Project Report

1. INTRODUCTION

1.1 ***Project Overview***

Househunt is a web-based rental home search platform built using the MERN stack.
It allows users to register, search, and book rental properties based on their preferences.
Property owners can add, update, and manage their listings through a dedicated dashboard.
The platform eliminates the need for brokers by directly connecting tenants and landlords.
Its responsive UI and filter-based search make the rental process efficient and user-friendly.

1.2 ***Purpose***

The purpose of the Househunt project is to simplify the rental home search process.
It provides a centralized platform for users to find properties based on location, budget, and amenities.
Property owners can easily list and manage their rental properties.
The system ensures direct communication between renters and owners without third-party agents.

2. IDEATION PHASE

2.1 Problem Statement

Finding rental homes is often stressful due to scattered listings, broker dependency, lack of transparency, and no real-time availability.
Tenants face issues like fake listings, high broker fees, and poor property information.
Property owners struggle to find genuine tenants and manage property listings efficiently.
There is a need for a centralized, user-friendly platform to streamline this entire process.

2.2 Empathy Map Canvas

2.3 Brainstorming

A platform with verified listings only
Filters for rent, location, and amenities
Owner dashboard for property management
User reviews and property ratings
Real-time chat between owner and user

3. REQUIREMENT ANALYSIS

3.1 Customer Journey map

Functional Needs:

Secure login, property listing by owners, booking system for users.

User Roles:

Role-based access: separate dashboards for users and owners.

Non-Functional Needs:

Mobile responsiveness, fast API responses, and clean, user-friendly UI.

3.2 Solution Requirement

Frontend to Backend:

Users interact via the React frontend, which sends data requests to the backend (Express API).

Backend Processing:

Node.js processes the request, handles authentication, and interacts with MongoDB.

Database Interaction:

MongoDB stores users, properties, and bookings; Mongoose handles schema and queries.

3.3 Data Flow Diagram

1. **Frontend:**
   ReactJS, Axios, React Router for building a responsive and interactive UI.

2. **Backend:**
   Node.js with Express.js for handling routes, logic, and APIs.

3. **Database & Auth:**
   MongoDB with Mongoose for data storage, and JWT for secure user authentication.

3.4 Technology Stack

Frontend:

ReactJS, Axios, React Router for building a responsive and interactive UI.

Backend:

Node.js with Express.js for handling routes, logic, and APIs.

Database & Auth:

MongoDB with Mongoose for data storage, and JWT for secure user authentication.

4. PROJECT DESIGN

4.1 Problem Solution Fit

1. **Problem:** Users face difficulty finding rental homes due to scattered listings and broker dependency.

2. **Problem:** Property owners lack a direct platform to manage listings and connect with genuine tenants.

3. **Problem:** Traditional methods lack filters, reviews, and transparency in pricing and property availability.

4. **Fit:** Househunt addresses these gaps by directly connecting owners and renters with verified property listings.

5. **Fit:** The platform ensures smoother communication, efficient filtering, and secure bookings—all in one place.

4.2 Proposed Solution

1. Online Rental Marketplace: A MERN stack web platform where users can search, filter, and book rental properties.

2. Dual Role System: Separate functionalities for owners (listing, managing) and users (searching, booking).

3. Booking Management: Owners can approve/reject bookings; users can track status in real-time.

4. Secure Access: JWT authentication ensures role-based secure login and restricted access to features.

5. Scalable Design: Modular codebase and database models allow easy addition of features like reviews or payments.

4.3 Solution Architecture

Frontend – ReactJS:

Manages UI with reusable components, routing using `react-router-dom`, and API calls via Axios.

Backend – Node.js with Express:

Handles REST API routes, authentication, booking logic, and property management using controllers.

Database – MongoDB with Mongoose:
Stores user data, property details, and bookings; relationships maintained using ObjectIDs.

Authentication – JWT & Middleware:
Auth tokens validate users/owners; role-based access control ensures secure feature access.

Deployment-Ready Stack:
Supports future deployment on platforms like Render (backend), Vercel (frontend), and MongoDB Atlas (database).

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

Requirement Gathering:
Identified user needs like property search, owner listing, booking system, and secure authentication.

Technology Selection:
Chose the MERN stack (MongoDB, Express, React, Node) for full-stack development due to its scalability and efficiency.

Task Breakdown:
Divided the project into modules: frontend UI, backend APIs, database models, authentication, and image upload.

Timeline & Milestones:
Set clear milestones: UI design → API integration → Testing → Bug fixes → Final deployment (weekly sprints).

