Project Documentation:

HouseHunt Rental App

1. Introduction

This document outlines the development procedure for the "HouseHunt" rental application, a MERN stack-based solution designed to facilitate property rentals. The application connects renters with property owners and is managed by an administrator.

1.1 Purpose

The purpose of the HouseHunt app is to provide a convenient and efficient platform for users to find rental properties, apartments, or houses.

1.2 Key Features

The HouseHunt application includes the following key features:

- **Property Listings:** A database of available rental properties with detailed descriptions, photos, location, and rent amount.
- **Search Filters:** Users can narrow down search results based on criteria like location, rent range, property type, number of bedrooms, and amenities.
- **Contact Landlords/Property Managers:** Users can directly contact property owners or managers via messaging or email through the app.
- **User Registration:** Supports registration for different user types (Renter, Owner).
- **Property Inquiry & Booking:** Renters can inquire about properties and send booking requests. Owners can approve booking requests.
- **Admin Approval:** Administrators review and approve new owner registrations to allow them to add properties.
- **Owner Management:** Approved owners can add, edit, and delete their properties, and update their status and availability.
- **Platform Governance:** The admin monitors activities and ensures adherence to platform policies, terms of service, and privacy regulations.
- **Transaction and Lease Agreement:** Facilitates negotiation and finalization of rental contracts and payment details within the app's messaging system.

2. Technical Architecture

The HouseHunt app follows a client-server model, leveraging the MERN stack for its components.

2.1 Frontend (UI)

- **Technology:** React.js.

- **Libraries:** Axios for connecting with the backend via RESTful APIs , Bootstrap and Material UI for a responsive and enhanced user interface , Ant Design for UI components , and Moment.js for date/time manipulation.
- **Functionality:** Handles user interface and presentation, displaying property listings, search filters, forms, and user-specific dashboards.

2.2 Backend

- **Technology:** Node.js.
- **Framework:** Express.js for server-side logic and API handling.
- **Database:** MongoDB for efficient and scalable storage of user data, property details, and booking information.
- **Libraries:**
    - cors: For enabling Cross-Origin Resource Sharing.
    - bcryptjs: For password hashing and security.
    - dotenv: For managing environment variables.
    - mongoose: An ODM (Object-Document Mapping) library for MongoDB interaction.
    - jsonwebtoken: For creating and verifying JWTs for authentication.
    - multer: For handling file uploads (e.g., property images).
    - nodemon: For automatically restarting the Node.js server during development.

2.3 Database Design (ER Diagram)

The database comprises three main collections:

- **users:** Stores user information including _id, name, email, password, and type (e.g., Renter, Owner, Admin).
- **property:** Stores details about rental properties, including _id, userID (foreign key to owner), prop.Type, prop.AdType, isAvailable, prop.Address, owner contact, prop.Amt, prop.images, and add.Info.
- **booking:** Records booking requests with _id, propertyId, userId, ownerId, and userName.

3. Project Setup and Configuration

3.1 Prerequisites

To set up and run the HouseHunt application, ensure the following are installed:

- **Node.js and npm:** Required for running JavaScript on the server-side and managing packages. Download from

https://nodejs.org/en/download/.

- **MongoDB:** A NoSQL database to store application data. Download from

https://www.mongodb.com/try/download/community.

- **Git:** For cloning the repository (if not manually downloaded).
- **Basic knowledge of HTML, CSS, and JavaScript:** Essential for understanding the structure, styling, and client-side interactivity of the app.

3.2 Project Installation Steps

1. **Download the Project:** Obtain the project code from the provided Google Drive link: https://drive.google.com/drive/folders/10OstrbGEPKtXDENGkerKBShejPJIbVgj?usp=sharing. Extract the ZIP file to your desired local directory.
2. **Navigate to Project Directory:** Open your command prompt or terminal and navigate to the extracted project folder (e.g., cd house-rent).
3. **Install Backend Dependencies:**
    - Navigate into the

backend folder: cd backend.

