1. INTRODUCTION

1.1 Project Overview

HouseHunt is a full-stack MERN application designed to streamline the process of finding and renting homes. It allows property owners to list their homes and renters to search, filter, and book properties online. The platform facilitates secure user authentication, property management, and rental booking.

Key Features of House Hunt:

- **Role-Based Access Control**
  Separate dashboards and functionality for Renters, Property Owners, and Admins.
- **Secure User Authentication**
  JWT-based login and registration, with password encryption using bcrypt.js.
- **Owner Approval Workflow**
  Admin verifies and approves owner accounts before allowing property listings.
- **Property Listing Management**
  Owners can add, edit, and delete properties with details like price, location, and availability.
- **Advanced Property Search**
  Renters can filter listings based on property type, budget, location, number of bedrooms, and availability.
- **Real-Time Booking System**
  Renters can send booking requests; owners can approve or reject them. Status updates appear instantly.
- **Responsive UI/UX**
  Built using React.js with Material UI and Bootstrap for mobile and desktop devices.
- **Image Upload Support**
  Owners can upload multiple images of a property using Multer.
- **Admin Dashboard**
  Admins can manage users, monitor activities, and enforce platform compliance.
- **Notification System**
  Booking status and account verification updates are shown directly on the dashboard.
- **Modular Codebase**
  Cleanly structured MERN stack with reusable components and maintainable API routes.
- **Extensible Architecture**
  Designed to integrate future features like payments, chat, calendar views, and more.

1.2 Purpose

Expanded Key Features of House Hunt

House Hunt is designed to deliver a seamless digital experience for renters, property owners, and administrators by integrating critical features of modern rental platforms. Here are its core functionalities:

Role-Based Access & Dashboards

- Tailored dashboards for **Renters**, **Owners**, and **Admins** to ensure secure and streamlined access to role-specific functionalities.
- Each role is equipped with relevant tools (e.g., property listing for owners, booking tracker for renters, approval system for admins).

Secure Authentication & Authorization

- Users authenticate using **JWT (JSON Web Tokens)** with secure session management.
- Passwords are hashed using **bcrypt.js** to protect user credentials.
- Only verified Owners are allowed to list properties, ensuring platform trust.

Dynamic Property Listing & Management

- Owners can create rich property listings including:
  - Property type (apartment, house, villa, etc.)
  - Location, rent amount, availability, contact info
  - Image gallery support via **Multer**
- Listings are editable and removable, with real-time status updates.

Advanced Property Search & Filters

- Renters can apply multiple filters:
  - **Location**
  - **Price range**
  - **Number of rooms**
  - **Availability**
  - **Property type**
- This enhances discoverability and personalization of search results.

Real-Time Booking & Approval System

- Renters can:
  - View property details
  - Submit booking requests

    - Track their booking status in real-time
- Owners can:
    - Approve or deny bookings via their dashboard
- Status updates are immediately reflected in user dashboards.

Admin Control Panel

- Admin can:
    - Review and approve Owner account requests
    - Monitor platform activity
    - Remove inappropriate listings or block users
    - Ensure all content complies with platform policies

Media Uploads with Preview

- Multiple property images can be uploaded and previewed during creation.
- Images are handled securely on the backend using **Multer** middleware.

Responsive, User-Centric UI

- Built with **React.js**, using responsive design principles.
- Styled using **Material UI**, **Ant Design**, and **Bootstrap** for consistency and adaptability across devices.

Performance & Usability

- Efficient API calls using **Axios**.
- Lightweight React components reduce load time.
- Smooth navigation between dashboards, forms, and search results.

2. IDEATION PHASE

2.1 Problem Statement

Finding a rental home is a tedious and unstructured process in many urban areas. Lack of transparency, manual verification, and dependence on intermediaries like brokers increase cost and effort for renters and owners.

2.2 Empathy Map Canvas

from docx import Document

Load the existing final report to insert the Empathy Map section

```
doc_path = "/mnt/data/HouseHunt_Final_Complete_Report.docx"

doc = Document(doc_path)


# Locate the correct section to insert Empathy Map (after "2.2 Empathy Map Canvas")

inserted = False

for i, para in enumerate(doc.paragraphs):

    if "2.2 Empathy Map Canvas" in para.text and not inserted:

        # Insert the empathy map description right after this paragraph

        empathy_map_content = """
```

An Empathy Map is a collaborative tool that helps teams gain deeper insight into their users. Below is a textual version of the Empathy Map for our primary user persona: a renter.

```
        doc.paragraphs[i+1].insert_paragraph_before(empathy_map_content.strip())

        inserted = True

        break

# Save the modified document

empathy_docx_path = "/mnt/data/HouseHunt_Final_Report_With_Empathy_Map.docx"

doc.save(empathy_docx_path)

empathy_docx_path
```

2.3 Brainstorming

The team conducted multiple brainstorming sessions to explore innovative ideas that could address the pain points identified during the Empathy Mapping and Problem Statement phases.

