Full Stack Development with MERN

Project Documentation format

1. Introduction

- **Project Title:** House Hunt
- **Team Members:**
  - **L.**Abhishek: Team Lead and Front-end
  - **L.**Chaitanya: Front-end development
  - **K.**Rama: Back-end development
  - **G.**Bhargav:Back-end development

2. Project Overview

- **Purpose:**

HouseHunt is an online platform designed to streamline the rental process by connecting landlords and tenants efficiently. The platform enables landlords to list their properties with comprehensive details, while providing tenants with an intuitive search experience to find properties that match their preferences.

Goals:

  - **Simplify Property Listing:** Allow landlords to easily list their properties with all relevant details, including rent, location, amenities, and photos.
  - **Enhance Search Experience:** Provide tenants with advanced search filters to find properties based on their specific needs and preferences.
  - **Facilitate Communication:** Enable tenants to contact landlords directly through the platform for inquiries or bookings, ensuring smooth and efficient communication.
- **Features:** Highlight key features and functionalities.

  - **Sign up:** Create an account to access personalized features and manage property listings.
  - **Login:** Securely log in to your account to view and manage your profile and interactions.
  - **Search listings:** Browse available rental properties using a comprehensive search tool.
  - **Filters for searching:** Apply filters such as location, price range, amenities, and property type to refine your search results.
  - **View listings:** View detailed property information, including photos, descriptions, rent, and amenities.
  - **Contacting property owners:** Directly message landlords for inquiries, schedule viewings, or book properties.
  - **Post the properties:** Landlords can easily list their properties by providing necessary details and uploading photos.

3. Architecture

- **Frontend: React**

1. **Component-Based Structure:**
    - The frontend is built using React, organized into reusable components such as `Header`, `Footer`, `PropertyCard`, `SearchBar`, and `FilterPanel`.
    - Each component handles a specific part of the UI, making the code modular and easy to manage.
2. **State Management:**
    - Uses React's built-in state management with hooks like `useState` and `useEffect`.
    - For complex state management, integrates Redux or Context API to handle global state.
3. **Routing:**
    - Uses React Router for client-side routing, enabling smooth navigation between pages like Home, Sign-Up, Login, Search Results, and Property Details.
4. **API Integration:**
    - Communicates with the backend via RESTful APIs using Axios or Fetch for operations like user authentication, fetching property listings, and posting new properties.

- **Backend: Node.js and Express.js.**

    - **Express Server:**

        - The backend is built using Node.js and Express.js, creating a robust server to handle API requests and serve static files.
        - Routes are defined to handle different endpoints such as `/signup`, `/login`, `/properties`, and `/contact`.

    - **Middleware:**

        - Uses middleware for tasks like parsing JSON bodies, handling CORS, logging requests, and managing authentication with JWT.

    - **Authentication:**

        - Implements JWT (JSON Web Tokens) for secure user authentication and authorization, ensuring that only logged-in users can access certain endpoints.

    - **Error Handling:**

        - Centralized error handling middleware to catch and manage errors consistently across the application.

    -
- **Database:** Detail the database schema and interactions with MongoDB.

    - **Database Schema:**

- The database schema is designed using Mongoose, an ODM (Object Data Modeling) library for MongoDB.
- **User Schema:** Includes fields like `username`, `email`, `passwordHash`, and `role` (landlord or tenant).
- **Property Schema:** Includes fields like `title`, `description`, `rent`, `location`, `amenities`, `photos`, `ownerId`, and `createdAt`.

- **CRUD Operations:**

  - Provides CRUD (Create, Read, Update, Delete) operations for both user and property data.
  - **Create:** Allows landlords to post new properties.
  - **Read:** Enables tenants to fetch property listings and view details.
  - **Update:** Allows landlords to edit property details.
  - **Delete:** Allows landlords to remove listings.

