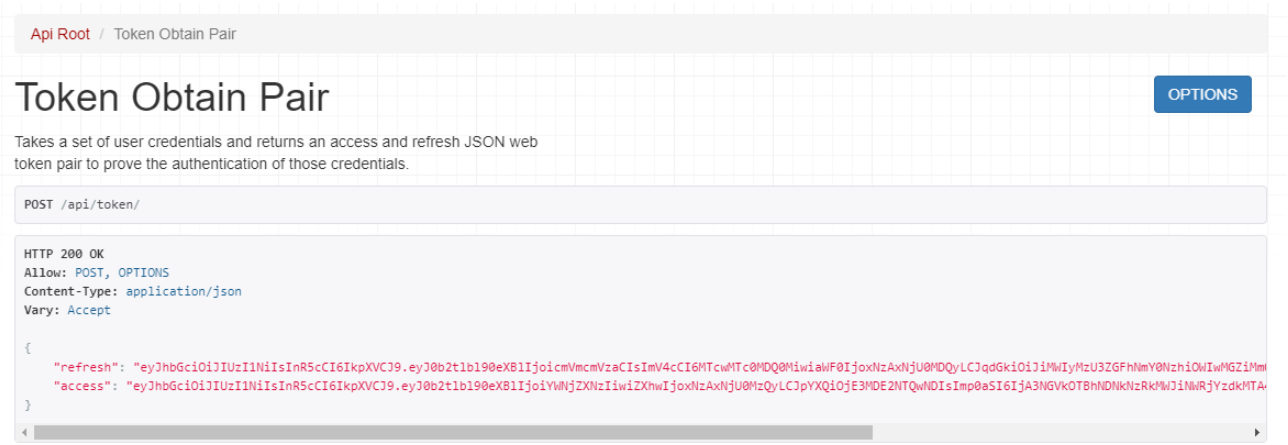

Рисунок 3.2 – Приклад генерації JWT токену

Вихідні дані (JWT токен) відправляється на frontend, до записуються у cookie-файл. Потім отриманий токен повторно використовується в системі, де опрацьовується у зворотному напрямку: дані розшифровуються у JWT за допомогою секретного ключу, попередньо перевіряються на актуальність. З цих даних обираються актуальні, та, за їх допомогою Django звертається до бази даних.

Структура бази даних була представлена раніше, на діаграмі. Зв’язок frontend складової з Django виконується за допомогою Django Rest Framework. Приклад налаштування стандартних endpoints наведено на рисунку 3.3.

```
from rest_framework_simplejwt.views import TokenVerifyView
router = routers.DefaultRouter()
router.register(r'users', views.UserViewSet)
router.register(r'materials', views.MaterialsViewSet)
router.register(r'roles', views.RoleViewSet)
router.register(r'tasks', views.TasksViewSet)
router.register(r'grades', views.GradesViewSet)
router.register(r'persons', views.PersonsViewSet)

urlpatterns = [
    path('', include(router.urls)),
    path('api-auth/', include('rest_framework.urls', namespace='rest_framework')),
    path('api/token/', TokenObtainPairView.as_view(), name='token_obtain_pair'),
    path('api/token/refresh/', TokenRefreshView.as_view(), name='token_refresh'),
    path('api/token/verify/', TokenVerifyView.as_view(), name='token_verify'),
]
```

Рисунок 3.3 – Endpoints Django Rest Framework

Для групування даних використовуються серіалізатори. Серіалізатори з боку SQL є запитами з групуванням та певним фільтром даних. Для зручної фільтрації та групування використовується вбудована мова запитів Django ORM. Приклад серіалізатора, з потребою даних авторизації наведено на рисунку 3.4.

```
class RoleSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = Role
        fields = ['id', 'name']
```

Рисунок 3.4 – Приклад серіалізатора Django

3.2 Frontend частина системи управління навчанням

Процес розробки візуальної складової є необхідною ланкою у циклі створення вебпродукту. При використанні фреймворку React JS передбачається виконання ряду передварних кроків, серед яких важливі наступні етапи:

– вибір середи розробки;
– вибір системи пакетів;
– завантаження необхідних пакетів для розробки;
– створення нового React JS проєкту.

Середа розробки може бути обрана залежно від особистих вподобань та потреб розробника. Система пакетів може бути будь-якою. Це може бути npm (Node Package Manager) або yarn. Перед початком роботи потрібно завантажити необхідні пакети, які будуть використовуватися під час розробки. Це можуть бути бібліотеки, фреймворки чи інші ресурси. Після попередніх підготовчих етапів слід створити новий проєкт на основі фреймворку React JS. Це включає налаштування початкових параметрів та структури проєкту.

У даному випадку, для зручності та продуктивності, обрано редактор коду Visual Studio Code, до якого були додані корисні плагіни: ES7+

React/Redux/React-Native snippets та Tailwind CSS (див. рис. 3.5). Ці додатки допомагають підтримувати структуру коду та виявляти помилки перед збереженням та компіляцією коду, сприяючи ефективному процесу розробки. Розглянуті етапи інтегровані для оптимального використання фреймворку React JS на початковому етапі розробки, щоб забезпечити ефективність та легкість управління проєктом.

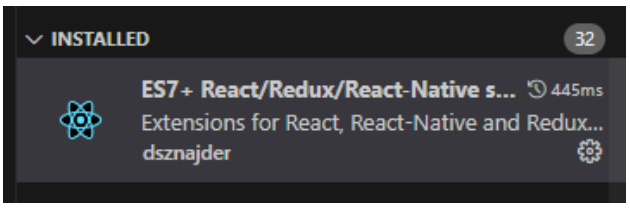

Рисунок 3.5 – Завантажені плагіни VS Code для створення вебпродукту

Для завантаження пакетів було обрано систему NPM, вона дає можливість завантажувати користувацькі пакети коду JavaScript та TypeScript актуальної версії, автоматично зберігаючи їх у папку «node_modules» [12].

На рисунку 3.6 продемонстрований файл «package.json» з усіма завантаженими у проєкт пакетами.

Рисунок 3.6 – Файл «package.json» для frontend проєкту

Найважливіший пакет цього проєкту – Tailwind CSS. Його основне призначення – майже повністю відмовитись від зовнішніх файлів стилів шляхом використання заздалегідь заготовлених класів. Потрібен мінімум 1 файл стилю, де імпортуються пресети tailwind. Візуально це спрощує розуміння де саме потрібно відредагувати або додати стилі, приклад використання цього пакету наведено на рисунку 3.7 [13].

```
<div className='w-[100vw] h-[100vh] bg-repeat bg-center bg-opacity-10 bg-login-img'>
    <div class="w-full h-full flex flex-col justify-center items-center">
        <form class="bg-white shadow-md rounded px-8 pt-6 pb-8 mb-4" onSubmit={serverSubmit}>
            <div class="mb-4">
                <label class="block text-gray-700 text-sm font-bold mb-2" for="username">
                    Username
                </label>
                <input class="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline" id="u
            </div>
            <div class="mb-6">
                <label class="block text-gray-700 text-sm font-bold mb-2" for="password">
                    Password
                </label>
                <input class="shadow appearance-none border border-red-500 rounded w-full py-2 px-3 text-gray-700 mb-3 leading-tight focus:outline-none focus
                <p class="text-red-500 text-xs italic">Please choose a password.</p>
            </div>
            <div class="flex items-center justify-between">
                <button class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded focus:outline-none focus:shadow-outline" type="submit">
                    Sign In
                </button>
            </div>
        </form>

    </div>
```

Рисунок 3.7 – Фрагмент коду з використанням Tailwind CSS

Використання подібного пакету для створення вебдодатку може призвести до збільшення обсягу коду, що, в свою чергу, може стати причиною втрати зручності сприйняття як самим розробником продукту, так і тими, хто відповідає за технічне супроводження.