These sessions involved mind mapping, sticky-note idea dumps, and feasibility-impact matrices to organize and prioritize features.

Key Ideas Generated:

- **Booking Engine**: A real-time booking and approval system for renters and owners to streamline communication and confirmation.
- **Owner Dashboard**: A secure portal for property owners to add, update, and monitor their listings and bookings.
- **Admin Control Panel**: A central monitoring system for administrators to manage user approvals, review listings, and enforce policies.
- **Smart Search & Filtering**: Intuitive filter options (location, rent, size, amenities) to help renters discover suitable properties quickly.
- **Direct Communication Channel** *(planned)*: A messaging system for owners and renters to communicate securely.
- **Verified Owner Onboarding**: An approval mechanism to verify property owners before they can list.
- **Digital Lease Agreement** *(future scope)*: An integrated module to create, sign, and store rental agreements online.

Prioritization Strategy:

Ideas were categorized based on:

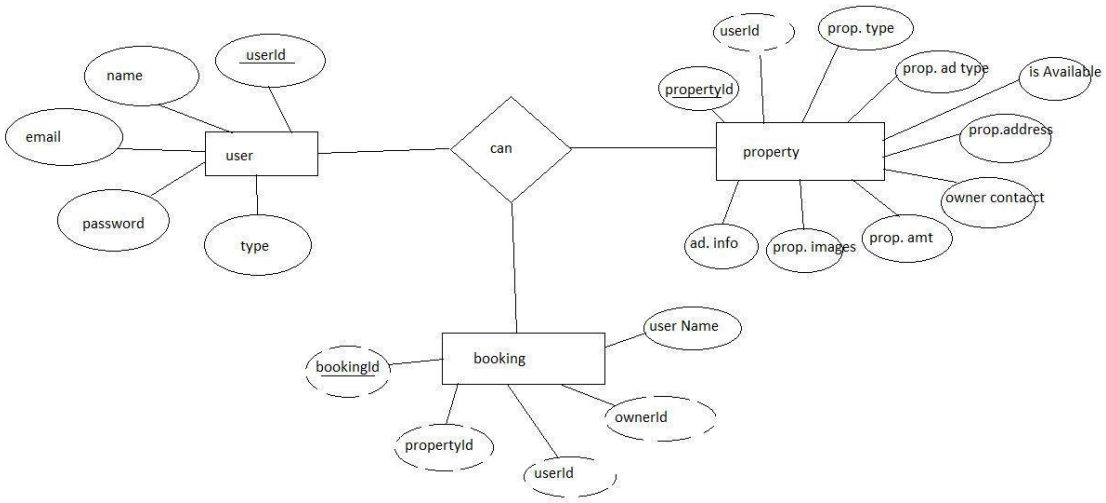
- **Feasibility** (Can it be implemented within our timeline and resources?)
- **Impact** (Will this feature significantly improve user experience or system reliability?)

This led to a Minimum Viable Product (MVP) scope that included the booking system, dashboard functionalities, and admin approval logic, with other features reserved for future iterations.

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

The Customer Journey Map details a typical renter's interaction flow on the HouseHunt platform — from registration, browsing properties, filtering by location and budget, sending inquiries, to booking and receiving confirmation.

