**Full Stack Development with MERN**

**House Rent Application with MERN Stack**

**1. Introduction**  
**Project Title:** House Rent Application using MERN

**Team Members:**

* Deepika Sarathy R
* Devi Pavithra G
* Ramya
* Santhiya T

**2. Project Overview**  
**Purpose:** This project is designed to simplify the process of renting and managing house listings. It allows property owners to post rentals, tenants to browse available properties, and both parties to communicate effectively.

**Features:**

* User registration and login using JWT authentication.
* Property management (create, update, delete listings).
* Tenant dashboard for browsing and applying for properties.
* Admin dashboard for managing property listings and user activity.

**3. Architecture**  
**Frontend:** Built using React, featuring components for user authentication, property browsing, and messaging. React Router ensures smooth navigation.  
**Backend:** Developed with Node.js and Express.js, handling requests, authentication, and MongoDB integration for data management.  
**Database:** MongoDB stores user details, property listings, and communication logs, enabling efficient data operations.

**4. Setup Instructions**  
**Prerequisites:**

* Node.js v14+
* MongoDB v4+
* npm or yarn for package management

**Installation:**

* Clone the repository: git clone <repository-url>
* Install dependencies in both frontend and backend directories:
* Backend: cd backend && npm install
* Frontend: cd ../frontend && npm install
* Set up environment variables in the backend for database and JWT configuration.

**5. Folder Structure**  
**Frontend:**

* src/components - Reusable React components.
* src/pages - Pages for property listings, login, and dashboard.
* src/styles - CSS files for styling.

**Backend:**

* config - Database connection configuration.
* controllers - Handles business logic.
* routers - Defines API endpoints.
* middlewares - Authentication and error handling.
* schemas - MongoDB schemas and models.

**6. Running the Application**

Start the frontend server: npm start

Start the backend server: npm start

**7. API Documentation**  
The backend exposes API endpoints, such as:

* POST /api/auth/login - User login with JWT authentication.
* POST /api/auth/register - User registration.
* GET /api/properties - Fetch available property listings.

**Example Response:**

{

"status": "success",

"data": {

"properties": [

{

"id": "property-id",

"title": "Spacious 2BHK Apartment",

"location": "Chennai",

"price": 15000

}

]

}

}

**8. Authentication**  
JWT tokens are implemented for secure login and user session management. Tokens are stored in local storage, and protected routes verify token validity using middleware.

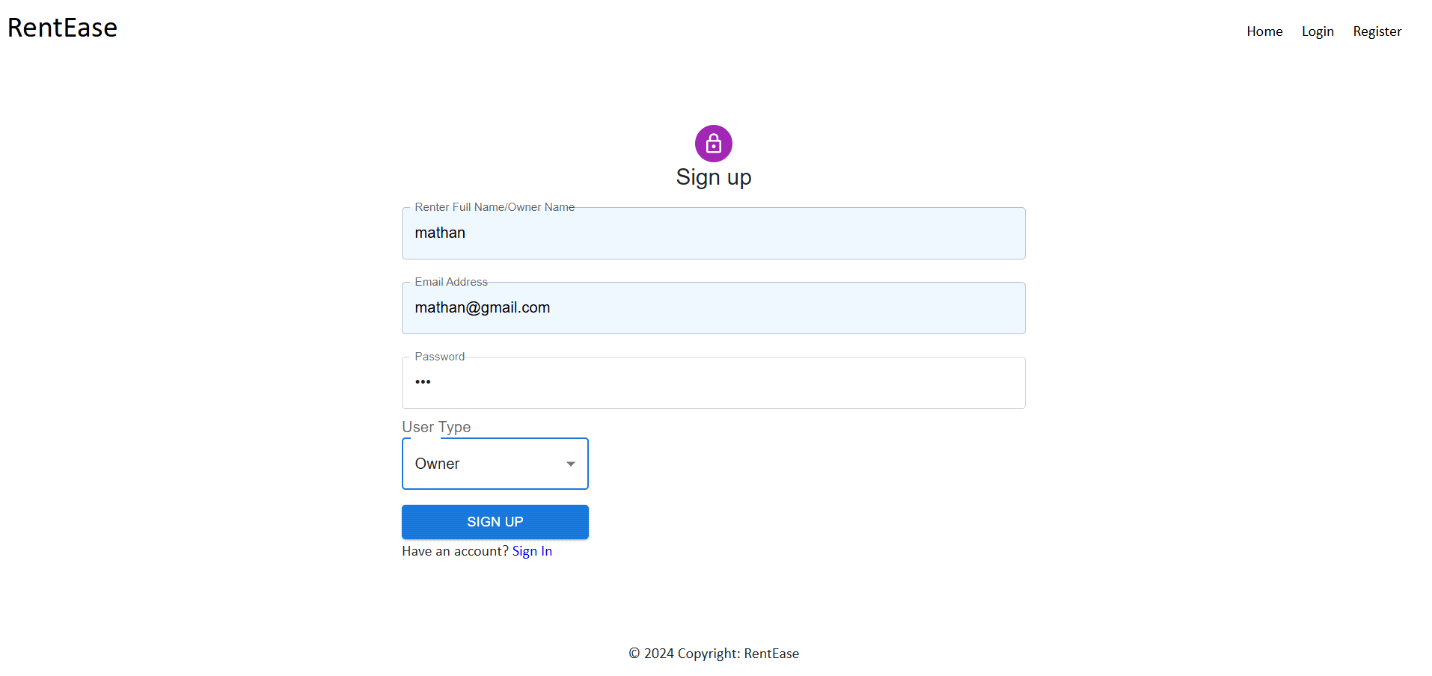
**9. User Interface**  
Includes the following:

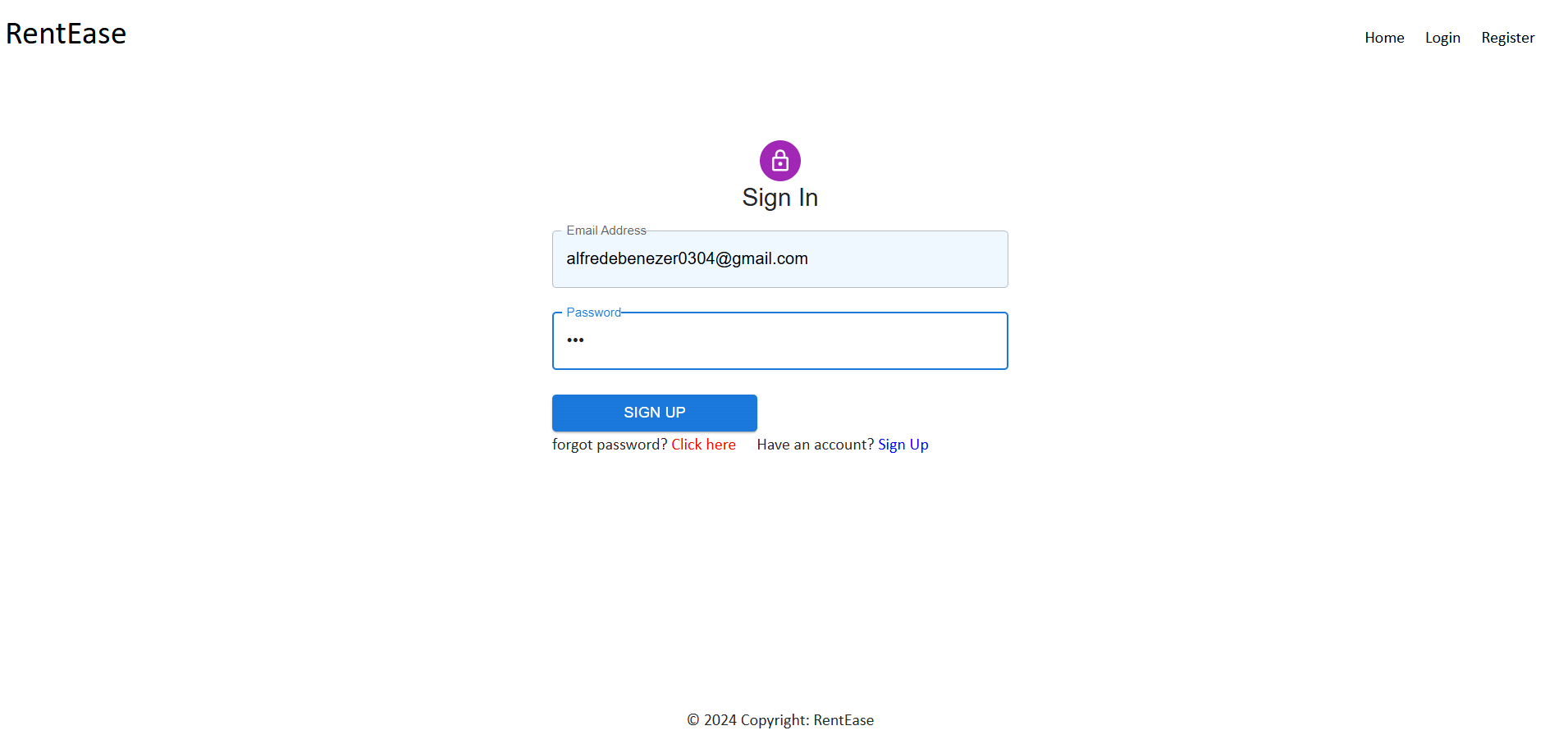
* Login and registration forms for tenants and property owners.
* Property listing page with filters for location, price, and property type.
* Dashboard for property owners to manage listings and for tenants to track applied rentals.

