A

PROJECT REPORT

ON

“House Rent

Application”

SUBMITTED IN FULFILLMENTS OF THE REQUIREMENTS

FOR THE FINAL YEAR OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE ENGINEERING

BY

ABISHEK

DERIN SHYLO

JEFFREY SAMSON R

KABILAN M

SRITHAR

**Purpose**

The **House Rent App** is designed to streamline the process of renting and leasing properties. Its purpose is to create an easy-to-use platform where landlords can list their properties, and tenants can search, view, and rent houses. The platform aims to simplify property management and enhance user experience for both tenants and landlords.

**Features**

* **User Authentication:** Secure registration and login for landlords and tenants.
* **Property Listings:** Landlords can list properties with details such as rent, location, size, and amenities.
* **Search and Filter:** Tenants can search for properties using filters like price range, location, and property type.
* **Favorites:** Tenants can bookmark properties for later review.
* **Booking Requests:** Tenants can send rental inquiries or booking requests.
* **Admin Dashboard:** Admin users can manage users, monitor activities, and resolve disputes.

**Architecture**

**Frontend**

The frontend is built using **React**, focusing on a responsive and user-friendly interface. It follows a component-based architecture for better reusability and scalability. State management is handled using **Redux**.

* **Technologies Used:** React, Redux, TailwindCSS/Bootstrap for styling, Axios for API calls.

**Backend**

The backend is developed using **Node.js** and **Express.js**, designed to handle API requests, authentication, and database interactions efficiently.

* **Technologies Used:** Node.js, Express.js, JWT for authentication, bcrypt for password hashing.

**Database**

The application uses **MongoDB** to store data. The database schema includes collections for:

* **Users:** Storing landlord and tenant data.
* **Properties:** Information about the listed properties.
* **Bookings:** Data related to tenant inquiries and rental requests.

**Setup Instructions**

**Prerequisites**

* Node.js
* MongoDB
* Git
* VS Code

**Installation**

1. Clone the repository:

bash

git clone https://github.com/username/house-rent-app.git

1. Navigate to the project directory:

bash

cd house-rent-app

1. Install dependencies for both client and server:

bash

cd client && npm install

cd ../server && npm install

1. Set up environment variables in .env files for both frontend and backend:
   * **Backend:**

makefile

PORT=5000

MONGO\_URI=mongodb+srv://<username>:<password>@cluster.mongodb.net/house-rent

JWT\_SECRET=your\_jwt\_secret

* + **Frontend:**

arduino

REACT\_APP\_API\_URL=http://localhost:5000

**Folder Structure**

**Client**

* **/src**
  + **components:** Reusable UI components (e.g., Navbar, Footer).
  + **pages:** Main application pages (e.g., Home, PropertyDetails).
  + **redux:** Redux store and slices for state management.

**Server**

* **/routes:** API routes (e.g., /api/properties, /api/auth).
* **/models:** Mongoose models for Users, Properties, and Bookings.
* **/controllers:** Logic for handling requests (e.g., PropertyController).

**Running the Application**

**Frontend**

Navigate to the client directory and run:

bash

npm start

**Backend**

Navigate to the server directory and run:

bash

npm start

**API Documentation**

**Endpoints**

**Authentication**

* **POST** /api/auth/register
  + Request: { username, email, password }
  + Response: { message: "User registered successfully" }
* **POST** /api/auth/login
  + Request: { email, password }
  + Response: { token, user }

**Properties**

* **GET** /api/properties
  + Response: [{ id, title, location, price, ... }]
* **POST** /api/properties (Landlord only)
  + Request: { title, description, location, price, amenities }

**Bookings**

* **POST** /api/bookings
  + Request: { propertyId, tenantId }

**Authentication**

The app uses **JWT (JSON Web Tokens)** for secure authentication and authorization. Tokens are issued at login and stored in the client as HTTP-only cookies. Passwords are securely hashed using **bcrypt** before storage.