Ще одним аспектом, який варто розглянути, є отримання даних авторизації додатку від серверної складової вебдодатку. Для взаємодії з сервером та отримання даних авторизації для підтримки сеансу користувача використовуються спеціальні функції, спрямовані на опрацювання таких запитів.

Незважаючи на те, що мова програмування JavaScript має стандартну функцію fetch, для спрощення цього процесу було розроблено пакет axios. Використання цього пакету дозволяє отримати аналогічний функціонал із меншою кількістю написаного коду.

Після успішного отримання даних авторизації, таких як сесії, важливо забезпечити їх збереження у браузері. Хоча запис у localstorage є простим та популярним варіантом, рекомендується використовувати cookies для більш точного та надійного збереження. У JavaScript відсутній стандартний метод для роботи з cookies, тому для цієї мети використовується допоміжний пакет – у конкретному випадку було обрано universal-cookie [14].

Додатковий нюанс полягає в тому, що запити до сервера, пов’язані з оновленням чи створенням токену, виконуються при кожному оновленні сторінки або при відправленні POST-запиту з логіном та паролем. При цьому важливим є використання пакету axios та universal-cookie, що спрощує взаємодію із сервером та керування авторизаційними даними, як показано на прикладі на рисунку 3.8.

Останнім важливим пакетом для створення системи управління навчанням є react-router. Цей пакет надає можливість створити систему переадресації сторінок та реалізації переміщення між компонентами в межах поточного вебдодатку, на рисунку 3.9 наведено приклад використання компоненту «Navigate» після успішної авторизації.

```
const serverSubmit = (e) => {
    e.preventDefault();

    axios.post('http://localhost:8000/api/token/', {
        username, password
    }).then(data => {
        if (data.status === 200) {
            cookies.set('code', data.data.access, { path: '/' });
            cookies.set('updateCode', data.data.update, { path: '/' });
            Navigate({ to: "/home" })
        }
        console.log(data);
    });


};
```

Рисунок 3.8 – Приклад вебдодатку з POST-запитом та cookies

```
<BrowserRouter>
  <Routes>

    <Route path="/" element={<SubjectListPage />} />
    <Route path="/test/:id" element={<TestPage test={test} />} />
    <Route path="/login" element={<LoginPage />} />
    <Route path="/journal" element={<SchoolJournal />} />
    <Route path="/admin" element={<TeacherSubjectPage />} />
    <Route path="/subject/:id" element={<SubjectDetailsPage />} />
    <Route path="/adminsubject/:id" element={<AdminSubjectPage />} />
    <Route path="/createtest" element={<CreateTestPage />} />
    <Route path="/tests" element={<TestListPage />} />

  </Routes>
```

Рисунок 3.9 – Налаштування навігації в LMS

BrowserRouter виступає як обгортка для всіх доступних наборів навігацій, забезпечуючи координацію між ними, а Routes використовується для зберігання масиву посилань на ці навігації. У кожному посиланні міститься адреса та елемент. Елемент вказує на компонент, який повинен бути відображений, та, при необхідності, параметри, які можуть бути передані цьому компоненту. Такий підхід до організації навігації дозволяє зручно керувати шляхами та компонентами, що виводяться для кожного шляху вебдодатку.

Вебдодаток, як правило, складається з різних сторінок, кожна з яких

виконує певну функцію. Ці сторінки можуть включати різні компоненти, які разом створюють функціональність сторінки. Компоненти можуть включати різні елементи, такі як форми введення даних, кнопки, списки та інше.

Кожна сторінка вебдодатку може складатися з певного набору цих компонентів. Наприклад, сторінка авторизації може включати компоненти для введення імені користувача та пароля, а головна сторінка може включати компоненти для відображення списку доступних курсів.

Самі сторінки, в свою чергу, розбиваються на файли. Це означає, що кожна сторінка вебдодатку представлена окремим файлом, який містить весь необхідний код для рендерингу сторінки. Це може включати HTML-код для структури сторінки, CSS-код для стилізації елементів сторінки та JavaScript-код для додавання інтерактивності.

Список сторінок проєкту наведено на рисунку 3.10. Цей список відображає всі сторінки, які входять до складу вебдодатку, та показує, як вони взаємодіють між собою. Це допомагає розробникам краще розуміти структуру вебдодатку та планувати його розробку.

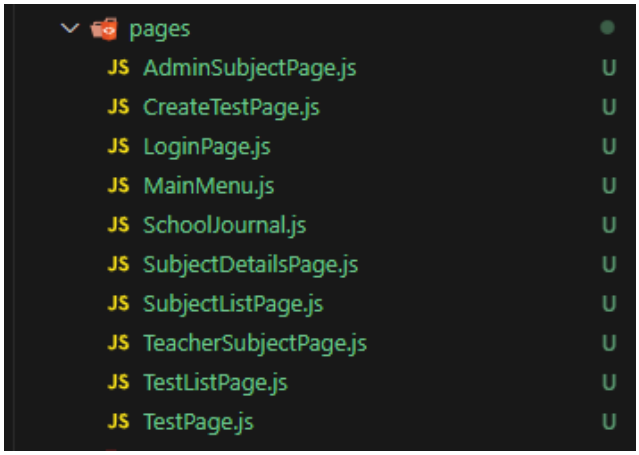

Рисунок 3.10 – Структура сторінок LMS «Cyber Educate»

У розробці вебдодатків компоненти є основними блоками, які використовуються для створення користувацьких інтерфейсів. Ці компоненти можуть бути класовими або функціональними, в залежності від стилю програмування.

У системі управління навчанням, про яку йдеться в цьому тексті, надавалась перевага саме функціональному стилю програмування. Функціональне програмування є парадигмою програмування, яка акцентує увагу на обчисленні результатів, а не на виконанні дій. Це означає, що логіка програми виражається через використання функцій, що робить код більш простим та зрозумілим.

Візуальну складову переносити у компоненти не було доцільним, а зв'язок з backend частково було перенесено у зовнішні компоненти. Це означає, що відповідальність за обробку даних та взаємодію з сервером було відокремлено від відповідальності за відображення даних користувачеві. Це допомагає забезпечити чистоту та організованість коду.

