Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра автоматизованих систем управління



**Звіт**

до лабораторної роботи №5

з дисципліни

“Веб-технології та розробка веб-застосувань”

на тему  
**“Розробка серверної частини додатку та її розгортання”**

Виконав:

студент групи ОІ-26

*Чабанов Павло*

Прийняла:

*Троян О. А.*

Львів – 2025

**Тема:** Розробка серверної частини додатку та її розгортання.

**Мета:** Навчитись розробляти серверну частину веб-додатків за допомогою Node.js та розгортати веб-додатки за допомогою хмарних сервісів.

**Лабораторне завдання**

**Завдання 1: Створення серверної частини за допомогою Node.js**

• Опис:

Розробити серверну частину додатку, створивши сервер використовуючи Node.js з Express. Цей сервер має обробляти запити API від клієнтської частини (фронтенду) та взаємодіятиме з базою даних для зберігання та отримання даних.

• Кроки:

1. Ініціалізувати новий проект Node.js та встановити необхідні залежності (наприклад, Express, CORS тощо).
2. Налаштувати Express-сервер для обробки вхідних запитів з клієнтської частини.
3. Визначити основні маршрути (наприклад, GET для отримання даних і POST для відправки даних).
4. Налаштувати маршрути автентифікації:

* POST /register: Створення нового користувача.
* POST /login: Автентифікація користувача та повернення токену JWT.
* GET /profile: Отримання даних користувача за допомогою токена.

5. Захистити приватні маршрути за допомогою проміжного програмного рішення для доступу тільки автентифікованих користувачів.

6. Протестувати сервер, налаштувавши прості маршрути, які повертають статичні дані або підтверджують успішні запити.

**Завдання 2: Робота з базою даних (зберігання та пошук даних)**

• Опис:

Інтегрувати хмарну базу даних (наприклад, Firebase Firestore) зі розробленою у завданні 1 серверною частиною для зберігання та отримання даних. Дані будуть використовуватися клієнтською частиною для динамічного відображення контенту.

• Кроки:

1. Налаштувати хмарну базу даних (Firebase Firestore) і підключити її до розробленого сервера Node.js за допомогою відповідного SDK або клієнтської бібліотеки.
2. Створити моделі даних (наприклад, дописи в блозі, інформацію про продукти, деталі подій) і визначити схему їх зберігання в базі даних.
3. Реалізувати маршрути API для зберігання даних у базі даних (POST) та їх отримання (GET).
4. Реалізувати завантаження клієнтською частиною збережених даних і динамічно відображати їх на веб-сторінці (наприклад, показувати список продуктів, дописів або деталі подій).

**Завдання 3: Розгортання веб-додатку на хмарному хостингу**

• Опис: Розгорнути веб-додаток на обраній хмарній хостинговій платформі.

• Кроки:

1. Обрати хостинг-провайдера

2. Налаштувати хмарне середовище:

▪ Розгорнути клієнтську частину додатка на статичному хостингу.

▪ Розгорнути серверну частину додатка за допомогою хмарної функції або сервера.

3. Налаштувати комунікацію між клієнтською та серверною частиною додатка:

▪ Переконатись, що маршрути API працюють з розгорнутою серверною частиною.

▪ Налаштувати CORS, якщо це необхідно.

**Хід виконання роботи**

**Завдання 1: Створення серверної частини за допомогою Node.js**

**Крок 1**. Ініціалізувати новий проект Node.js та встановити необхідні залежності (наприклад, Express, CORS тощо).

Слідуючи інструкціям я ініціалізував новий проект Node.js та встановив необхідні залежності(Express). Зробив я це за допомогою консольних команд:

1. npm init -y
2. npm install express cors

**Крок 2**. Налаштувати Express-сервер для обробки вхідних запитів з клієнтської частини.

Для виконання цього завдання я створив файл server.js в корені програми. Всередині цього файлу я налаштував Express-сервер для обробки вхідних запитів з клієнтської частини, що видно з Рис. 1.1.

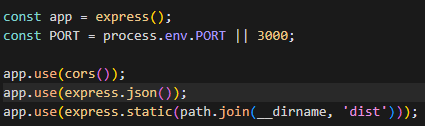


Рис. 1.1. Налаштування Express-сервера.

