МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ВЕБСАЙТУ ДЛЯ ОРГАНІЗАЦІЇ РОБОТИ РЕПЕТИТОРА»**

Виконав: студент 4 курсу, групи 6.1210-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
Б.Р. Ковбаса
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.т.н., Мухін В.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н., Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

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

Ковбасі Богдану Руслановичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка вебсайту для організації роботи репетитора

керівник роботи Мухін Віталій Вікторович, к.т.н, доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 21 » грудня 2023 року № 2180-с

2. Строк подання студентом роботи 03.06.2024 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. Реалізація серверної та вебчастини програмного забезпечення.

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

6. Консультанти розділів роботи

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

7. Дата видачі завдання 25.12.2023 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.	22.01.2024	
2.	Збір вихідних даних.	12.02.2024	
3.	Обробка методичних та теоретичних джерел.	04.03.2024	
4.	Розробка першого та другого розділу.	15.04.2024	
5.	Розробка третього та четвертого розділу.	17.05.2024	
6.	Оформлення та нормоконтроль	27.05.2024	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	19.06.2024	

Студент _____ Б.Р.Ковбаса
(підпис) (ініціали та прізвище)

Керівник роботи _____ В.В. Мухін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

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

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка вебсайту для організації роботи репетитора»: 53 с., 43 рис., 19 джерел, 2 додатки.

БАЗА ДАНИХ, ІНТЕРФЕЙС, MONGO, NODE JS, REACT JS.

Об’єкт дослідження – процес організації репетиторської діяльності, управління розкладом, обробка комунікацій та зберігання даних користувачів.

Мета роботи – розробка клієнтської та серверної частин інформаційної системи для управління взаємодією між репетиторами та студентами.

Методи дослідження – аналіз предметної області, моделювання предметної області, об’єктно-орієнтований підхід.

Результат роботи – розроблений програмний модуль, який дозволяє створювати профілі репетиторів, керувати розкладом, полегшувати комунікацію між репетиторами та студентами, а також забезпечувати безпечне зберігання даних.

SUMMARY

Bachelor’s qualifying paper “Development of a Website for a Tutor’s Work Organizing”: 53 pages, 43 figures, 19 references, 2 supplements.

DATABASE, INTERFACE, MONGO, NODE JS, REACT JS.

Object of the study – the process of organizing tutoring activities, managing schedules, handling communication, and storing user data.

Aim of the study is development of the client and server parts of the information system for managing tutor-student interactions.

Methods of research is domain analysis, domain modeling, object-oriented approach.

The result of the work – a developed software module that allows creating tutor profiles, managing schedules, facilitating communication between tutors and students, and ensuring secure data storage.

ЗМІСТ

ВСТУП

У сучасному світі значна частина діяльності перейшла в цифровий формат, зокрема і освіта. Організація роботи репетиторів через вебсайти стає все більш популярною та необхідною. Згідно з останніми дослідженнями, все більше студентів та викладачів обирають онлайн-платформи для навчання, що сприяє зручності та ефективності навчального процесу [1, 2].

Розробка вебсайту для організації роботи репетитора є важливим кроком у розвитку сучасної освітньої системи. Такий вебсайт не тільки полегшує взаємодію між викладачем і студентом, але й робить процес навчання більш доступним та гнучким. Вебсайти для репетиторів дозволяють точно планувати заняття, швидко вносити зміни до розкладу та забезпечують безперервність навчального процесу [3].

Основні переваги вебсайту для організації роботи репетитора включають точність планування, оперативний доступ до розкладу та матеріалів, безперервність навчального процесу, поліпшену комунікацію з клієнтами, та гнучкість у адаптації розкладу занять відповідно до потреб кожного студента. Вони сприяють прозорості навчального процесу, прискорюють комунікацію та підтримують розвиток освітньої сфери [4].

Метою цієї кваліфікаційної роботи є розробка вебсайту для організації роботи репетитора. Цей вебсайт буде основною платформою для взаємодії викладачів та студентів, забезпечуючи їм зручний та ефективний інструмент для проведення занять. У роботі розглядаються основні функціональні можливості вебсайту, методи їх реалізації, а також питання безпеки та захисту даних користувачів [5].

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галзуі

Ефективне управління діяльністю репетиторів є ключовим елементом сучасної освітньої системи. Вебсайти для репетиторів не лише спрощують навчальний процес, але й роблять його більш доступним для широкої аудиторії. Такі вебсайти сприяють прозорості навчального процесу, прискорюють комунікацію та підтримують розвиток освітньої сфери. На сьогодні вони повинні бути основою будь-якої стратегії, спрямованої на розвиток репетиторської діяльності в майбутньому [6].

Основні переваги вебсайту для організації роботи репетитора:

– точність: вебсайт забезпечує точне планування та відстеження занять, дозволяючи репетиторам та студентам уникати непорозумінь;
– швидкість: оперативний доступ до розкладу, матеріалів та можливість швидко вносити зміни до розкладу;
– безперервність: онлайн платформа дозволяє проводити заняття в будь-який час та з будь-якого місця, забезпечуючи безперервність навчального процесу;
– комунікація з клієнтом: вбудовані чати та повідомлення дозволяють швидко реагувати на запити студентів, покращуючи взаємодію [7];
– гнучкість: можливість адаптувати розклад занять та методи навчання відповідно до потреб кожного студента.

Ефективне управління уроками та комунікація з учнями сьогодні неможливі без використання сучасних технологій, і в майбутньому ця роль стане ще важливішою. Вебсайти для репетиторів підтримують не тільки індивідуальних викладачів, але й освітні центри, надаючи їм можливість швидко адаптуватися до нових умов та вимог ринку. У сфері освіти транзакції та комунікації через Інтернет або мобільні пристрої стали незамінними [8].

1.2 Ідентифікація проблем та розробка актуальних рішень

Одна з основних проблем вебсайтів для репетиторів – це відсутність точної системи планування. Репетитори та студенти часто стикаються з непорозумінням щодо розкладу занять. Для вирішення цієї проблеми необхідно впровадити функціонал для точного планування та відстеження занять. Наприклад, інтерактивний календар з можливістю синхронізації з іншими календарними системами забезпечить точність та зручність у плануванні [9, 10].

Недостатня комунікація з клієнтами є ще однією проблемою, яку можна вирішити за допомогою вбудованих чатів та системи повідомлень для швидкого реагування на запити студентів [11]. Це також може включати створення форумів або спільнот, де студенти та репетитори можуть обмінюватися інформацією та досвідом.

Високі транзакційні витрати є ще однією важливою проблемою. Автоматизація процесів бронювання та оплати допоможе зменшити витрати на адміністрування. Інтеграція з платіжними системами для зручного та безпечного проведення оплат, а також можливість налаштування автоматичних рахунків та нагадувань дозволять значно знизити ці витрати.

Недостатня зворотний зв’язок є ще однією проблемою, яку можна вирішити шляхом впровадження системи оцінок та відгуків. Це сприятиме підвищенню якості послуг, дозволяючи студентам залишати відгуки та оцінки після кожного заняття, а також аналізувати зібрані дані для покращення роботи репетиторів [12].

Захист даних та конфіденційність також є важливими питаннями для вебсайтів репетиторів. Використання сучасних методів шифрування та безпеки для захисту даних користувачів допоможе вирішити цю проблему. Регулярні перевірки безпеки, використання двофакторної аутентифікації та регулярне оновлення системи безпеки є необхідними заходами для забезпечення конфіденційності та безпеки даних.

1.3 Постановка задачі

Аналіз предметної області виявив потребу у створенні вебсайту для огранізації роботи репетиторів.

Фунціональні можливості для відвідувачів вебсайту:

– реєстрація;
– авторизація;
– пошук репетиторів;
– огляд профілів репетиторів.

Фунціональні можливості для репетиторів:

– реєстрація;
– авторизація;
– створення профіля;
– створення оголошення;
– редагування профіля;
– редагування оголошення;
– перегляд профілів інших репетиторів;
– комунікація з учнями за допомогою чату;
– можливість вести планер.

Фунціональні можливості для учні:

– реєстрація;
– авторизація;
– створення профіля;
– редагування профіля;
– перегляд профілів репетиторів;
– комунікація з репетиторами за допомогою чату.

2 РОЗРОБКА ВИМОГ ДО СИСТЕМИ

2.1 Розробка системних вимог до інформаційної системи

Інформаційна система для платформи репетиторів має включати такі компоненти:

– вебзастосунок;
– серверну частину;
– базу даних;
– вебзастосунок повинен забезпечувати наступний функціонал:
– реєстрація та авторизація користувачів (студентів та викладачів);
– створення та редагування профілю користувача;
– пошук та фільтрація оголошень репетиторів;
– створення та управління оголошеннями для репетиторів;
– відправлення та отримання повідомлень між студентами та викладачами;
– керування розкладом репетиторів через планувальник;
– можливість завантаження та відображення сертифікатів.

Серверна частина відповідатиме за обробку та відповідь на запити від вебклієнта, міститиме бізнес-логіку та запити до бази даних. Вона має включати наступні компоненти:

– реєстрація та авторизація користувачів;
– обробка даних профілів користувачів;
– обробка даних оголошень;
– обробка повідомлень чату;
– обробка даних планувальника;
– завантаження та зберігання сертифікатів.

Для зберігання даних слід використовувати NoSQL базу даних (наприклад, MongoDB), яка забезпечить масштабованість та гнучкість

реалізації. База даних має зберігати:

– дані користувачів (студентів та викладачів);
– дані оголошень;
– історію повідомлень чату;
– дані планувальника;
– посилання на завантажені сертифікати.

Комунікація між вебзастосунком та серверною частиною повинна здійснюватися через API-інтерфейс. Всі дані, що передаються між компонентами, мають бути зашифровані для забезпечення захищеності зв’язку [13].

Система повинна бути здатною до масштабування для обробки збільшеного навантаження та підтримувати додавання нових функцій без значних змін у вже існуючій інфраструктурі.

Інтерфейс повинен бути інтуїтивно зрозумілим та зручним для користувачів різного віку та технічного рівня [14].

Система повинна мати можливість отримувати оновлення без переривання роботи користувачів та забезпечувати технічну підтримку для вирішення проблем користувачів.

2.2 Функціональні вимоги

Фунціональні можливості для відвідувачів вебсайту:

– реєстрація;
– авторизація;
– пошук репетиторів;
– огляд профілів репетиторів.

Фунціональні можливості для репетиторів:

– реєстрація;
– авторизація;

– створення профіля;
– створення оголошення;
– редагування профіля;
– редагування оголошення;
– перегляд профілів інших репетиторів;
– комунікація з учнями за допомогою чату;
– можливість вести планер.

Фунціональні можливості для учні:

– реєстрація;
– авторизація;
– створення профіля;
– редагування профіля;
– перегляд профілів репетиторів;
– комунікація з репетиторами за допомогою чату.

2.3 Нефунціональні вимоги

Безпека:

– паролі всіх користувачів повинні зберігатися в зашифрованому вигляді з використанням криптографічних алгоритмів;
– база даних має бути захищена від будь-яких зовнішніх запитів, крім запитів з серверної частини.

Супроводжуваність:

– реалізація системи повинна забезпечувати легке масштабування без необхідності змінювати існуючі компоненти;
– код повинен бути добре задокументований для полегшення обслуговування та підтримки;
– використання модульної архітектури для спрощення оновлень та розширення функціональних можливостей.

Доступність:

– система повинна правильно функціонувати в браузерах safari, chrome, firefox та opera;
– вебзастосунок має бути адаптивним для різних пристроїв, включаючи настільні комп’ютери, планшети та смартфони.

Вимоги до користувацького інтерфейсу:

– інтерфейс має бути виконаний у єдиному стилі з використанням повторно використовуваних компонентів;
– інтерфейс повинен враховувати найкращі практики ui/ux дизайну для забезпечення зручності користування;
– інтерфейс має бути реалізований у світлих та темних тонах для забезпечення комфортного перегляду вебсторінок в різних умовах освітлення.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Схема проєктування програмного забезпечення

Система включає три категорії користувачів: гість сайту, репетитор і учень. На рисунку 3.1 зображена схема, що демонструє функціональність і поведінку цих користувачів.