На рисунку 3.11 наведено приклад компонента, який відправляє дані у базу даних при зміні оцінки користувача. Цей компонент використовує API-інтерфейс для взаємодії з сервером та відправки даних. Це демонструє, як компоненти можуть бути використані для створення високофункціональних вебдодатків, які взаємодіють з сервером та забезпечують користувачам потужні та гнучкі інструменти.

```
import axios from 'axios';

const sendGradesToAPI = async (students) => {
  try {
    const apiEndpoint = 'http://127.0.0.1:8000/';

    const gradesData = students.map((student) => {
      return {
        studentId: student.ID,
        grades: student.grades,
      };
    });

    const response = await axios.post(apiEndpoint, { grades: gradesData });

    console.log(response.data);

  } catch (error) {
    console.error('Error sending grades to API:', error);
  }
};

export default sendGradesToAPI;
```

Рисунок 3.11 – Компонент sendGradesToAPI в LMS «Cyber Educate»

Найбільші особливості фреймворку React – реалізована система стану компонентів та ефектів. Кожний компонент має своє «життя» і зміни в компоненті оновлюють тільки компонента-власника, не всю сторінку. Це має як переваги так і недоліки.

Основний недолік такої архітектури – при оновленні навіть одного числа весь компоненті проходить повторний рендер, що може створити додаткове навантаження на всю скомпоновану сторінку. Але недолік частково стає перевагою при вдалому деструктуруванні компонента.

Виходячи з опису недоліків, становляться більш зрозумілі переваги – при правильному розміщенні та комбінуванні компонентів оновлювання маленьких фрагментів значно зменшить навантаження на пристрій користувача, навіть при великій кількості компонентів на поточній сторінці вебдодатку.

Розглянемо приклад на рисунку 3.12.

```
return (
  <div className="container mx-auto mt-8">
    <h2 className="text-3xl font-bold mb-4 text-center text-blue-700">Teacher Subjects</h2>

    <div className="flex mb-4">
      <input
        type="text"
        placeholder="Subject Name"
        value={newSubject.name}
        onChange={(e) => setNewSubject({ ...newSubject, name: e.target.value })}
        className="border rounded p-2 mr-2"
      />
      <input
        type="text"
        placeholder="Description"
        value={newSubject.description}
        onChange={(e) => setNewSubject({ ...newSubject, description: e.target.value })}
        className="border rounded p-2 mr-2"
      />
      {isEditing ? (
        <>
          <button onClick={handleEditSubject} className="bg-blue-500 text-white px-4 py-2 rounded">
            Update
          </button>
          <button onClick={handleCancelEdit} className="bg-gray-500 text-white px-4 py-2 rounded ml-2">
            Cancel
          </button>
        </>
      ) : (
        <button onClick={handleAddSubject} className="bg-green-500 text-white px-4 py-2 rounded">
          Add
        </button>
      )}
    </div>
```

Рисунок 3.12 – Приклад компоненту TeacherSubjectPage.js

На рисунку наведено фрагмент компоненту адміністраторської панелі викладача курсу. Форма реалізована за технологіями звичайної верстки на HTML, але з елементами React. При будь-якому вводі даних в Input викликається трігер «onChange», який викликає функцію «setNewSubject». Це сеттер, який було створено за допомогою функціїї useState, який складається з початкового значення (null, false, об’єкт, тощо) та записується в змінні геттера та сеттера (див. рис. 3.13). В сеттер передається об’єкт, в який передається спред-оператор та новий елемент об’єкту, що теж є особливістю мови програмування JavaScript.

Геттер також використовується у цій структурі, в параметрі «value». Загалом структура наступна: спочатку в формі відображається стандартний параметр хуку, при будь-якому вводі з клавіатури викликається сеттер, а значення геттера, відповідно, оновлюється через поле «value». Таким чином зберігається стан конкретного поля форми.

```
const [selectedSubject, setSelectedSubject] = useState(null);
const [isEditing, setIsEditing] = useState(false);
const [newSubject, setNewSubject] = useState({ name: '', description: '' });
```

Рисунок 3.13 – Приклад використання хуків React

У кнопках також є зміни, відносно стандартної верстки чистим HTML. Параметр «onClick» викликається майже завжди, тому що це один з найпопулярніших способів надати ефект натиску на кнопки. На рисунку 3.14 наведено наступні приклади: оновлення матеріалу та створення матеріалу.

Перша функція – «handleAddSubject». Тут використовується сеттер «setSubjects», котрий очікує масив об’єктів (в даному випадку – список матеріалів). Аналогічно до попередніх прикладів, використовується спред-оператор, завдяки якому до масиву попередніх матеріал додається новий, який складається з назви та опису.

Друга функція – «handleEditSubject». Вона майже ідентична до

попередньої функції, окрім параметрів сеттеру. Так як виконується оновлення конкретного об’єкту масиву, спочатку його треба обрати за допомогою функції «map», у якому фільтром є параметр «id». Якщо його знайдено – використовуємо два спред-оператори для оновлення елементу, якщо цей елемент не підходить – залишаємо його в початковому стані. Для порівняння використовується тернарний оператор, але практично можна замінити його стандартним оператором «if».

```
const handleAddSubject = () => {
  setSubjects((prevSubjects) => [
    ...prevSubjects,
    { id: prevSubjects.length + 1, ...newSubject },
  ]);
  setNewSubject({ name: '', description: '' });
};

const handleEditSubject = () => {
  if (selectedSubject) {
    setSubjects((prevSubjects) =>
      prevSubjects.map((subject) =>
        subject.id === selectedSubject.id ? { ...subject, ...newSubject } : subject
      )
    );
    setNewSubject({ name: '', description: '' });
    setSelectedSubject(null);
    setIsEditing(false);
  }
};
```

Рисунок 3.14 – Функції створення та оновлення матеріалу

Останній популярний хук в React JS – «useEffect». В цьому ж компонтенті при оновленні даних, вони повинні відправитися на сервер, щоб оновити базу даних. Для цього використовуємо знову пакет «axios» і кінцеву точку «material» (див. рис. 3.15). Для цієї функції додатково використовується блок try-catch, задля відображення помилки. Додатково, якщо не використовувати такий блок, то при виникненні помилки з API сайт повністю стане недоступним. Це може бути критичним, якщо в API помилка виникає саме на цій сторінці, тому кожний запит треба обов’язково ізолювати від всієї логіки сторінки. Весь код компоненту продемонстровано в додатку А.

```
useEffect(() => {
  try {
    axios.post('http://127.0.0.1:8000/materials', subjects);
  } catch (e) {
    console.error('Error API Materials:', e)
  }
}, [subjects]);
```

Рисунок 3.15 – Відправлення нових матеріалів на вебсервер

Фільтрація також може виконуватися функцією «filter». Випадки використання цієї функції різноманітні, але найчастіше це стосується відбору елементів, без необхідності заміни. Приклад використання цієї функції наведено на рисунку 3.16.

```
const calculateAverage = (grades) => {
  const filteredGrades = Object.values(grades).filter((grade) => grade !== '0');
  if (filteredGrades.length === 0) return 0;

  const sum = filteredGrades.reduce((acc, grade) => acc + parseInt(grade), 0);
  return (sum / filteredGrades.length).toFixed(2);
};
```

Рисунок 3.16 – Функція розрахунку середньої оцінки

В математичних розрахунках популярним рішенням є використання reduce функцій, які повертають не об’єкт або масив, а число. На рисунку 3.16 також наведено приклад такої функції, де всі елементи масиву попередньо переводяться в цілі числа, додаються один до одного. Якщо елементів 0 – повертається число 0, інакше в змінну «sum» повертається середнє арифметичне.

