Full Stack Development with MERN: Househunt Project Report

---

1. Introduction

- **Project Title:** Househunt: Finding Your Perfect Rental Home
- **Team Members:** Mahesh Kumar Reddy, Rohith Kumar, Bharat Kumar, Manjunath

2. Project Overview

Purpose:

- To provide a user-friendly platform for finding rental homes based on location, price, and preferences.
- Users can register, search, view details, and book rental houses easily.
- Owners can list their properties and manage bookings.
- The platform connects tenants and owners directly, removing the need for brokers.
- Efficient filtering and responsive design enhance usability across devices.

Features:

- Property search with filters (location, rent, rooms).
- User registration and login with JWT authentication.
- Owners can add, update, or delete property listings.
- Users can book properties and view booking status.
- Responsive design ensures usability on mobile and desktop.

3. Architecture

Frontend (React):

- Component-Based Design: Reusable components like Navbar, PropertyCard, Login, Dashboard.
- React Router Integration: Enables smooth navigation.
- State Management: `useState`, `useEffect`, Context API.
- Modular File Structure: `components/`, `pages/`, `services/`, `context/`.
- API Integration: Uses Axios to communicate with backend APIs.

Backend (Node.js & Express):

- Express.js handles routing and middleware.
- Modular Structure: `controllers/`, `routes/`, `models/`, `middlewares/`.
- MongoDB with Mongoose for schema definition.
- JWT for secure authentication.
- Multer for property image uploads.

Database (MongoDB):

- **Collections:** Users, Properties, Bookings.
- **User Schema:** name, email, password, role.
- **Property Schema:** ownerId, title, location, rent, description, images.
- **Booking Schema:** userId, propertyId, date, status.
- ObjectId relationships enable referencing between collections.

4. Setup Instructions

Prerequisites:

- Node.js and npm

- MongoDB or MongoDB Atlas
- ReactJS
- Postman or Web Browser
- VS Code and Git

Installation:

1. Clone the Repository:
   ```
   git clone https://github.com/Mahesh-7989/Househunt-Finding-Your-Perfect-Rental-Home.git
   ```

Install Frontend:

```
cd frontend
npm install
```

2.

Install Backend:

```
cd backend
npm install
```

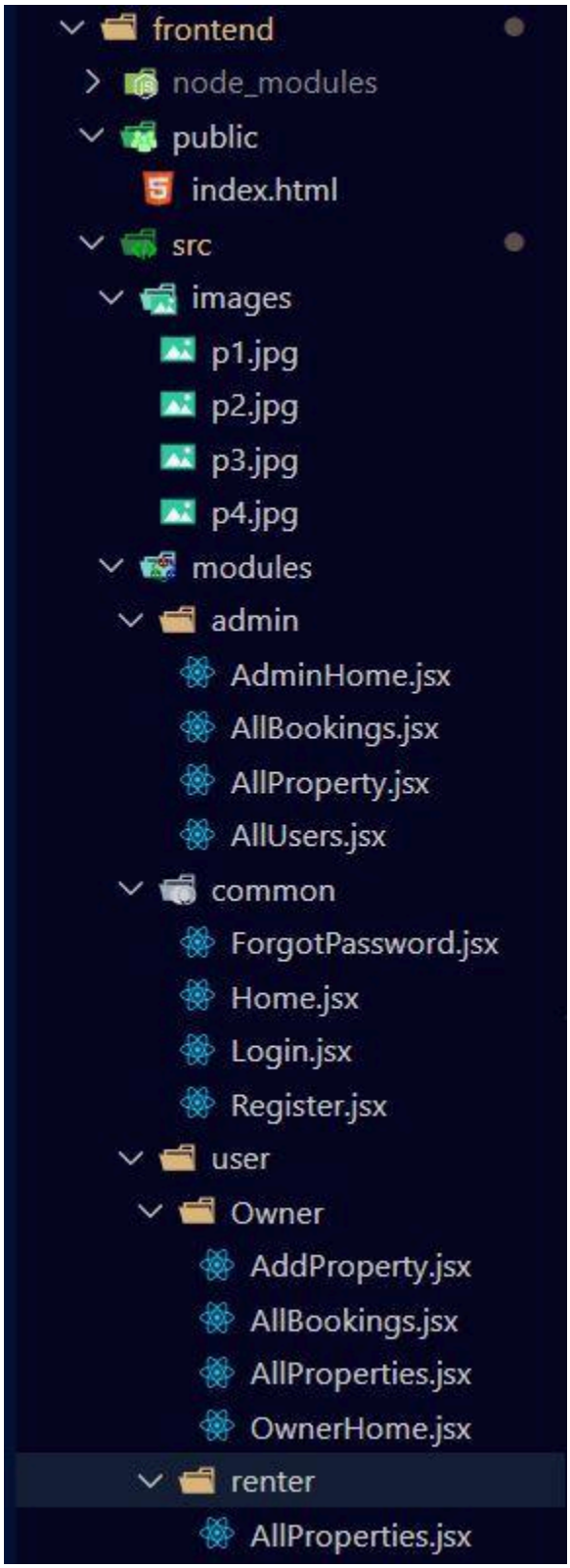
3.

