Project Report

HouseHunt - Finding Your perfect Rental Home using MERN

Submitted to the APSCHE

In partial fulfillment of the requirements for the degree of Bachelor of TechnologyInComputer Science and EngineeringSchool of Engineering and Sciences

Submitted By:

- Prameela Saketi
- Virekha Vykuntam
- Pujari Sushmitha
- Poojitha

Under the guidance of faculty mentor:

- M. Anji babu Sir

July 2025

Acknowledgements

We would like to express our sincere gratitude to the Andhra Pradesh State Skill Development Corporation (APSSDC) for providing us with the opportunity to undertake this project.

We extend our deepest thanks to our guide, **M. Anji Sir**, for his invaluable guidance, continuous support, and constructive feedback throughout the duration of this project. His expertise and encouragement were instrumental in shaping this report and the successful completion of the "HouseHunt" application.

We are also grateful to our faculty members and the School of Engineering and Sciences for providing the necessary resources and an environment conducive to learning and development.

Finally, we would like to thank our families and friends for their unwavering support and encouragement during this academic endeavor.

Abstract

This report details the design and development of "HouseHunt - Find Your Rental Home," a full-stack web application developed using the MERN (MongoDB, Express.js, React.js, Node.js) stack. The project addresses the complexities of the real estate rental market by providing a user-friendly and comprehensive platform that connects renters, property owners, and administrators. Key features include advanced property search with filters, interactive listings with virtual tours, and a robust user management system supporting distinct roles. The application aims to streamline the process of finding and listing rental properties, offering a transparent and efficient marketplace. This report covers the problem identification, system requirements, design, implementation details, testing, and future enhancements, showcasing a practical application of MERN stack technologies in solving a real-world problem.

Table of Contents

1. INTRODUCTION 1.1 General Introduction 1.2 Problem Identification
2. LITERATURE SURVEY
3. SYSTEM REQUIREMENTS 3.1 Hardware Requirements 3.2 Software Requirements
4. SYSTEM DESIGN 4.1 High Level Design 4.2 Modules 4.3 ER Diagram
5. CODING & IMPLEMENTATION 5.1 Technologies Used 5.2 Project Structure 5.3 Project Flow / Milestones
6. TESTING & ANALYSIS 6.1 Functional Testing 6.2 Performance Analysis
7. RESULT ANALYSIS 7.1 Output Screenshots
8. CONCLUSION & FUTURE ENHANCEMENT 8.1 Conclusion 8.2 Future Enhancement
9. APPENDIX 9.1 Source Code 9.2 Project Demo Link 9.3 GitHub & Project Demo Link (Combined with 9.2)

List of Figures

Figure 4.1: High-Level Technical Architecture of HouseHunt
Figure 4.2: Entity-Relationship (ER) Diagram for HouseHunt
Figure 5.1: Frontend Project Structure
Figure 5.2: Backend Project Structure
Figure 7.1: HouseHunt Register / Sign Up Page
Figure 7.2: HouseHunt Login Page
Figure 7.3: HouseHunt Booking History Page

---

1. INTRODUCTION

1.1 General Introduction The process of searching for a rental home, be it a luxury apartment or a spacious villa, is often a mix of excitement and challenge. To secure a residence that aligns perfectly with individual needs, meticulous planning and a clear understanding of personal priorities are paramount. This involves a careful assessment of factors such as budget, which includes not only the rental cost but also utilities and other living expenses. For premium rentals, additional charges like maintenance fees or parking might also be a consideration. Equally important is the location, requiring users to consider its proximity to workplaces, educational institutions, shopping centers, and recreational facilities, ensuring it complements their desired lifestyle. The "HouseHunt" project aims to simplify this complex and time-consuming process by offering a comprehensive, user-friendly digital platform.

1.2 Problem Identification The traditional methods of searching for rental properties can be inefficient, fragmented, and time-consuming. Renters often face challenges in finding properties that precisely match their criteria, accessing high-quality information (like reliable photos, virtual tours, and floor plans), and connecting directly with legitimate property owners or agents. Property owners, conversely, struggle with effectively marketing their listings, managing inquiries, and tracking potential renters. The lack of a centralized, transparent, and interactive platform leads to missed opportunities, prolonged search cycles, and potential for misinformation. This project identifies the need for a streamlined, digital solution that bridges these gaps, enhancing efficiency and transparency for all stakeholders in the rental market.

