Full Stack Development with MERN

Project Documentation format

1. Introduction

Project Title: HouseHunt - Finding Your Perfect Rental Home

Team Members & Roles:

S.No	Name	Role	Responsibilities
1	B Bhuvaneswari	Team Lead / Frontend Developer	React UI, UX, routing, responsive design
2	Atthuluru Charan	Backend Developer	API development, database integration with MongoDB
3	Athuluru Shreya Sree	Full Stack Developer	Overall coordination, frontend with React, backend with Node.js
4	Arepalli Vishnu Vardhan	Database Administrator	MongoDB schema design, indexing, backup

2. Project Overview

Purpose: HouseHunt is a MERN stack application that streamlines the process of finding, listing, and booking rental properties. It caters to renters, property owners, and admins for a seamless housing experience.

Features:

- User registration and login (email, Gmail)
- Role-based dashboards (renter, owner, admin)
- Add, view, update, and delete properties
- Booking requests and approval system
- Image upload via URL or local file
- Responsive UI for mobile and desktop

3. Architecture

Frontend: Built with React.js using functional components, hooks, React Router, and Axios for API communication. Context API is used for authentication state management.

Backend: Node.js and Express.js power the REST API. The architecture follows the MVC pattern with clearly separated routes, controllers, and models.

Database: MongoDB is used to store user details, properties, and booking records. Mongoose ODM handles data schema and validation.

4. Setup Instructions

Prerequisites:

- Node.js (v18+)
- MongoDB Community Edition or Atlas
- Git

Installation:

```
# Clone the repository
git clone https://github.com/yourusername/househunt.git
cd househunt

# Set up backend
cd server
npm install
cp .env.example .env  # Fill in variables like DB_URI, JWT_SECRET

# Set up frontend
cd ../client
npm install
```

5. Folder Structure

Client (Frontend):

```
client/
|-- node_modules
|-- public/
|-- src/
|   |-- api/
|   |-- components/
|   |-- pages/
|   |-- style/
|   |-- App.jsx
|   |-- index.js
```

Server (Backend):

```
server/
|-- config/
|-- controllers/
|-- middleware/
|-- models/
|-- node_modules
|-- routes/
|-- .env
|-- server.js
```

6. Running the Application

To run locally:

```
# In server directory
npm start

# In client directory
```

```
npm start
```

7. API Documentation

Example: **POST /api/users/login**

- Request Body: `{ email, password }`
- Response: `{ token, user }`

GET /api/properties

- Fetches all listed properties
- Requires: None (public)

More endpoints documented in Postman collection.

8. Authentication JWT-based authentication:

- On login, JWT is issued and stored in frontend context
- Protected routes require token verification via middleware
- Role-based access control for dashboards and actions

9. User Interface

- Clean, minimalistic UI
- Mobile responsive using Flexbox/Grid
- Screens include: Login, Register, Dashboard, Property Details, Add Property

10. Testing

- Manual testing using browser and Postman
- Chrome DevTools for performance & responsiveness
- Error simulation for testing fallback UI

11. Screenshots / Demo

- https://drive.google.com/file/d/1pMtQcqROzjHuy_zML-xa-EuE5pTtIpLL/view?usp=drive_link

12. Known Issues

- OTP-based email confirmation pending implementation
- No profile picture support yet
- Booking calendar integration is planned

13. Future Enhancements

- Admin dashboard with analytics
- Filter/search functionality for renters
- Payment gateway integration

- Google Maps integration for property location
- Progressive Web App (PWA) version