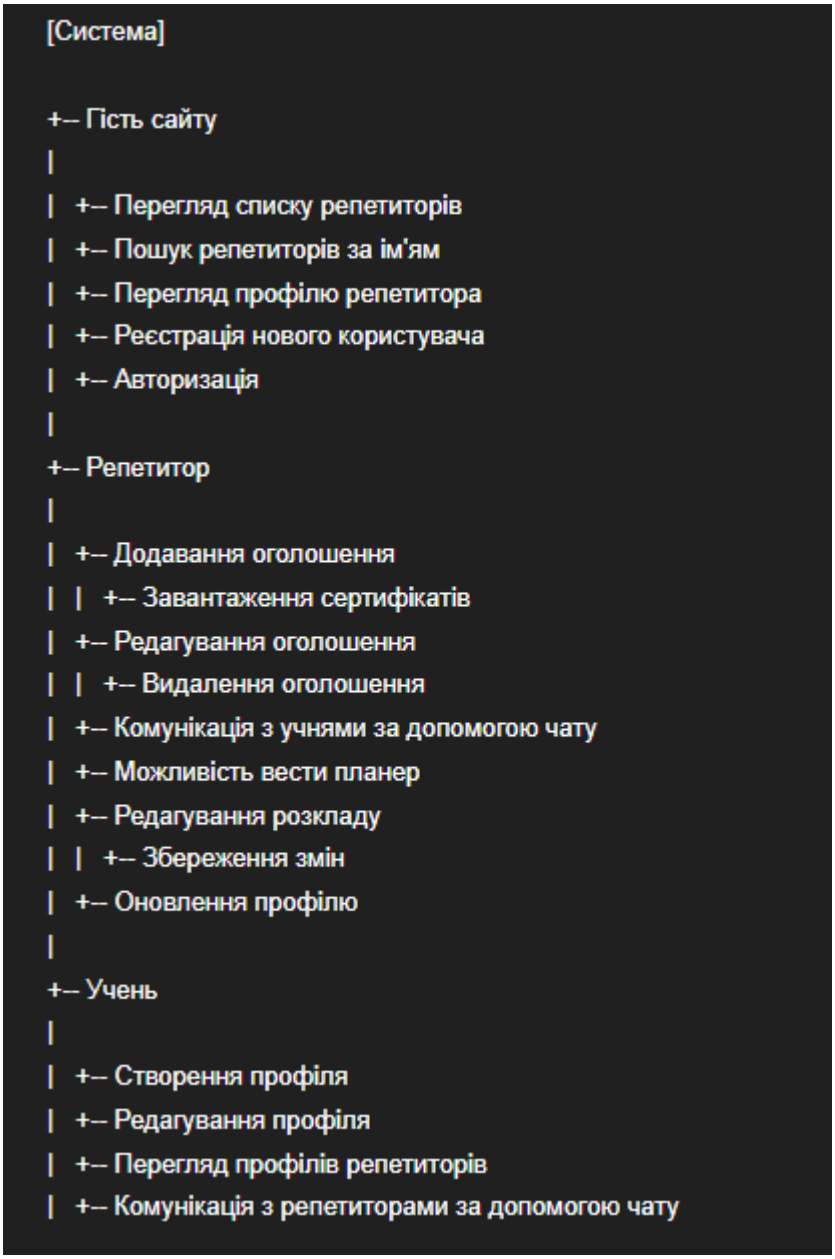

Рисунок 3.1 – Схема функціональності та поведінки

Фунціональні можливості для відвідувачів вебсайту:

– реєстрація;
– авторизація;
– пошук репетиторів;
– пошук репетиторі за ім’ям;
– огляд профілів репетиторів.

Фунціональні можливості для репетиторів:

– реєстрація;
– авторизація;
– створення профіля;
– створення оголошення;
– редагування профіля;
– редагування оголошення;
– перегляд профілів інших репетиторів;
– комунікація з учнями за допомогою чату;
– можливість вести планер.

Фунціональні можливості для учні:

– реєстрація;
– авторизація;
– створення профіля;
– редагування профіля;
– перегляд профілів репетиторів;
– комунікація з репетиторами за допомогою чату.

3.2 Проєктування архітектури програмного забезпечення

Клієнтська частина:

– користувачі взаємодіють з вебдодатком через браузер;
– вебдодаток на react забезпечує динамічний інтерфейс користувача;

– запити з вебдодатку надсилаються до серверної частини через http-протокол.

Серверна частина:

– сервер на node.js отримує запити від клієнтської частини;

– серверна частина взаємодіє з базою даних через бібліотеку Mongoose.

База даних:

– MongoDB Atlas використовується для зберігання даних про користувачів, об’яви та інші важливі дані;

– всі запити до бази даних здійснюються виключно через серверну частину для забезпечення безпеки.

Взаємодія компонентів:

– взаємодія між клієнтською частиною та сервером відбувається через HTTP-протокол;

– взаємодія між сервером та базою даних здійснюється через бібліотеку Mongoose з використанням TCP/IP-протоколу.

На рисунку 3.2 ви можете побачити діаграму розвертання проєкту.

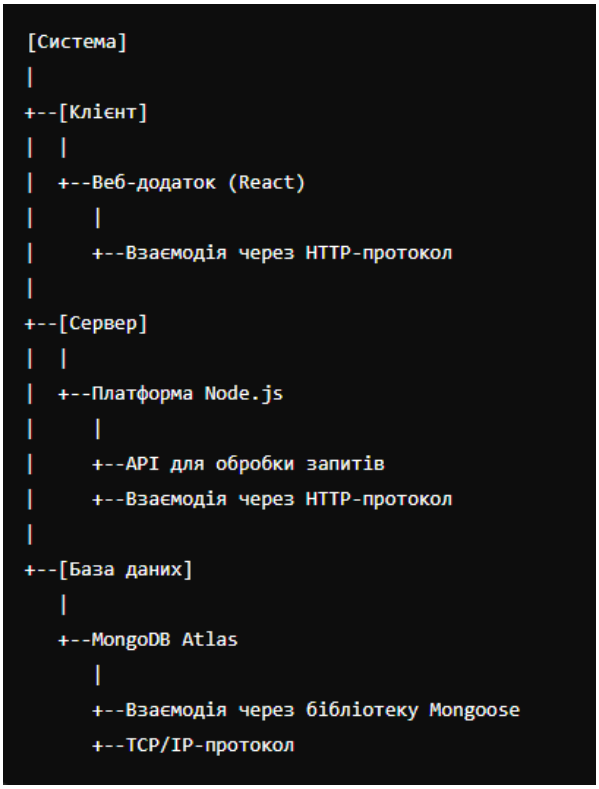

Рисунок 3.2 – Діаграма розгортання

Ця схема показує взаємозв’язок між компонентами програмної системи та їх взаємодію через визначені протоколи та технології.

3.3 Проєктування структури зберігання даних

У системі використовується база даних MongoDB, яка є документо-орієнтованою базою даних. Вона зберігає дані, що мають структуру, схожу на JSON-файли. Такий тип баз даних дозволяє легко масштабувати базу даних та підтримувати цілісність даних без порушення роботи системи при зміні структури.

MongoDB Atlas забезпечує постійний доступ до бази даних і захищає від несанкціонованого доступу. Для цього потрібно точно визначити потенційних користувачів системи та дозволені IP-адреси для входу. Середовище MongoDB Atlas також пропонує автоматичне резервне копіювання та відновлення даних, що підвищує надійність і безпеку зберігання інформації.

Крім того, MongoDB підтримує реплікацію даних, що забезпечує високу доступність та відмовостійкість системи. Це дозволяє системі залишатися доступною навіть у випадку збоїв окремих серверів.

На рисунку 3.3 ви можете побачити структуру бази даних.

Опис діаграми:

– користувачі зберігають основні дані про користувачів платформи, включаючи їх особисті дані та роль;
– оголошення містять інформацію про уроки або послуги, які пропонують репетитори;
– повідомлення зберігають дані про спілкування між користувачами у чатах;
– чати включають списки користувачів і повідомлень, пов’язаних з конкретним чатом;
– планувальники містять розклад занять або подій для користувачів, організованих за днями тижня та годинами.

```
[Users]
|-- _id: ObjectId
|-- name: String
|-- email: String
|-- password: String
|-- role: String
|-- phoneNumber: String
|-- photo: String

[Adverts]
|-- _id: ObjectId
|-- user: ObjectId (ref: Users)
|-- description: String
|-- subjects: [String]
|-- youInTwoWords: String
|-- price: Number
|-- certificates: [String]

[Messages]
|-- _id: ObjectId
|-- chat: ObjectId (ref: Chats)
|-- sender: ObjectId (ref: Users)
|-- content: String
|-- timestamp: Date

[Chats]
|-- _id: ObjectId
|-- users: [ObjectId] (ref: Users)
|-- messages: [ObjectId] (ref: Messages)

[Planners]
|-- _id: ObjectId
|-- user: ObjectId (ref: Users)
|-- planner: [[String]] (Масив, що містить розклад на тиждень)
```

Рисунок 3.3 – Структура бази даних

3.4 Створення UI / UX дизайну системи

При розробці інтерфейсу слід враховувати досвід користувачів та дотримуватися основних принципів проєктування інтерфейсів.

Основні вимоги до проєктування включають:

– чітка структура всіх елементів інтерфейсу;
– логічне групування елементів інтерфейсу;
– вирівнювання всіх компонентів інтерфейсу;
– уніфікований стиль оформлення всіх сторінок і елементів інтерфейсу.

Ви можете побачити головну сторінку сайту на рисунку 3.4.

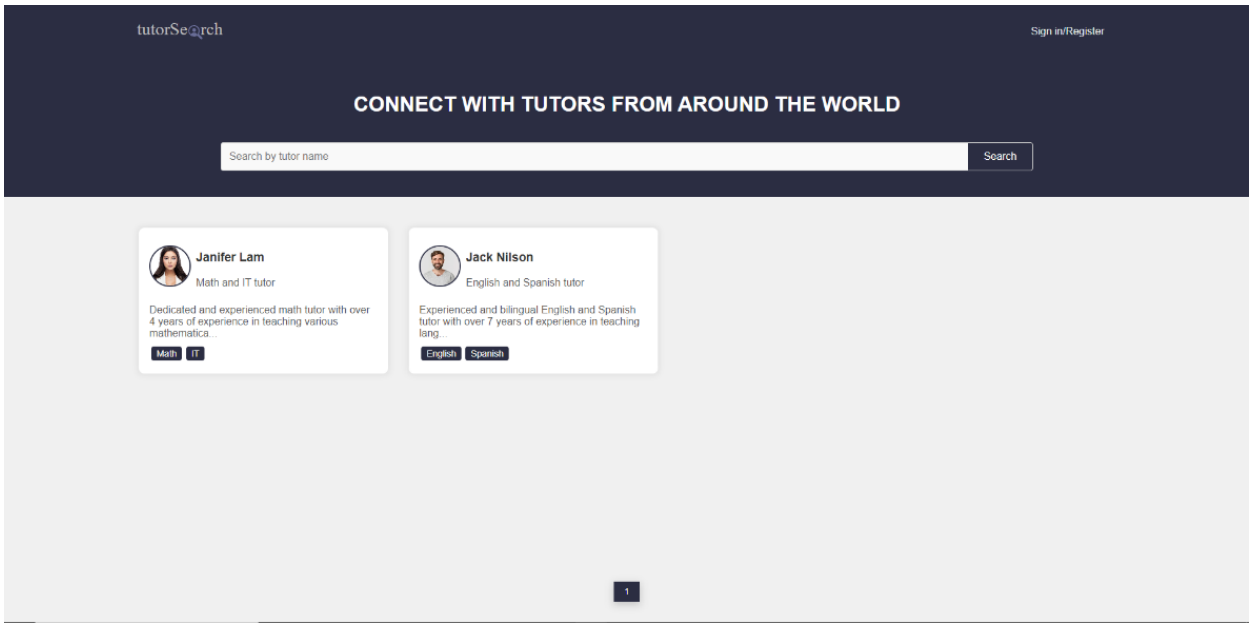

Рисунок 3.4 – Головна сторінка сайту

Головна сторінка сайту розроблена з метою полегшення пошуку репетиторів для користувачів. Основний фокус сторінки полягає в забезпеченні простого та інтуїтивно зрозумілого інтерфейсу для взаємодії з сервісом.

Верхня панель:

– логотип: розміщений у верхньому лівому куті сайту, що допомагає користувачам швидко впізнавати бренд;
– заклик до дії: великий заголовок “CONNECT WITH TUTORS FROM AROUND THE WORLD” привертає увагу користувачів і наголошує на основній меті сайту;
– панель авторизації: у верхньому правому куті знаходиться кнопка

“Sign in/Register”, яка дозволяє користувачам зареєструватися або увійти в систему.