---

2. LITERATURE SURVEY

The development of the "HouseHunt" application is informed by a comprehensive review of existing online real estate and rental platforms, both domestic and international. This survey aims to understand current industry trends, identify best practices, and pinpoint areas for innovation.

Major platforms like Zillow, Apartments.com, Realtor.com, MagicBricks, and 99acres (as referenced in the project documentation) demonstrate the success of centralized property listing services. These platforms typically excel in:

- **Extensive Databases:** Offering a wide array of properties, from apartments to villas, catering to diverse needs.
- **Advanced Search Functionality:** Providing filters for location, price, property type, amenities, and more.
- **Rich Media Content:** Utilizing high-quality photos, videos, and sometimes 360-degree virtual tours to enhance property viewing.
- **User Accounts:** Enabling users to save searches, favorite properties, and track inquiries.

However, the survey also highlights common areas for improvement and opportunities for differentiation:

- **Trust and Transparency:** While listing services are common, ensuring the legitimacy of listings and direct communication with verified owners/agents can still be a challenge.
- **Transaction Simplification:** Beyond just listing, there's scope for integrating booking, application, and even payment processes more seamlessly.
- **Role-Specific Dashboards:** While some platforms have different interfaces, truly tailored experiences for owners and renters that go beyond simple listing/searching are less common.
- **Performance and Responsiveness:** Older platforms might struggle with modern web performance demands or mobile responsiveness.
- **Integrated Professional Connections:** Facilitating connections with legal and financial experts directly within the platform could add significant value.

This literature survey validates the demand for a comprehensive online rental platform and guides the design choices for "HouseHunt," emphasizing its full-stack MERN implementation to address performance, scalability, and the creation of a seamless, role-specific user experience.

---

3. SYSTEM REQUIREMENTS

3.1 Hardware Requirements To effectively develop and run the HouseHunt application, the following hardware specifications are recommended:

- **Processor:** A multi-core processor (e.g., Intel Core i5 or equivalent AMD Ryzen series) is advisable to handle the compilation and execution of both frontend and backend development environments simultaneously.
- **RAM:** A minimum of 8GB RAM is required, with 16GB or more highly recommended for optimal performance when running various development tools, multiple browser tabs, and local server instances concurrently.
- **Storage:** At least 256GB Solid State Drive (SSD) is crucial for faster operating system boot times, quicker application loading, and efficient compilation of project files.
- **Internet Connection:** A stable and high-speed internet connection is necessary for downloading development dependencies, accessing online documentation, and interacting with cloud-based MongoDB instances if used.

3.2 Software Requirements The HouseHunt project relies on a modern software stack to achieve its functionalities:

- **Operating System:** Compatible with cross-platform operating systems such as Windows 10/11, macOS, or popular Linux distributions (e.g., Ubuntu, Fedora).
- **Runtime Environment: Node.js** (LTS version recommended) for executing server-side JavaScript code.
- **Package Manager: npm** (Node Package Manager) for managing project dependencies and running scripts.
- **Backend Framework: Express.js** for building the RESTful APIs and handling server-side routing and middleware.
- **Database: MongoDB** (Community Server Edition) for flexible NoSQL data storage.

- **Object-Document Mapper (ODM): Mongoose.js** for simplifying interactions with MongoDB from Node.js.
- **Frontend Library: React.js** for building the dynamic and interactive user interface.
- **Build Tool: Vite** (for React project creation) for fast development server and optimized production builds.
- **UI Component Libraries: Material UI**, **Bootstrap**, and **Antd** for pre-built, responsive, and aesthetically pleasing UI components.
- **HTTP Client: Axios** for making HTTP requests from the React frontend to the Express.js backend.
- **Date/Time Library: Moment.js** for parsing, validating, manipulating, and formatting dates and times.
- **Authentication/Security: bcryptjs** for password hashing and **jsonwebtoken (JWT)** for token-based authentication.
- **Development Environment:** A code editor like **Visual Studio Code** (recommended), Sublime Text, or WebStorm.
- **Version Control System: Git** for tracking changes and managing code revisions.
- **Browser:** Modern web browsers (Chrome, Firefox, Edge, Safari) for testing the frontend application.