3.3 Модульне тестування системи управління навчанням

Розробка вебдодатків – це складний процес, який включає в себе ряд послідовних етапів. Перший з них – це збір даних. Цей етап включає в себе

визначення вимог до додатка, вивчення потреб користувачів, а також аналіз ринку і конкурентів.

Наступний етап – проєктування. На цьому етапі розробники визначають архітектуру додатка, проєктують його інтерфейс і взаємодію з користувачем. Важливою частиною цього етапу є створення прототипів і макетів, які допомагають визначити зовнішній вигляд додатка і його функціональність.

Третій етап – програмування. На цьому етапі розробники пишуть код, який реалізує всі заплановані функції додатка. Вони також створюють базу даних, яка зберігає всю необхідну інформацію.

Останній етап – тестування. Цей етап включає в себе перевірку роботи додатка, виявлення і виправлення помилок. Одним з видів тестування є автоматизоване тестування, яке дозволяє автоматично перевіряти роботу додатка і виявляти помилки. У контексті розробки системи управління навчанням використовувалися модульне тестування та додатково застосовувалося ручне тестування, яке охоплювало перевірку функціоналу кожного компоненту.

Під час ручного тестування, яке було проведено паралельно з розробкою, виявлялися різноманітні помилки, які невідкладно виправлялися. Хоча повне усунення всіх помилок є практично неможливим завданням, акцент робився на їх мінімізації, яка є важливим результатом успішного проходження етапів тестування.

Перед початком розробки системи автоматизованого тестування необхідно визначити комплект тестових випадків, оскільки вони є важливими для створення автоматичних тестів. Важливо перевірити, як система поводить себе в позитивних та негативних сценаріях, наприклад, якщо було введено недійсні дані. Тому було створено тестові випадки для кожного модуля.

Перший модуль – авторизація здобувача освіти, викладача, або адміністратора. Для початку роботи з системою управління навчанням користувачеві потрібно зайти в особистий кабінет. При позитивному результаті користувач авторизується в LMS. Тесткейс наведено у таблиці 3.1.

Таблиця 3.1 – Перевірка валідації полей на сторінці авторизації

№	Крок	Очікуваний результат
1	Відкрити сторінку “Login”	Відображається сторінка “Login”
2	Заповнити поле “Email” невірними даними. Приклади: tester@@gmail.com, testergmail.com, tester@gmail	Нижче поля “Email” відображається помилка “Некоректний email”, а рамка поля підсвічується червоним кольором.
3	Залишите поле “Password” пустим та натиснути “Login”	Нижче поля “Password” відображається помилка “Введіть пароль”, а рамка поля підсвічується червоним кольором.
4	Заповнити поле “Email” вірними даними, а в поле “Password” вести невірний пароль, натиснути “Login”.	Вище кнопки “Login” відображається помилка “Невірні дані авторизації”, а рамки полей “Email” та “Password” підсвічується червоним кольором.

Другий важливий модуль – здача роботи здобувачем освіти.Для здачі роботи кожному здобувачу освіти потрібно заповнити форму даними, основна задача – відхилити здачу роботи якщо файл завеликий, або всі поля порожні. Тесткейс наведено у таблиці 3.2.

Таблиця 3.2 – Перевірка валідації полей на сторінці здачі роботи

№	Крок	Очікуваний результат
1	Відкрити сторінку “Здача роботи”	Відображається сторінка “Здача роботи”
2	Пропустити заповнення полей “Текст” та “Файл”	Над кнопкою “Здача” відображається помилка “Відправлення порожньої роботи неможливо”, а рамка всіх полів підсвічується червоним кольором.

Продовження таблиці 3.2

№	Крок	Очікуваний результат
3	Натиснути на кнопку “Оберіть файл”, обрати файл розміром вище 15мб, натиснути “Обрати”	Поле “Файл” підсвічується червоним кольором. Над полем відображається помилка “Завантажено завеликий файл”. Завантаження файлу відхиляється.
4	Заповнити поле “Email” вірними даними, а в поле “Password” вести невірний пароль, натиснути “Login”.	Вище кнопки “Login” відображається помилка “Невірні дані авторизації”, а рамки полей “Email” та “Password” підсвічуються червоним кольором.

Існують також поля створення робіт у викладача або адміністратора, але валідація виконується аналогічно, тому відображати їх додатково не є доцільним.

Також, було протестовано журнал оцінок користувачів. Єдина важлива в цьому компоненті валідація – перевірка введеної викладачем оцінки, щоб вона не була менше нуля та не більше системного обмеження (у тестовому випадку – 12 балів).

Зазначено, що у процесі розробки системи було використано модульне тестування компонентів, зокрема, фокус був спрямований на розрахунковий функціонал frontend складової, а саме – компоненту журналу оцінок студентів. Для цього використовувалися вбудовані пакети «testing-library» в середовищі React JS.

Приклад модульного тесту розрахування середнього арифметичного для кожного студента наведено у вигляді прикладу на рисунку 3.17 [15].

Модульне тестування відрізняється від ручного тестування тим, що воно дозволяє перевіряти кожну окрему функцію програми. Це особливо корисно для функцій, які виконують технічні розрахунки і мають передбачувані результати.

```
test('calculateAverage function calculates average grade correctly', () => {
  const grades = {
    '2023-01-10': '9',
    '2023-01-12': '11',
    '2023-01-15': '7',
    '2023-01-20': '10',
    '2023-01-25': '8',
    '2023-02-01': '9',
    '2023-02-05': '7',
    '2023-02-10': '8',
    '2023-02-15': '11',
  };

  const average = SchoolJournal.calculateAverage(grades);

  expect(average).toBe('9.22'); // Замініть значення залежно від вашого розрахунку
});
```

Рисунок 3.17 – Тестування середньої оцінки в LMS

Кожен тест має свою унікальну назву, що допомагає розробникам краще розуміти, як працює програма, і виявляти помилки. Перед початком тестування визначаються початкові параметри даних, такі як змінні, об’єкти, масиви тощо. Потім викликається тестована функція, а отримані результати автоматично порівнюються з очікуваними.