    - Install the required Node.js packages:

npm install.

        - *Dependencies include:* cors, bcryptjs, express, dotenv, mongoose, jsonwebtoken, multer, nodemon.

4. **Configure Backend Environment Variables:**
    - In the backend folder, create a file named .env.
    - Add the following lines to the .env file (replace placeholders as needed):
    - MONGO_DB=mongodb://localhost:27017/househunt
    - JWT_SECRET=your_secret_jwt_key
    - PORT=5000
        - MONGO_DB: Connects to your MongoDB instance.
        - JWT_SECRET: Used for authentication.
        - PORT: The port on which the backend server will run.
5. **Start the Backend Server:**
    - While still in the

backend folder, run: npm start.

    - This will start the Node.js server, and you should see a "Connected to MongoDB" message in your terminal. Keep this terminal window open.
6. **Install Frontend Dependencies:**
    - Open a *new* command prompt or terminal window.
    - Navigate back to the main project directory and then into the frontend folder:

Bash

cd ../frontend

- Install the required React.js packages:

npm install.

  - *Dependencies include:* react, react-dom, react-router-dom, axios, bootstrap, moment, antd, @mui/material, mdb-react-ui-kit, react-bootstrap.

7. **Start the Frontend Development Server:**

   - While in the

frontend folder, run: npm start.

   - This will launch the React application in your default web browser, typically at

http://localhost:3000.

4. Application Flow and Roles

The application supports three main roles: Renter, Owner, and Admin.

4.1 Renter/Tenant

- **Registration & Login:** Creates an account and logs in using email and password.
- **Browse Properties:** Views available properties on the dashboard with detailed descriptions, photos, and rental information.
- **Search & Filter:** Applies filters to narrow down search results based on location, rent range, and number of bedrooms.
- **Property Inquiry:** Clicks on a property to view details and owner contact information, then fills out a form to send inquiry details to the owner.
- **Booking Status:** Views booking status (initially "pending") in their dashboard, which is updated by the owner.
- **Move-in Process:** Marks the completion of the rental process after moving into the apartment.

4.2 Owner

- **Account Approval:** Signs up for an Owner account and awaits admin approval.
- **Property Management:** After approval, can add, edit, and delete properties in their account.
- **Property Status:** Updates the status and availability of their properties based on occupancy.
- **Booking Confirmation:** Receives renter inquiries and approves booking requests.
- **Lease Agreement:** Negotiates lease terms and finalizes rental contracts and payment details through the app's messaging system.

4.3 Admin

- **User Approval:** Reviews and approves legitimate user requests to become "Owner" accounts, allowing them to add properties to the app.
- **Platform Governance:** Implements and enforces platform policies, terms of service, and privacy regulations.
- **Activity Monitoring:** Monitors activities to maintain a safe and trustworthy environment for all users.

5. Project Implementation Details

5.1 Backend Development

- **Server Setup:** Creation of index.js file in the backend folder to define port number, MongoDB connection string, and JWT key in a .env file, and configure CORS and body-parser.
- **Authentication:** Implementation of authMiddleware.js in the middlewares folder for project authentication.
- **Database Configuration:**
    - Importing Mongoose for MongoDB connection.
    - Database connection established from

config.js.

    - Creation of a

model folder to store database schemas for users, properties, and bookings.

5.2 Frontend Development

- **Tool Installation:** Installation of React, Bootstrap, Material UI, Axios, Moment, Antd, MDB React UI Kit, and React-Bootstrap.
- **User Interface:** Development of user-facing components for entering booking information, checking booking status, and admin dashboards. Usage of Material UI and Bootstrap for enhanced UI.

6. Project Execution

After completing the development, the application is run to verify all functionalities and check for bugs.

- **Access:** The application is accessible at http://localhost:3000 once both frontend and backend servers are running.
- **User Interface Examples:** Screenshots are provided for the register/sign-up page, login page, admin panel, owner panel, and tenant panel.