- **Data Relationships:**

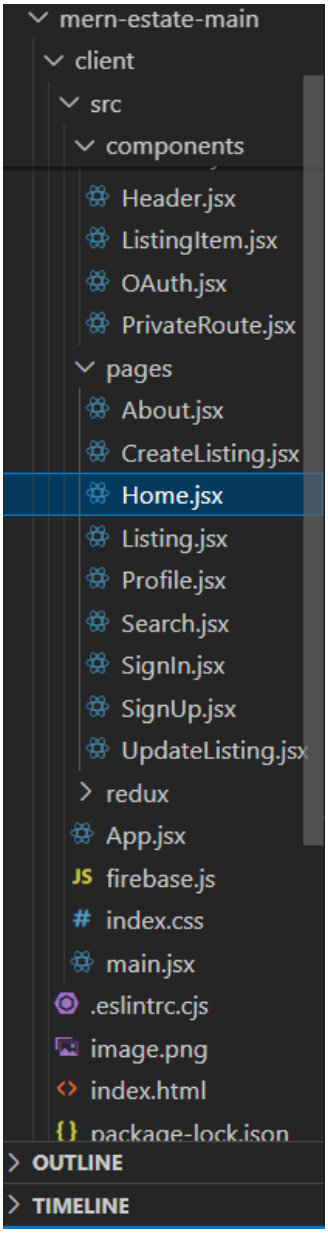
  - Establishes relationships between users and properties, linking `ownerId` in the Property schema to the `User` schema.
  - Uses population methods to fetch related data, such as getting landlord details when retrieving property listings.

4. Setup Instructions

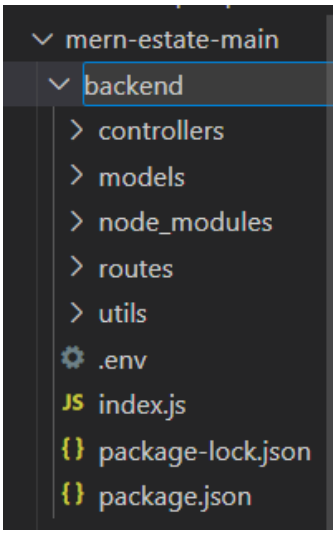
- **Prerequisites:** Software dependencies are React,Node js,Express js and Mongo DB
- **Installation:**
  - Install the all the software dependencies.
  - Set up the environment variable.
  - Establish connection with MONGO DB.

5. Folder Structure

- **Client:**


mern-estate-main
client
src
components
Header.jsx
ListingItem.jsx
OAuth.jsx
PrivateRoute.jsx
pages
About.jsx
CreateListing.jsx
Home.jsx
Listing.jsx
Profile.jsx
Search.jsx
SignIn.jsx
SignUp.jsx
UpdateListing.jsx
redux
App.jsx
firebase.js
index.css
main.jsx
.eslintrc.cjs
image.png
index.html
package-lock.json
OUTLINE
TIMELINE

- **Server:**


mern-estate-main
backend
controllers
models
node_modules
routes
utils
.env
index.js
package-lock.json
package.json

6. Running the Application

- Provide commands to start the frontend and backend servers locally.
    - **Frontend:** `npm run dev`.
    - **Backend:** `nodemon index.js`

7. API Documentation

Endpoint: `/api/houses`

Description: Get a list of available houses for rent.

Request:

- **Method:** GET
- **Parameters:** None

Example Request: `get/api/houses`

Example Response:

- **Status:** 200 OK
- **Content-Type:** application/json

```
json
[
  {
    "id": 1,
    "title": "Cozy Apartment",
    "price": 1500,
    "location": "City Center",
    "description": "A comfortable apartment close to amenities.",
    "image": "https://example.com/images/apartment.jpg"
  },
]
```

8. Authentication

Authentication:

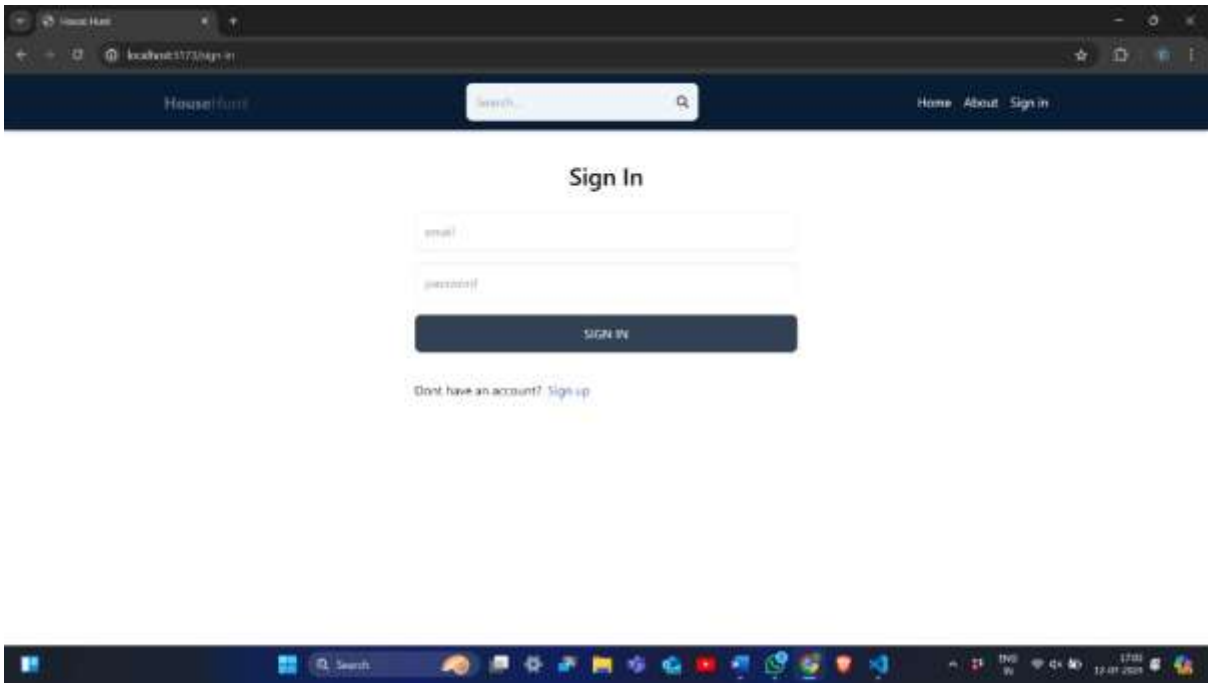
Authentication verifies the identity of users accessing your application.

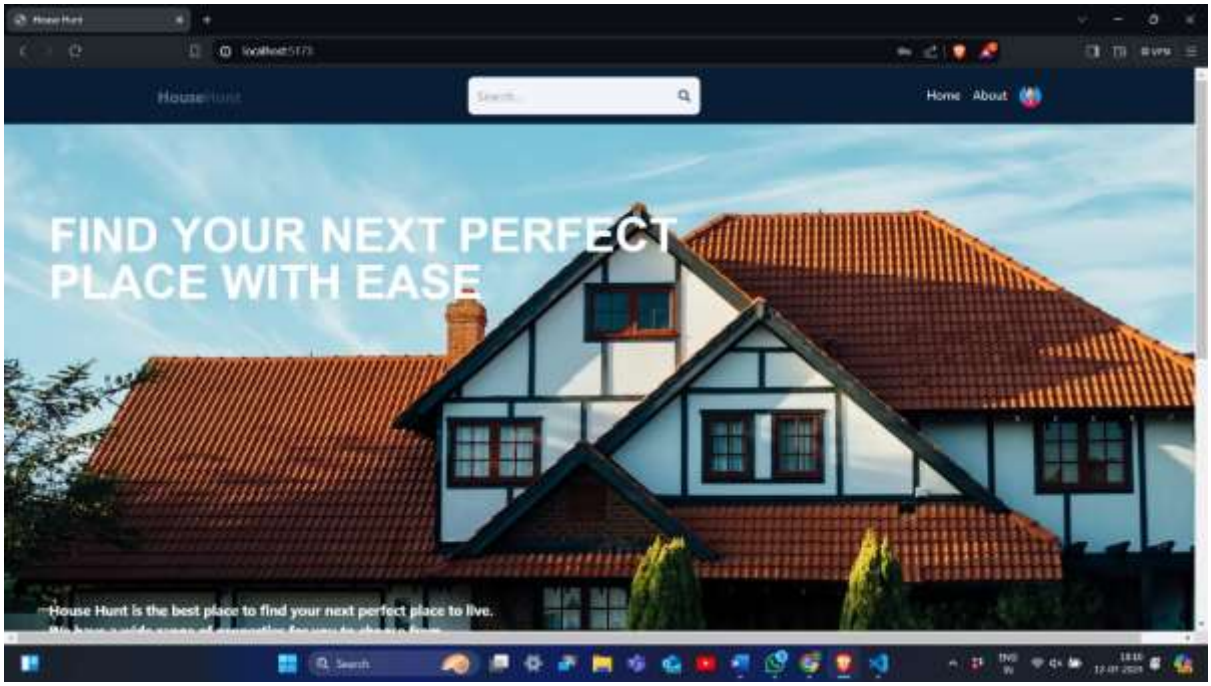
- **JWT (JSON Web Tokens):**
    - When a user logs in with valid credentials, the server creates a JWT containing a payload (like user ID) and signs it with a secret key known only to the server.
    - This JWT is sent back to the client and stored (usually in local storage or cookies).
    - For subsequent requests to protected endpoints, the client sends this JWT in the `Authorization` header.