Цей підхід дозволяє ефективно контролювати правильність роботи кожної функції і швидко виявляти можливі невідповідності. Наприклад, на рисунку 3.18 показано результати тестування розрахунку середнього арифметичного для кожного студента.

```
PASS src/tests/schoolJournal.test.js (5.266 s)
  √ calculateAverage function calculates average grade correctly (2 ms)

A worker process has failed to exit gracefully and has been force exited. This is likely caused by tests leaking due to improper teardown. Try running with --dete
ctOpenHandles to find leaks. Active timers can also cause this, ensure that .unref() was called on them.
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        8.547 s
Ran all test suites related to changed files.

Watch Usage: Press w to show more.
```

Рисунок 3.18 – Приклад результату модульного тестування

Для тестування backend частини був використаний пакет unittest, який надає широкий спектр можливостей та зручні mock [16]. Приклад тесту для Django-об’єкту наведено на рисунку 3.19.

```python
class UserTestCase(TestCase):
    def setUp(self):
        Person.objects.create(first_name="Kyrylo", last_name="Mytroshyn")
        Person.objects.create(first_name="Test", last_name="User")

    def user_full_name(self):
        first_person = Person.objects.get(first_name="Kyrylo") # we have only 1 kyrylo
        second_person = Person.objects.get(first_name="Test") # we have only 1 test user
        self.assertEqual(first_person.getFullName(), 'Kyrylo Mytroshyn')
        self.assertEqual(second_person.getFullName(), 'Test User')
```

Рисунок 3.19 – Приклад модульного тестування Django

Кожний case (тест) реалізується у вигляді окремого класу зі своєю функцією setup та функцію безпосередньо тестування. У setup функції передаються тестові дані для бази даних, задля подальшої роботи з ними. Ці дані видаляться по завершенню тестування, тим самим забезпечуючи безпеку даних в базі даних. У додаткових функціях, аналогічно до тестування у React, виконується якась функція, у даному випадку – отримання об’єктів користувачів. Остання дія кожного тесту – використання функції assert (assertEqual, assertGreater, assertTrue, тощо), у параметри якої зазвичай передається функція, яка повертає дані та очікуваний результат виконання функції.

Модульне тестування є важливим інструментом для забезпечення якості програмного забезпечення. Воно дозволяє розробникам виявляти і виправляти помилки на ранніх стадіях розробки, що може значно зменшити вартість і час розробки. Крім того, модульне тестування може допомогти забезпечити, що програма буде працювати правильно під час її використання.

Важливо зазначити, що, хоча модульне тестування може бути дуже корисним, воно не може замінити інші форми тестування, такі як інтеграційне тестування, системне тестування або приймальне тестування. Кожен з цих видів тестування має свою специфіку і важливість, і всі вони разом допомагають забезпечити високу якість кінцевого продукту.

Таким чином, модульне тестування є важливою частиною процесу розробки програмного забезпечення. Воно допомагає розробникам виявляти і

виправляти помилки на ранніх стадіях розробки, що може значно підвищити якість кінцевого продукту і зменшити вартість його розробки. Однак, необхідно пам’ятати, що модульне тестування – це лише один з інструментів, які доступні розробникам для забезпечення якості програмного забезпечення, і його слід використовувати в поєднанні з іншими методами тестування.

3.4 Огляд розробленої системи управління навчанням

Система управління навчанням, про яку йдеться в цьому тексті, є високофункціональним вебдодатком, що надає користувачам широкий спектр можливостей. Вона була розроблена з метою забезпечення ефективного управління навчальним процесом і має ряд ключових особливостей.

Перш за все, користувачі можуть авторизуватися в системі. Це означає, що вони можуть створити власний обліковий запис, який буде зберігати всю їхню інформацію та дозволить їм отримати доступ до всіх функцій системи. Це включає можливість перегляду оцінок, виконання завдань та створення домашніх завдань та тестів. Ці функції дозволяють користувачам активно взаємодіяти з системою і отримувати максимальну користь від її використання.

Електронний журнал є однією з ключових складових системи. Він надає вчителям можливість виставляти оцінки від 1 до 12, а діапазон оцінок може бути адаптований до вимог конкретного навчального закладу. Журнал також надає докладну інформацію про користувача, включаючи його ПІБ, групу або клас, а також унікальний ідентифікатор. Ця інформація допомагає вчителям краще розуміти потреби своїх студентів і адаптувати свої методики навчання відповідно.

Остання колонка журналу автоматично розраховує середній бал за всіма оцінками користувача для даного курсу чи предмету. Це дозволяє вчителям легко відстежувати успішність студентів і виявляти області, які можуть вимагати додаткової уваги. Ця функція є важливим інструментом для

моніторингу прогресу студентів і допомагає вчителям забезпечити, що всі студенти розвиваються належним чином.

Вчителі також можуть редагувати вже виставлені оцінки, що спрощує процес ведення журналу. Зміни зберігаються на backend та в базі даних, гарантуючи точність та надійність даних. Це означає, що вчителі можуть бути впевнені, що їхні оцінки будуть збережені безпечно і точно. Зразок журналу інтегровано в інтерфейс та відображено на рисунку 3.20.

Students	2023-01-10	2023-01-12	2023-01-15	2023-01-20	2023-01-25	2023-02-01	2023-02-05	2023-02-10	2023-02-15	Average Grade
John Doe Class: 10A Student UID: 1	10	8	11	7	9	12	8	10	9	9.33
Jane Smith Class: 10A Student UID: 2	9	11	7	10	8	9	7	8	11	8.89
Test User Class: 10A Student UID: 3	9	11	7	10	8	9	7	8	11	8.89
Kyrylo Mytroshyn Class: 10A Student UID: 4	9	11	7	10	8	9	7	8	11	8.89

Рисунок 3.20 – Журнал оцінок LMS «Cyber Education»

Також важливо відзначити, що при виставленні оцінки 0 балів за певний матеріал, вона автоматично розглядається як невиставлена, і цей бал не враховується у розрахунках середнього. Ця логіка може бути змінена відповідно до вимог навчального закладу, особливо якщо використовується система кумулятивного підрахунку. Це дозволяє вчителям гнучко налаштовувати систему оцінювання відповідно до своїх потреб.

Щодо системи тестування, вона побудована на принципі «Питання – 4 варіанта відповіді». Цей підхід може бути гнучко налаштований адміністратором системи, що дозволяє адаптувати тести до особливостей навчального процесу. Це означає, що вчителі можуть створювати тести, які

відповідають конкретним вимогам їхніх курсів або предметів.

Для створення нового тесту адміністратор повинен ввести його назву та створити відповіді до питань. Це робить процес створення тестів простим і зручним, що сприяє ефективності навчального процесу. Крім того, це дозволяє вчителям створювати тести, які відповідають конкретним вимогам їхніх курсів або предметів, що допомагає забезпечити, що студенти отримують найбільш відповідне тестування.

Всі ці функції разом створюють потужний інструмент для управління навчальним процесом, який може бути адаптований до вимог будь-якого навчального закладу. Це робить систему управління навчанням важливим інструментом для підтримки якості освіти. Завдяки цим функціям вчителі можуть краще відстежувати прогрес своїх студентів, а студенти можуть отримати більше від свого навчального досвіду. Загалом, ця система управління навчанням є важливим кроком вперед у сфері освіти, який допомагає підвищити якість навчання та забезпечити кращі результати для студентів.

Інтерфейс системи тестування зі сторони адміністратор відображено на рисунку 3.21, підкреслюючи його зручність та легкість використання.