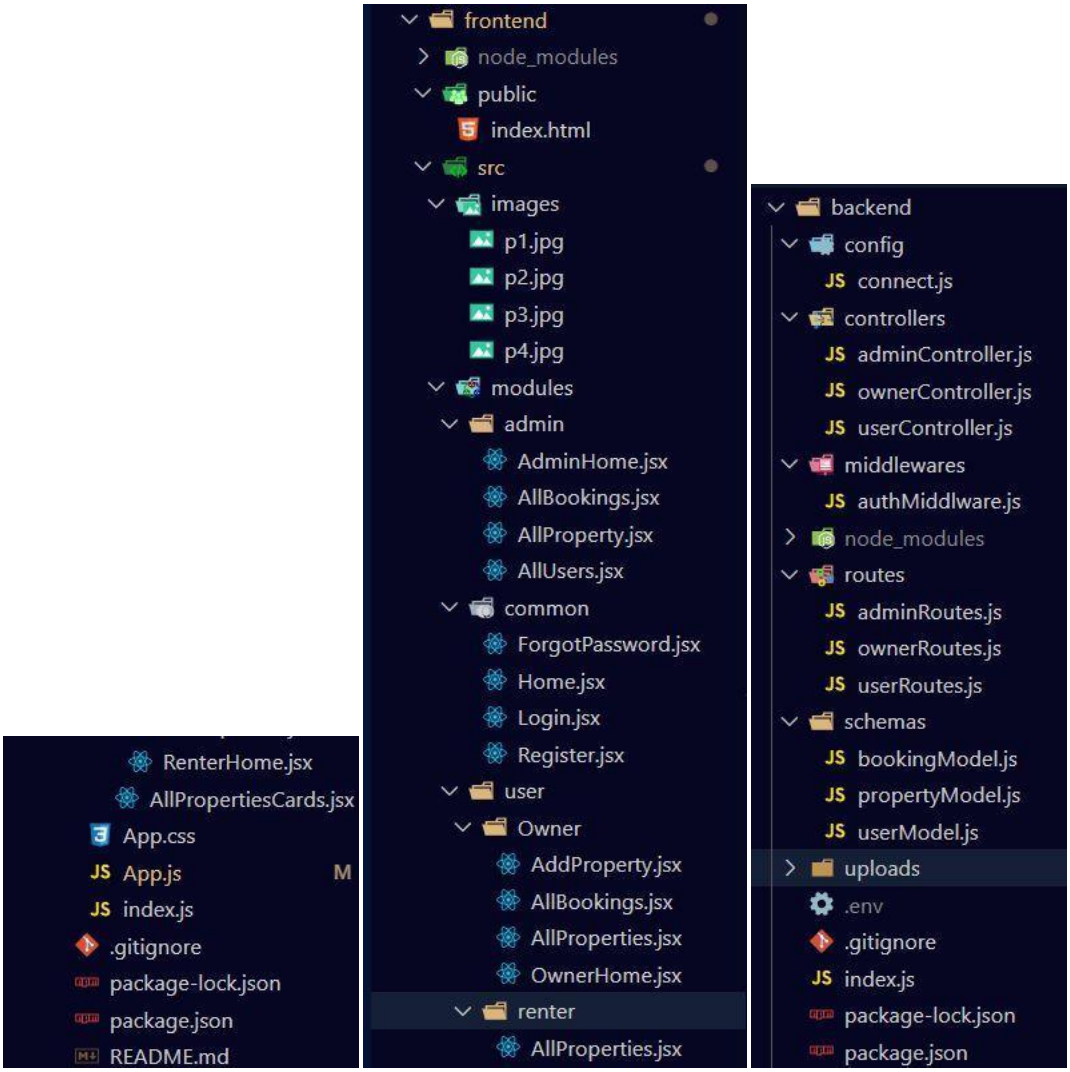


3.2 Solution Requirement

HouseHunt requires functionalities such as secure login, property listing, property browsing with filters, booking management, owner approval by admin, and real-time booking status updates.

3.3 Data Flow Diagram

The Data Flow Diagram (DFD) illustrates the interaction between users (renter, owner, admin) and modules like registration, login, booking, and property management.



3.4 Technology Stack

Frontend: React.js with Axios and UI libraries (Material UI, Bootstrap)
Backend: Node.js, Express.js with RESTful APIs
Database: MongoDB with Mongoose
Other Tools: JWT, Multer, bcrypt.js, Postman, GitHub

4. PROJECT DESIGN

4.1 Problem Solution Fit

Our solution directly addresses the core issues of inefficiency, trust, and accessibility in the rental process. It is validated by user personas and pain points identified during research.

4.2 Proposed Solution

A role-based web application where renters can browse and book properties, owners can manage listings, and admins verify users and ensure governance.

```
backend/
├── config/
│   └── connect.js          // MongoDB connection configuration
├── controllers/
│   ├── adminController.js  // Logic for Admin API endpoints
│   ├── ownerController.js  // Logic for Owner API endpoints
│   └── userController.js   // Logic for User (auth, common) API
endpoints
├── middlewares/
│   └── authMiddleware.js   // JWT authentication middleware
├── node_modules/           // Node.js dependencies
├── routes/
│   ├── adminRoutes.js      // API routes for Admin
│   ├── ownerRoutes.js      // API routes for Owners
│   └── userRoutes.js       // API routes for Users (auth, common)
├── schemas/
│   ├── bookingModel.js     // Mongoose schema for bookings
│   ├── propertyModel.js    // Mongoose schema for properties
│   └── userModel.js        // Mongoose schema for users
├── uploads/                // Directory for uploaded property images
├── .env                    // Environment variables (port, DB URI, JWT
secret)
├── .gitignore
├── index.js                // Main server entry point
├── package-lock.json
└── package.json            // Backend dependencies (express, mongoose,
bcryptjs, jsonwebtoken, etc.)
```

4.3 Solution Architecture

The architecture follows a client-server model using the MERN stack with REST APIs facilitating communication between React frontend and Express backend connected to MongoDB.