**Крок 3**. Визначити основні маршрути (наприклад, GET для отримання даних і POST для відправки даних).

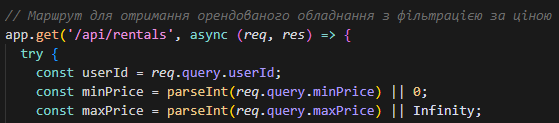


Рис. 1.2. Маршрут для отримання обладнання з фільтрацією.

Для того щоб отримувати орендоване обладнання з фільтрацією за ціною потрібно створити маршрут. Для цього я використав GET, тобто, серверна частина отримує дані, після чого маніпулює ними. Це показано на Рис.1.2.

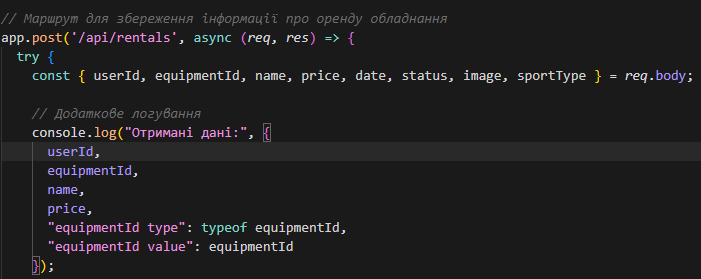


Рис. 1.3. Маршрут для збереження інформації про оренду обладнання.

Я написав код, що зображений на Рис 1.3, який за допомогою POST зберігає інформацію про нову оренду обладнання.

**Крок 4**. Налаштувати маршрути автентифікації.

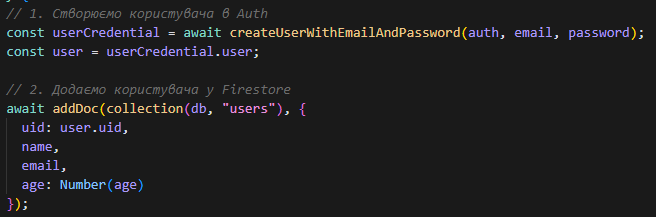


Рис 1.4. Реєстрація (POST /register).

Я створив реєстрацію за допомогою серверної частини, вона зображена на Рис 1.4. Реєстрація реалізована у Register.jsx через Firebase.

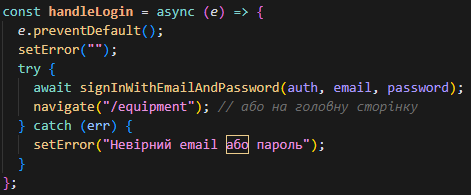


Рис. 1.5. Логін (POST /login).

Для входу в акаунт за допомогою серверної частини я створив функцію з використанням **Firebase Auth.** Функціонал реалізований у Login.jsx та зображений на Рис. 1.5.

**Крок 5**. Захистити приватні маршрути за допомогою проміжного програмного рішення для доступу тільки автентифікованих користувачів.

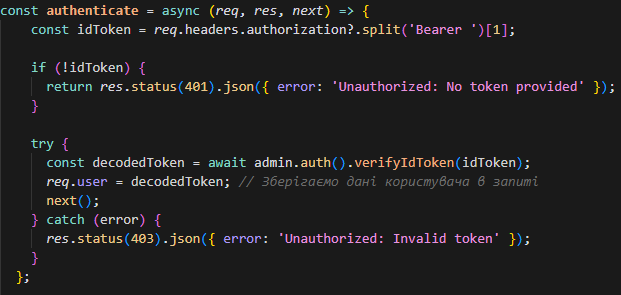


Рис. 1.6. Захист приватних маршрутів.

Я створив окремий middleware, який перевіряє автентифікацію. Він зображений на Рис. 1.6.

**Крок 6**. Протестувати сервер, налаштувавши прості маршрути, які повертають статичні дані або підтверджують успішні запити.



Рис. 1.7. Тест сайту.

Я додав до свого коду частину, що зображена на Рис. 1.7. Цей код додає новий тестовий маршрут для перевірки працездатності сервера. Для того щоб протестувати, потрібно перейти по посиланню <http://localhost:3000/test>. Після переходу, бачимо результат, що зображений на Рис. 1.8.

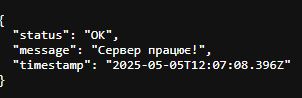


Рис. 1.8. Тестове повідомлення.

**Завдання 2. Робота з базою даних (зберігання та пошук даних)**

**Крок 1**. Налаштувати хмарну базу даних (Firebase Firestore) і підключити її до розробленого сервера Node.js за допомогою відповідного SDK або клієнтської бібліотеки.

