

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-застосунків”
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 взаємодіяти 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 токена 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:
 - Переконайтесь, 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 проєкт Node.js та встановити необхідні залежності (наприклад, Express, CORS тощо).

Слідуючи інструкціям я ініціалізував новий проєкт 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.

```
const app = express();
const PORT = process.env.PORT || 3000;

app.use(cors());
app.use(express.json());
app.use(express.static(path.join(__dirname, 'dist')));
```

Рис. 1.1. Налаштування Express-сервера.

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

```
// Маршрут для отримання орендованого обладнання з фільтрацією за ціною
app.get('/api/rentals', async (req, res) => {
  try {
    const userId = req.query.userId;
    const minPrice = parseInt(req.query.minPrice) || 0;
    const maxPrice = parseInt(req.query.maxPrice) || Infinity;
```

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.

```
// Маршрут для збереження інформації про оренду обладнання
app.post('/api/rentals', async (req, res) => {
  try {
    const { userId, equipmentId, name, price, date, status, image, sportType } = req.body;

    // Додаткове логування
    console.log("Отримані дані:", {
      userId,
      equipmentId,
      name,
      price,
      "equipmentId type": typeof equipmentId,
      "equipmentId value": equipmentId
    });
  }
});
```

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.

```
// 1. Створюємо користувача в Auth
const userCredential = await createUserWithEmailAndPassword(auth, email, password);
const user = userCredential.user;

// 2. Додаємо користувача у Firestore
await addDoc(collection(db, "users"), {
  uid: user.uid,
  name,
  email,
  age: Number(age)
});
```

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.

```
const handleLogin = async (e) => {
  e.preventDefault();
  setError("");
  try {
    await signInWithEmailAndPassword(auth, email, password);
    navigate("/equipment"); // або на головну сторінку
  } catch (err) {
    setError("Невірний email або пароль");
  }
};
```

Рис. 1.5. Логін (POST /login).

Для входу в акаунт за допомогою серверної частини я створив функцію з використанням **Firestore 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.

```
const authenticate = async (req, res, next) => {
  const idToken = req.headers.authorization?.split('Bearer ')[1];

  if (!idToken) {
    return res.status(401).json({ error: 'Unauthorized: No token provided' });
  }

  try {
    const decodedToken = await admin.auth().verifyIdToken(idToken);
    req.user = decodedToken; // Зберігаємо дані користувача в запиті
    next();
  } catch (error) {
    res.status(403).json({ error: 'Unauthorized: Invalid token' });
  }
};
```

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.

```
// Тестовий GET-маршрут для перевірки роботи сервера
app.get('/test', (req, res) => {
  res.json({
    status: 'OK',
    message: 'Сервер працює!',
    timestamp: new Date().toISOString()
  });
});

// Тестовий POST-маршрут
app.post('/test', (req, res) => {
  const { data } = req.body;
  res.json({
    status: 'OK',
    receivedData: data || 'Немає даних у тілі запиту',
    method: 'POST'
  });
});
```

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.

```
{
  "status": "OK",
  "message": "Сервер працює!",
  "timestamp": "2025-05-05T12:07:08.396Z"
}
```

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.

```
const serviceAccount = require('./firebase-service-account.json');
admin.initializeApp({
  credential: admin.credential.cert(serviceAccount)
});
const db = admin.firestore();
```

Рис. 2.1. Підключення до Firestore через Firebase Admin SDK.

Для підключення бази даних до сервера я використав Firebase Admin SDK. Для цього в кабінеті Firebase потрібно створити новий секретний ключ. Він

буде збережений в json форматі. Його потрібно перекинути в корінь проекту. Після цього в server.js вписати код, що зображений на Рис. 2.1.

Крок 2. Створити моделі даних (наприклад, дописи в блозі, інформацію про продукти, деталі подій) і визначити схему їх зберігання в базі даних.