Пошукова панель:

– поле пошуку: широке поле для введення імені репетитора, розташоване під заголовком, дозволяє користувачам швидко знайти потрібного репетитора;
– кнопка пошуку: кнопка “Search” праворуч від поля введення дозволяє виконати пошук за введеними даними.

Список репетиторів:

– фото репетитора: допомагає користувачам візуально ідентифікувати репетиторів;
– ім’я та спеціалізація: відображають ім’я репетитора та предмети, які він викладає.

Короткий опис: опис досвіду репетитора.

Теги предметів: перелік предметів, які викладає репетитор, представлені у вигляді тегів.

Пагінація: внизу сторінки розміщена панель пагінації, що дозволяє користувачам перемикатися між сторінками з репетиторами.

На рисунку 3.5 наведено скриншот сторінки «Register».

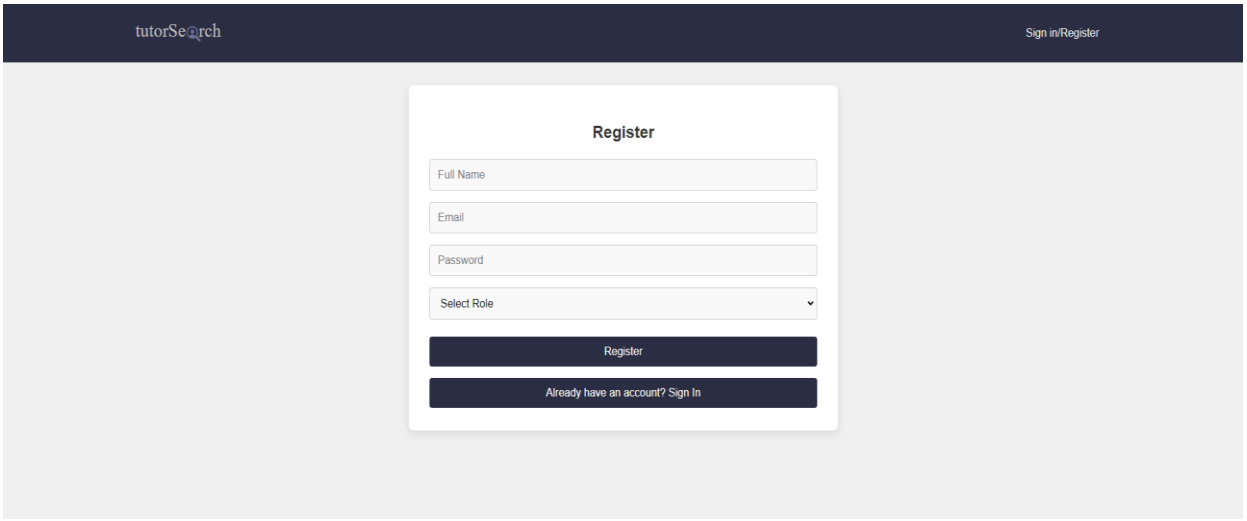

Рисунок 3.5 – Сторінка «Register»

Сторінка реєстрації сайту створена з метою забезпечення користувачам легкого та швидкого процесу реєстрації на платформі. Інтерфейс має бути інтуїтивно зрозумілим та забезпечувати необхідний функціонал для введення даних.

Заголовок:

– великий заголовок “Register” чітко вказує на мету сторінки.

Поля введення:

– Full Name: поле для введення повного імені користувача;
– Email: поле для введення електронної пошти;
– Password: поле для введення паролю;
– Select Role: випадаючий список для вибору ролі користувача (Student або Tutor).

Кнопки:

– Register**:** кнопка для підтвердження реєстрації;
– Switch to Sign In: кнопка для переходу до форми входу, що забезпечує зручність для користувачів, які вже мають обліковий запис.

На рисунку 3.6 наведено скриншот сторінки “Profile” для ролі “tutor”.

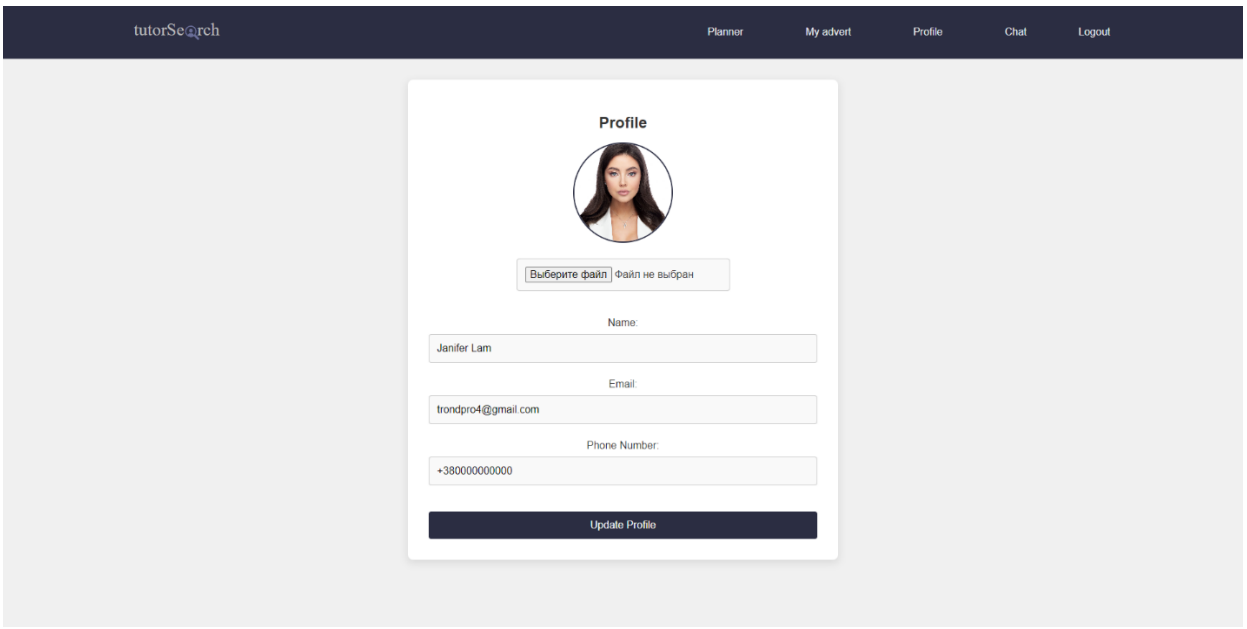

Рисунок 3.6 – Сторінка “Profile” для ролі “tutor”

Сторінка профілю репетитора (Profile) створена для забезпечення можливості редагування особистої інформації та завантаження аватарки користувача. Інтерфейс має бути простим у використанні та інтуїтивно зрозумілим.

Верхня панель містить кнопки для переходу до різних розділів сайту:

– Planner: перехід до сторінки планувальника;
– My advert: перехід до сторінки з оголошенням користувача;
– Profile: поточна сторінка профілю;
– Chat: перехід до чату;
– Logout: кнопка для виходу з облікового запису.

Форма редагування профілю:

– користувач може завантажити або змінити свій аватар, використовуючи кнопку «Виберіть файл»;
– поточна фотографія відображається у верхній частині форми.

Поля введення:

– Name: поле для введення імені користувача;
– Email: поле для введення електронної пошти;
– Phone Number: поле для введення номера телефону;

Кнопка оновлення профілю:

– Update Profile: кнопка для збереження змін.

На рисунку 3.7 наведено приклад програмного рішення для головної сторінки сайту.

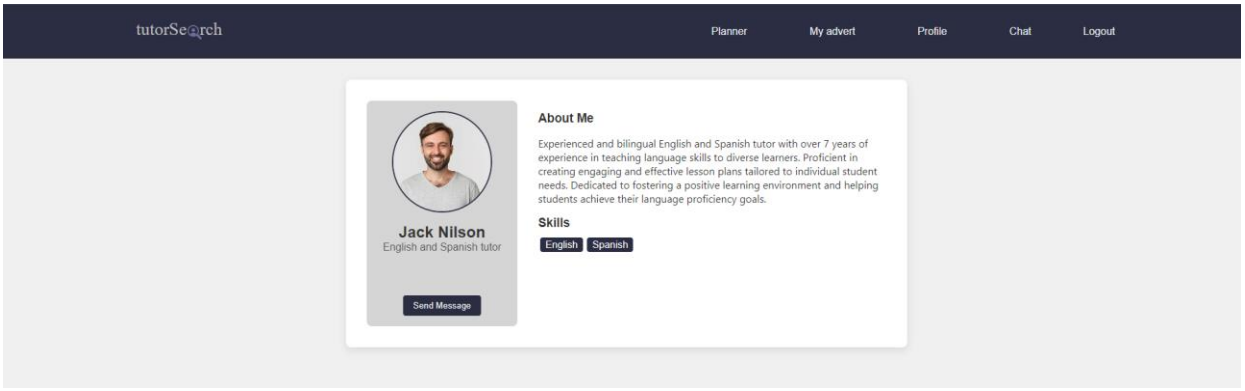

Рисунок 3.7 – Сторінка оголошення репетитора

Сторінка оголошення репетитора створена для детального представлення інформації про конкретного репетитора. Ця сторінка містить інформацію про ім’я, спеціалізацію, досвід роботи та навички репетитора.

Інформація про репетитора:

– фотографія: ліва частина сторінки містить фотографію репетитора;
– ім’я та спеціалізація: під фотографією вказано ім’я та спеціалізацію репетитора;
– кнопка “Send Message”: кнопка для надсилання повідомлення репетитору, що дозволяє користувачам легко зв’язатися з ним.

Про мене (About Me):

– опис досвіду: права частина сторінки містить розділ “About Me”, де детально описано досвід роботи репетитора, його навички та підхід до навчання.

Навички (Skills):

– мітки навичок: під розділом “About Me” наведено мітки з основними навичками репетитора, що дозволяє користувачам швидко ознайомитися з його компетенціями.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Реалізація серверної частини програмного забезпечення

Серверна складова даного проєкту відповідає за обробку запитів від клієнтської частини, забезпечення безпеки та взаємодію з базою даних.

Розглянемо основні компоненти серверної частини, представлені на на рисунку 4.1.

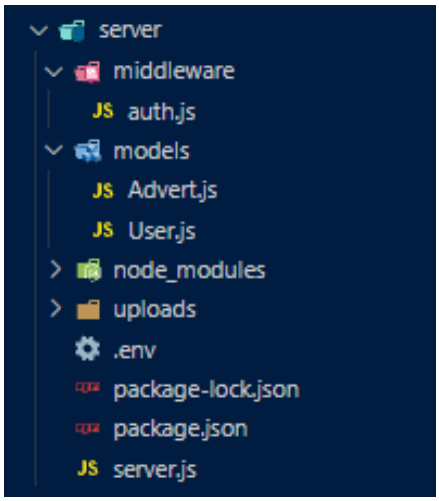

Рисунок 4.1 – Структура проєкту серверної частини

Серверний файл (server.js):

– головний файл серверної частини, що відповідає за ініціалізацію серверу, підключення до бази даних MongoDB, налаштування середовища та маршрутизацію запитів;

– в цьому файлі налаштовуються різні середовища (наприклад, cors для налаштування крос-доменних запитів), а також підключаються маршрути для обробки запитів.

Middleware:

– папка містить проміжне програмне забезпечення, яке забезпечує додаткові функції для обробки запитів: у даному випадку, у файлі

auth.js реалізовані функції аутентифікації та авторизації користувачів за допомогою JWT (JSON Web Token).

Models:

– папка містить моделі даних, які використовуються для взаємодії з базою даних MongoDB;

– **Advert.js**: модель оголошень, що описує структуру оголошення (опис, предмети, ціна, тощо);

– **User.js**: модель користувачів, що описує структуру користувача (ім'я, електронна пошта, пароль, роль, тощо).

uploads:

– папка для зберігання завантажених користувачами файлів, таких як зображення профілів та сертифікати.

Файл .env:

– файл конфігурації, який містить змінні середовища, такі як налаштування підключення до бази даних та секретні ключі для JWT.

Файли конфігурації (package.json та package-lock.json):

– ці файли містять інформацію про залежності проєкту та скрипти для запуску серверу.