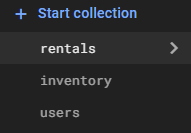


Рис. 2.1. Всі таблиці в базі даних.

Я перейшов в Firebase Console > Firestore Database > Create Database та. Ці таблиці показані на Рис. 2.1.

**Крок 2**. Імпортувати Firebase SDK і налаштувати Firestore для роботи з базою даних.

Цей крок я виконав на кроці 3 минулого завдання.

**Крок 3**. Створити функцію для запису даних у Firestore, наприклад, для збереження інформації про користувача.



Рис. 2.3. Функція для збереження даних про користувача.

Для збереження даних про користувача я скористався функціоналом реєстрації. При реєстрації користувача, програма перевіряє, чи не існує такої ж почти у базі даних, і якщо ні, то додає користувача та введені дані про нього у базу даних.

**Крок 4**. Реалізувати функцію для читання даних з Firestore, щоб отримати збережену інформацію.

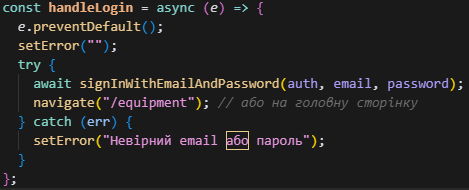


Рис. 2.4. Функція для читання даних про користувача.

Для читання даних про користувача я скористався функціоналом входу в систему. При вході в систему програма перевіряє та співспавляє почти та паролі. Якщо знайдено таку почту та пароль, то користувач успішно входить в систему.

**Крок 5**. Додати виклики цих функцій для перевірки працездатності: додавання даних до бази та отримання їх з бази.

Виклики цих функцій додані при натисканні відповідних кнопок “Вхід” та “Зареєструватися”.

**Хід роботи**

**Варіант 23**

Веб-сайт платформи для оренди спортивного обладнання

1. Створити форми реєстрації та входу користувачів на сайт та впровадити функції автентифікації.
2. Реалізувати можливість оформлювати оренду обладнання доданого у кошик і переглядати активні та минулі оренди тільки для автентифікованих користувачів.
3. Створити хмарну базу даних для зберігання інформації про обладнання. Використовувати інформацію з бази даних для відображення на клієнтській частині.

**Крок 1**. Створити форми реєстрації та входу користувачів на сайт та впровадити функції автентифікації.

Форми входу та реєстрації користувачів на сайт були описані відповідно на Рис. 1.4.1 та Рис. 1.4.3.

**Крок 2**. Реалізувати можливість оформлювати оренду обладнання доданого у кошик і переглядати активні та минулі оренди тільки для автентифікованих користувачів.

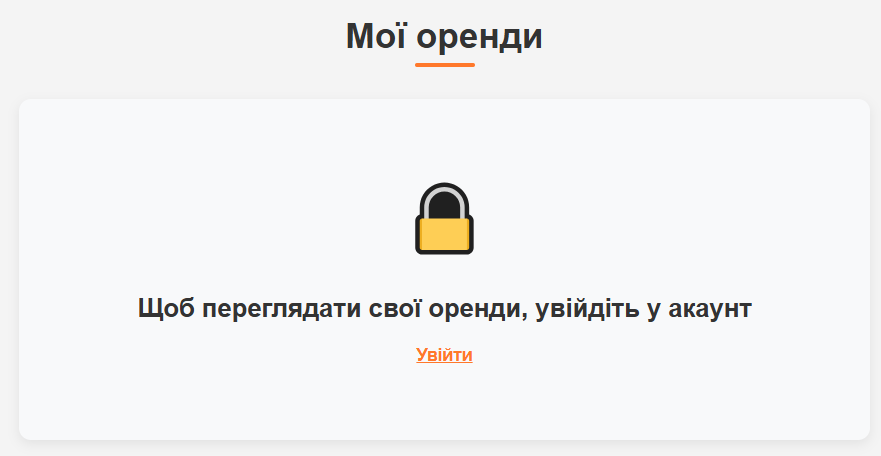


Рис. 3.1. Мої оренди для неавтентифікованих користувачів.

Перегляд історії активних та минулих оренд відбувається на сторінці Мої оренди. Для неавтентифікованих користувачів вміст сторінки Мої оренди буде приховано, доки вони не ввійдуть в систему. Також на цій сторінці є заклик та посилання для входу.