У системі управління навчанням, процес тестування для студентів розроблено таким чином, що вони отримують лише одне питання за раз. Це означає, що після того, як студент надає відповідь на поточне питання, система автоматично видає наступне питання. Цей процес продовжується до тих пір, поки тестування не буде завершено.

Після завершення тестування система відображає остаточну оцінку студента. Це дозволяє студентам отримати негайний зворотний зв’язок про свою успішність, що може бути корисним для їхнього навчального процесу.

Однією з ключових особливостей цієї системи є те, що правильні відповіді не відображаються після тестування. Це робиться для мінімізації небажаного розповсюдження правильних відповідей серед студентів. Ця функція допомагає забезпечити, що кожен студент має рівні можливості під час проходження тесту.

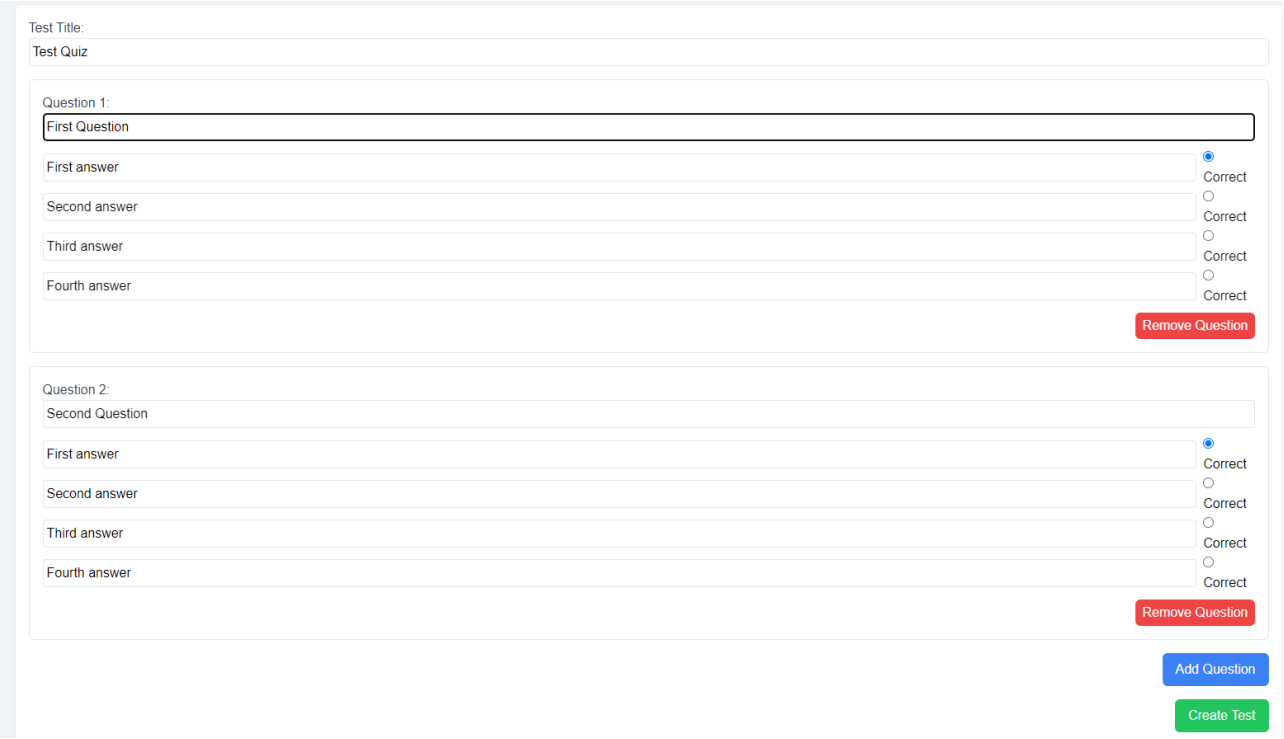

Рисунок 3.21 – Створення або редагування тесту в LMS «Cyber Education»

Інтерфейс тестування є простим і зручним для користувачів. Він був розроблений з урахуванням потреб користувачів, щоб забезпечити максимальну зручність під час проходження тестів (див. рис. 3.22). Це допомагає студентам зосередитися на відповідях на питання, а не на навігації по системі.

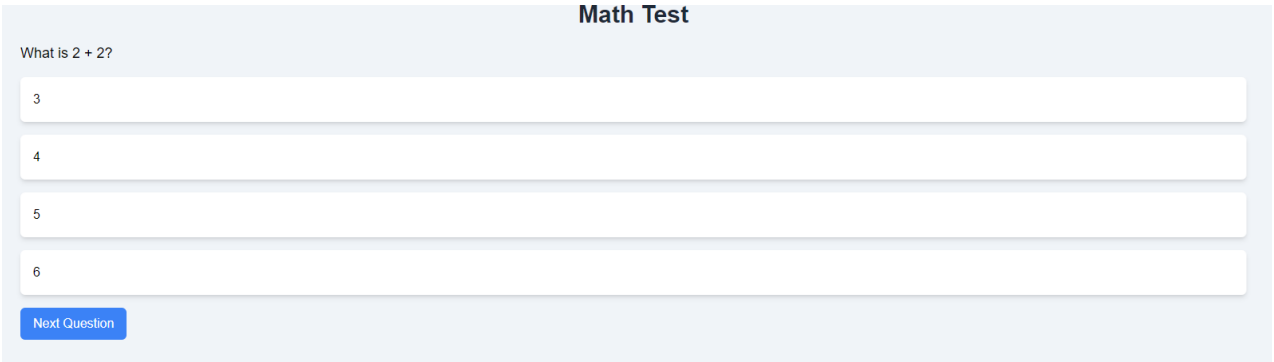

Рисунок 3.22 – Сторінка тесту для студента в LMS «Cyber Education»

Всі ці функції разом створюють потужний інструмент для управління навчальним процесом, який може бути адаптований до вимог будь-якого

навчального закладу. Це робить систему управління навчанням важливим інструментом для підтримки якості освіти. Завдяки цим функціям вчителі можуть краще відстежувати прогрес своїх студентів, а студенти можуть отримати більше від свого навчального досвіду. Загалом, ця система управління навчанням є важливим кроком вперед у сфері освіти, який допомагає підвищити якість навчання та забезпечити кращі результати для студентів.

Кожен викладач має можливість створити новий курс або видалити існуючий. При створенні нового курсу та додаванні нових студентів до нього, цей курс автоматично додається до списку курсів, які доступні для студентів. В самому курсі викладач може розміщувати різноманітний контент, такий як завдання, відеоматеріали, тести, з якими студенти можуть взаємодіяти та здійснювати навчання. Приклад сторінки адміністратора та його можливостей з управління курсами наведено на рисунку 3.23.