У мене в базі даних є такі таблиці з полями:

1. rentals

- userId
- equipmentId
- name
- price
- date
- status
- image
- sportType

2. inventory

- name
- price
- quantity
- sportType
- image

Крок 3. Реалізувати маршрути API для зберігання даних у базі даних (POST) та їх отримання (GET).

```
app.get('/api/rentals', async (req, res) => {  
  const { userId, minPrice, maxPrice } = req.query;  
  // ... фільтрація даних з Firestore  
});
```

Рис. 2.2. GET /api/rentals.

На Рис. 2.2 зображено маршрут для отримання оренд з бази даних з фільтрацією за ціною.

```
app.post('/api/rentals', async (req, res) => {  
  const { userId, equipmentId, name, price } = req.body;  
  // ... збереження в Firestore  
});
```

Рис. 2.3. GET /api/rentals.

На Рис. 2.3 зображено маршрут для додавання нової оренди в базу даних.

Крок 4. Реалізувати завантаження клієнтською частиною збережених даних і динамічно відображати їх на веб-сторінці (наприклад, показувати список продуктів, дописів або деталі подій).

```
const querySnapshot = await getDocs(collection(db, "inventory"));  
const items = querySnapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }));
```

Рис. 2.4. Завантаження даних на клієнті.

Цей крок виконаний у клієнтських компонентах. Equipment.jsx - отримує список обладнання з Firestore, що зображений на Рис. 2.3

```
const response = await fetch(`/api/rentals?userId=${user.uid}`);  
const data = await response.json();
```

Рис. 2.5. Завантаження даних на клієнті.

Rentals.jsx - отримує оренди через API (/api/rentals). Це зображено на Рис. 2.5.

Завдання 3. Розгортання веб-додатку на хмарному хостингу.

Крок 1. Обрати хостинг-провайдера.

Крок 2. Налаштувати хмарне середовище:

Розгорнути серверну частину додатка за допомогою хмарної функції або сервера.

Для розгортання серверної частини додатку я скористався сервісом render. Для того щоб заhostити на ньому свій сервер, спочатку потрібно створити акаунт. Після створення акаунту потрібно налаштувати хостинг. Після успішного налаштування, деплоїмо проект. Результат показано на Рис 3.1.

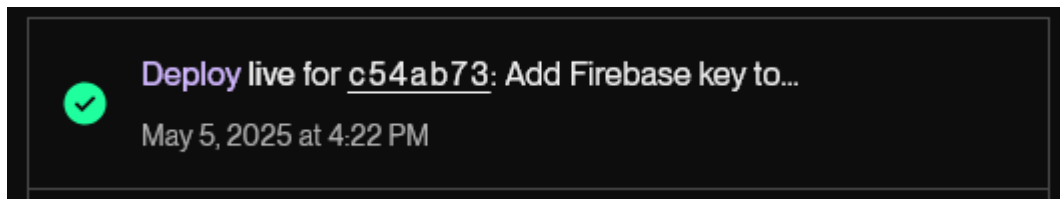


Рис. 3.1. Деплой проекту серверу.

Розгорнути клієнтську частину додатка на статичному хостингу.

Для розгортання фронтенду я обрав такий сервіс як Vercel. Він надає безплатний варіант хостингу для клієнтської частини. Після введення всіх даних про збірку, проект готовий до відвідування.