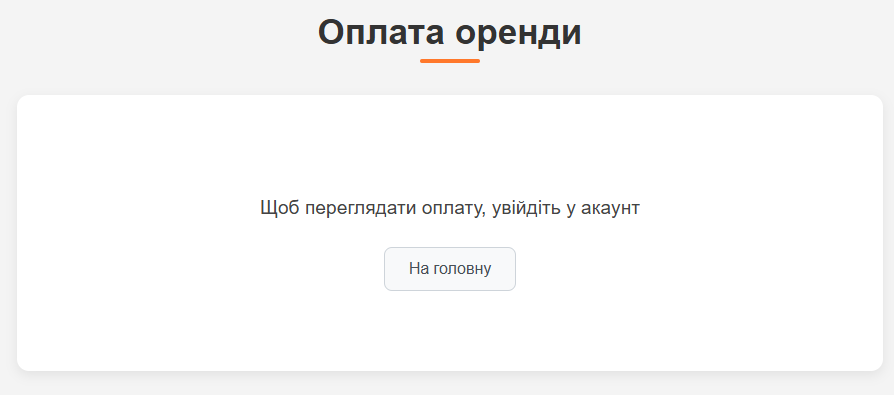


Рис. 3.2. Оплата для неавтентифікованих користувачів

Можливість оплати для неавтентифікованих користувачів також буде заблокованою, так як вони не мають змоги орендувати речі, а лише переглядати, то і оплачувати їм нічого не потрібно.

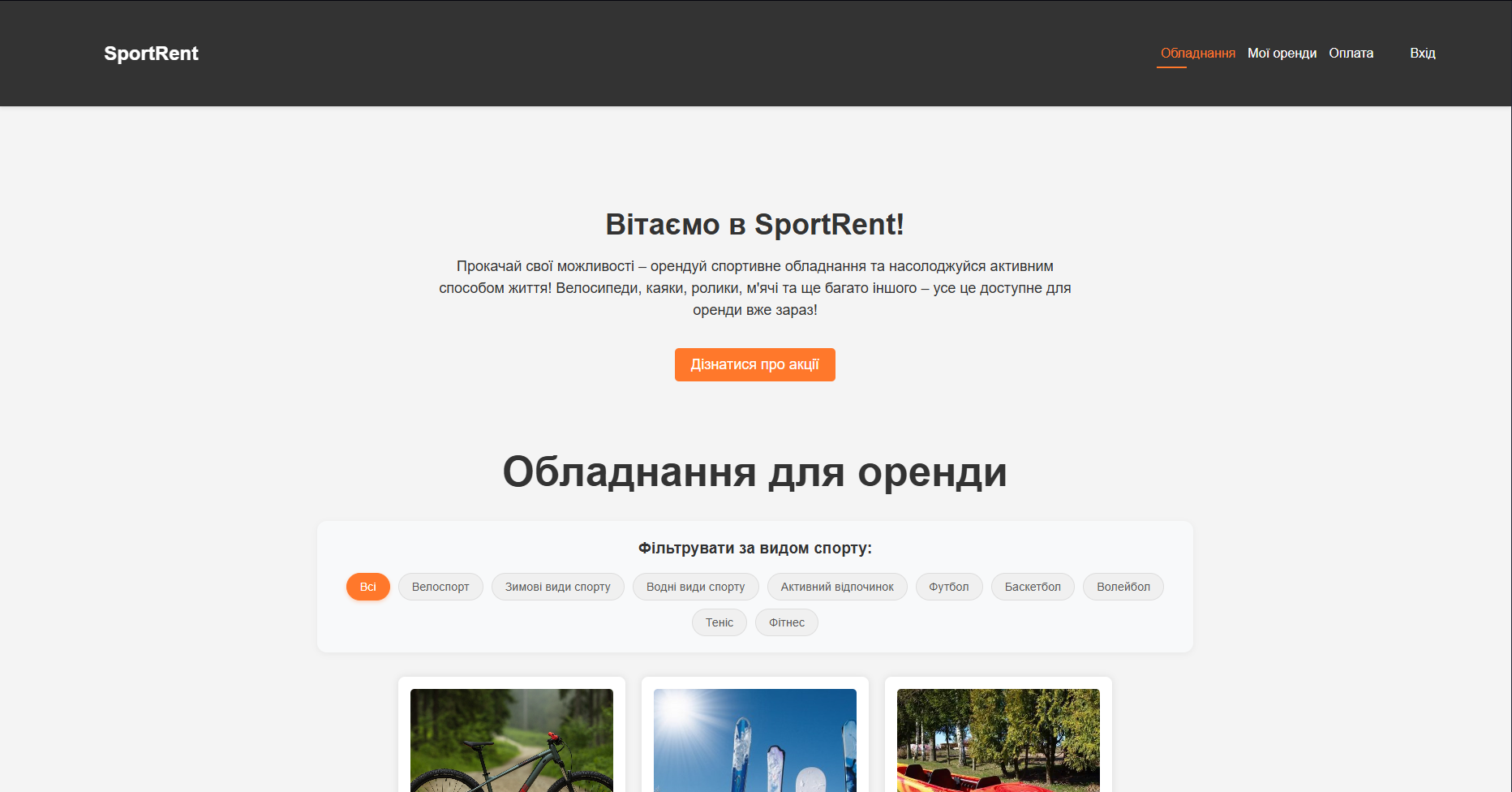
**Крок 3**. Створити хмарну базу даних для зберігання інформації про обладнання. Використовувати інформацію з бази даних для відображення на клієнтській частині.