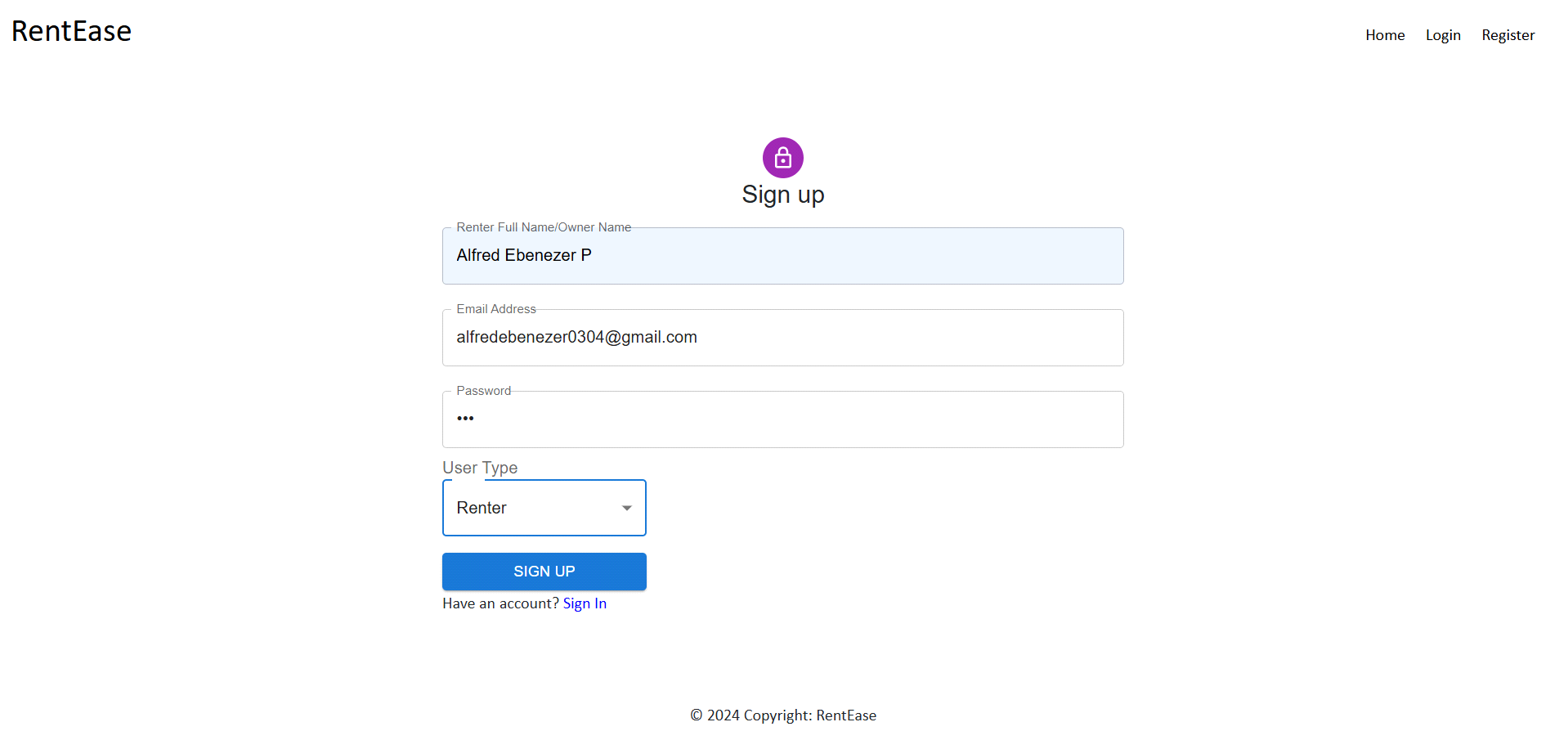
**10. Testing**  
Testing is performed using tools like Jest for unit tests and Postman for API validation. Tests cover API endpoints, authentication, and CRUD operations for properties.