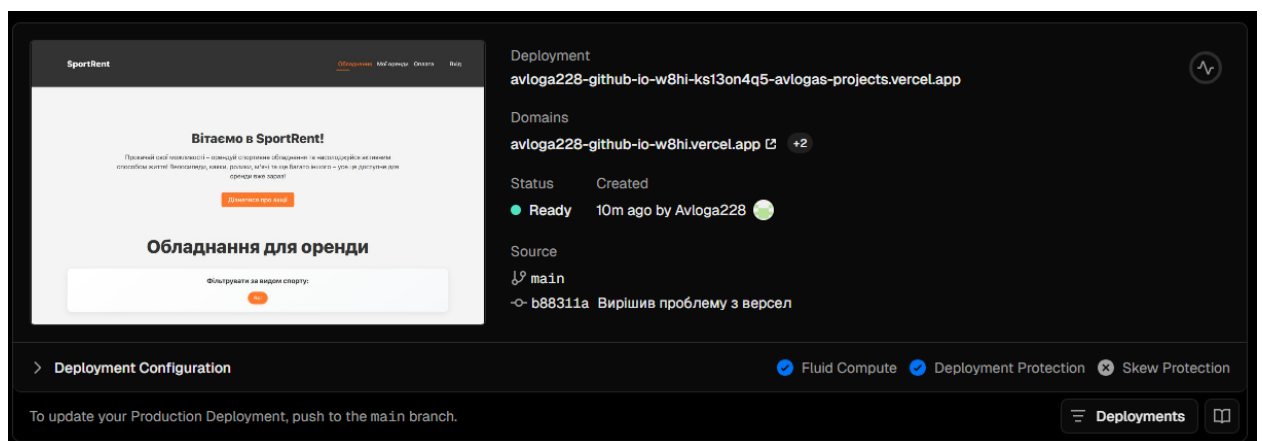


Рис. 3.2. Деплой проекту клієнту.

Хід роботи

Варіант 23

Веб-сайт платформи для оренди спортивного обладнання

Веб-сайт платформи для оренди спортивного обладнання

1. Налаштувати серверну частину з використанням Node.js/Express для хостингу статичних файлів сайту.
2. Створити хмарну базу даних для збереження інформації про оформлені користувачем оренди обладнання.
3. На сервері створити маршрут (HTTP GET) для отримання інформації про оформлені користувачем оренди обладнання відфільтровані по ціні. Використати створений маршрут у клієнтській частині для відображення інформації про оформлені користувачем оренди обладнання на сторінці "Мої оренди".
4. Реалізувати на сервері маршрут (HTTP POST) для збереження інформації про оформлені користувачем оренди обладнання та ціну оренди. Використати створений маршрут у клієнтській частині, для збереження змін інформації про оформлені користувачем оренди обладнання.
5. Розгорнути веб-сайт платформи для оренди спортивного обладнання на хостингу.

Крок 1. Налаштувати серверну частину з використанням Node.js/Express для хостингу статичних файлів сайту.

Всі пункти налаштування серверної частини відображені на рисунках 1.2 та 1.3.

Крок 2. Створити хмарну базу даних для збереження інформації про оформлені користувачем оренди обладнання.

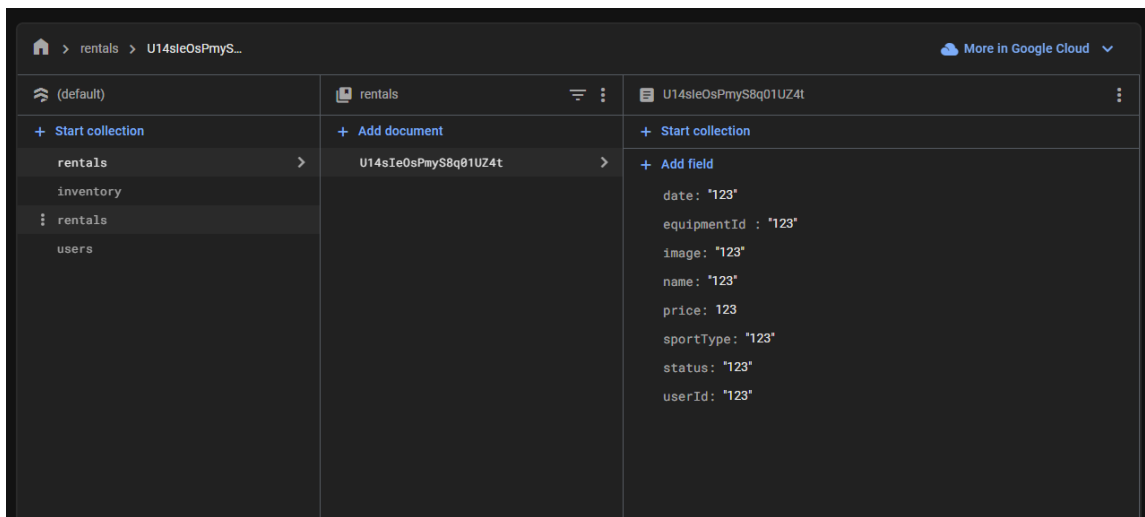


Рис. 4.1. Хмарна база даних Firestore.