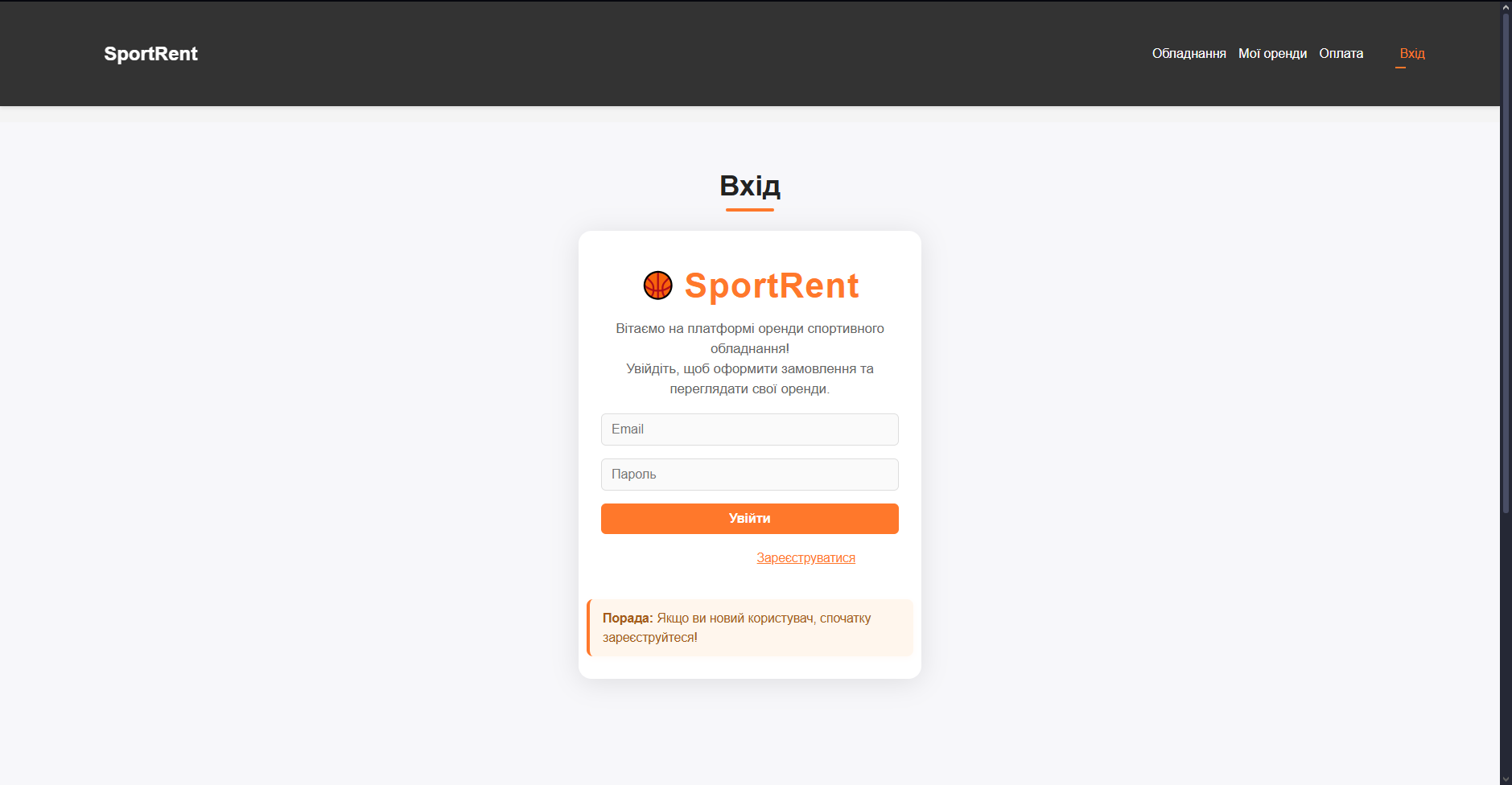
Я створив хмарну базу даних на сайті Firebase, де зберігаю всю важливу інформацію, таку як інформація про користувачів, інвентар та оренди. Подробніше я розповідаю про це в завданні 2 крок 1.