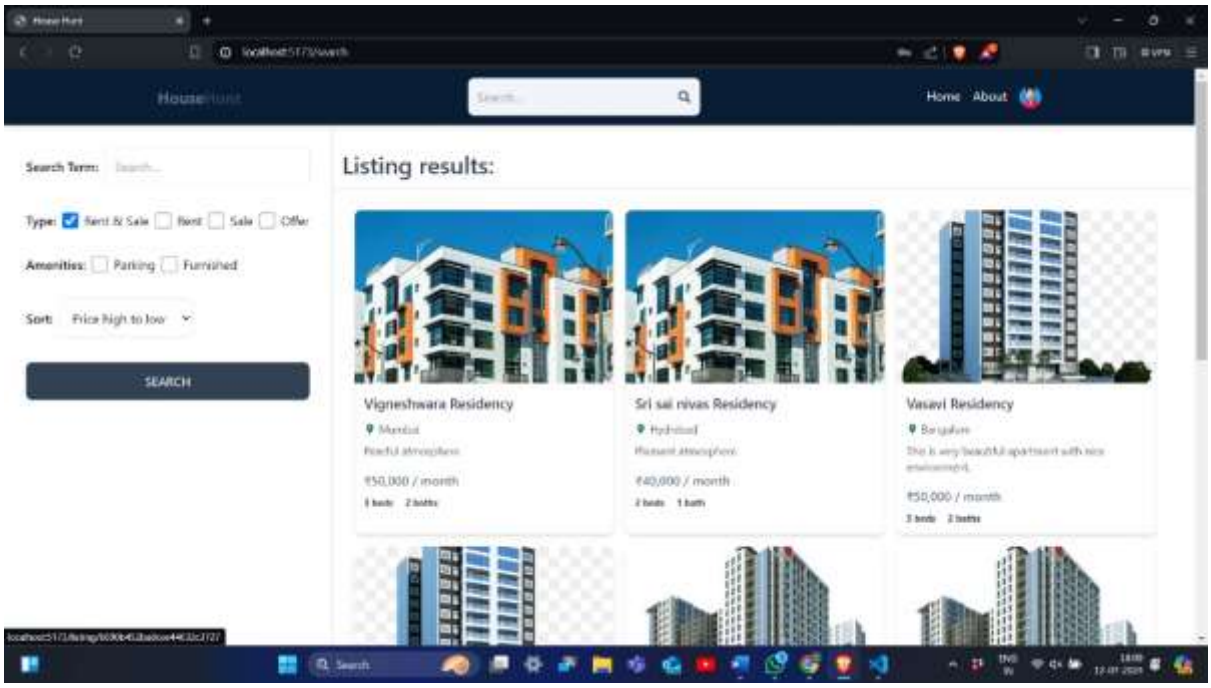
- The server verifies the JWT's signature using the secret key and checks if the user has the necessary permissions.