Я взяв хмарну базу даних Firestore з попередньої лабораторної роботи. Я її уже підключав до свого проекту. І так як вона хмарна, то ніякого додаткового налаштування більше проводити не потрібно.

Крок 3-4. На сервері створити маршрут (HTTP GET) для отримання інформації про оформлені користувачем оренди обладнання відфільтровані по ціні. Використати створений маршрут у клієнтській частині для відображення інформації про оформлені користувачем оренди обладнання на сторінці "Мої оренди".

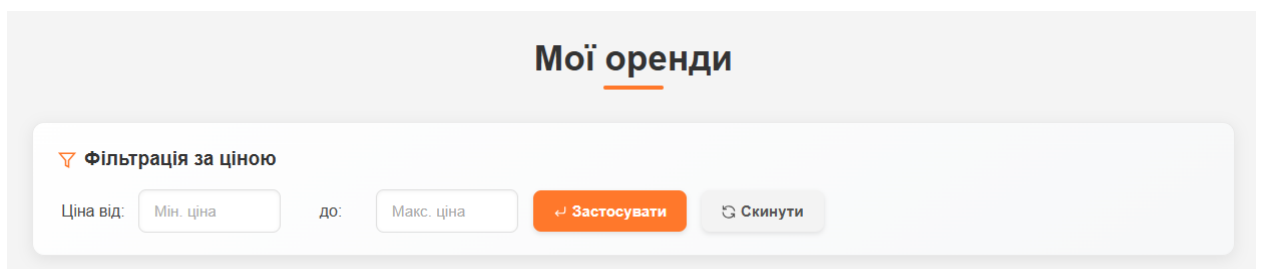


Рис. 4.2. Фільтрація за ціною.

Я створив можливість фільтрування усіх оренд за ціною, яка відбувається на серверній частині. Я зробив це за допомогою HTTP GET. Потім за допомогою HTTP POST я зберіг інформацію про оформлені користувачем оренди обладнання.

Крок 5. Розгорнути веб-сайт платформи для оренди спортивного обладнання на хостингу.

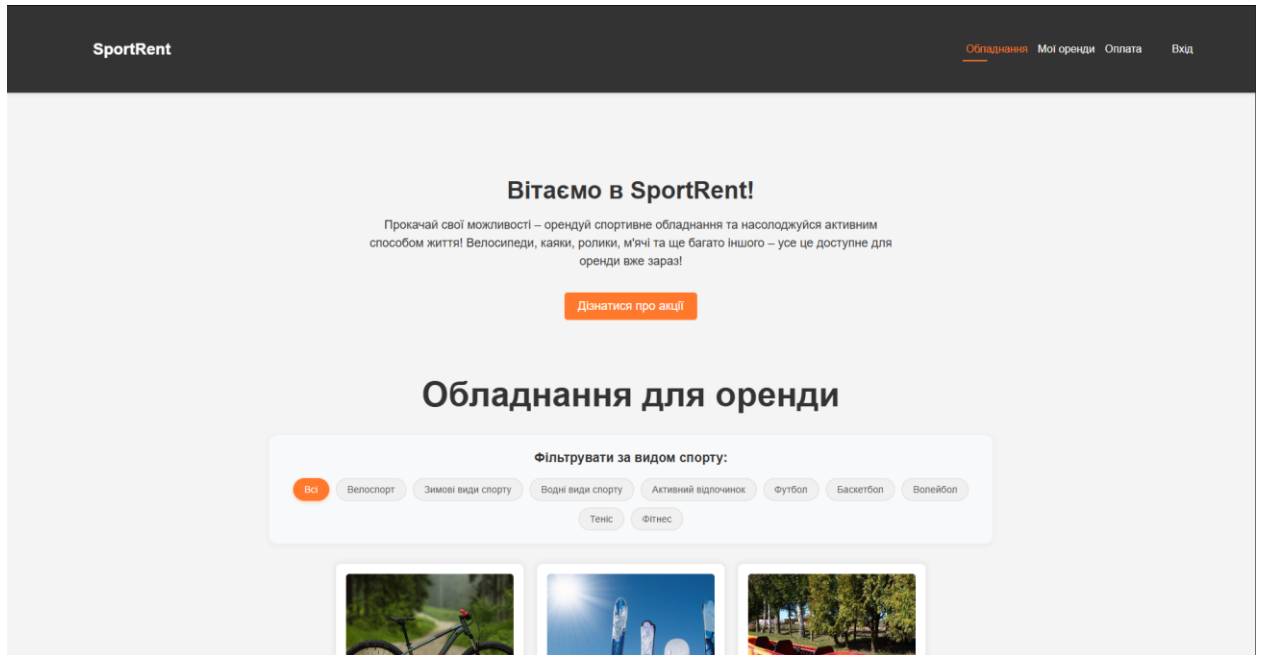
В кінці розробки я розгорнув свій веб-сайт на платформах Vercel та Render для клієнтської частини та серверної частини відповідно.