* **11. Screenshots**

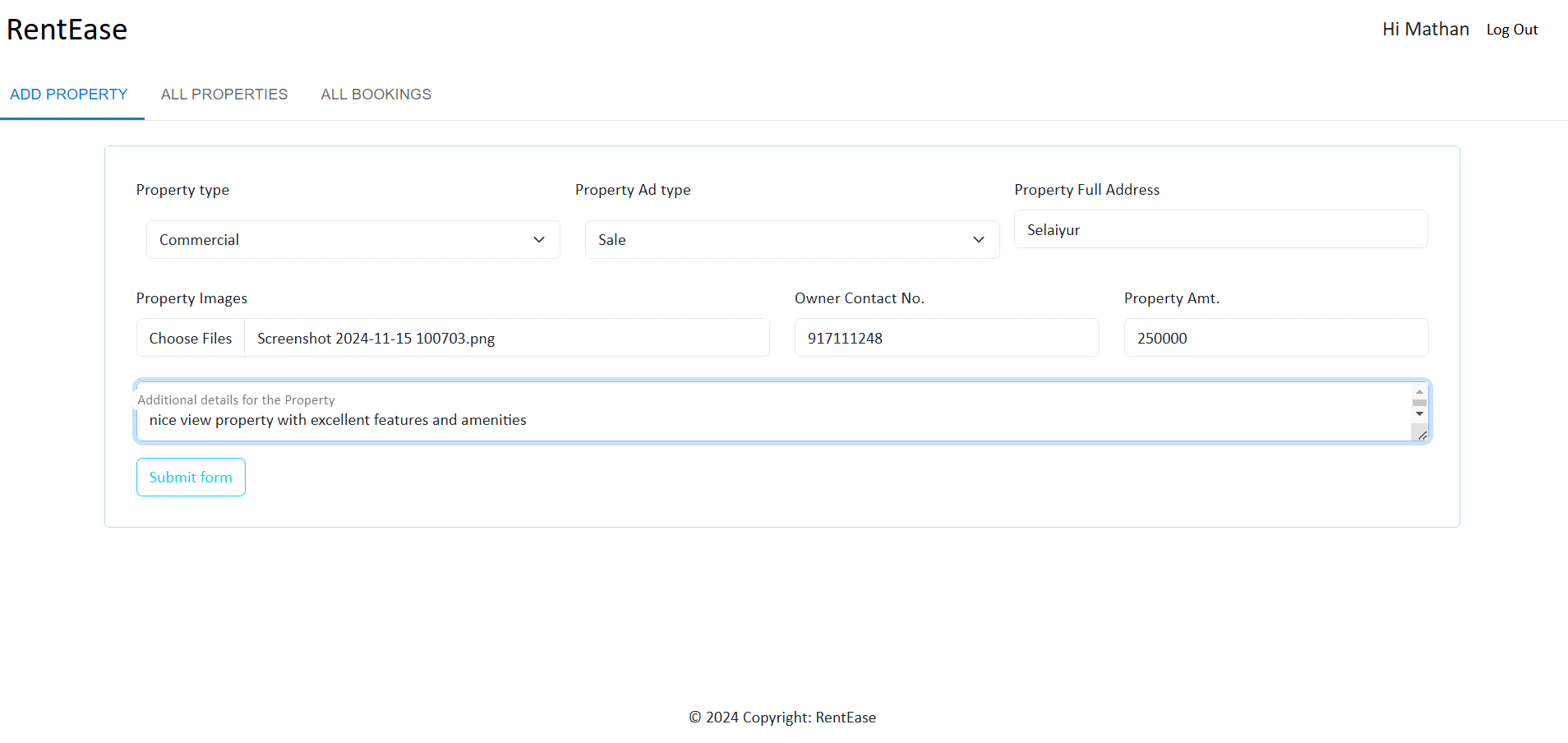
A Login/ Registration page



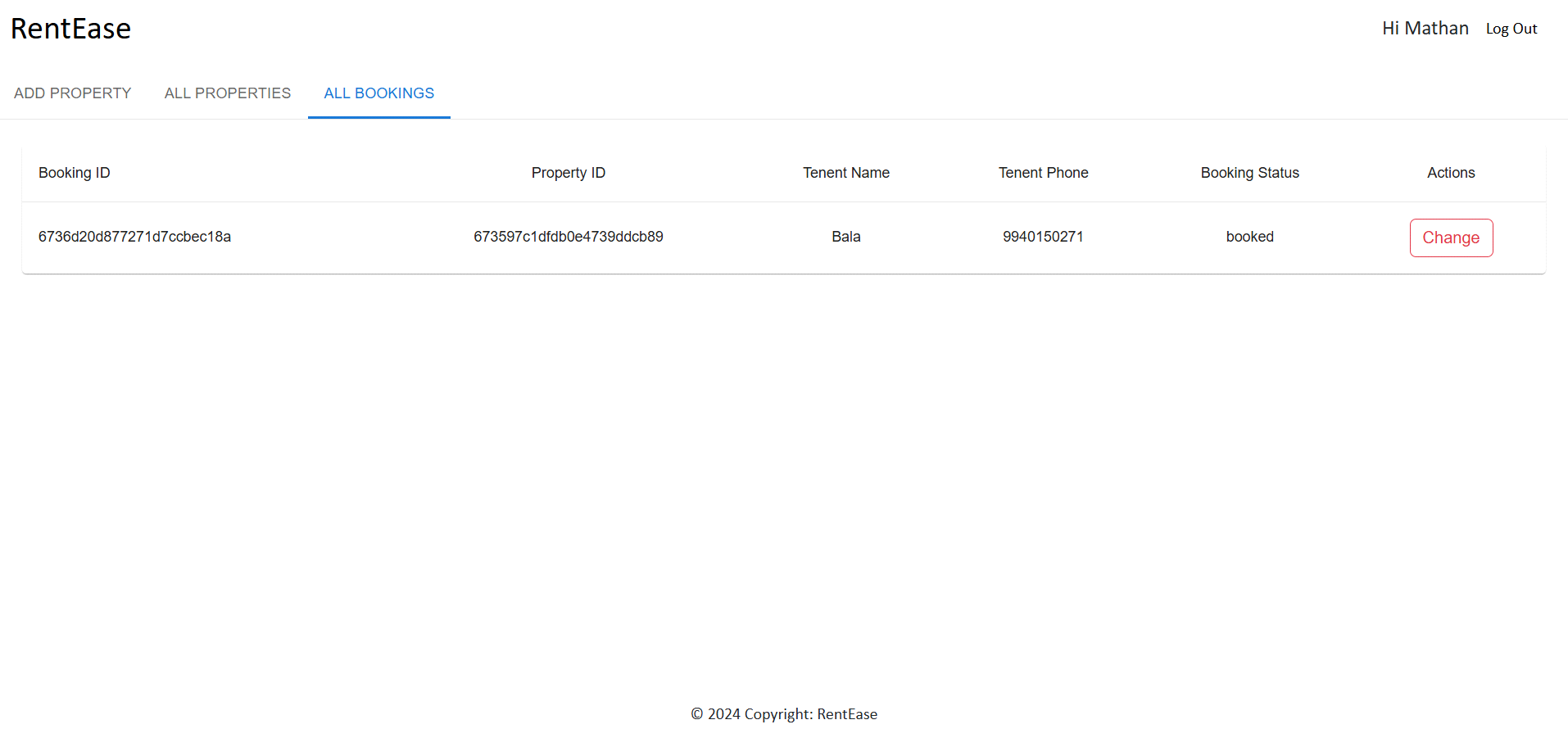