**Висновок**

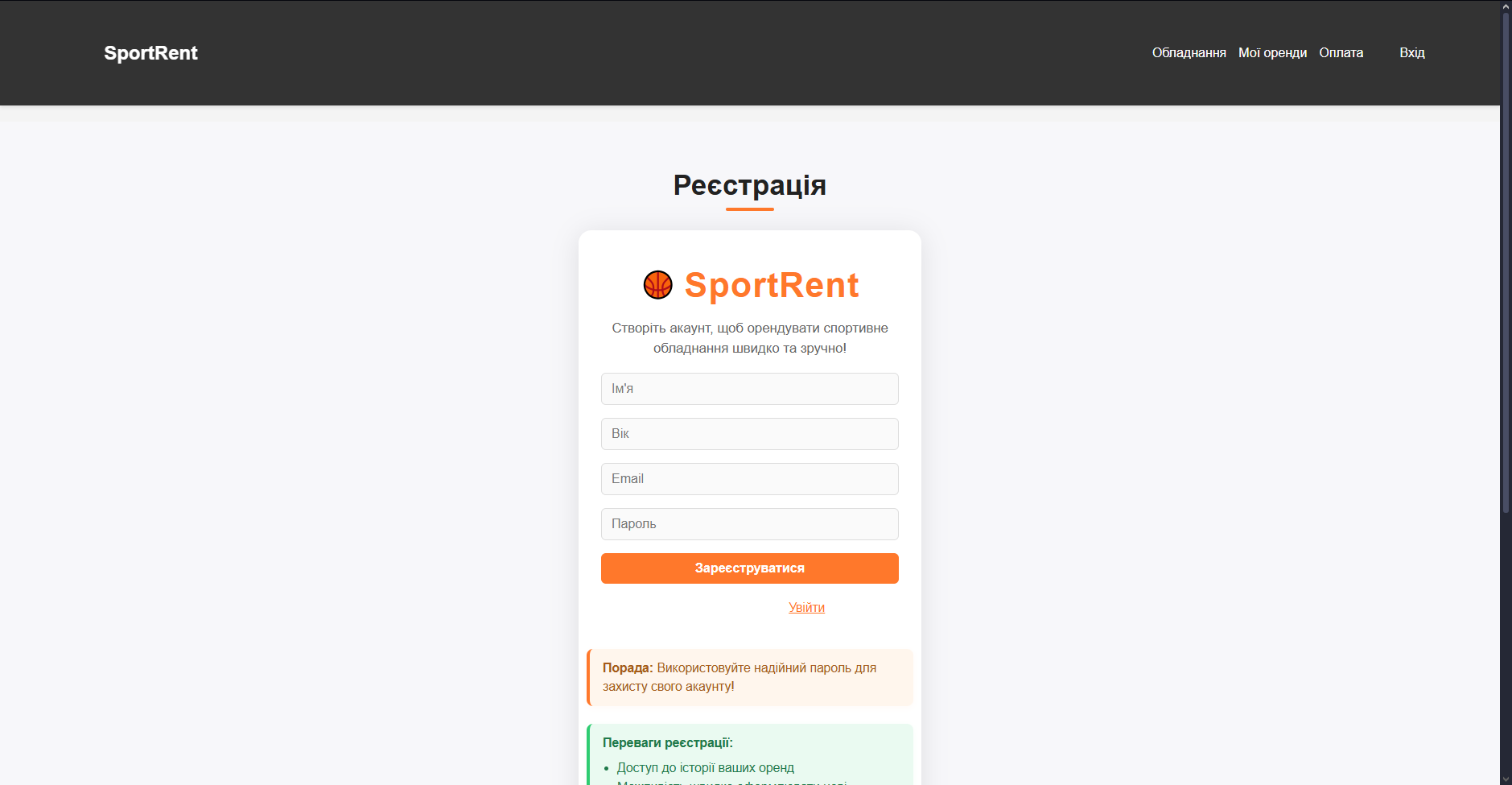
Отже, на цій лабораторній роботі я навчився реалізовувати автентифікацію користувачів. Навчився реалізовувати підключення до бази даних для зберігання та управління даними.