Розглянемо код файлу auth.js, представлений на рисунку 4.2.

```
1  const jwt = require("jsonwebtoken");
2
3  module.exports = function (req, res, next) {
4    const token = req.header("x-auth-token");
5    if (!token) {
6      return res.status(401).send("No token, authorization denied");
7    }
8
9    try {
10     const decoded = jwt.verify(token, "secretkey");
11     req.user = decoded;
12     next();
13   } catch (err) {
14     res.status(401).send("Token is not valid");
15   }
16 };
17
```

Рисунок 4.2 – Код файлу auth.js

У цьому фрагменті коду реалізовано проміжне програмне забезпечення (middleware) для аутентифікації користувачів за допомогою JSON Web Token (JWT). Ось покрокове пояснення того, що відбувається в цьому коді:

- спочатку ми імпортуємо бібліотеку jsonwebtoken, яка дозволяє працювати з JWT;
- експортується функція, яка приймає три аргументи: req (запит), res (відповідь) та next (функція, яка передає керування наступному мідлвару);
- отримується токен із заголовку x-auth-token у запиті. Якщо токен відсутній, повертається статус 401 (не авторизовано) з повідомленням “No token, authorization denied’;
- у блоці try здійснюється перевірка та декодування токена за допомогою методу jwt.verify(token, “secretkey”). Якщо токен валідний, він декодується і зберігається в req.user;
- викликається функція next(), яка передає керування наступному мідлвару або кінцевій точці маршруту;
- якщо токен недійсний або не може бути перевірений, виконується блок catch, який повертає статус 401 з повідомленням “Token is not valid”.

Таким чином, цей код забезпечує перевірку автентичності користувача за допомогою JWT. Якщо токен відсутній або недійсний, доступ до захищених ресурсів не надається, і користувач отримує повідомлення про відсутність авторизації.

Розглянемо код файлу Advert.js, представлені на на рисунку 4.3.

У цьому фрагменті коду реалізовано модель оголошення (Advert) для бази даних MongoDB за допомогою бібліотеки Mongoose. Таким чином, цей код визначає структуру даних для оголошень у системі, включаючи необхідні поля та їх типи, а також зв’язок між оголошеннями та користувачами. Модель Advert дозволяє створювати, читати, оновлювати та видаляти документи оголошень у колекції MongoDB.

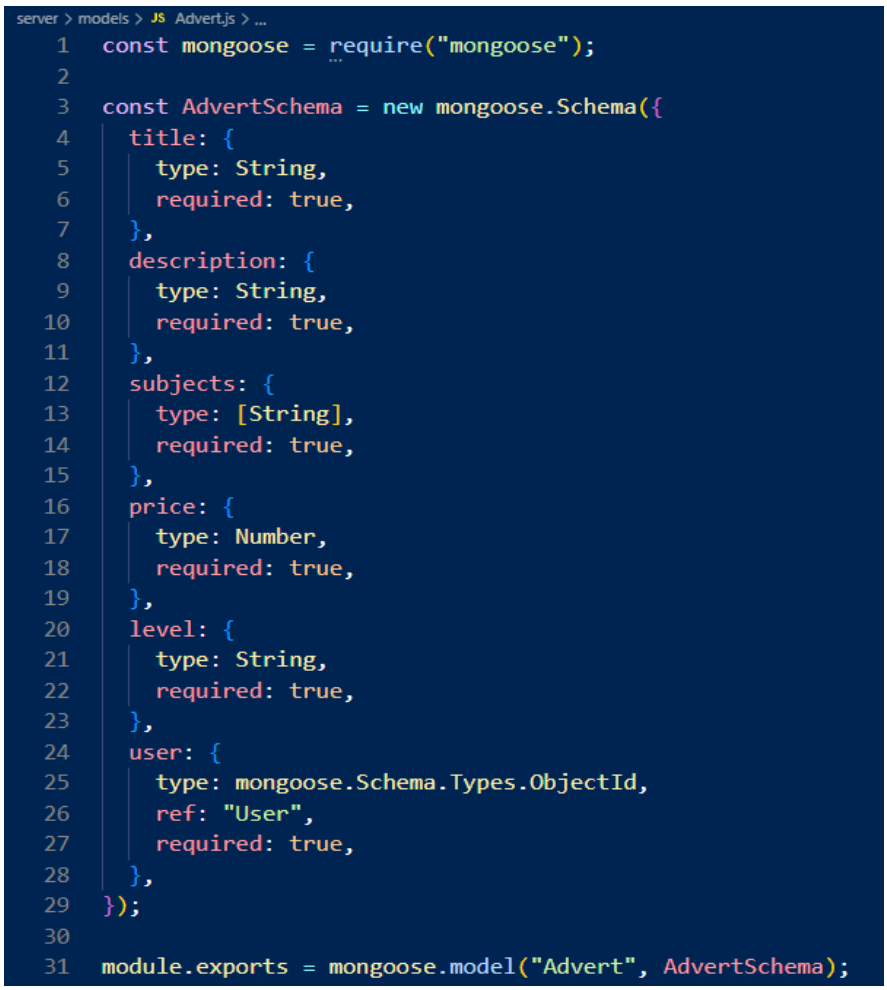

Рисунок 4.3 – Код файлу Advert.js

Розглянемо код файлу User.js, представлені на на рисунку 4.4.

У цьому фрагменті коду реалізовано модель користувача (User) для бази даних MongoDB за допомогою бібліотеки Mongoose. Ось покрокове пояснення того, що відбувається в цьому коді.

Спочатку ми імпортуємо бібліотеку mongoose, яка дозволяє нам працювати з базою даних MongoDB у Node.js.

Далі створюється нова схема для користувача (UserSchema) з такими полями:

– **fullName**: повне ім’я користувача, типу String, обов’язкове (required: true);
– **email**: адреса електронної пошти користувача, типу String, обов’язкова (required: true) та унікальна (unique: true), що означає, що два користувачі не можуть мати однакову електронну пошту;

– **password**: пароль користувача, типу String, обов’язковий (required: true);
– **role**: роль користувача, типу String, обов’язкова (required: true) та обмежена значеннями “Student” або “Tutor” (enum: [“Student”, “Tutor”]);
– **phoneNumber**: номер телефону користувача, типу String, не обов’язкове поле.
– **photo**: посилання на фотографію користувача, типу String, не обов’язкове поле.
– експортується модель User, створена на основі схеми UserSchema (це дозволяє використовувати цю модель в інших частинах додатка для взаємодії з колекцією користувачів у базі даних).

server > models > JS User.js > ...

```
1  const mongoose = require("mongoose");
2
3  const UserSchema = new mongoose.Schema({
4    fullName: {
5      type: String,
6      required: true,
7    },
8    email: {
9      type: String,
10     required: true,
11     unique: true,
12   },
13   password: {
14     type: String,
15     required: true,
16   },
17   role: {
18     type: String,
19     required: true,
20     enum: ["Student", "Tutor"],
21   },
22   phoneNumber: {
23     type: String,
24   },
25   photo: {
26     type: String,
27   },
28 });
29
30 module.exports = mongoose.model("User", UserSchema);
```

Рисунок 4.4 – Код файлу User.js

Таким чином, цей код визначає структуру даних для користувачів у системі, включаючи необхідні поля та їх типи, а також обмеження, такі як унікальність електронної пошти та дозволені значення для ролі. Модель User дозволяє створювати, читати, оновлювати та видаляти документи користувачів у колекції MongoDB.

Розглянемо фрагменти коду файлу server.js для імпорту модулів, наведені на рисунку 4.5. На рисунку 4.6 імпортуються бібліотеки для роботи з Express, MongoDB, JWT, шифрування паролів, обробки файлів та інші.

Рисунок 4.5 – Імпорт необхідних модулів

Рисунок 4.6 – Налаштування додатка та середовища

Конфігурація Express, CORS, обробка JSON та налаштування статичного шляху, підключення до MongoDB та обробку помилок з’єднання можна побачити на рисунку 4.7.

```
47  // Auth middleware
48  const auth = (req, res, next) => {
49    const token = req.header("x-auth-token");
50    if (!token)
51      return res.status(401).json({ msg: "No token, authorization denied" });
52
53    try {
54      const decoded = jwt.verify(token, process.env.JWT_SECRET);
55      req.user = decoded.user;
56      next();
57    } catch (err) {
58      res.status(401).json({ msg: "Token is not valid" });
59    }
60  };
```

Рисунок 4.7 – Middleware для аутентифікації

Цей middleware перевіряє наявність та дійсність JWT токена. Файл server.js відповідає за серверну частину вебдодатку. Основні його функції:

– налаштування середовища та підключення до MongoDB: підключення до бази даних MongoDB за допомогою бібліотеки mongoose, обробка помилок з’єднання;
– моделі даних: визначення схем для користувачів, оголошень, чатів, повідомлень та планувальників;
– маршрутизація: обробка HTTP-запитів для реєстрації, логіну користувачів, створення, оновлення та видалення оголошень, обробка чатів та повідомлень;
– аутентифікація: використання JWT токенів для забезпечення безпеки користувачів;
– обробка файлів: налаштування завантаження та зберігання файлів на сервері.

Таким чином, файл server.js обробляє запити від клієнтів, виконує операції з базою даних та забезпечує безпеку даних.

4.2 Реалізація вебчастини програмного забезпечення

Вебчастина нашого проєкту реалізована за допомогою сучасного фреймворку React, що забезпечує високу продуктивність, масштабованість та зручність у використанні. Реалізацію файлу components можна побачити на рисунку 4.8.

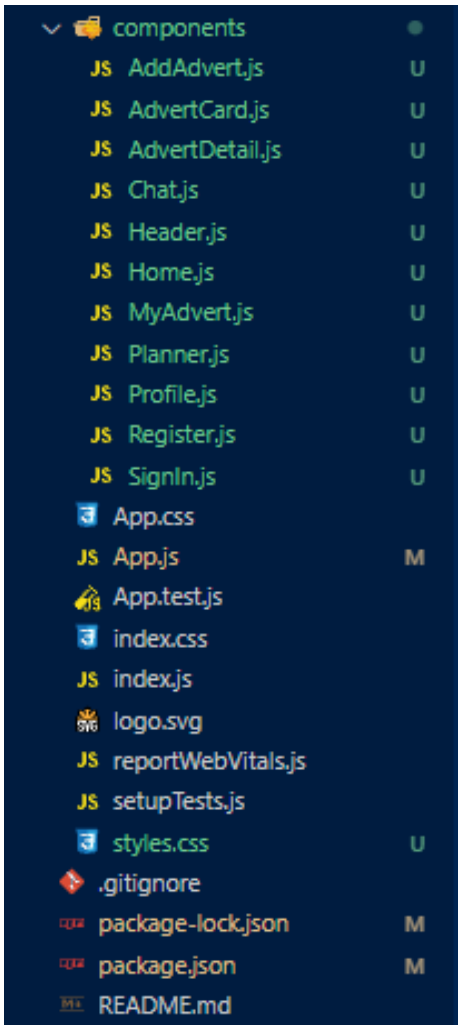

Рисунок 4.8 – Нутро файлу components

Ця частина включає в собі основні компоненти, які відповідають за різні функціональні можливості сайту, такі як управління оголошеннями, профілями користувачів, комунікація через чат та планування розкладу для репетиторів. На рисунку 4.9 представлено приклад оголошення репетитора.

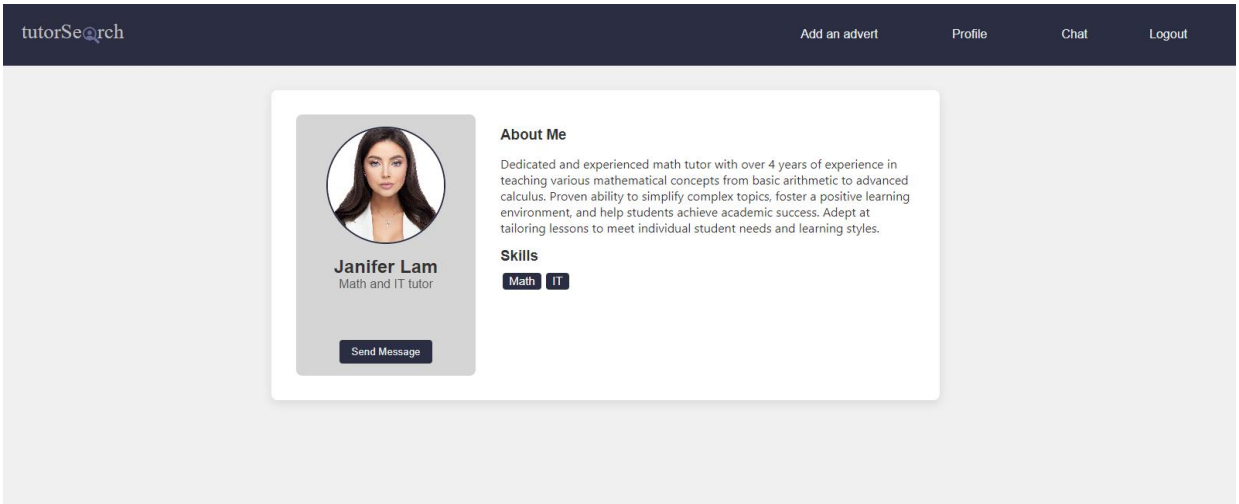

Рисунок 4.9 – Оголошення репетитора

Цей компонент, AdvertDetail, відповідає за відображення детальної інформації про конкретне оголошення репетитора на сайті. Реалізація імпортів та ініціалізації представлена на рисунку 4.10.

```
1  import React, { useState, useEffect } from "react";
2  import { useParams, useNavigate } from "react-router-dom";
3  import axios from "axios";
4  import "../styles.css";
5
```

Рисунок 4.10 – Імпорти та ініціалізація

Давайте детальніше розглянемо кожну частину цього коду:

– імпорт основних бібліотек і стилів;
– використовуються React хуки useState та useEffect;
– використання useParams для отримання параметрів URL (в даному випадку ID оголошення);
– useNavigate використовується для навігації по сайту;
– Axios для HTTP-запитів.