Collaboration & Tools:
Used GitHub for version control, VS Code for development, Postman for testing, and Google Docs/Sheets for planning and documentation.

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

1. **API Response Time Testing:**
   Used Postman to test all major backend APIs (login, add property, booking) and measured response times under different loads.

2. **Load Testing with Apache JMeter (Optional Tool):**
   Simulated multiple users accessing the site simultaneously to observe backend behavior under stress (optional but recommended for production).

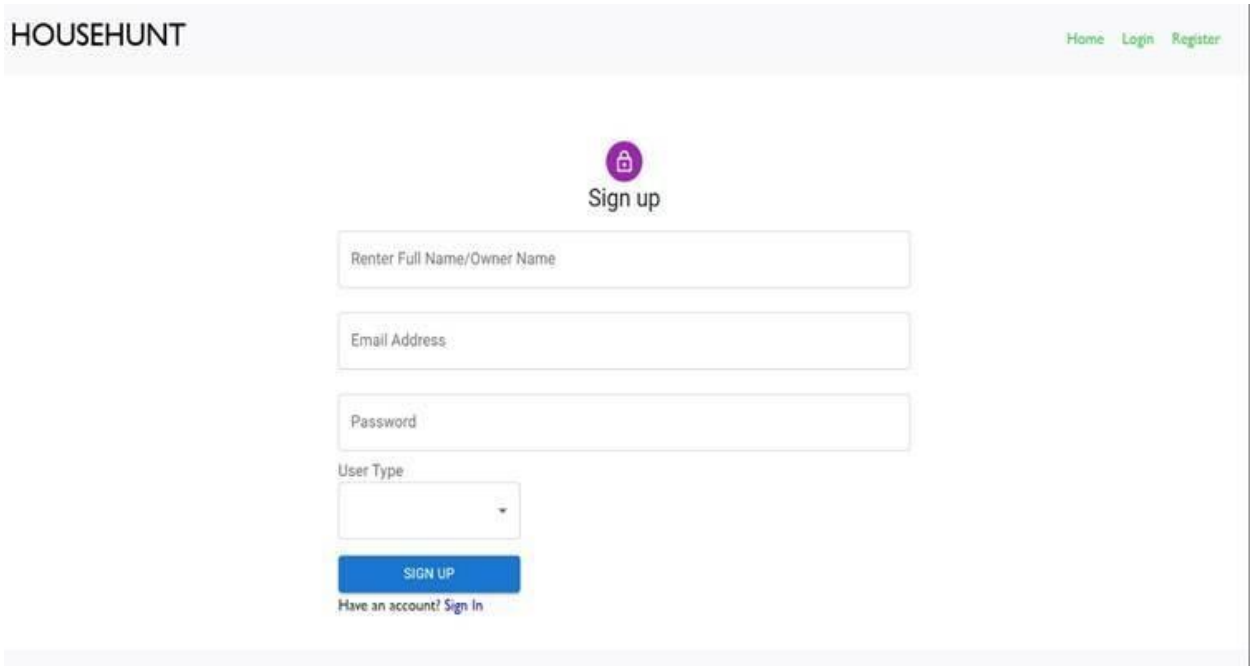
3. **Frontend Load Speed:**
   Verified that the React frontend loads efficiently on browsers with minimal lag using Chrome DevTools → Performance tab.