Висновок

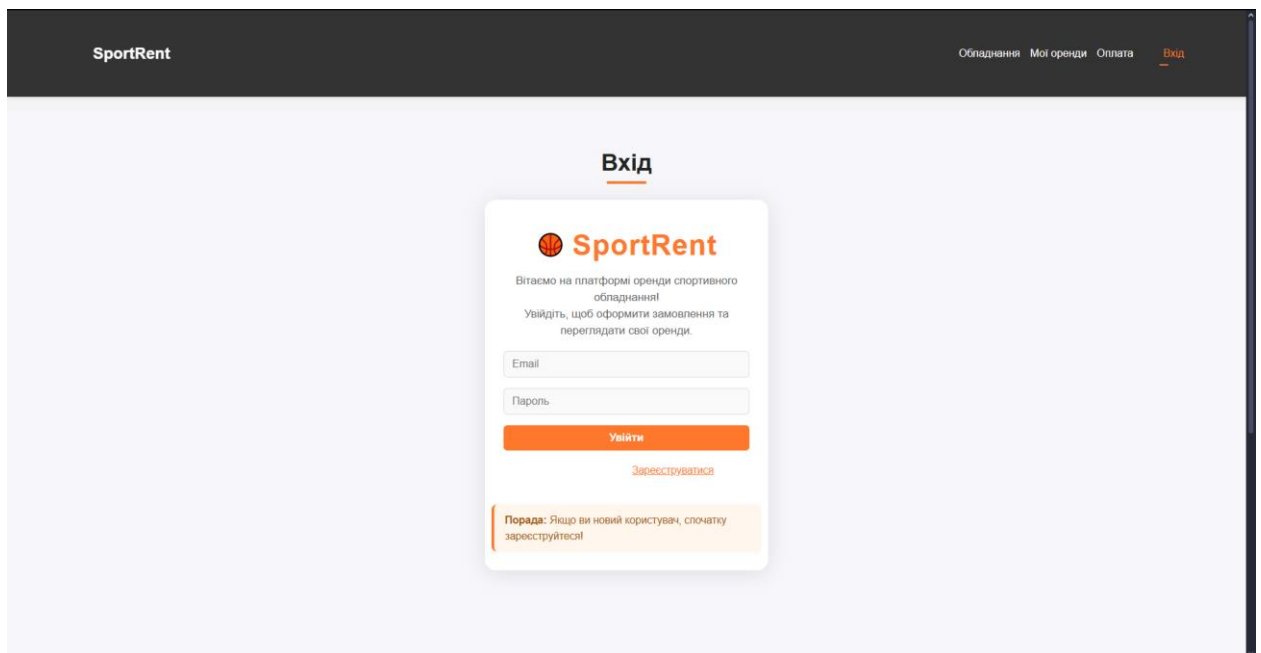
Отже, на цій лабораторній роботі я навчився розробляти серверну частину веб-додатків за допомогою Node.js та розгортати веб-додатки за допомогою хмарних сервісів.