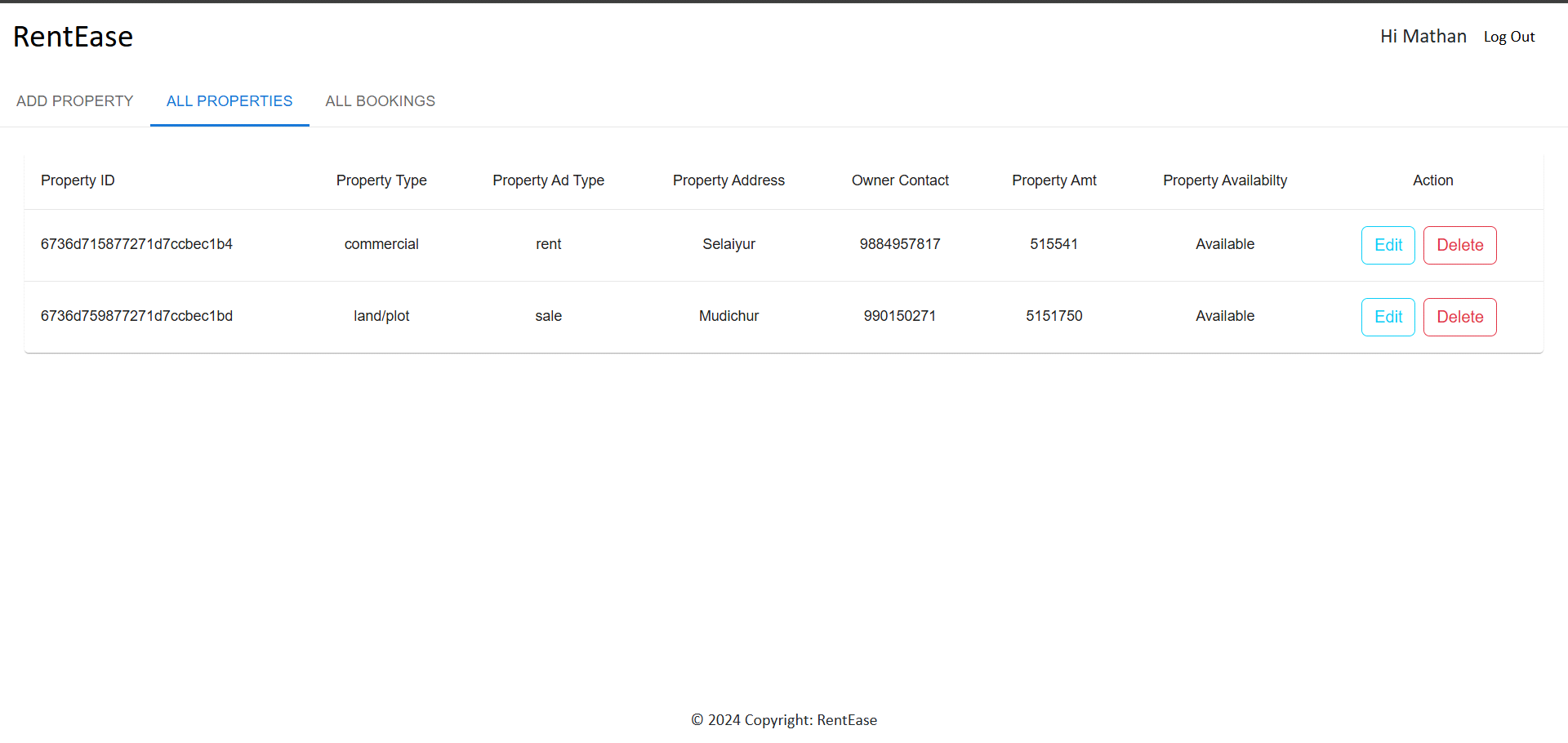
Add Property



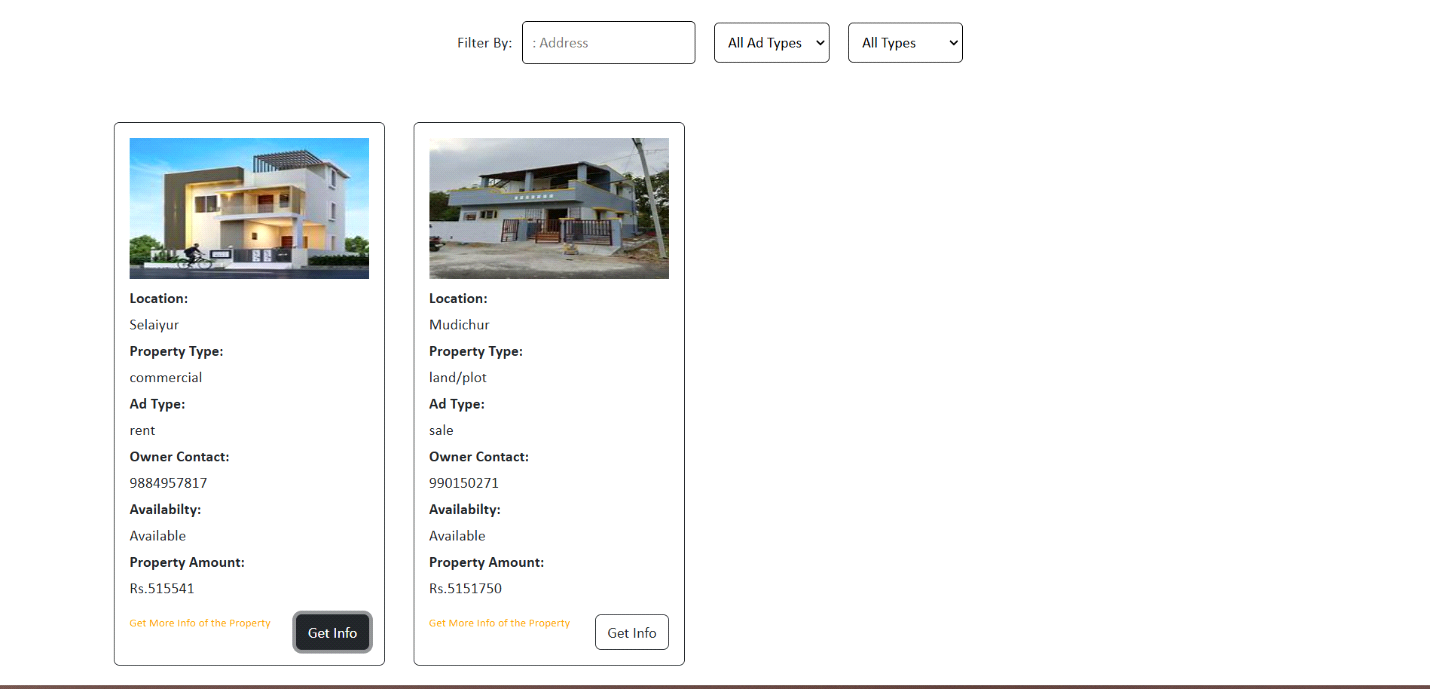
Add Bookings



All Properties



List of properties



**12. Known Issues**

* Token expiration may log out users unexpectedly.
* Occasional delays in updating property listings in real-time.

**13. Future Enhancements**

* Add features like payment integration for rent deposits.
* Enable virtual property tours and enhanced search filters.