Реалізація завантаження даних оголошення представлена на рисунку 4.11.

```
const AdvertDetail = () => {
  const { id } = useParams();
  const [advert, setAdvert] = useState({});
  const [certificates, setCertificates] = useState([]);
  const navigate = useNavigate();

  useEffect(() => {
    const fetchAdvert = async () => {
      try {
        const res = await axios.get(`http://localhost:5000/adverts/${id}`);
        setAdvert(res.data);
        setCertificates(res.data.certificates || []);
      } catch (err) {
        console.error("Error fetching advert: ", err);
      }
    };

    fetchAdvert();
  }, [id]);
```

Рисунок 4.11 – Завантаження даних оголошення

В цьому коді відбувається наступне:

– використання useEffect для завантаження даних оголошення при завантаженні компонента;
– функція fetchAdvert виконує HTTP-запит до сервера для отримання даних оголошення за його ID;
– дані оголошення зберігаються у стан advert, сертифікати – у стан certificates.

Реалізація відправки повідомлення представлена на рисунку 4.12.

```
  const handleSendMessage = async () => {
    const token = localStorage.getItem("token");

    if (!token) {
      navigate("/signin");
      return;
    }

    try {
      const res = await axios.post(
        "http://localhost:5000/chats",
        { userId: advert.user._id },
        { headers: { "x-auth-token": token } }
      );

      navigate(`/chat/${res.data._id}`);
    } catch (err) {
      console.error("Error creating or finding chat: ", err);
    }
  };
```

Рисунок 4.12 – Відправка повідомлення

В цьому коді відбувається наступне:

– функція handleSendMessage відправляє запит на створення або пошук чату з репетитором;
– якщо користувач не авторизований, перенаправляє на сторінку входу;
– якщо авторизований, відправляє запит на сервер для створення чату та переходить до чату.

Реалізація відображення компонента представлена на рисунку 4.13.

```
47   return (
48     <div className="advert-detail-container">
49       <div className="advert-detail-left">
50         <img
51           className="advert-detail-photo"
52           src={`http://localhost:5000${advert.user?.photo}`}
53           alt={advert.user?.name}
54         />
55         <div className="advert-detail-user-info">
56           <h2>{advert.user?.name}</h2>
57           <p>{advert.youInTwoWords}</p>
58         </div>
59         <button className="send-message-button" onClick={handleSendMessage}>
60           Send Message
61         </button>
62       </div>
63       <div className="advert-detail-right">
64         <div className="advert-detail-content">
65 ›         <h3>About Me</h3>···
90         </div>
91       </div>
92     </div>
93   );
94 };
95
96 export default AdvertDetail;
97
```

Рисунок 4.13 – Відображення компонента

В цьому коді відбувається наступне:

– компонент повертає структуру для відображення інформації про оголошення;
– ліва частина (advert-detail-left) містить фотографію та коротку інформацію про репетитора;
– права частина (advert-detail-right) містить детальну інформацію про репетитора, включаючи опис, навички та сертифікати;
– кнопка для відправки повідомлення репетитору.

Цей компонент забезпечує зручний інтерфейс для перегляду детальної інформації про репетитора та дозволяє швидко почати з ним спілкування. На рисунку 4.14 представлено приклад сторінки оголошення репетитора.

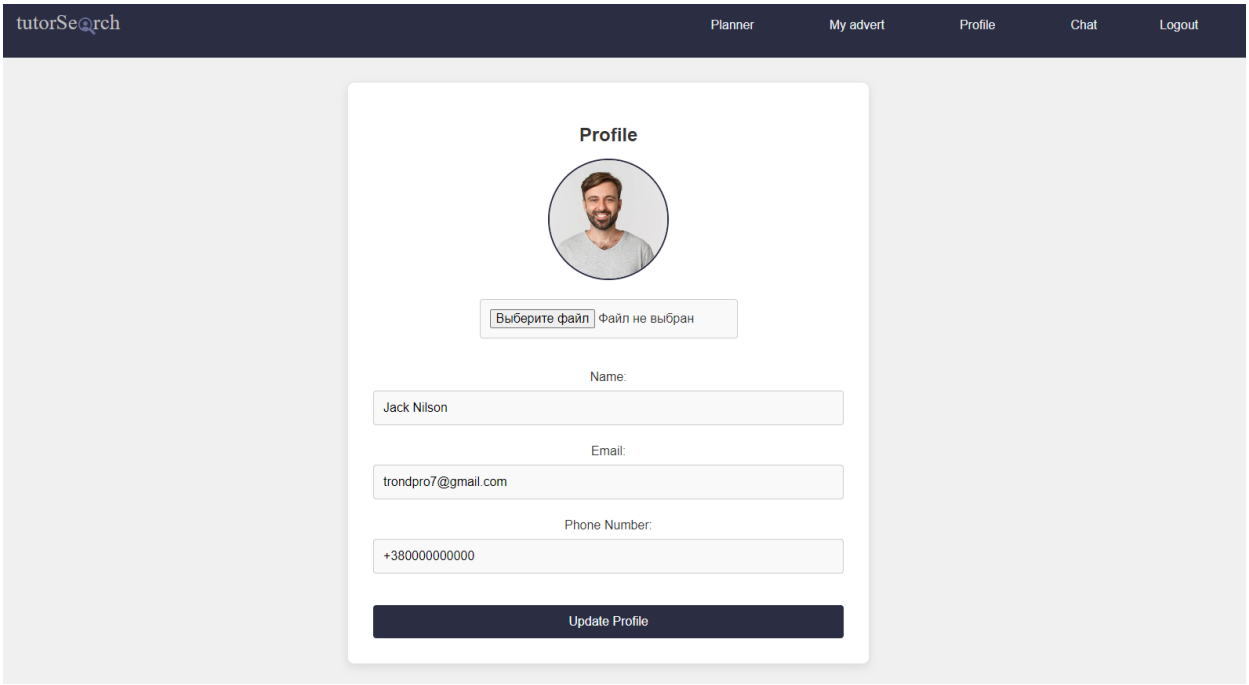

Рисунок 4.14 – Сторінка оголошення репетитора

Цей код представляє компонент Profile в React, який відповідає за відображення та оновлення профілю користувача на сайті. Давайте детальніше розглянемо кожну частину цього коду. Код представлено на рисунку 4.15.

```
client > src > components > JS Profile.js > [e] Profile > useEffect() callback > [e] fetchProfile
1  import React, { useState, useEffect } from "react";
2  import axios from "axios";
3  import { useNavigate } from "react-router-dom";
4  import "../styles.css";
5
6  const Profile = () => {
7    const [user, setUser] = useState({
8      name: "",
9      email: "",
10     phoneNumber: "",
11     photo: "",
12   });
13   const [loading, setLoading] = useState(true);
14   const [photo, setPhoto] = useState(null);
15   const [message, setMessage] = useState("");
16   const navigate = useNavigate();
```

Рисунок 4.15 – Імпорти та ініціалізація стану

В цьому коді відбувається наступне:

– імпортуються необхідні бібліотеки та стилі;
– ініціалізується стан компоненту, включаючи інформацію про користувача, стан завантаження, вибране фото та повідомлення.

Код завантаження даних профілю представлено на рисунку 4.16.

```
18  useEffect(() => {
19    const fetchProfile = async () => {
20      const token = localStorage.getItem("token");
21      if (!token) {
22        navigate("/signin");
23        return;
24      }
25
26      try {
27        const res = await axios.get("http://localhost:5000/profile", {
28          headers: { "x-auth-token": token },
29        });
30        setUser(res.data);
31        setLoading(false);
32      } catch (err) {
33        console.error("Error fetching user: ", err);
34        setLoading(false);
35      }
36    };
37
38    fetchProfile();
39  }, [navigate]);
```

Рисунок 4.16 – Завантаження даних профілю

В цьому коді відбувається наступне:

– використовується хук useEffect для завантаження даних профілю при першому завантаженні компонента;
– якщо токен не знайдено в локальному сховищі, користувача перенаправляють на сторінку входу;
– якщо токен знайдено, виконується запит на сервер для отримання даних профілю користувача.

На рисунку 4.17 наведено реалізацію обробки змін у формах.

Функції handleChange і handleFileChange обробляють зміни в текстових полях і вибір файлу відповідно.

```
41  const handleChange = (e) => {
42    setUser({ ...user, [e.target.name]: e.target.value });
43  };
44
45  const handleFileChange = (e) => {
46    setPhoto(e.target.files[0]);
47  };
```

Рисунок 4.17 – Обробка змін у формах

На рисунку 4.18 наведено реалізацію функціоналу відправки даних профілю.

```
49  const handleSubmit = async (e) => {
50    e.preventDefault();
51    const token = localStorage.getItem("token");
52    const formData = new FormData();
53    formData.append("name", user.name);
54    formData.append("email", user.email);
55    formData.append("phoneNumber", user.phoneNumber);
56    if (photo) {
57      formData.append("photo", photo);
58    }
59
60    try {
61      const res = await axios.put("http://localhost:5000/profile", formData, {
62        headers: {
63          "x-auth-token": token,
64          "Content-Type": "multipart/form-data",
65        },
66      });
67      setMessage("Profile updated successfully");
68      setUser(res.data);
69    } catch (err) {
70      console.error("Error updating profile: ", err);
71      setMessage("Error updating profile");
72    }
73  };
```

Рисунок 4.18 – Відправка даних профілю

В цьому коді відбувається наступне:

– функція handleSubmit відправляє оновлені дані профілю на сервер;
– дані відправляються у форматі FormData, що дозволяє передавати файли;
– після успішного оновлення даних, відповідь з сервера використовується для оновлення стану компонента та відображення повідомлення про успіх.

Реалізацію відображення компоненту наведено на рисунку 4.19.

client > src > components > JS Profile.js > Profile

```
75    if (loading) {
76      return <div className="loading">Loading...</div>;
77    }
78
79    return (
80      <div className="profile-container">
81        <h2>Profile</h2>
82        {message && <p>{message}</p>}
83        <form onSubmit={handleSubmit} className="profile-form">
84          <div className="profile-photo">
85            <img
86              src={
87                user.photo
88                  ? `http://localhost:5000${user.photo}`
89                  : "/img/default-avatar.png"
90              }
91              alt="Profile"
92            />
93            <input type="file" onChange={handleFileChange} />
94          </div>
95          <div className="profile-details">
96            <div>
97              <label>Name:</label>
98              <input
99                type="text"
100               name="name"
101               value={user.name}
102               onChange={handleChange}
103               required
104             />
105           </div>
106           <div>
107             <label>Email:</label>
108             <input
109               type="email"
110               name="email"
111               value={user.email}
112               onChange={handleChange}
113               required
114             />
115           </div>
116           <div>
117             <label>Phone Number:</label>
118             <input
119               type="text"
120               name="phoneNumber"
121               value={user.phoneNumber}
122               onChange={handleChange}
123               required
124             />
125           </div>
126         </div>
127         <button type="submit" className="update-button">
128           Update Profile
129         </button>
130       </form>
131     </div>
132   );
133 };
```

Рисунок 4.19 – Відображення компонента

В даному коді описано наступне:

– якщо дані ще завантажуються, відображається повідомлення “Loading...”;
– після завантаження даних відображається форма з полями для редагування і можливістю завантажити нову фотографію профілю;
– форма включає поля для імені, електронної пошти, номера телефону та кнопки для збереження змін.

