Full Stack Development with MERN

House Hunt

**1. Introduction**

**Project Title:** House Hunt — Finding Your Perfect Rental Home

**Team Members:**

Katta Abhilash— Frontend Developer

Kare Amith Josh— Full Stack Developer & Database Specialist

Karra Geethika— Project Coordinator & Backend Developer

Karre Himakar— UI/UX Designer

**2. Project Overview**

🏡 House Hunt – Final Year Project Documentation

✨ Project Overview

🎯 Purpose

Finding a rental home is never easy — too many listings, scattered platforms, and often, lack of reliable information. That’s where House Hunt comes in! 🏠💡

The goal of this project is to create a simple, easy-to-use platform that connects people looking for rental homes with property owners. Whether you're a student, a working professional, or a family looking to relocate — House Hunt provides a centralized and smart way to browse, filter, and find your perfect home. ✅

It’s built to be efficient, mobile-friendly, and scalable — ensuring a smooth experience for both tenants and owners.

🌟 Core Features

🔐 Secure user registration and login with authentication

🏘️ Add & manage property listings (for owners)

🔍 Advanced property search with filters (location, rent, availability, property type)

📸 Image preview for properties

💬 Contact owner button (basic communication support)

🧑‍💼 Admin access to manage users and properties

📱 Fully responsive UI for mobile and desktop

🧱 Architecture Overview

🖥️ Frontend (React.js + Tailwind CSS)

Dynamic, component-based structure using React

Clean UI designed with Tailwind CSS

Pages:

Home

Search Results

Login/Register

Owner Dashboard

Property Details Page

⚙️ Backend (Node.js + Express.js)

RESTful APIs built using Express.js

Middleware for authentication and error handling

Controllers and routes separated for maintainability

🗄️ Database (MongoDB)

Stores users, properties, admin logs, and image URLs

Uses Mongoose models for validation and schema

🧰 Setup Instructions

📦 Prerequisites

Make sure the following are installed on your system:

✅ Node.js & npm

✅ MongoDB or MongoDB Atlas account

✅ Git & VS Code

📁 Folder Structure

house-hunt/

│

├── client/ # Frontend (React)

│ ├── src/

│ └── public/

│

├── server/ # Backend (Node + Express)

│ ├── models/

│ ├── routes/

│ ├── controllers/

│ └── index.js

│

├── .env # For MongoDB URI, JWT secret

└── README.md

▶️ Running the Application

🔧 Frontend

cd client

npm install

npm start

🔧 Backend

cd server

npm install

node index.js

✅ Ensure MongoDB is running and the .env file has your DB connection string and secret keys.

📡 API Documentation

🔑 Authentication APIs

POST /api/auth/register → Registers a new user

POST /api/auth/login → Logs in and returns a token

🏘️ Property APIs

GET /api/properties → List all properties

POST /api/properties → Add a new property

GET /api/properties/:id → View single property

👤 User APIs

GET /api/users/:id → Get user info

DELETE /api/users/:id → Remove user (admin only)

🔒 Authentication

Authentication is handled using JWT (JSON Web Tokens) 🛡️. Once a user logs in, a token is issued and stored in localStorage, which is then used to access protected routes (like posting a property or viewing owner dashboard).

Middleware in the backend verifies this token before allowing any sensitive action.

🎨 User Interface

The UI was designed keeping real users in mind. It’s responsive, visually clean, and intuitive to use 📱💻

🖼️ Properties displayed with thumbnail + key info

🧭 Navigation is simple with a sticky navbar

💬 User feedback for actions like “Property added” or “Login failed”

✨ Dark mode support (optional)

🧪 Testing

✅ Manual Testing

Each feature tested on Chrome, Firefox, and mobile devices

🧪 Tools Used

Postman for testing backend APIs

React Testing Library for key components

Console testing for error handling and edge cases

📸 Screenshots

> (Add the following screenshots to your document or presentation):

Home Page

Property Listings

Property Details Page

Login/Register Screen

Owner Dashboard

🧩 Known Issues

🔍 Search is limited to simple filters (e.g., rent, city)

🗂️ Images are currently stored as URLs, not cloud-uploaded

👥 No advanced roles yet (admin panel is basic)

🧠 No ML-based recommendations yet (future goal)

🚀 Future Enhancements

📍 Google Maps API integration for precise location

🧑‍🤝‍🧑 Role-based access (admin, agent, tenant, owner)

💬 In-app messaging between owner and user

📊 Admin dashboard with analytics

📤 Upload property images via Cloudinary or Firebase

🧠 Smart search with auto-suggestions and personalization

📱 Launch mobile app version with Flutter