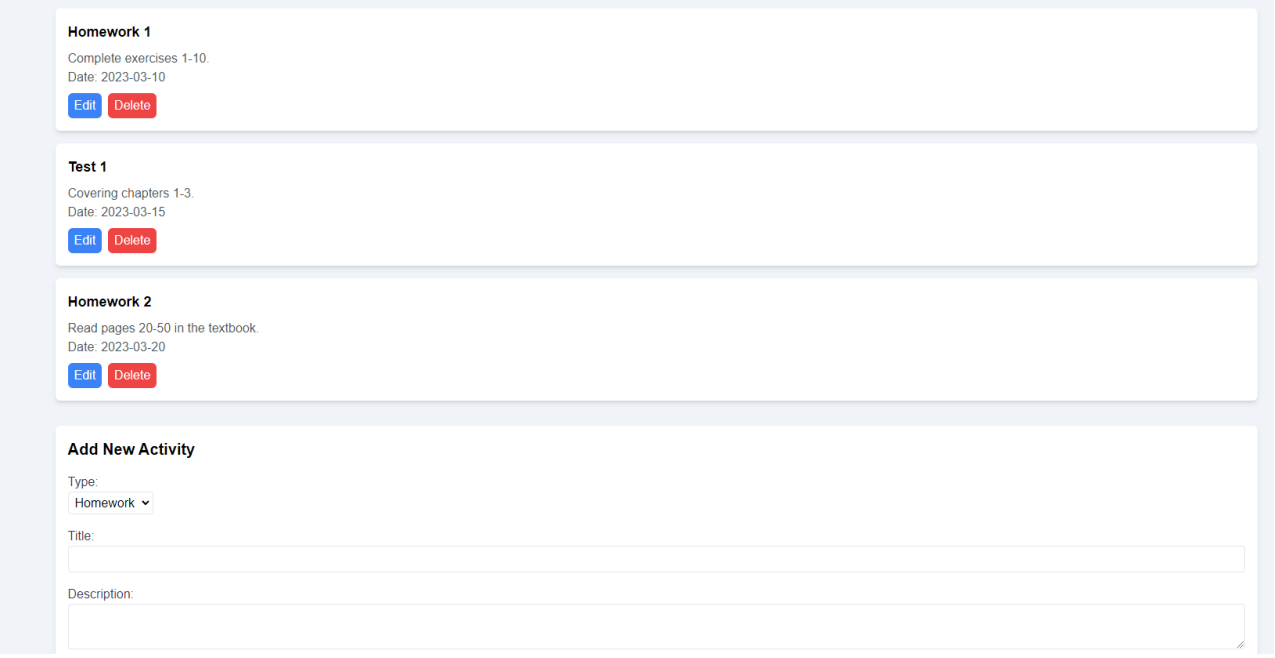

Рисунок 3.23 – Сторінка курсу користувача-адміністратора

ВИСНОВКИ

Проєктування та розробка системи управління навчанням включили в себе комплексний підхід до використання передових технологій. В процесі аналізу програмних засобів для вебдодатків, особливу увагу приділено системам авторизації та управління сеансами користувачів. Результатом став високофункціональний продукт, що базується на потужних інструментах розробки, таких як React JS, PostgreSQL, Python та фреймворки Django та Django Rest Framework.

Глибокий аналіз вимог до системи був проведений на всіх етапах проєкту. Починаючи від концептуального проєктування, де створено детальний макет Learning Management System (LMS) з урахуванням потреб користувачів, до розробки діаграми прецедентів, яка відображає концепцію роботи системи та технічні можливості кожної ролі користувача.

Одним з ключових елементів системи є авторизація, яку детально проілюстровано в другому розділі та реалізовано з використанням «Json Web Token». Принципи підтримки користувача в сесії та видача токену після успішної авторизації були детально вивчені та реалізовані в рамках проєкту.

Логічне проєктування включало розробку діаграми класів, що визначила основні об’єкти системи та їх взаємодію. Фізичне проєктування, у свою чергу, було підтримано діаграмою компонентів, що визначає архітектурну структуру додатку та його взаємодію з технічним оснащенням.

Для реалізації програмного забезпечення були завантажені та налаштовані наступні інструменти та фреймворки: «PyCharm», «VS Code», «React JS», «Django», «Django Rest Framework». Окрім того, проведено підготовку бази даних, включаючи завантаження та актуалізацію драйверів для забезпечення ефективної роботи.

Після підготовки системи до розробки застосунку, були реалізовані візуальні компоненти, як-от «TeacherSubjectPage.js», «SchoolJournal.js»,

«AdminSubjectPage.js», виконана налаштування бази даних та заповнення тестовими даними для проведення ефективного тестування. Кожен компонент був предметом ручного тестування, а також пройшов модульне автоматичне тестування як на рівні backend, так і на рівні frontend.

Узагальнюючи підхід до розробки, було не лише наголошено на функціональності, але й віддано важливість гарантії високої надійності та ефективності системи. Кінцевий результат не тільки відзначається технічною вдосконаленістю, але й повністю відповідає всім поставленим вимогам і стандартам, що гарантує успішне впровадження та стабільну роботу створеної системи управління навчанням.

ПЕРЕЛІК ПОСИЛАНЬ

1. Митрошин К. В., Кривохата А. Г. Розробка інформаційної системи управління навчанням засобами React. *Актуальні проблеми математики та інформатики* : збірка тез доповідей Чотирнадцятої Всеукраїнської, двадцять першої регіональної наукової конференції молодих дослідників. (Запоріжжя, 27–28 квітня 2023). Запоріжжя : ЗНУ, 2023. С. 65–66. URL: https://www.znu.edu.ua//faculty/math/confirences/tezi_apm___23.pdf (дата звернення: 23.09.2023).
2. Система управління навчанням Moodle. URL: https://moodle.org/ (дата звернення: 20.06.2023).
3. Система управління навчанням Google Class. URL: https://classroom.google.com/ (дата звернення: 20.06.2023).
4. Система управління навчанням BlackBoard. URL: https://www.blackboard.com/ (дата звернення: 20.06.2023).
5. Система управління навчанням Talent LMS. URL: https://www.talentlms.com/ (дата звернення: 20.06.2023).
6. Система управління навчанням Nearpod. URL: https://nearpod.com/ (дата звернення: 20.06.2023).
7. React JS. Overview. URL: https://react.dev/reference/react (дата звернення: 20.07.2023).
8. JSON Web Token Introduction. URL: https://jwt.io/introduction (дата звернення: 19.08.2023).
9. Django Docs. URL: https://www.djangoproject.com/ (дата звернення: 20.09.2023).
10. Django Rest Framework. URL: https://www.django-rest-framework.org/ (дата звернення: 23.10.2023).
11. Simple JWT Package. URL: https://django-rest-framework-simplejwt.readthedocs.io/en/latest/ (дата звернення 20.09.2023).

12. NPM | Home. URL: https://www.npmjs.com/ (дата звернення: 20.08.2023).
13. Tailwind CSS. URL: https://tailwindcss.com/ (дата звернення: 23.09.2023).
14. Universal Cookie Package. URL: https://www.npmjs.com/package/universal-cookie (дата звернення: 10.10.2023).
15. React Testing Library. URL: https://testing-library.com/docs/react-testing-library/intro/ (дата звернення: 11.10.2023).
16. Unittest library. URL: https://docs.python.org/3/library/unittest.html#module-unittest (дата звернення: 23.10.2023).