---

4. SYSTEM DESIGN

4.1 High Level Design The HouseHunt application follows a three-tier client-server architecture, typical of a full-stack web application built with the MERN stack.

- **Client-Side (Frontend):** This layer is the user interface, built using **React.js**. It is responsible for rendering the UI, handling user interactions, and making API calls to the backend. It uses libraries like **Axios** for HTTP requests and **Bootstrap/Material UI** for styling and responsive design.
- **Server-Side (Backend):** This layer is powered by **Node.js** and the **Express.js** framework. It acts as the application's brain, handling business logic, user authentication and authorization, processing API requests from the frontend, and interacting with the database. It exposes RESTful APIs to the client.
- **Database Layer:** This layer consists of **MongoDB**, a NoSQL database. It stores all application data, including user profiles, property listings, and booking details. **Mongoose.js** is used on the backend to facilitate structured data modeling and interaction with MongoDB.

This layered architecture ensures clear separation of concerns, scalability, and ease of maintenance.

High-Level Technical Architecture of HouseHunt: The diagram below illustrates the fundamental flow of data and interaction between the three core components. \begin{figure}[h!] \centering \includegraphics[width=0.8\textwidth]{/tmp/assets/WhatsApp Image 2025-06-26 at 10.12.30_15bfad89.jpg-89a7ac3d-16c8-4812-be8a-78c9acb2254b} \caption{High-Level Technical Architecture of HouseHunt} \label{fig:high_level_architecture} \end{figure}

4.2 Modules The HouseHunt system is designed with distinct modules corresponding to the primary user roles and core functionalities, ensuring clear separation of concerns and efficient management:

- 1. **User Authentication Module:**
    - Handles user registration (Renter, Owner, Admin).
    - Manages user login and session management using JWT.
    - Implements password hashing (bcryptjs) for security.
    - Facilitates "Forgot Password" functionality.
- 2. **Renter/Tenant Module:**
    - **Property Browse:** Displays all available properties with search and filter options.
    - **Property Details View:** Allows viewing comprehensive information about a property, including images, address, price, and owner contact.
    - **Inquiry/Contact:** Functionality to send details/inquiries to property owners.
    - **Booking Management:** Views booking history and status (pending, confirmed).
- 3. **Owner Module:**
    - **Admin Approval Gateway:** Requires initial approval from the Admin to gain full owner privileges.
    - **Property Management (CRUD):** Allows owners to Add, View, Update, and Delete their property listings.
    - **Property Status Management:** Ability to change property availability (e.g., mark as rented).
    - **Booking Oversight:** View booking requests from renters.
- 4. **Admin Module:**
    - **User Management:** Approves new owner registrations.
    - **Property Oversight:** Monitors all properties listed on the platform.
    - **Booking Monitoring:** Oversees all booking requests.
    - **Policy Enforcement:** Manages and enforces platform policies.

4.3 ER Diagram The Entity-Relationship (ER) diagram below visually represents the database schema, illustrating the entities (User, Property, Booking) and their relationships within the HouseHunt application.

\begin{figure}[h!] \centering \includegraphics[width=0.8\textwidth]{/tmp/assets/WhatsApp Image 2025-06-26 at 10.12.30_15bfad89.jpg-89a7ac3d-16c8-4812-be8a-78c9acb2254b/563} \caption{Entity-Relationship (ER) Diagram for HouseHunt} \label{fig:er_diagram} \end{figure}

Entities and their Attributes:

- **User:**
    - userId (Primary Key, unique identifier)

    - name
    - email
    - password
    - type (e.g., "renter", "owner", "admin")
- **Property:**
    - propertyId (Primary Key, unique identifier)
    - userId (Foreign Key, links to the User who owns the property)
    - prop.type (e.g., "apartment", "house")
    - prop.ad.type (e.g., "rent")
    - isAvailable (boolean)
    - prop.address
    - owner.contact
    - ad.info
    - prop.images (array of image URLs)
    - prop.amt (amount/price)
- **Booking:**
    - bookingId (Primary Key, unique identifier)
    - propertyId (Foreign Key, links to the booked Property)
    - userId (Foreign Key, links to the Renter who made the booking)
    - userName (Name of the renter for convenience)
    - status (e.g., "pending", "confirmed", "cancelled")