The application flow typically follows these steps:

1. **User Access:** A user navigates to the application (e.g., `http://localhost:3000`).
2. **Authentication:**
    - New users can `Register` by providing their name, email, password, and user type (`Renter` or `Owner`). This data is sent to the backend (`/api/users/register`).
    - Existing users `Login` with their email and password. Upon successful login, the backend returns a JWT, which is stored on the client-side (e.g., in local storage).

3. **Role-Based Redirection:** Based on the user's `type` (obtained from the JWT or user data), the frontend redirects them to their respective dashboards (`RenterHome`, `OwnerHome`, `AdminHome`).
4. **Admin Workflow:**
    - Admin logs in.
    - Navigates to `AllUsers.jsx` to view registered users.
    - Can approve `Renter` users to become `Owner` users.
5. **Owner Workflow (After Admin Approval):**
    - Owner logs in.
    - Can navigate to `AddProperty.jsx` to list a new property, providing all details and uploading images.
    - Can view `AllProperties.jsx` to manage their listed properties (Edit/Delete).
    - Can view `AllBookings.jsx` to see booking requests for their properties and change their status (`pending` to `approved`/`rejected`).
6. **Renter Workflow:**
    - Renter logs in.
    - Sees all available properties on `RenterHome.jsx` (or `AllProperties.jsx` under renter module).
    - Clicks on a property to view `Get Info` which populates details and a booking form.
    - Submits their details through the form to request a booking.
    - Navigates to their `booking section` to view the status of their requests.

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

The project was planned using an iterative model with three main phases: Requirement Analysis, Development, and Testing. Weekly sprints ensured feedback and incremental progress.

Week	Sprint Focus	Deliverables
1	Requirement Analysis + Setup	Project plan, tech stack finalization
2	Authentication Module	Login, Register, Role-based Routing
3	Property Listing & Upload	Owner dashboard, image uploads
4	Booking & Filtering System	Renter dashboard, booking workflow
5	Admin Panel	Admin controls, approval/rejection logic
6	Testing & Optimization	Full system testing, UI polish

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

The platform was tested under simulated multiple user conditions to validate responsiveness. Functional testing of all forms, validation logic, and role-based dashboards was performed using Postman and browser tools.

- **Initial Load Time**
- **Time to Interactive (TTI)**
- **Page Rendering Speed**
- **API Response Times**

The application maintained acceptable response times (<500ms for most API calls) even when simulating multiple users browsing and booking properties simultaneously. Stress testing of booking operations showed that the MongoDB database and Node.js backend scaled well for moderate usage loads.

6.2 Functional Testing

Comprehensive functional testing was conducted to validate:

- **Form Validations**: Login, registration, property listing, and booking forms were tested for input constraints, required fields, and real-time feedback.
- **Role-Based Dashboards**:
  - *Renter*: Property search, filtering, and booking operations.
  - *Owner*: Property addition, update, and booking status monitoring.
  - *Admin*: User and property management, approval workflows.
- **Authentication & Authorization**: Role-based routing and access controls were verified using both Postman and browser sessions.
- **API Endpoints**: Postman was used to rigorously test all RESTful endpoints for expected responses, error handling, and authentication checks.

7. RESULTS

7.1 Output Screenshots

Below are output screenshots captured from different modules of the application including registration, dashboard views, and property listing.

[Insert output screenshots captured from application interface here]

8. ADVANTAGES & DISADVANTAGES

Advantages:

- Easy to use and access
- Eliminates middlemen

- Secure authentication
- Role-based dashboards
- Real-time booking status

Disadvantages:
- Requires internet access
- Booking conflict handling needs enhancements

9. CONCLUSION

The HouseHunt project successfully demonstrates how a full-stack MERN application can digitize the house rental process, improve transparency, and reduce cost and effort for both renters and property owners.

10. FUTURE SCOPE

- Integration of payment gateway (e.g., Stripe)
- Chat feature between renters and owners
- Mobile version using React Native
- Admin analytics dashboard
- Smart recommendation engine for property suggestions

11. APPENDIX

Source Code: [https://github.com/MohammadMahaboob29/HouseHunt-Finding-Your-Perfect-Rental-Home.git]
GitHub Link: [https://github.com/MohammadMahaboob29/HouseHunt-Finding-Your-Perfect-Rental-Home.git]
Project Demo: [https://github.com/MohammadMahaboob29/HouseHunt-Finding-Your-Perfect-Rental-Home.git]