ДОДАТОК А

Компонент TeacherSubjectPage

```
import axios from 'axios';
import React, { useEffect, useState } from 'react';

const TeacherSubjectPage = (subjectsData) => {
  const [subjects, setSubjects] = useState(subjectsData);

  const [selectedSubject, setSelectedSubject] = useState(null);
  const [isEditing, setIsEditing] = useState(false);
  const [newSubject, setNewSubject] = useState({ name: '', description: '' });



  useEffect(() => {
    try {
      axios.post('http://127.0.0.1:8000/materials', subjects);
    } catch (e) {
      console.error('Error API Materials:', e)
    }
  }, [subjects]);

  const handleAddSubject = () => {
    setSubjects((prevSubjects) => [
      ...prevSubjects,
      { id: prevSubjects.length + 1, ...newSubject },
    ]);
    setNewSubject({ name: '', description: '' });
  };
```

```
  const handleEditSubject = () => {
    if (selectedSubject) {
      setSubjects((prevSubjects) =>
        prevSubjects.map((subject) =>
          subject.id === selectedSubject.id ? { ...subject, ...newSubject } : subject
        )
      );
      setNewSubject({ name: '', description: '' });
      setSelectedSubject(null);
      setIsEditing(false);
    }
  };

  const handleSelectSubject = (subject) => {
    setSelectedSubject(subject);
    setNewSubject({ name: subject.name, description: subject.description });
    setIsEditing(true);
  };

  const handleCancelEdit = () => {
    setSelectedSubject(null);
    setNewSubject({ name: '', description: '' });
    setIsEditing(false);
  };

  return (
    <div className="container mx-auto mt-8">
      <h2 className="text-3xl font-bold mb-4 text-center text-blue-700">Teacher
Subjects</h2>
```

```
    <div className="flex mb-4">
      <input
        type="text"
        placeholder="Subject Name"
        value={newSubject.name}
        onChange={(e) => setNewSubject({ ...newSubject, name: e.target.value })}
        className="border rounded p-2 mr-2"
      />
      <input
        type="text"
        placeholder="Description"
        value={newSubject.description}
        onChange={(e) => setNewSubject({ ...newSubject, description: e.target.value
})}
        className="border rounded p-2 mr-2"
      />
      {isEditing ? (
        <>
          <button onClick={handleEditSubject} className="bg-blue-500 text-white
px-4 py-2 rounded">
            Update
          </button>
          <button onClick={handleCancelEdit} className="bg-gray-500 text-white
px-4 py-2 rounded ml-2">
            Cancel
          </button>
        </>
      ) : (
```

```
        <button onClick={handleAddSubject} className="bg-green-500 text-white px-4 py-2 rounded">
          Add
        </button>
      )}
    </div>

    <ul className="divide-y divide-yellow-400">
      {subjects.map((subject) => (
        <li key={subject.id} className="py-4">
          <div className="bg-gradient-to-b from-yellow-400 to-orange-500 rounded-lg shadow-md p-4">
            <h3 className="text-2xl font-semibold text-white mb-2">{subject.name}</h3>
            <p className="text-gray-800">{subject.description}</p>
            <div className="mt-2">
              <button
                onClick={() => handleSelectSubject(subject)}
                className="bg-blue-500 text-white px-3 py-1 rounded mr-2"
              >
                Edit
              </button>
              <button
                onClick={() =>
                  setSubjects((prevSubjects) =>
                    prevSubjects.filter((s) => s.id !== subject.id)
                  )
                }
                className="bg-red-500 text-white px-3 py-1 rounded"
              >
```

```
                Delete
              </button>
            </div>
          </div>
        </li>
      ))}
    </ul>
  </div>
 );
};

export default TeacherSubjectPage;
```

ДОДАТОК Б

Компонент SchoolJournal

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

const StudentInfo = ({ student }) => {
  return (
    <div className="bg-blue-200 p-4 mb-4">
      <h3 className="text-lg font-bold">{student.name}</h3>
      <p>Class: {student.class}</p>
      <p>Student UID: {student.ID}</p>
    </div>
  );
};

const DateGrade = ({ grade, onGradeChange }) => {
  return (
    <div className="mb-2">
      <input
        type="number"
        min="1"
        max="12"
        value={grade}
        onChange={(e) => onGradeChange(e.target.value)}
        className="border rounded p-1 text-center w-12"
      />
    </div>
  );
};
```

```
const calculateAverage = (grades) => {
  const filteredGrades = Object.values(grades).filter((grade) => grade !== '0');
  if (filteredGrades.length === 0) return 0;

  const sum = filteredGrades.reduce((acc, grade) => acc + parseInt(grade), 0);
  return (sum / filteredGrades.length).toFixed(2);
};

const SchoolJournal = (studentsData) => {
  const [students, setStudents] = useState(studentsData);

  const [currentDateIndex, setCurrentDateIndex] = useState(0);

  const handleGradeChange = (studentIndex, newGrade) => {
    setStudents((prevStudents) => {
      const updatedStudents = [...prevStudents];
      const updatedStudent = { ...updatedStudents[studentIndex] };
      const currentDate = Object.keys(updatedStudent.grades)[currentDateIndex];
      updatedStudent.grades[currentDate] = newGrade;
      updatedStudents[studentIndex] = updatedStudent;
      return updatedStudents;
    });
  };

  return (
    <div className="container mx-auto mt-8">
      <h2 className="text-2xl font-bold mb-4 text-center text-blue-700">
        School Electronic Journal
      </h2>
```

```
<input
  type="range"
  min="0"
  max={Object.keys(students[0].grades).length - 1}
  step="1"
  value={currentDateIndex}
  onChange={(e) => setCurrentDateIndex(parseInt(e.target.value))}
  className="slider mb-4"
/>

<table className="w-full border-collapse border">
  <thead>
    <tr>
      <th className="border p-2 bg-blue-500 text-white">Students</th>
      {Object.keys(students[0].grades).map((date, index) => (
        <th key={index} className="border p-2 bg-blue-500 text-white">
          {date}
        </th>
      ))}
      <th className="border p-2 bg-blue-500 text-white">Average Grade</th>
    </tr>
  </thead>
  <tbody>
    {students.map((student, index) => (
      <tr key={index}>
        <td className="border p-2 w-48">
          <StudentInfo student={student} />
        </td>
        {Object.keys(student.grades).map((date, dateIndex) => (
```

```
            <td
              key={dateIndex}
              className={`border p-2 ${
                dateIndex === currentDateIndex ? 'bg-yellow-200' : ''
              }`}
            >
              <DateGrade
                grade={student.grades[date]}
                onGradeChange={(newGrade) =>
                  handleGradeChange(index, newGrade)
                }
              />
            </td>
          ))}
          <td className="border p-2">{calculateAverage(student.grades)}</td>
        </tr>
      ))}
    </tbody>
  </table>
 </div>
 );
};

export default SchoolJournal;
```