Relationships:

- A User can own zero or many Property listings. (One-to-Many: User to Property via userId)
- A User (as a Renter) can make zero or many Bookings. (One-to-Many: User to Booking via userId)
- A Property can have zero or many Bookings. (One-to-Many: Property to Booking via propertyId)

5. CODING & IMPLEMENTATION

5.1 Technologies Used The HouseHunt application is built entirely on the MERN (MongoDB, Express.js, React.js, Node.js) stack, leveraging JavaScript across both the frontend and backend.

- **Frontend (React.js):** Utilized for building dynamic and interactive user interfaces. React's component-based architecture facilitated modular development and reusability. Key libraries used include:
    - Axios: For making HTTP requests to the backend APIs.
    - Bootstrap & Material UI: For responsive design and pre-styled UI components.
    - Antd: Another UI library providing additional components.
    - Moment.js: For handling date and time formatting.

- **Backend (Node.js & Express.js):** Node.js provides a robust runtime environment for server-side JavaScript, and Express.js simplifies the creation of RESTful APIs. Core modules and libraries include:
    - cors: For enabling Cross-Origin Resource Sharing.
    - body-parser: For parsing incoming request bodies.
    - dotenv: For managing environment variables securely.
    - jsonwebtoken (JWT): For token-based user authentication.
    - bcryptjs: For hashing passwords securely.
    - multer: For handling file uploads (e.g., property images).
    - nodemon: A development utility that automatically restarts the Node.js server upon code changes.
- **Database (MongoDB):** A NoSQL database chosen for its flexibility and scalability. It stores data in JSON-like documents, which integrates seamlessly with JavaScript.
    - Mongoose: An ODM (Object Data Modeling) library for Node.js and MongoDB, providing a schema-based solution for application data.

5.2 Project Structure The project adheres to a clear folder structure, separating frontend and backend logic for maintainability and scalability.

Frontend Structure: The frontend directory contains all client-side code: \begin{figure}[h!] \centering \includegraphics[width=0.8\textwidth]{/tmp/assets/WhatsApp Image 2025-06-26 at 10.12.30_15bfad89.jpg-89a7ac3d-16c8-4812-be8a-78c9acb2254b/649} \caption{Frontend Project Structure} \label{fig:frontend_structure} \end{figure}

- node_modules: Contains all installed npm packages for the frontend.
- public: Public assets, including the index.html file.
- src: Main source code.
    - images: Stores static images like property photos.
    - modules: Potentially for highly modular, reusable components.
    - admin: React components specific to the Administrator's dashboard and functionalities (e.g., AdminHome.jsx, AllBookings.jsx, AllProperty.jsx, AllUsers.jsx).
    - common: Shared React components used across multiple user roles (e.g., ForgotPassword.jsx, Home.jsx, Login.jsx, Register.jsx).
    - user: Contains components tailored for end-users, further subdivided into Owner and renter specific components.
        - Owner: For property owners (e.g., AddProperty.jsx, AllBookings.jsx, AllProperties.jsx, OwnerHome.jsx).
        - renter: For renters/tenants (e.g., AllProperties.jsx, RenterHome.jsx, AllPropertiesCards.jsx).
    - App.css, App.js, index.js: Main application styling, root component, and entry point.

Backend Structure: The backend directory houses all server-side code: \begin{figure}[h!] \centering

\includegraphics[width=0.8\textwidth]{/tmp/assets/WhatsApp Image 2025-06-26 at 10.12.30_15bfad89.jpg-89a7ac3d-16c8-4812-be8a-78c9acb2254b/649} \caption{Backend Project Structure} \label{fig:backend_structure} \end{figure}

- config: Database connection configuration (e.g., connect.js).
- controllers: Contains the logic that handles requests from routes and interacts with models (e.g., adminController.js, ownerController.js, userController.js).
- middlewares: Houses custom middleware functions, notably authMiddleware.js for authentication.
- node_modules: All installed npm packages for the backend.
- routes: Defines API endpoints and links them to corresponding controller functions (e.g., adminRoutes.js, ownerRoutes.js, userRoutes.js).
- schemas: Contains Mongoose schema definitions for database models (e.g., bookingModel.js, propertyModel.js, userModel.js).
- uploads: Directory for storing uploaded files, such as property images.
- .env: Environment variables.
- index.js: The main entry point for the backend server.