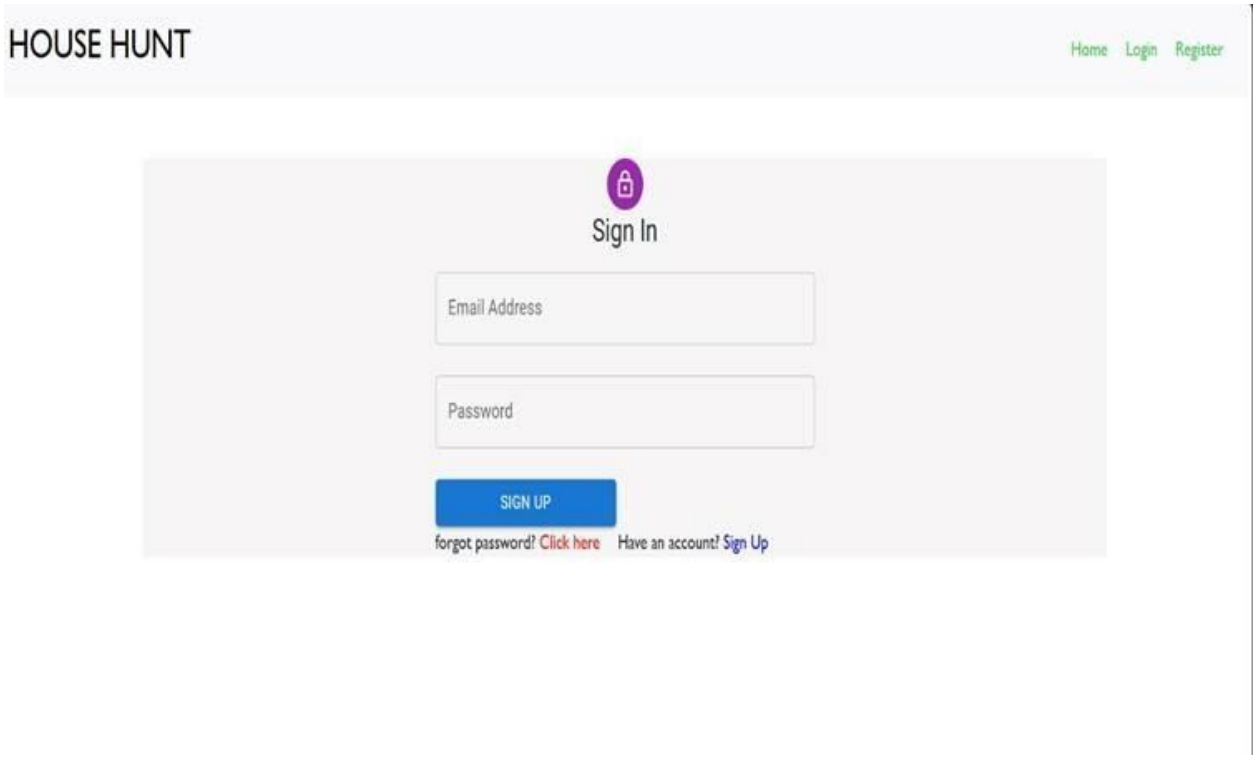
4. **Image Upload Handling:**
   Tested file upload with Multer for various image sizes to ensure server can process without crashing or long delays.

5. **Database Query Efficiency:**
   Observed MongoDB query times using Mongoose logs to ensure indexes and queries were optimized for large data sets.

7. RESULTS

7.1 Output Screenshots





8. ADVANTAGES & DISADVANTAGES

Advantages

User-Friendly Interface:

Clean and responsive design allows users to easily search and book rental

homes across devices.

Direct Owner-Tenant Interaction:
Removes the need for brokers or third-party agents, saving time and cost.

Secure Access:
JWT-based authentication ensures that user and owner data is protected.

Role-Based Dashboards:
Owners and users get personalized features suited to their roles.

Scalable & Maintainable Codebase:
Built with the MERN stack, which supports modular development and future upgrades.

Disadvantages

1. **No Real-Time Chat:**
   Lack of live communication may delay conversations between tenants and owners.

2. **No Payment Integration:**
   Users can't make secure advance payments through the platform (currently manual).

3. **Basic Error Handling:**
   Limited user feedback or alerts for backend/server failures.

4. **Filters Reset on Refresh:**
   Search filters don’t persist after page reload, affecting user experience.

5. **No Admin Panel:**
   There's no central system for admin moderation or control over content and misuse.

9. CONCLUSION

Househunt provides a digital solution to simplify the rental home search process by directly connecting tenants and property owners.

The platform ensures user convenience through features like property filtering, secure booking, and role-based access.

Developed using the MERN stack, it offers scalability, responsiveness, and modularity for future upgrades.

Manual and functional testing confirmed that the core features work effectively across different user roles and devices.

Future enhancements such as real-time chat, payment gateway, and admin panel can further improve usability and system robustness.

10. FUTURE SCOPE

Real-Time Chat Integration:

Implementing live chat between users and owners for instant communication and faster decision-making.

Payment Gateway Support:

Adding secure online payment options (Razorpay, Stripe, etc.) for rent deposits or booking confirmations.

Property Reviews and Ratings:

Enabling users to leave feedback on properties, helping future tenants make better choices.

Admin Dashboard:

Creating an admin panel to manage users, properties, bookings, and moderate content.

Mobile App Development:

Expanding to Android/iOS platforms using React Native for wider accessibility and push notifications.

11. APPENDIX

Github link: https://github.com/rrahul226/HouseHunt

Demo link : ***https://drive.google.com/file/d/1m0BplVbdjwsdut2bHwwl67b84-p8yZdV/view?usp=sharing***