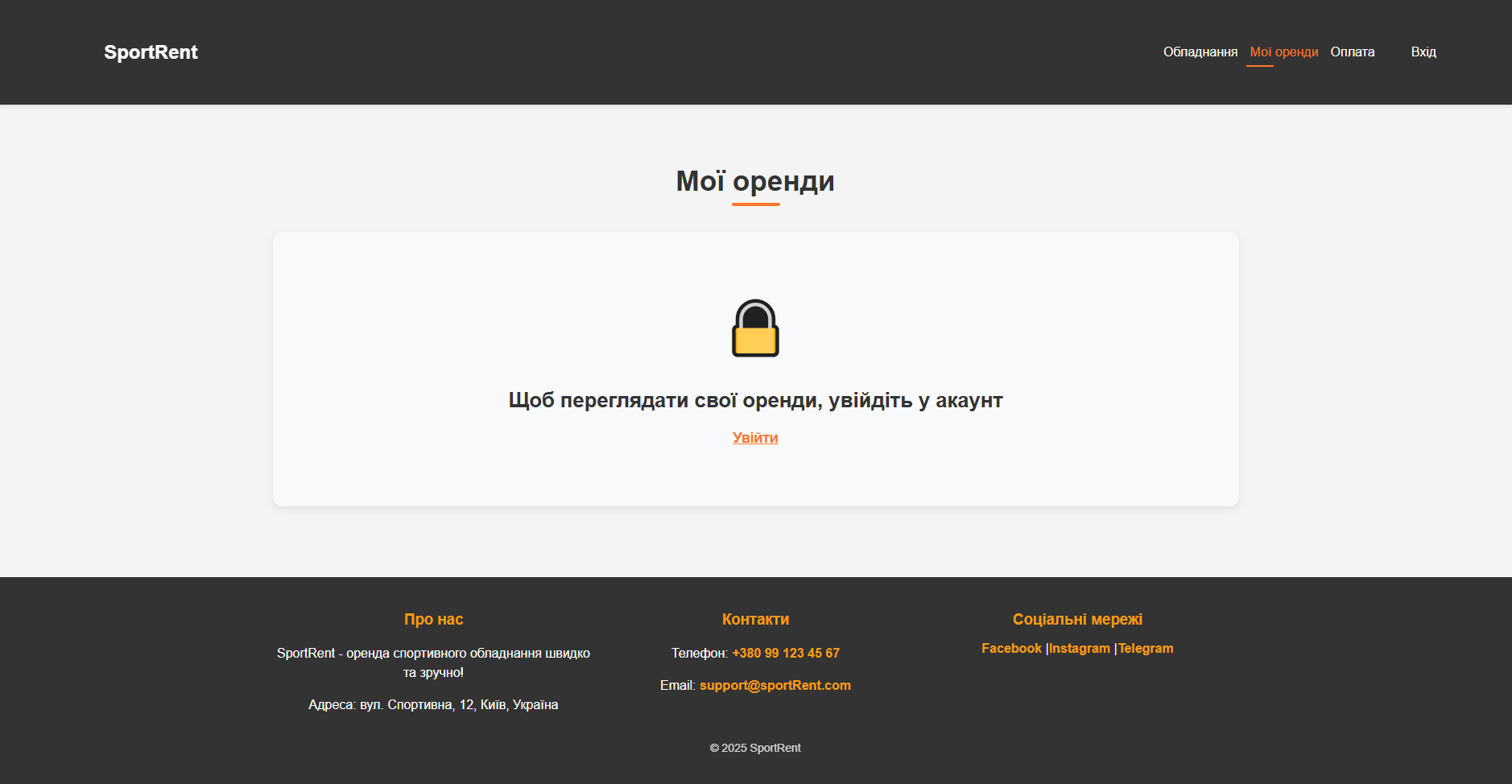
**Додатки**

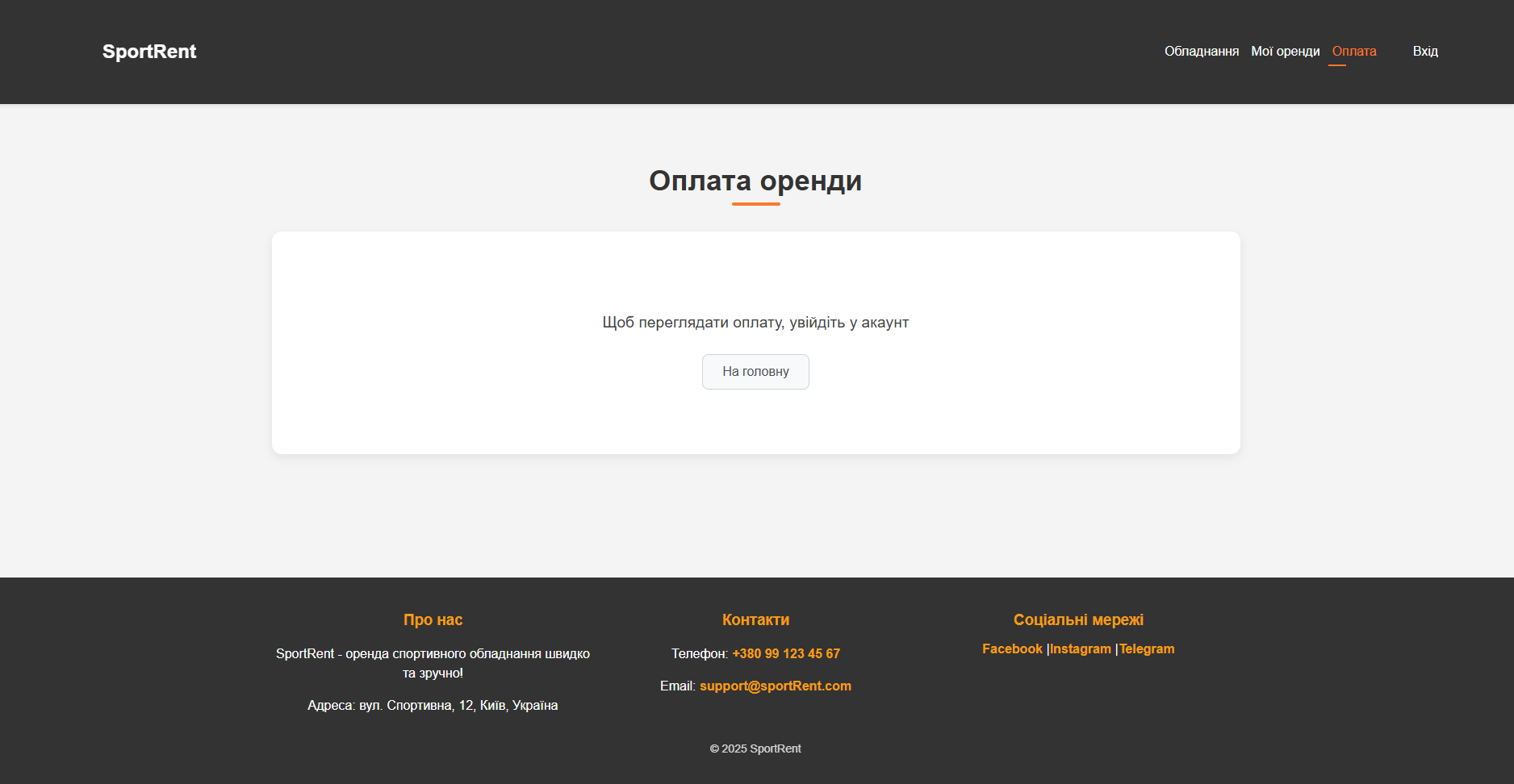
**Дотаток А (скріншот головної сторінки)  
  
  
Додаток Б ( скріншот сторінки входу)**

****

**Додаток В ( скріншот сторінки реєстрації)**

****

**Додаток Г (скріншот сторінки “Мої оренди” без входу)  
**

**Додаток Ґ (скріншот сторінки “Оплата” без входу)  
**