5.3 Project Flow / Milestones The development of HouseHunt followed a phased milestone approach:

- **Milestone 1: Project Setup and Configuration**
  - Created frontend and backend directories.
  - Installed all necessary npm packages for both frontend (React, Bootstrap, Material UI, Axios, Moment, Antd, mdb-react-ui-kit, react-bootstrap) and backend (cors, bcryptjs, express, dotenv, mongoose, Moment, Multer, Nodemon, jsonwebtoken).
  - Configured package.json for both environments.
- **Milestone 2: Backend Development**
  - Set up the Express server in index.js.
  - Configured environment variables (.env) for sensitive data like port, MongoDB connection string, and JWT key.
  - Integrated cors and body-parser middleware.
  - Developed authMiddleware.js for robust project authentication.
- **Milestone 3: Database Development**
  - Configured MongoDB connection using Mongoose from config.js.
  - Designed and implemented database schemas for users, properties, and bookings in the schemas folder.
- **Milestone 4: Frontend Development**
  - Initialized the React application.
  - Designed and implemented UI components, focusing on reusability and responsiveness using Bootstrap and Material UI.
  - Integrated a navigation system.
  - Connected frontend with backend APIs using Axios for data fetching and submission.
  - Implemented data binding to display dynamic content.

- **Milestone 5: Project Implementation**
    - Implemented core functionalities: User Registration/Sign Up, Login, Property listing/viewing, and Booking History.
    - Thoroughly verified all functionalities and performed bug fixing.
    - The application was made accessible locally for testing.

---

6. TESTING & ANALYSIS

6.1 Functional Testing Throughout Milestone 5, comprehensive functional testing was performed to ensure that all features of the HouseHunt application operate as intended according to the defined requirements. This testing involved:

- **User Authentication Flow:** Verifying the success of user registration (for all types), login, and password management.
- **Role-Based Access Control:** Confirming that each user type (Renter, Owner, Admin) has access only to their designated functionalities and dashboards.
- **Property Management:** Testing that owners can correctly add, view, update, and delete properties; and that property availability status can be changed.
- **Property Search and Filtering:** Validating that users can effectively search for properties and apply filters.
- **Booking Process:** Ensuring renters can send inquiries/booking requests, and owners/admins can view and manage these requests.
- **Data Integrity:** Checking that data is correctly stored in MongoDB and retrieved accurately by the frontend.
- **API Validation:** Verifying that all backend API endpoints respond correctly to requests and handle various inputs/errors gracefully. The application was run locally, and each function was manually tested to ensure a smooth user experience and adherence to project specifications.

6.2 Performance Analysis While formal, dedicated performance testing (e.g., load testing, stress testing) was not extensively detailed in the project documentation, the architectural choices of the MERN stack inherently contribute to the application's performance characteristics.

- **Node.js:** Its non-blocking, event-driven architecture is highly efficient for handling concurrent requests, which is beneficial for an application with multiple users accessing and modifying data. This makes the backend responsive even under moderate load.
- **React.js:** The use of a Virtual DOM minimizes direct manipulation of the actual DOM, leading to faster UI updates and a smoother user experience. This contributes to perceived performance on the client-side.
- **MongoDB:** As a NoSQL database, MongoDB offers high scalability and performance for large datasets, especially for read operations. Its flexible schema also allows for quick iteration and data modeling.

- **Efficient Data Flow:** The unified JavaScript language across the stack (Node.js, Express.js, React.js, MongoDB via Mongoose) reduces serialization/deserialization overhead and simplifies the overall data flow, which can contribute to better application responsiveness. The implementation ensures that common operations like property listing, searching, and booking are handled efficiently. Further performance optimizations could involve caching mechanisms, database indexing, and CDN integration for static assets in a production environment.

