**House rent app using MERN**

**1. Introduction**

**Project Title: House Rent Application**

**Team Members: Indris P, Immanuel Joshua P, Lingam S, Manoj Kumar L**

**2. Project Overview**

**Purpose:**

* **The House Rent App aims to streamline the rental process by connecting property owners with potential tenants. It provides a platform for users to browse, list, and manage rental properties efficiently.**

**Features:**

* **User authentication and authorization.**
* **Property listing with images, descriptions, and pricing.**
* **Search and filter functionality for finding properties.**
* **User profiles for landlords and tenants.**
* **Real-time messaging between landlords and tenants.**

**3. Architecture**

**Frontend:**

* **Built using React, the frontend handles the user interface, routing, and interactions with backend APIs.**

**Backend:**

* **Node.js and Express.js are used for server-side logic, API creation, and handling requests.**

**Database:**

* **MongoDB stores user data, property details, and messages.**

**4. Setup Instructions**

**Prerequisites:**

* **Node.js**
* **MongoDB**
* **NPM/Yarn**

**Installation:**

* **Clone the repository.**

**Bash: git clone [repository URL]**

* **Navigate to the project directory.**

**Bash:cd house-rent-app**

* **Install dependencies for both client and server.**
* **Bash:**
  + **cd client && npm install**
  + **cd ../server && npm install**
  + **Set up environment variables (.env) for both frontend and backend.**

**5. Folder Structure**

* **Client:**
  + **/src: Contains React components, routes, and styles.**
  + **/public: Public assets like images and static files.**
* **Server:**
  + **/routes: API endpoints for users, properties, and messages.**
  + **/models: Mongoose schemas for MongoDB collections.**
  + **/controllers: Business logic for API endpoints.**

**6. Running the Application**

* **Frontend:Navigate to the client folder and start the React app.**

**Bash: npm start**

* **Backend: Navigate to the server folder and start the Node.js server.**

**Bash: npm start**

**7. API Documentation**

* **Endpoints:**
  + **User Authentication:**
    - **POST /api/users/login**
      * **Body: { email, password }**
      * **Response: { token, user }**
* **Property Management:**
  + **GET /api/properties**
    - **Query Params: { location, priceRange }**
    - **Response: List of properties.**
* **POST /api/properties**
  + **Body: { title, description, price, images }**
  + **Response: Newly created property.**

**8. Authentication**

* **Method:**
  + **JWT (JSON Web Tokens) is used for authentication.**
  + **Passwords are hashed using bcrypt.**
* **Flow:**
  + **Users log in or sign up to receive a token.**
  + **Protected routes require a valid token.**

**9. User Interface**

* **Key Screens:**
  + **Login/Sign-Up**
  + **Dashboard (for browsing properties)**
  + **Details Page**
  + **Profile Management**
  + **Add/Edit Property Form**

**10. Testing**

* **Tools:**
  + **Jest and React Testing Library for frontend.**
  + **Mocha and Chai for backend.**
* **Strategy:**
  + **Unit tests for components and API endpoints.**
  + **Integration testing for end-to-end functionality.**

**11. Screenshots or Demo**

* **Screenshots:**
  + **Add screenshots of the main pages (e.g., Home Page, Property Details).**
* **Demo Link:**
  + **Provide a link to the hosted application or a video demo.**

**12. Known Issues**

* **Potential performance issues with large datasets.**
* **Missing payment integration for booking properties.**

**13. Future Enhancements**

* **Adding payment gateway integration.**
* **Implementing AI-based recommendations for properties.**

**14. Enabling multi-language support.**

* **Feel free to update the content**