Цей компонент забезпечує зручний інтерфейс для редагування профілю користувача на сайті.

Реалізація вебчастини програмного забезпечення даного проєкту виконана на високому технологічному рівні, що забезпечує ефективність, зручність та безпеку використання. Загалом, вебчастина програмного забезпечення проєкту відповідає сучасним стандартам розробки, забезпечуючи високий рівень функціональності, безпеки та зручності використання. Це дозволяє користувачам ефективно взаємодіяти з системою, задовольняючи їхні потреби у пошуку та наданні послуг репетиторства.

ВИСНОВКИ

Кількість людей, які користуються онлайн-сервісами для пошуку та замовлення репетиторів, постійно зростає, що робить впровадження таких платформ все більш актуальним [2]. Важливими аспектами, які впливають на успішність подібних платформ, є зручність використання, безпека та надійність системи, а також можливість швидкого та ефективного взаємодії між репетиторами та студентами [3]. Для досягнення цих цілей необхідно було провести всебічний аналіз предметної галузі, визначити основні проблеми та запропонувати можливі шляхи їх вирішення.

Під час виконання кваліфікаційної роботи було проведено детальний аналіз потреб користувачів, що дозволило визначити ключові функції та вимоги до системи. Аналіз конкурентів надав можливість виділити їхні сильні та слабкі сторони, що допомогло сформувати більш ефективну та конкурентоспроможну платформу [15].

Було визначено основні функції системи, які відповідають поставленим завданням та опису проєкту [16]. Під час розробки програмного забезпечення були сформульовані загальні принципи роботи системи та детально описані вимоги до неї. У процесі створення програми були визначені основні компоненти, описані їх функціональні можливості та поведінка, а також спроєктована структура зберігання даних і користувацький інтерфейс [17].

В результаті виконання проєкту було розроблено комплексний вебдодаток, який включає в себе клієнтську та серверну частини, а також базу даних [18]. Впровадження таких сучасних технологій, як React, Node.js та MongoDB, забезпечило високу продуктивність, гнучкість та масштабованість системи. Інтеграція технологій безпеки, таких як JWT для аутентифікації та шифрування паролів, гарантує надійний захист даних користувачів [19].

Таким чином, розроблена платформа відповідає всім основним вимогам до сучасних вебдодатків для репетиторів, забезпечуючи зручність, безпеку та

ефективність взаємодії між користувачами. Проєкт продемонстрував успішну інтеграцію передових технологій та підходів, що сприяє розвитку освіти через інноваційні та доступні платформи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Gonţa I., Tripon C. Students' perspectives on online learning – are their expectations met by current teaching practices? *Journal of Pedagogy*. 2021. Vol. 1. P. 73–91. URL: https://doi.org/10.26755/RevPed/2021.1/73 (дата звернення: 27.01.2024).
2. Howard M. Global E-Learning Market Research: Size, Share, Growth, Trends, Insights, and Opportunities. URL: https://www.linkedin.com/pulse/global-e-learning-market-research-size-share-growth-trends-howard (дата звернення: 28.01.2024).
3. Gautam P. Advantages And Disadvantages Of Online Learning. URL: https://elearningindustry.com/advantages-and-disadvantages-online-learning (дата звернення: 27.01.2024).
4. Stubber T. H. Uncovering the Many Benefits of Online Tutoring. URL: https://tutorcruncher.com/tutoring-online/what-are-the-benefits-of-online-tutoring/ (дата звернення: 29.01.2024).
5. Huivan N. Web Development Best Practices in 2024. URL: https://atlasiko.com/blog/web-development/web-development-best-practices/ (дата звернення: 28.01.2024).
6. Samsuri N. N. A Study on the Student's Perspective on the Effectiveness of Using e-learning. *Procedia – Social and Behavioral Sciences*. 2014. Vol. 123. P. 139–144. URL: https://doi.org/10.1016/j.sbspro.2014.01.1407 (дата звернення: 01.04.2024).
7. Blackburn G. How Chatbots Could Be The Future Of Learning. URL: https://elearningindustry.com/chatbots-future-learning (дата звернення: 01.04.2024).
8. Technology-Driven Education: A New Era Of Learning. URL: https://www.forbes.com/sites/forbestechcouncil/2024/02/01/technology-driven-education-a-new-era-of-learning/ (дата звернення: 06.04.2024).

9. Czerwonka E. Common Scheduling Challenges and How to Fix Them. URL: https://buddypunch.com/blog/common-scheduling-software-challenges-and-how-to-fix-them/ (дата звернення: 09.04.2024).
10. What is calendar integration. URL: https://www.nylas.com/products/calendar-api/what-is-calendar-integration/ (дата звернення: 09.04.2024).
11. Your Pathway to Tutoring Success: Strategies on Managing Clients As a Tutor. URL: https://practice.do/blog/your-pathway-to-tutoring-success-strategies-on-managing-clients-as-a-tutor (дата звернення: 11.04.2024).
12. The Importance of Effective Feedback During Tutoring Sessions. URL: https://www.peardeck.com/blog/the-importance-of-effective-feedback-during-tutoring-sessions (дата звернення: 11.04.2024).
13. API Security Best Practices. URL: https://curity.io/resources/learn/api-security-best-practices/ (дата звернення: 12.04.2024).
14. Що таке дизайн взаємодії з користувачем (UX). URL: https://www.lyssna.com/blog/what-is-user-experience-design/ (дата звернення: 12.04.2024).
15. Larson J. What is user experience (UX) design. URL: https://www.lyssna.com/blog/what-is-user-experience-design/ (дата звернення: 12.05.2024).
16. Software Development Plan Template. URL: https://www.teamgantt.com/software-development-templates/software-development-plan (дата звернення: 12.05.2024).
17. Життєвий цикл розробки програмного забезпечення. URL: https://www.maxzosim.com/software-development-life-cycle-sdlc/ (дата звернення: 12.05.2024).
18. Altynpara E., Bestaieva D. Web-applications architectures: components, layers, and types. URL: https://www.cleveroad.com/blog/web-application-architecture/ (дата звернення: 12.05.2024).
19. What is JSON Web Token. URL: https://jwt.io/introduction (дата звернення: 12.05.2024).

ДОДАТОК А

Реалізація бази даних MongoBD

В цій кваліфікаційній роботі використовується база даних MongoDB для зберігання інформації про користувачів, оголошення, чати, повідомлення та планери. Для взаємодії з MongoDB використовується бібліотека Mongoose, яка забезпечує простий і ефективний спосіб роботи з даними в NoSQL базі даних. Реалізацію моделі користувача наведено на рисунку А.1.

server > models > JS User.js > ...

```
1  const mongoose = require("mongoose");
2
3  const UserSchema = new mongoose.Schema({
4    fullName: {
5      type: String,
6      required: true,
7    },
8    email: {
9      type: String,
10     required: true,
11     unique: true,
12   },
13   password: {
14     type: String,
15     required: true,
16   },
17   role: {
18     type: String,
19     required: true,
20     enum: ["Student", "Tutor"],
21   },
22   phoneNumber: {
23     type: String,
24   },
25   photo: {
26     type: String,
27   },
28 });
29
30 module.exports = mongoose.model("User", UserSchema);
31
```

Рисунок А.1 – Реалізація моделі користувача

Реалізацію моделі оголошення наведено на рисунку А.2.

```
server > models > JS Advert.js > ...
const mongoose = require("mongoose");

const AdvertSchema = new mongoose.Schema({
  title: {
    type: String,
    required: true,
  },
  description: {
    type: String,
    required: true,
  },
  subjects: {
    type: [String],
    required: true,
  },
  price: {
    type: Number,
    required: true,
  },
  level: {
    type: String,
    required: true,
  },
  user: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "User",
    required: true,
  },
});

module.exports = mongoose.model("Advert", AdvertSchema);

```

Рисунок А.2 – Реалізація моделі оголошення

Реалізацію моделі чату наведено на рисунку А.3.

```
const chatSchema = new mongoose.Schema({
  users: [{ type: mongoose.Schema.Types.ObjectId, ref: "User" }],
  messages: [{ type: mongoose.Schema.Types.ObjectId, ref: "Message" }],
});

```

Рисунок А.3 – Реалізація моделі чату

Реалізацію моделі повідомлення наведено на рисунку А.4.

```
const messageSchema = new mongoose.Schema({
  chat: { type: mongoose.Schema.Types.ObjectId, ref: "Chat" },
  sender: { type: mongoose.Schema.Types.ObjectId, ref: "User" },
  content: String,
  timestamp: { type: Date, default: Date.now },
});
```

Рисунок А.4 – Реалізація моделі повідомлення

Реалізацію моделі планера наведено на рисунку А.5.

```
const plannerSchema = new mongoose.Schema({
  user: { type: mongoose.Schema.Types.ObjectId, ref: "User" },
  planner: { type: Array, default: Array(7).fill(Array(24).fill("")) },
});
```

Рисунок А.5 – Реалізація моделі планера

Реалізацію підключення до бази даних наведено на рисунку А.6.

```
24  // MongoDB connection
25  mongoose.set("debug", true); // Enable mongoose debug mode
26  mongoose
27    .connect(process.env.MONGODB_URI, {
28      useNewUrlParser: true,
29      useUnifiedTopology: true,
30      serverSelectionTimeoutMS: 50000, // 50 seconds
31      socketTimeoutMS: 60000, // 60 seconds
32    })
33    .then(() => console.log("MongoDB connected"))
34    .catch((err) => {
35      console.error("MongoDB connection error:", err);
36      process.exit(1); // Exit process with failure
37    });
38
39  mongoose.connection.on("error", (err) => {
40    console.error("MongoDB connection error:", err);
41  });
42
43  mongoose.connection.once("open", () => {
44    console.log("Connected to MongoDB");
45  });
46
```

Рисунок А.6 – Реалізація підключення до бази даних

Приклад таблиці users бази даних MongoDB наведено на рисунку А.7.

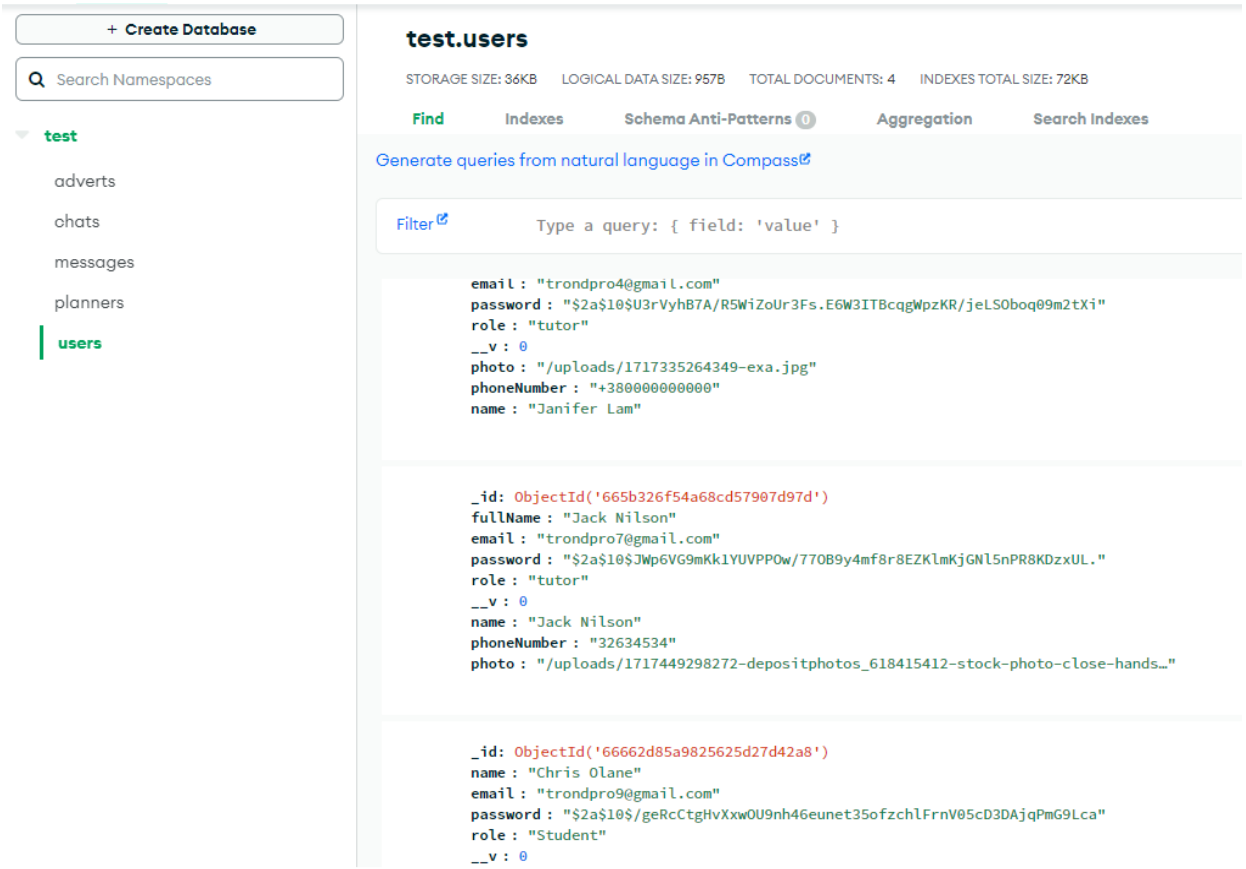

Рисунок А.7 – Таблиця users бази даних MongoDB

Моделі бази даних, створені з використанням бібліотеки Mongoose, забезпечують простий та ефективний спосіб взаємодії з MongoDB, дозволяючи швидко та безпечно зберігати та отримувати необхідну інформацію. Реєстрація користувачів, створення оголошень, обробка повідомлень чату та управління планерами – всі ці функціональні можливості реалізовані завдяки MongoDB.