Додатки

Додаток А (скріншот головної сторінки)



Додаток Б (скріншот сторінки входу)




Додаток В (скріншот сторінки реєстрації)

SportRent

ОбладнанняМои орендыОплатаВхід

Регістрація

 **SportRent**

Створіть акаунт, щоб орендувати спортивне обладнання швидко та зручно!

Зареєструватися

Увійти

Порада: Використовуйте надійний пароль для захисту свого акаунту!

Переваги реєстрації:


- Доступ до історії ваших оренд

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

SportRent

ОбладнанняМои орендыОплатаВхід

Мої оренди



Щоб переглядати свої оренди, увійдіть у акаунт

Увійти

Про нас

SportRent - оренда спортивного обладнання швидко та зручно!
Адреса: вул. Спортивна, 12, Київ, Україна

Контакти

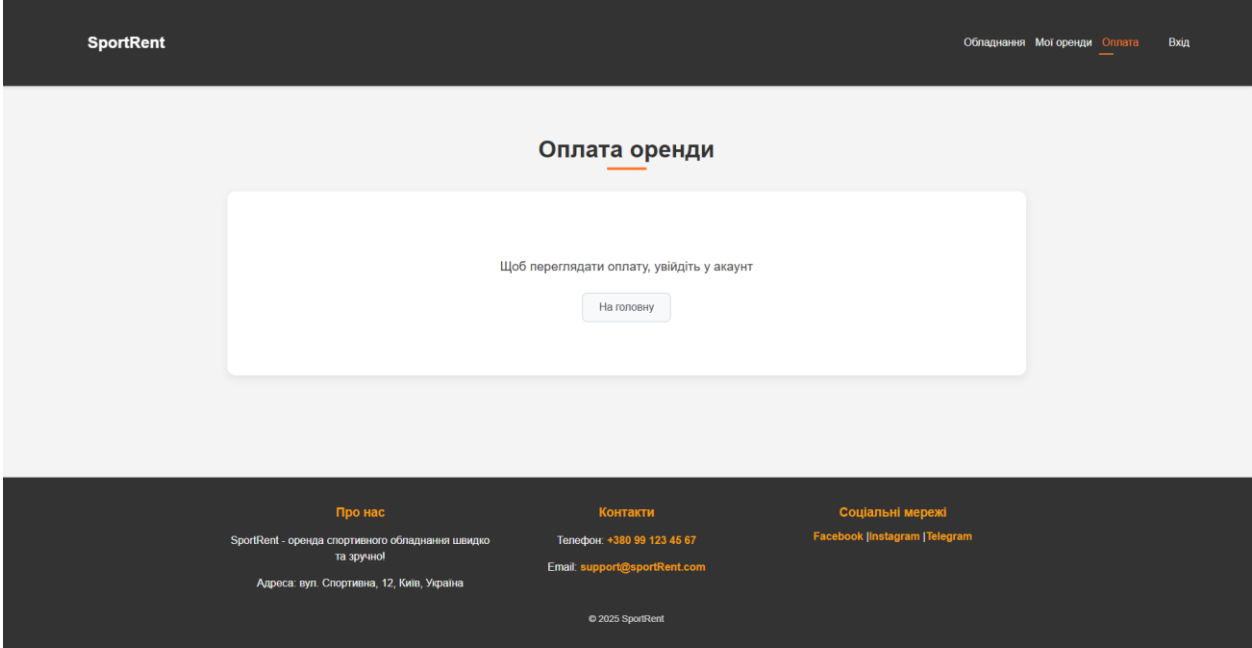
Телефон: +380 99 123 45 67
Email: support@sportRent.com

Соціальні мережі

[Facebook](#) | [Instagram](#) | [Telegram](#)

© 2025 SportRent

Додаток Г (скріншот сторінки “Оплата” без входу)



Додаток Д (скріншот сторінки “Мої оренди” з фільтрацією)

