

Full Stack Development with MERN

Project Documentation

1. Introduction

- **Project Title:** HouseHunt: Finding Your Perfect Rental Home
- **Team Members:** Syed Shameem(Project setup& Configuration,Backend)
Pannangi Praneetha (Database and Frontend)
Vankadari Meghana(Project Implementation &Exeution)

2. Project Overview

- **Purpose:**

The purpose of **House Hunt** is to create a simple and efficient web-based platform that allows users to **search for rental or sale homes** based on their preferred **location, budget, and number of bedrooms**. It helps users find the most suitable homes without physically visiting each one.

Goals:

1. **User-Friendly Interface:** Provide an easy-to-use interface for users to search for houses.
2. **Dynamic Search:** Enable users to filter homes based on criteria like location, price, and bedroom count.
3. **Database Management:** Store and retrieve house listings using a MySQL database.
4. **Real-Time Results:** Display matching house listings immediately upon search.
5. **Efficient Access:** Help users find their ideal home faster, reducing manual effort and time.

- **Features:**



. Property Search Functionality

- Users can search houses by **location, price, and number of bedrooms**.
- Real-time results are displayed based on search criteria.



. Property Details View

- Each house listing provides:
 - **Location**
 - **Price**
 - **Number of bedrooms**
 - **Detailed description**

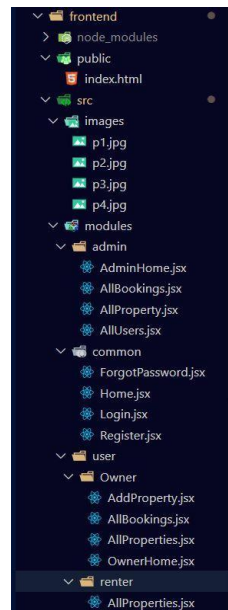


Responsive and Simple Design

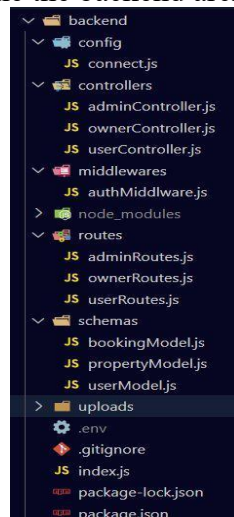
- Built with **HTML, CSS, and JavaScript** for a clean, mobile-friendly interface.
- Easy navigation for users of all skill levels.

3. Architecture

- **Frontend:** Frontend architecture using React.



- **Backend:** Outline the backend architecture using Node.js and Express.js.



- **Database:** Detail the database schema and interactions with MongoDB
- Schema-less flexibility for changing house data structures.
- Easily handles image URLs, tags, and nested data.
- Fast querying with find(), sort(), and limit().

4. Setup Instructions

- **Prerequisites:** List software dependencies (e.g., Node.js, MongoDB)

Node.js Runs the backend server & npm

npm Package manager for dependencies npm

MongoDB NoSQL database to store house listings

MongoDB Compass GUI for managing and viewing MongoDB data Latest

React.js Frontend framework for building UI

Express.js Web framework for Node.js backend

Mongoose ODM (Object Data Modeling) for MongoDB

Installation: Step-by-step guide to clone, install dependencies, and set up the environment variables.

Step 1: Clone the Project


```
git clone https://github.com/your-username/house-hunt.git
cd house-hunt
```

Step 2: Project Structure Overview

```
house-hunt/
├── client/      # React Frontend
├── server/      # Node.js + Express Backend
└── README.md
```

Step 3: Backend Setup (Node.js + MongoDB)

```
cd server
npm install
```

 Create .env File in /server Directory:

```
PORT=5000
MONGO_URI=mongodb://localhost:27017/househunt
```

Step 4: Frontend Setup (React)

```
cd ../client
npm install
```

Step 5: Start the Servers

● Start the Backend:

```
cd ../server
npm start
```

By default, the backend runs at: <http://localhost:5000>

● Start the Frontend:

```
cd ../client
npm start
```

By default, the frontend runs at: <http://localhost:3000>

Step 6: Test the App

1. Open your browser and visit: <http://localhost:3000>
 2. Use the search filters to find houses.
 3. The frontend will call the backend API and fetch data from MongoDB.
-

Step 7: Verify MongoDB is Connected

You should see a log in the backend terminal:
MongoDB Connected Successfully
Server running on port 5000

5.Folder Structure

- **Client:** Describe the structure of the React frontend.

client/

```
├── public/           # Static files
│   └── index.html
├── src/              # Source code
│   ├── assets/       # Images and icons
│   ├── components/   # Reusable UI components
│   │   ├── NavBar.js
│   │   ├── SearchBar.js
│   │   ├── HouseList.js
│   │   ├── HouseCard.js
│   │   └── Footer.js
│   ├── pages/        # Page-level components
│   │   ├── Home.js
│   │   └── HouseDetails.js
│   ├── App.js        # Main App component (routes and layout)
│   ├── index.js       # App entry point
│   └── styles/       # Optional: CSS or Tailwind setup
│       └── app.css
└── package.json

└── .env              # Environment variables (e.g., API URL)
```