ДОДАТОК Б

Зовнішня частина кваліфікаційної роботи (фронтенд)

Зовнішня частина кваліфікаційної роботи (фронтенд) побудована з використанням фреймворку React. React є однією з найпопулярніших бібліотек для розробки інтерфейсів користувача, що забезпечує високу продуктивність та зручність у створенні динамічних вебдодатків. Вона дозволяє розробникам створювати компоненти, які можна повторно використовувати, що значно спрощує процес розробки та обслуговування проєкту. Головну сторінку сайту наведено на рисунку Б.1.

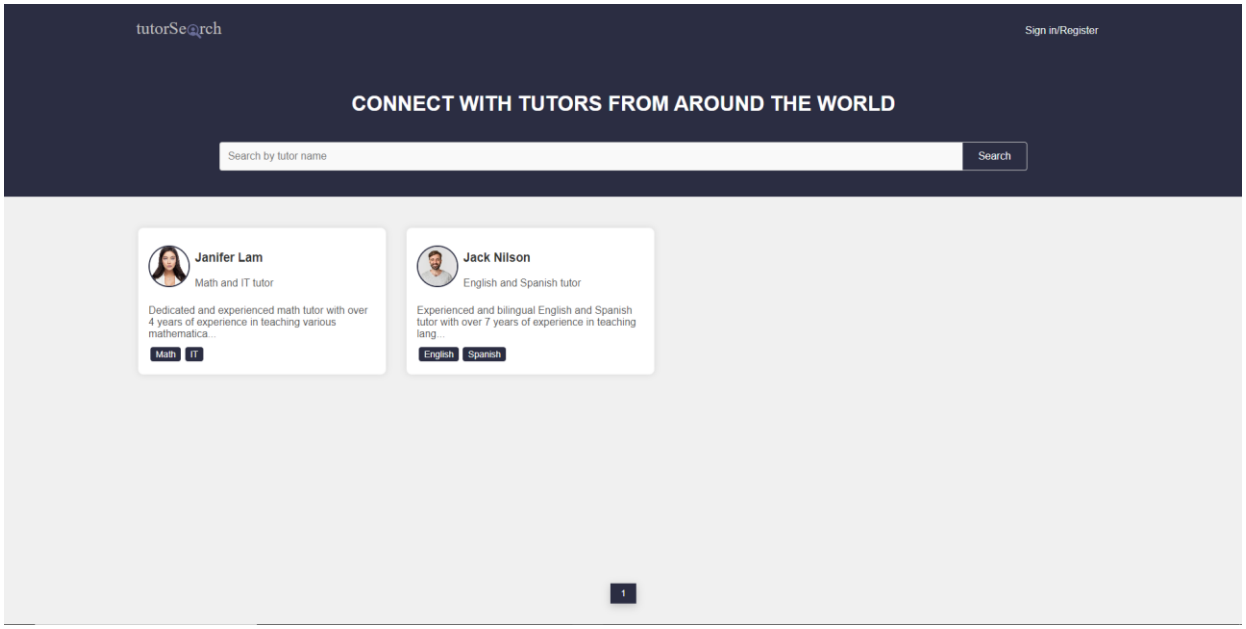

Рисунок Б.1 – Головна сторінку сайту

Головну сторінку сайту, з аккаунту ролі tutor наведено на рисунку Б.2.

Сторінку реєстрації наведено на рисунку Б.3.

Роботу робота пошукової служби наведено на рисунку Б.4.

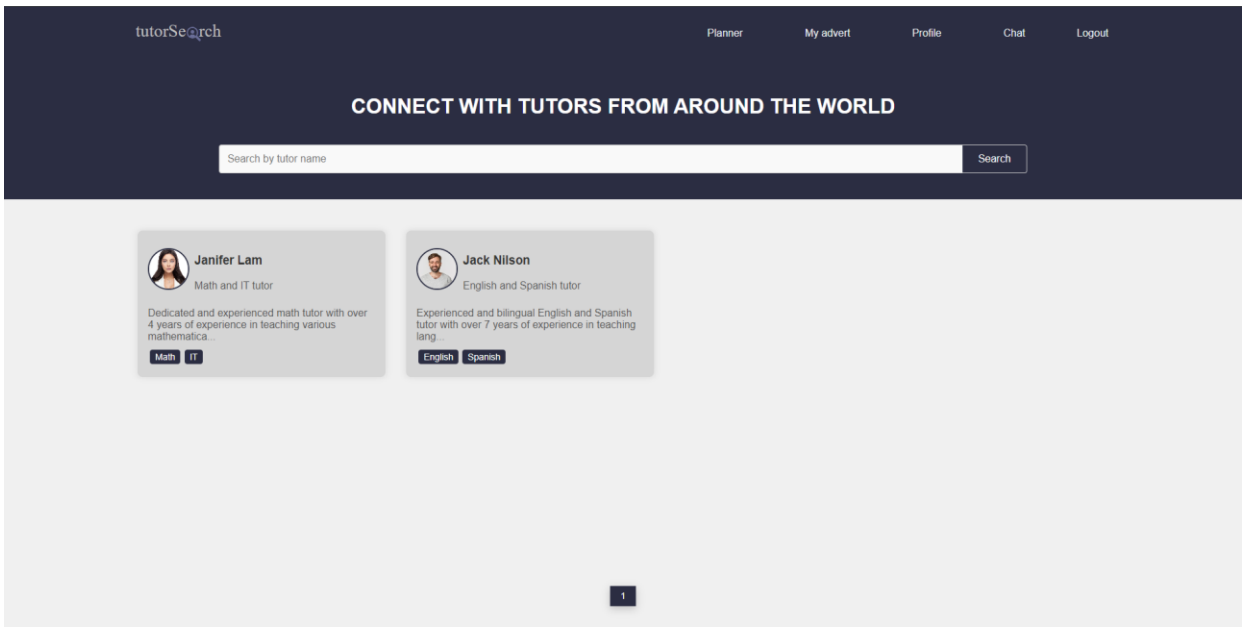

Рисунок Б.2 – Головна сторінку сайту, з аккаунту ролі tutor

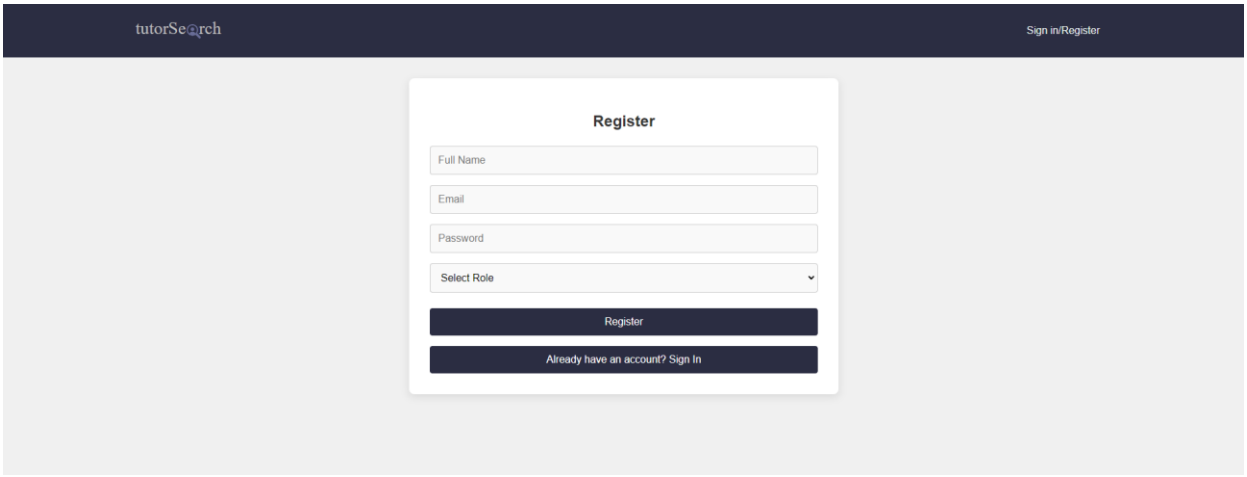

Рисунок Б.3 – Сторінка реєстрації

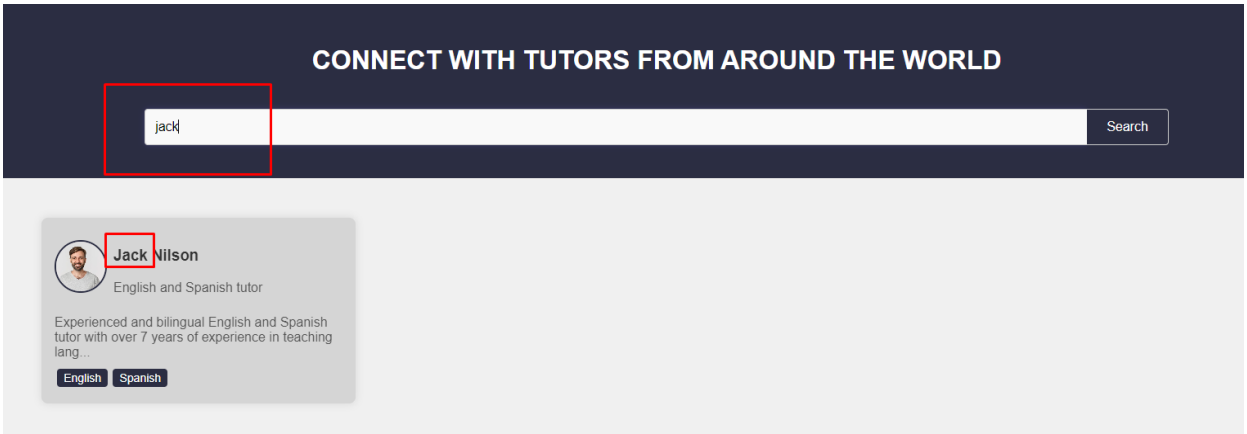

Рисунок Б.4 – Робота пошукової служби

Сторінку входу наведено на рисунку Б.5.

Сторінку додавання оголошення наведено на рисунку Б.6.

Сторінку редагування профілю наведено на рисунку Б.7.