Create `.env` in backend:

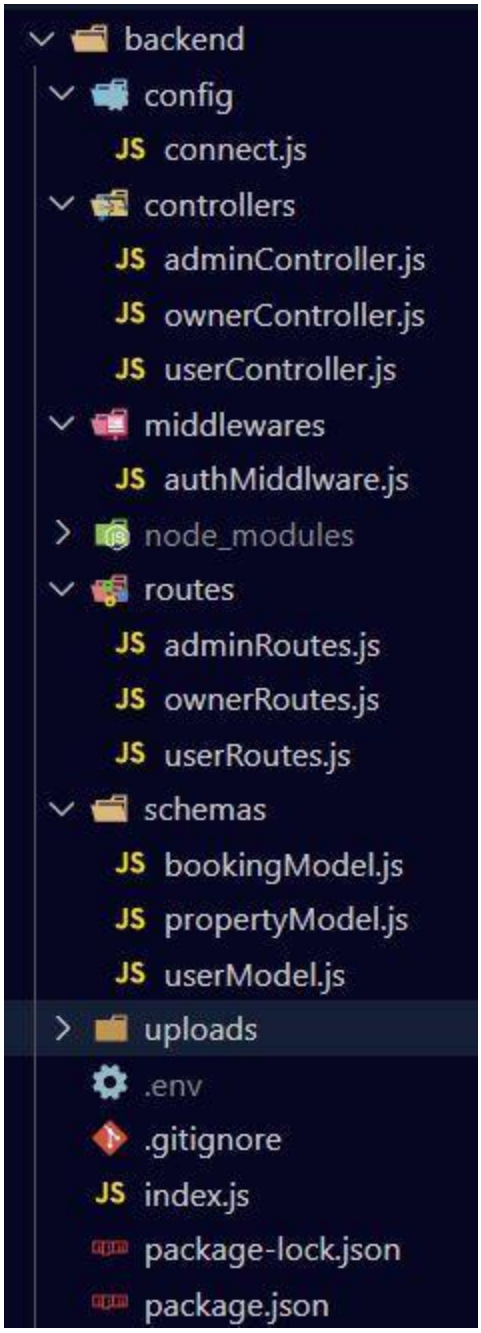
```
MONGO_URI=your_mongodb_connection_string
JWT_SECRET=your_jwt_secret
PORT=5000
```

4.
5. Run:
   - Backend: `npm start` (in backend folder)
   - Frontend: `npm start` (in frontend folder)

5. Folder Structure




RenterHome.jsx
AllPropertiesCards.jsx
App.css
App.js
M
index.js
.gitignore
package-lock.json
package.json
README.md



Client (React):

- Reusable components
- Pages directory (Home, Login, Dashboard)
- React Router for navigation

- Context API for global state
- Axios for API requests

Server (Node.js):

- Express-based routing
- Organized routes/controllers/models
- Middleware for auth and uploads
- Mongoose models for MongoDB integration

6. Running the Application

- **Frontend:** `npm start` inside `frontend/`
- **Backend:** `npm start` inside `backend/`

7. API Documentation

- **POST /api/auth/register** – User registration
- **POST /api/auth/login** – User login
- **POST /api/owner/add-property** – Add property
- **GET /api/owner/get-owner-properties** – List properties
- **DELETE /api/owner/delete-property/:id** – Delete property
- **PUT /api/owner/update-property/:id** – Update property
- **GET /api/owner/get-all-bookings** – Owner bookings
- **PUT /api/owner/booking-status/:id** – Approve/reject bookings
- **GET /api/user/search-properties** – Search properties

- **POST /api/user/book-property/:propertyId** – Book a property

8. Authentication

- JWT-based authentication
- Tokens stored in `localStorage`
- `Authorization` headers used in API calls
- Middleware checks token validity and role
- Stateless session
- Optional token expiry for added security

9. User Interface

- Clean UI with React and Bootstrap/MUI
- Responsive design across devices
- Separate views for users and owners
- Screenshot link: UI Demo

10. Testing

- Manual functional testing
- User Acceptance Testing (UAT)
- Test case documentation (positive + negative)
- Bug tracking via Excel/Sheets
- Postman used for API testing
- DevTools for debugging

- Mobile responsiveness tested
- Access control verified (JWT roles)
- Edge case and validation testing
- Performance monitored via response time

11. Known Issues

- Occasional image upload failure
- Filters reset on refresh
- Missing loading indicators
- Toast messages overlap/disappear quickly
- Logout redirect inconsistency
- No date validation for bookings
- No delete confirmation
- Limited mobile responsiveness on some pages
- Poor error handling on server crash
- Booking not reflected immediately in owner panel

12. Future Enhancements

- Property rating and reviews
- Advanced filters (Wi-Fi, pets, etc.)
- Real-time chat between user and owner
- Google Maps integration
- Booking calendar view

- Email/SMS notifications
- Admin dashboard
- Payment integration
- React Native mobile app
- Accessibility improvements and dark mode