---

7. RESULT ANALYSIS

7.1 Output Screenshots The successful implementation of the HouseHunt application is evident through its functional user interfaces for different user roles and core processes. The following screenshots demonstrate key aspects of the application:

-

Register or Sign Up Page: This interface allows new users to create an account by providing their name, email, password, and selecting their user type (Renter or Owner). \begin{figure}[h!] \centering \includegraphics[width=0.8\textwidth]{/tmp/assets/Full_stack_mern_stackn_documentation[1].docx/718} \caption{HouseHunt Register / Sign Up Page} \label{fig:register_page} \end{figure}

-
-

Login Page: This page provides existing users with fields to enter their registered email address and password to gain access to the platform. \begin{figure}[h!] \centering \includegraphics[width=0.8\textwidth]{/tmp/assets/Full_stack_mern_stackn_documentation[1].docx/719} \caption{HouseHunt Login Page} \label{fig:login_page} \end{figure}

-
-

Booking History Page: This screenshot illustrates the booking history section, where renters can view details of their property bookings, including Booking ID, Property ID, Tenant Name, Phone number, and the current Booking Status. \begin{figure}[h!] \centering \includegraphics[width=0.8\textwidth]{/tmp/assets/Full_stack_mern_stackn_documentation[1].docx/721} \caption{HouseHunt Booking History Page} \label{fig:booking_history} \end{figure}

-

These screenshots highlight the intuitive design and functional capabilities implemented within the HouseHunt application.

---

8. CONCLUSION & FUTURE ENHANCEMENT

8.1 Conclusion The "HouseHunt - Find Your Rental Home" project successfully developed a comprehensive, full-stack web application leveraging the MERN (MongoDB, Express.js, React.js, Node.js) stack. The platform effectively addresses the complexities of the rental real estate market by providing a streamlined, user-friendly interface for renters, property owners, and administrators. Key functionalities such as advanced property search, interactive listings, robust user authentication, and role-based dashboards were successfully implemented. The project demonstrated the power of the MERN stack in building scalable, efficient, and interactive web solutions, proving its capability to simplify complex real-world processes. All defined functional requirements were met, resulting in a stable and operational application that holds significant potential for enhancing the rental experience.

8.2 Future Enhancement The HouseHunt application has substantial scope for future development and integration of advanced features:

- **Interactive Maps:** Integrate Google Maps or similar services to allow users to search for properties directly on a map, view nearby amenities, and calculate commute times.
- **360° Virtual Tours & High-Quality Media:** Enhance property listings with embedded high-resolution photos, video walkthroughs, and immersive 360° virtual tours for a more detailed viewing experience.
- **Direct Messaging/Chat:** Implement an in-app messaging system using WebSockets (e.g., Socket.IO) to facilitate real-time communication between renters and owners/agents.
- **Online Application & Payment Processing:** Integrate a secure online application form and a payment gateway to allow renters to apply and make rental payments directly through the platform.
- **Automated Property Alerts:** Develop more sophisticated notification systems for new listings, price drops, or changes in property availability matching saved search criteria.
- **Property Management Tools for Owners:** Provide owners with advanced tools for managing multiple properties, tracking inquiries, scheduling viewings, and generating reports.
- **Review and Rating System:** Allow renters to leave reviews and ratings for properties and owners, enhancing transparency and trust within the community.
- **AI-Powered Recommendations:** Implement machine learning algorithms to provide personalized property recommendations to renters based on their search history and preferences.
- **Mobile Application:** Develop native iOS and Android mobile applications to provide an optimized user experience on mobile devices and expand reach.

- **Legal & Financial Integration:** Partner with legal and financial experts to offer services like lease agreement templates, credit checks, and mortgage advice directly through the platform.

---

9. APPENDIX

9.1 Source Code The complete source code for the "HouseHunt" project is available in the following GitHub repository and Google Drive folder:

- **GitHub Repository:** House Rent GitHub
- **Code Drive Link:** House Rent Code Drive

9.2 Project Demo Link A live demonstration of the "HouseHunt" application's key functionalities can be viewed at the following link:

- **Project Demo Video:** House Rent Project Demo