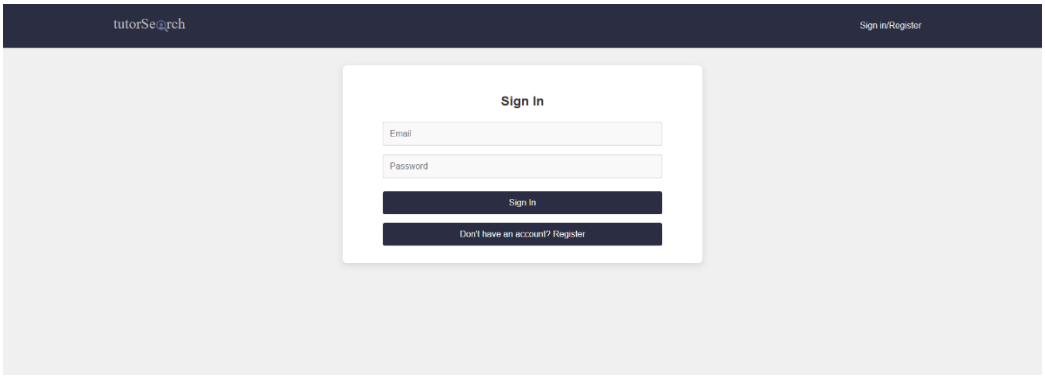

Рисунок Б.5 – Сторінка входу

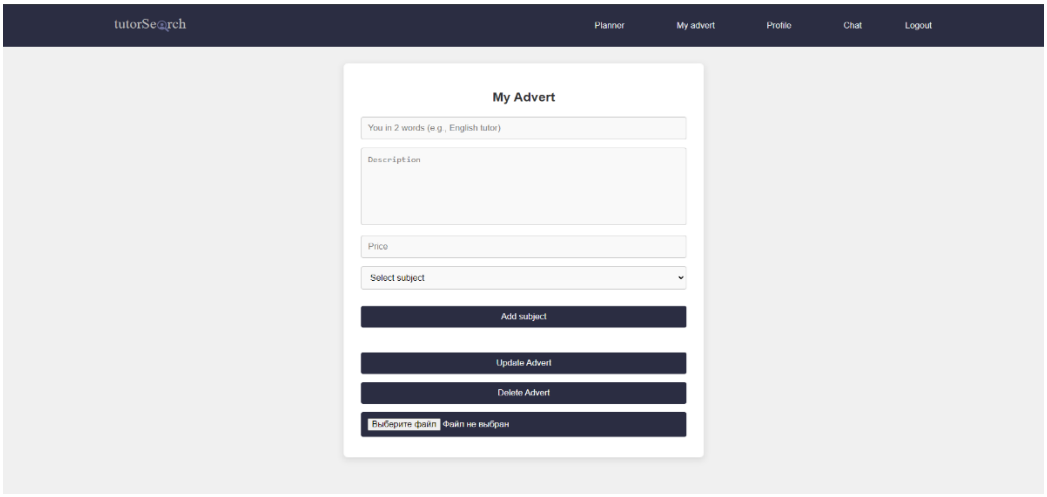

Рисунок Б.6 – Сторінка додавання оголошення

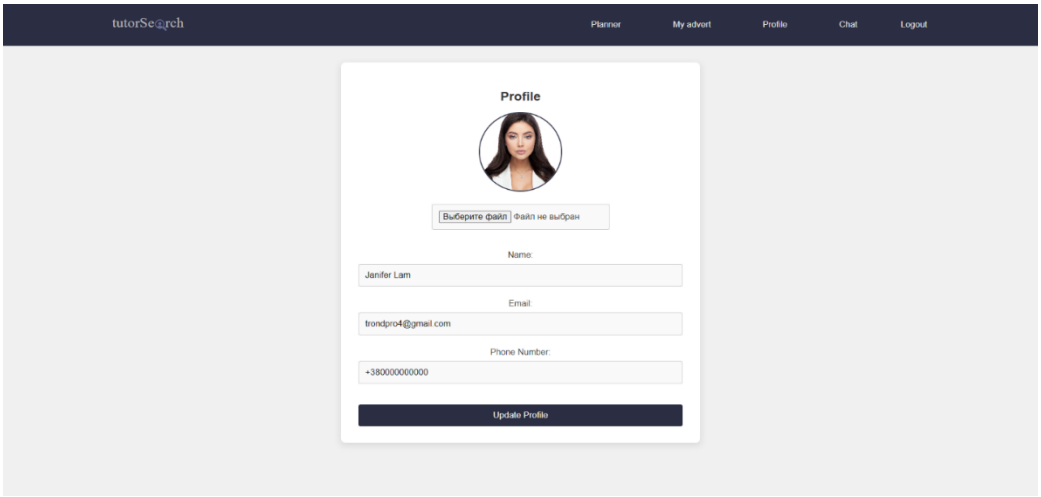

Рисунок Б.7 – Сторінка редагування профілю

Сторінку оголошення репетитора наведено на рисунку Б.8.

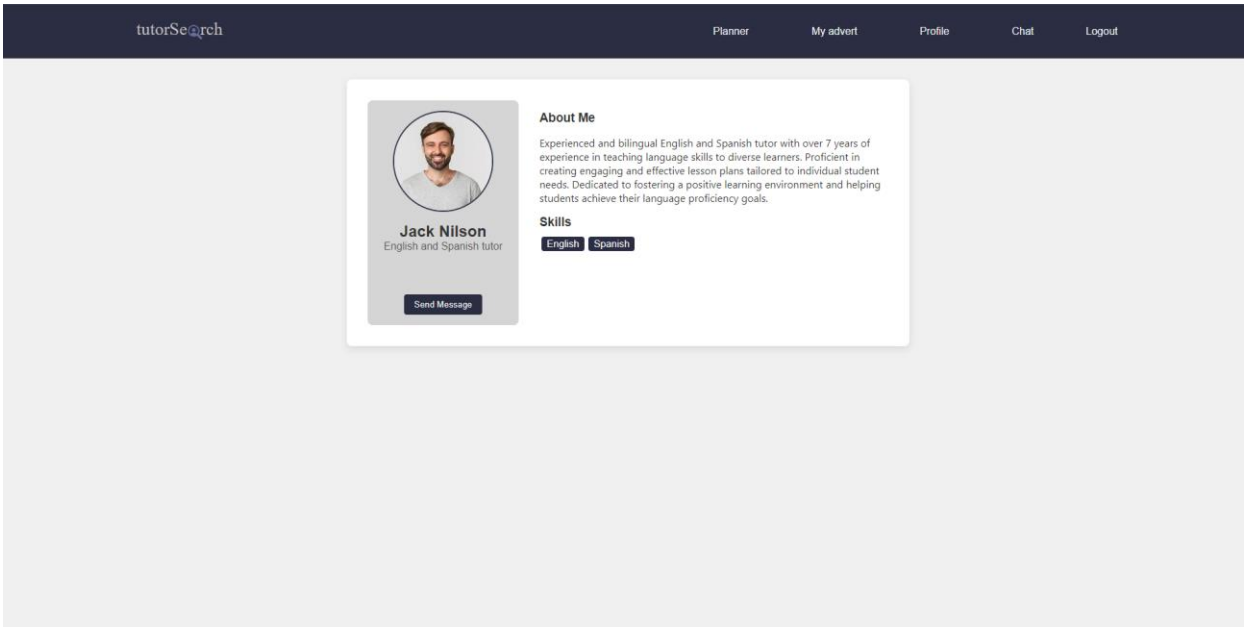

Рисунок Б.8 – Сторінка оголошення репетитора

Сторінку чата наведено на рисунку Б.9.

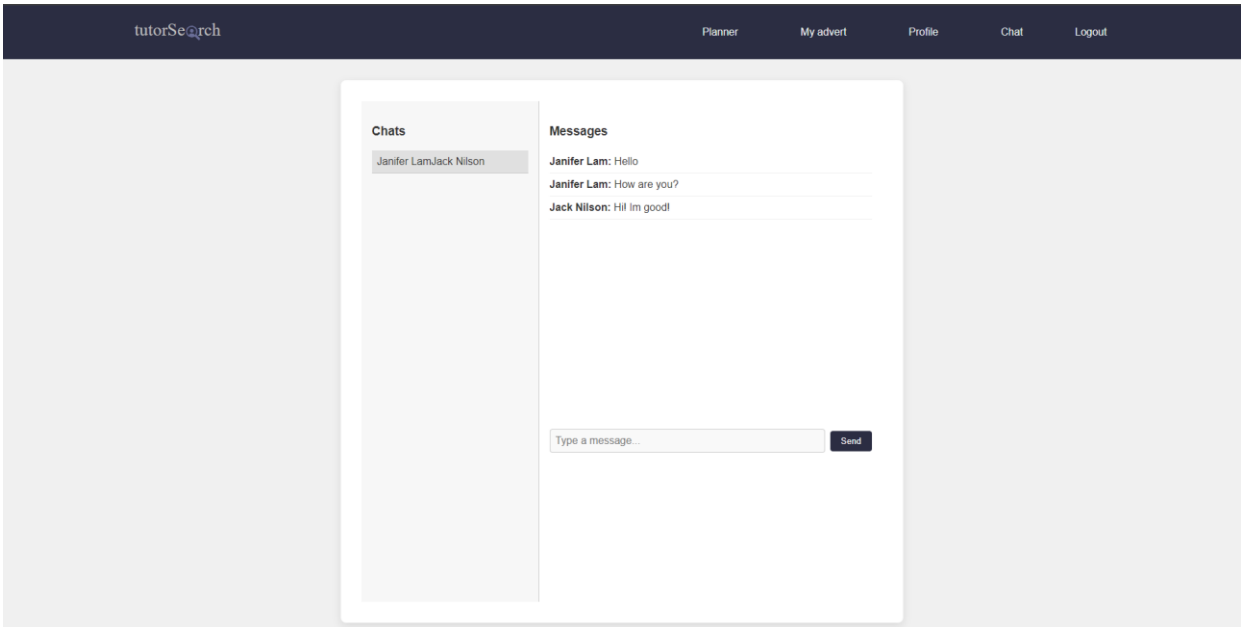

Рисунок Б.9 – Сторінка чата

Сторінку планера для репетитора наведено на рисунку Б.10.

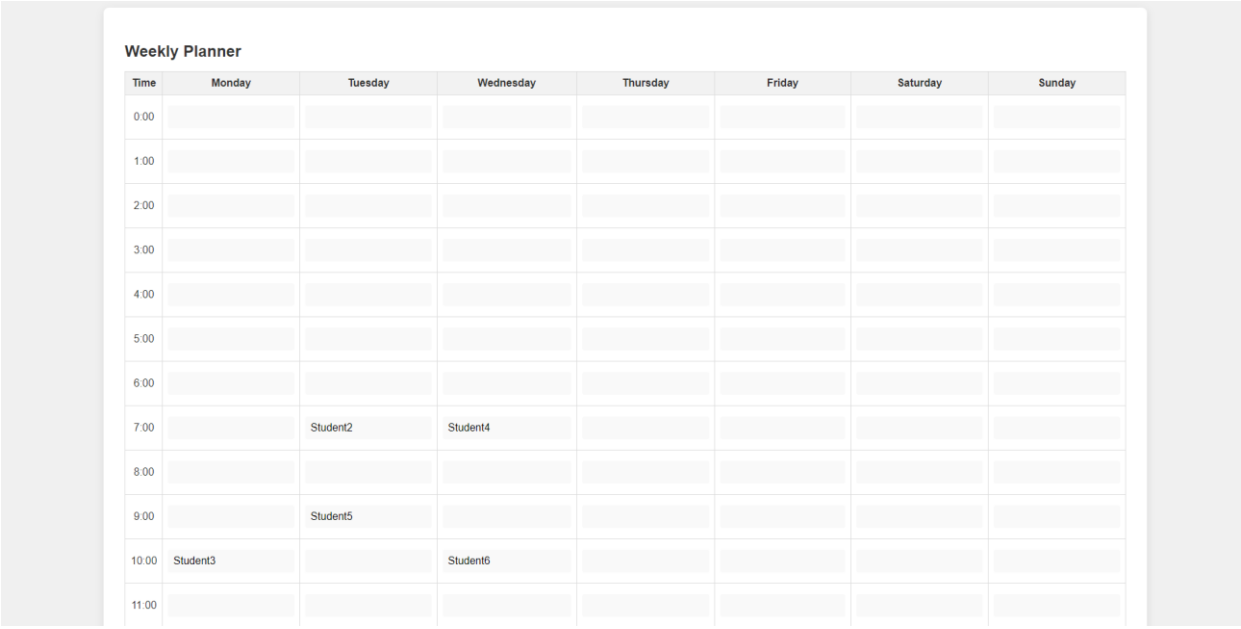

Рисунок Б.10 – Сторінка планера для репетитора

Завдяки використанню React кваліфікаційна робота досягає високого рівня продуктивності та зручності користування, оскільки компоненти можуть бути повторно використані, а оновлення даних відбувається без перезавантаження сторінки. Інтеграція з бібліотекою Axios забезпечує надійний зв’язок з серверною частиною для обробки даних та виконання запитів.

Дизайн інтерфейсу враховує найкращі практики UI/UX, забезпечуючи інтуїтивно зрозумілий та привабливий вигляд. Це сприяє кращій взаємодії користувачів із системою та підвищує задоволення від користування платформою.