**User Interface**

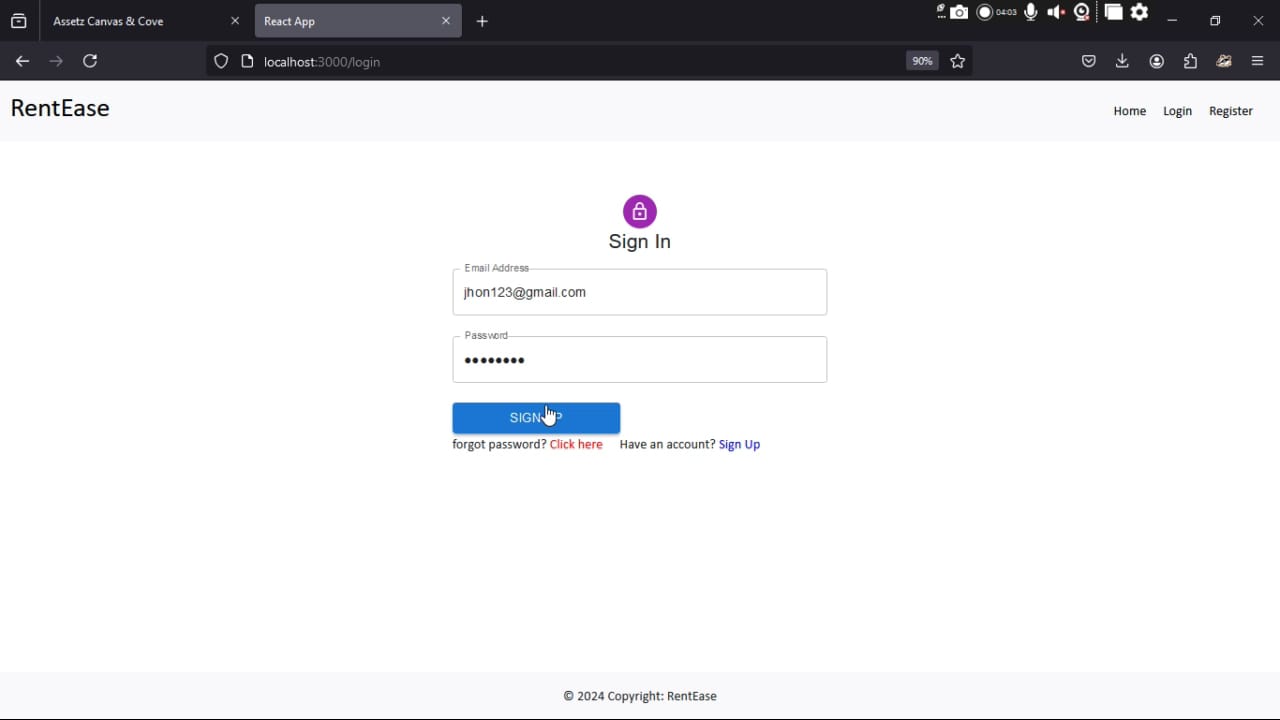
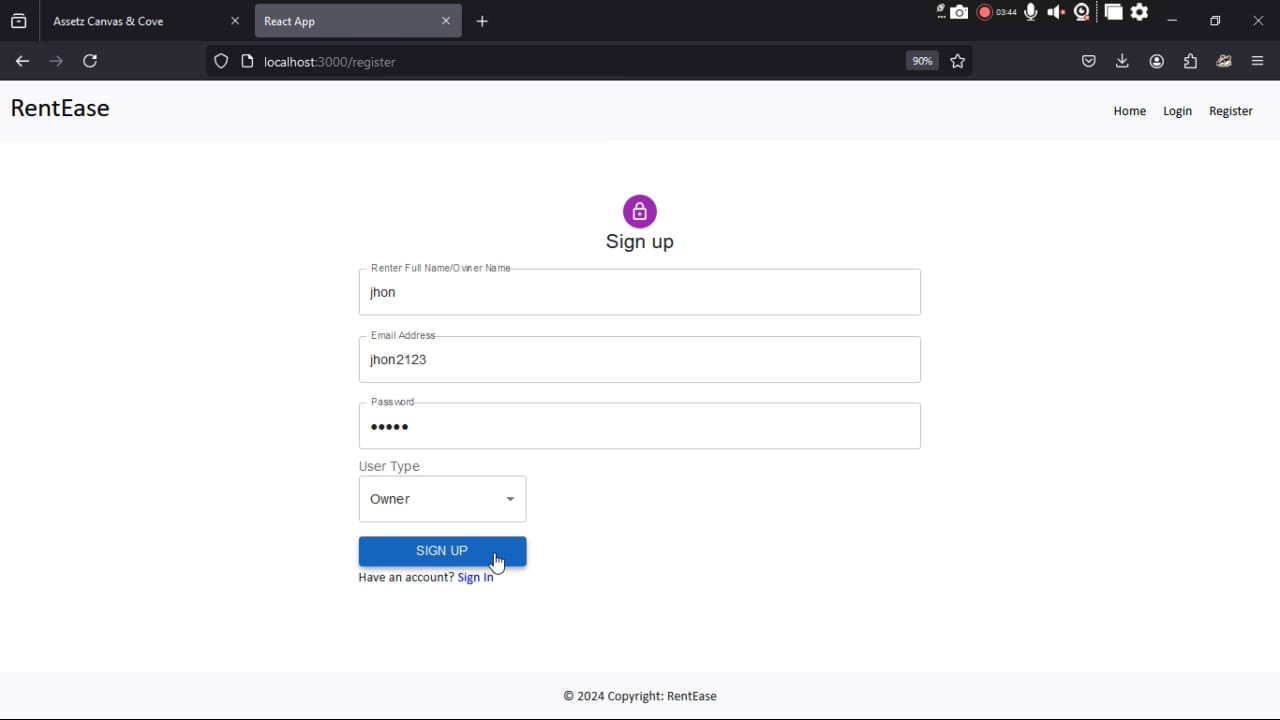
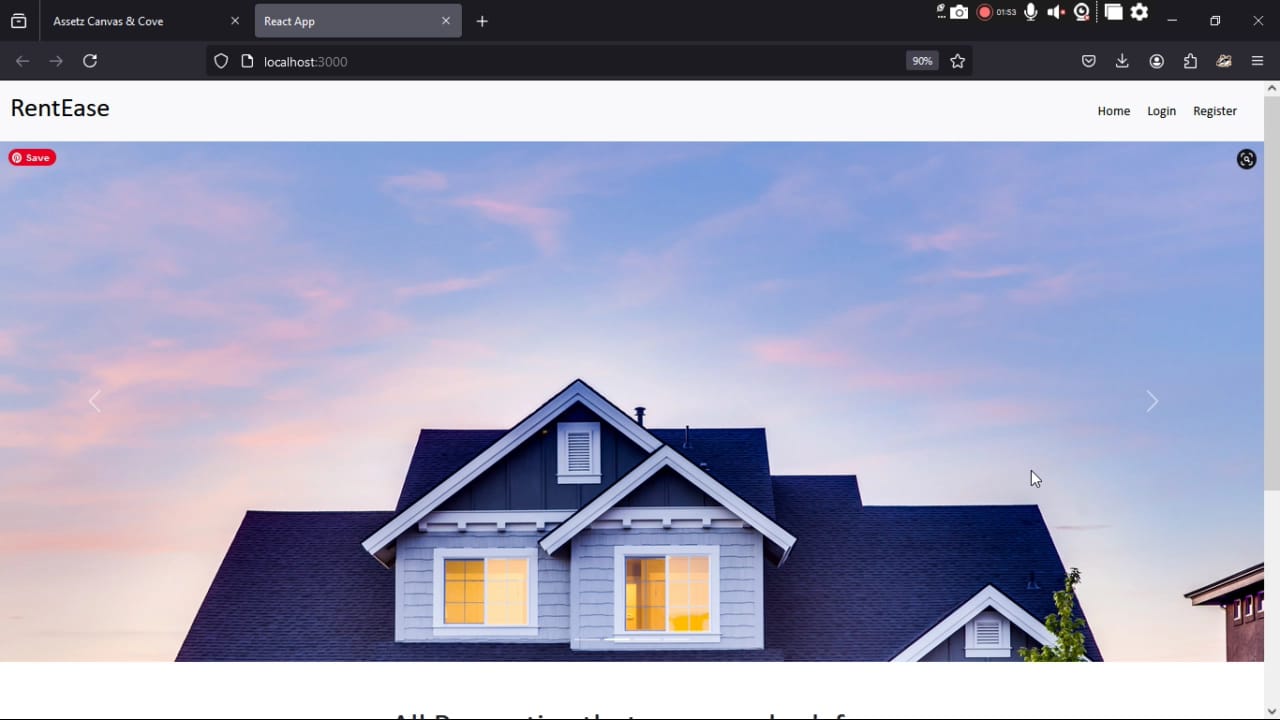
The user interface is designed for simplicity and clarity:

* **Homepage:** Displays a list of available properties.
* **Property Details Page:** Shows detailed information about a selected property.
* **Dashboard:** Separate dashboards for tenants and landlords.

**Testing**

* **Frontend Testing:** Performed using Jest and React Testing Library.
* **Backend Testing:** API tests are implemented using Mocha and Chai.

**Screenshots or Demo**



**Known Issues**

* Search functionality could be optimized for larger datasets.
* Booking requests lack real-time notifications.

**Future Enhancements**

* Add **real-time chat** between landlords and tenants.
* Integrate payment gateway for online rent payments.
* Mobile application using React Native.