- **Server:** Explain the organization of the Node.js backend.

server/

```
├── models/           # Mongoose schemas
│   └── House.js
├── routes/           # API route handlers
│   └── houseRoutes.js
├── controllers/      # Route logic separated from routes
│   └── houseController.js
├── config/           # Configuration files
│   └── db.js         # MongoDB connection logic
├── .env              # Environment variables
├── server.js         # Main entry point
└── package.json
```

6. Running the Application

Provide commands to start the frontend and backend servers locally.

- **Frontend:** `npm start` in the client directory.
 `cd client`
 `npm install`
 `npm start`
- **Backend:** `npm start` in the server directory.
 `cd server`
 `npm install`
 `npm start`

7. API Documentation

- Document all endpoints exposed by the backend.

Method	Endpoint	Description	Parameters
GET	/api/houses	Get all/filter houses	location, price, bedrooms
POST	/api/houses	Add a new house	Body (JSON)
GET	/api/houses/:id	Get a house by ID	URL param :id
PUT	/api/houses/:id	Update house by ID	URL param + body
DELETE	/api/houses/:id	Delete house by ID	URL param

- Include request methods, parameters, and example responses.

GET

```
[
  {
    "_id": "667c91c19b9e2c123abc4567",
    "location": "Delhi",
    "price": 280000,
    "bedrooms": 2,
    "description": "Spacious 2BHK apartment with modern amenities.",
    "image_url": "https://example.com/delhi-house.jpg",
    "posted_date": "2025-06-25T10:30:00.000Z"
  }
]
```

DELETE

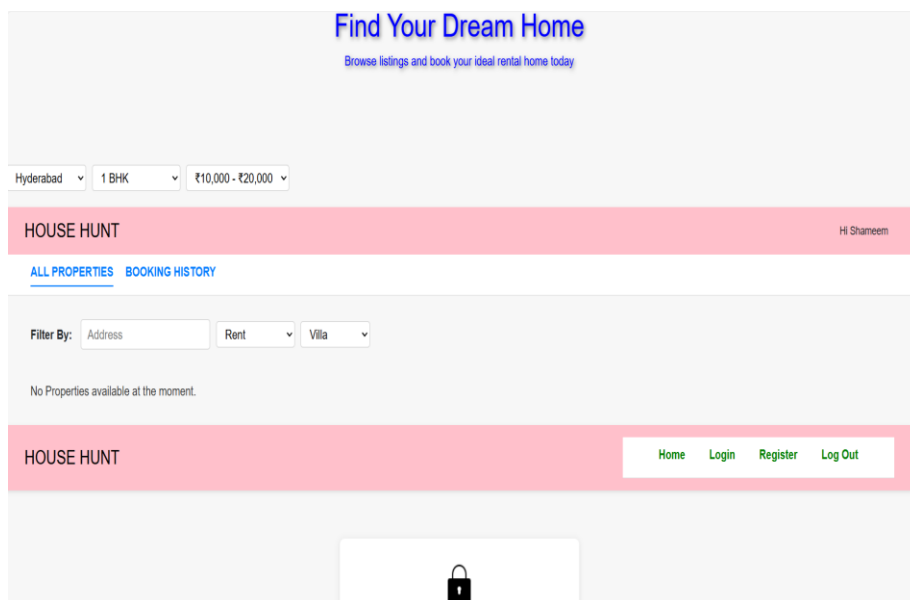
```
{
  "message": "House deleted successfully"
}
```

8.Authentication

- Explain how authentication and authorization are handled in the project.

Aspect	Method
Authentication	JWT (stateless, signed tokens)
Authorization	Role-based (user, admin)
Token Storage	LocalStorage or HTTP-only cookie
Protected Routes	Middleware authenticate, authorizeAdmin
Password Security	Bcrypt hashing

9.User Interface



10.Testing

Tools used.

- VS Code / Sublime Text
- HTML
- CSS
- JAVA SCRIPT
- NODE.js
- MONGODB

11.Screenshots or Demo

- A link to a demo to showcase the application.

https://drive.google.com/file/d/1RWDc8DtD_d0ZfhR5ZEcEHnF361YZnQBi/view?usp=drivesdk

12.Known Issues

- Document any known bugs or issues that users or developers should be aware of.

Issue: Some pages or images don't scale well on smaller screens.

Impact: Poor user experience on mobile devices.

Solution: Use responsive CSS (e.g., Bootstrap or media queries).

13.Future Enhancements

- Outline potential future features or improvements that could be made to the project.

Description: Show property location using Google Maps API.

Benefit: Helps users visualize where the property is located.