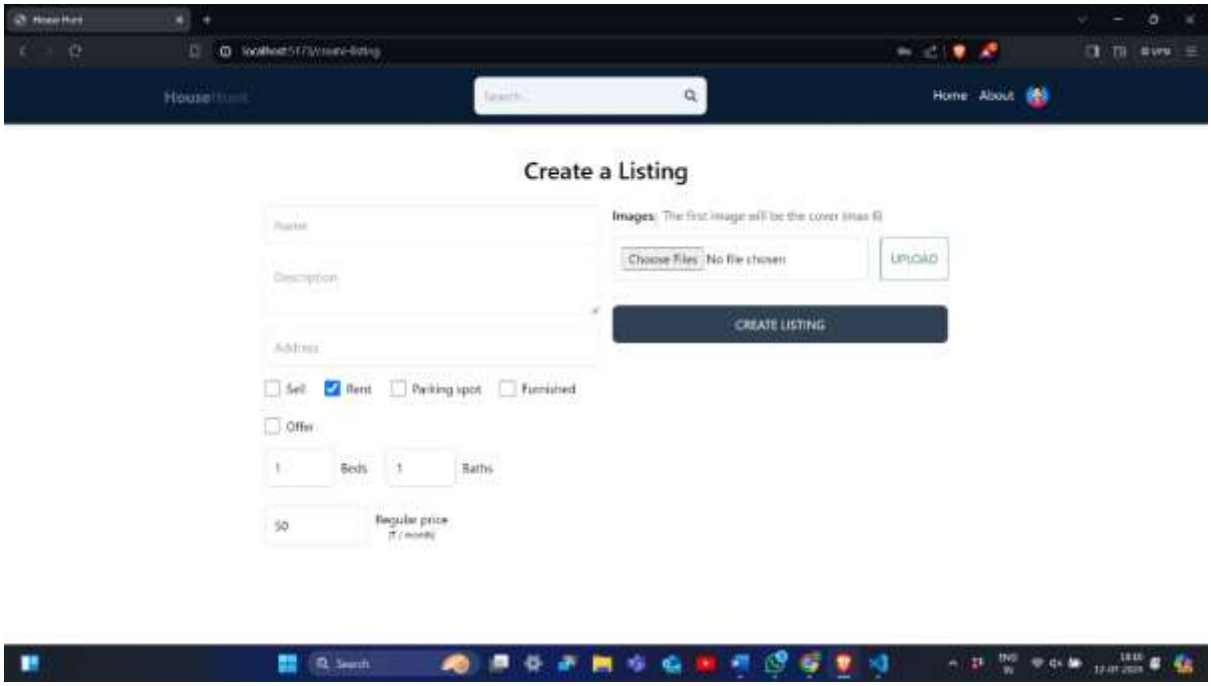
- **Authorization:**
  - The only user who have added the properties will be able to delete the property and the edit the property.
  - The user need to be successfully login into his account for adding new property.

9. User Interface:




Home
About
FIND YOUR NEXT PERFECT PLACE WITH EASE
House Hunt is the best place to find your next perfect place to live.


Home
About
Search Term:
Type:
Rent & Sale
Rent
Sale
Offer
Amenities:
Parking
Furnished
Sort:
Price high to low
SEARCH
Listing results:
Vigneshwara Residency
Sri sai nivas Residency
Vasavi Residency
₹50,000 / month
₹40,000 / month
₹50,000 / month



10. Testing:

- We have tested our project by creating Test Acceptance criteria. In this we have done for both positive and negative scenarios.And compare our actual result with the expected result.

11. Screenshots or Demo

https://drive.google.com/file/d/18cNa_CyEIxsohZyu9EMf2PeB3yh8Bxek/view?usp=drive_link

12. Known Issues

- There is no issues in our project knownly.

13. Future Enhancements

- **Real-Time Notifications:** Integrating real-time notifications (e.g., using WebSocket technology) for users to receive alerts on new listings, messages, or updates related to their account.
- **Booking and Payment Integration:** Implement a booking system where users can schedule property viewings or make rental bookings directly through the platform, integrated with payment gateways for secure transactions.