**Full Stack Development with MERN – Project**

**FINAL REPORT**

**Project Title**:

MERN House Rent App - HouseHunt

**Team Members**:

Thanuja gogada (22Q71A0553)   
Anvitha Durgam (22Q71A0545)

# 1. INTRODUCTION

In an era where mobility is integral to our lives, finding rental properties has become increasingly challenging. This holds especially true for students, professionals, and families who often face a maze of options and unreliable methods when moving to unfamiliar cities. Conventional approaches to house hunting, such as relying on brokers, verbal referrals, or navigating through countless social media posts, often prove to be time-consuming and frustrating. These methods frequently lack transparency, leading to misinformation and unnecessary delays in securing a rental property.

## 1.1 Project Overview

To address these challenges, **HOUSE HUNT**, a modern web-based application built on the MERN stack (MongoDB, Express.js, React.js, and Node.js), emerges as a solution aimed at redefining the house rental process. Leveraging cutting-edge technology, HOUSE HUNT offers a seamless experience for both landlords and tenants. Landlords can list properties with verified details, ensuring credibility and reliability. Tenants, on the other hand, gain access to a streamlined search feature that allows them to explore homes based on their preferences, including location, budget, and required amenities.

The application goes beyond basic functionalities by integrating advanced features that enhance convenience and security. These include user authentication to protect personal information, secure channels for communication between landlords and tenants, and real-time updates that foster transparency and trust. By bridging the gap between landlords and tenants, HOUSE HUNT eliminates common barriers in the rental process, such as miscommunication or delays, creating a unified digital platform.

The primary goal of HOUSE HUNT is to deliver convenience, trust, and efficiency in one comprehensive solution. This project aspires to redefine how people approach rental property searches, empowering users with accurate information and cutting-edge tools. By combining innovative design with practical usability, HOUSE HUNT promises to transform house hunting into a hassle-free and rewarding experience for all.

## 1.2 Purpose

The primary purpose of **HOUSE HUNT** is to revolutionize the way people search for and list rental properties by offering a centralized and trustworthy digital platform. In an era where renting a home can often be a stressful and time-consuming process, HOUSE HUNT aims to streamline the journey for both property owners and seekers by addressing core challenges and inefficiencies in the current rental market.

Specifically, the objectives of HOUSE HUNT include:

* **Centralized Rental Listings:** To provide a single, easy-to-use digital platform where landlords can list their properties and tenants can explore available options without hopping across multiple platforms or relying on word-of-mouth.
* **Broker-Free Interactions:** To eliminate the reliance on brokers and middlemen by enabling direct communication between property owners and prospective tenants. This helps in reducing additional brokerage fees and ensures transparency in negotiations and agreements.
* **Verified Listings for Trust and Safety:** To promote user trust and safety by ensuring all property listings undergo a verification process. This minimizes the risk of fraud, fake listings, or misleading property information that often misguides tenants.
* **Advanced Filtering and Real-Time Availability:** To empower users with intelligent filtering options based on location, budget, property type (e.g., apartment, shared accommodation, independent house), and real-time availability status, allowing for efficient property discovery.
* **Enhanced User Experience:** To make the renting process faster, more secure, and convenient, especially for users relocating due to job transfers, educational opportunities, or family requirements. The platform is designed with a user-first approach, ensuring intuitive navigation and reliable support.

By meeting these goals, HOUSE HUNT strives to create a transparent, efficient, and user-friendly rental ecosystem that benefits all stakeholders—property owners, tenants, and communities—while fostering trust, speed, and convenience in the home renting experience.

# 2. IDEATION PHASE

## 2.1 Problem Statement

**Customer Problem Statement :**

A customer problem statement helps the team focus on solving real-world issues that target users face while looking for or listing rental properties. These statements were built by putting ourselves in the shoes of tenants and owners, analyzing their behavior, and understanding their frustrations.

Finding and renting a house in today’s fast-paced world can be an overwhelming experience for both tenants and property owners. Existing rental platforms often fail to address the real pain points of users, such as trust, convenience, and transparency. Most tenants struggle to find verified listings, end up browsing multiple unorganized websites, or even fall victim to fake or misleading advertisements. On the other hand, property owners face difficulty in listing their properties, managing tenant communication, and filtering out genuine inquiries from spam.

This project aims to develop a modern, user-friendly house rental platform – **HOUSE HUNT** – using the MERN stack (MongoDB, Express.js, React.js, Node.js). The app will centralize verified property listings, simplify communication between tenants and owners, and ensure transparency and trust throughout the process.

**Key Problems Identified:**

* **Scattered Listings**: Rental listings are distributed across multiple websites with no standardization or centralized access.
* **Lack of Trust**: Users often encounter fake listings or misleading information, leading to safety and financial concerns.
* **Poor Communication**: Tenants and owners rely on calls or third-party brokers, which adds friction and delay.
* **No Real-time Updates**: Listings often remain outdated or unavailable, wasting users’ time and effort.
* **Limited Dashboard for Owners**: Owners have minimal tools to manage their properties, inquiries, or tenant data efficiently.
* **Overwhelming User Experience**: The rental journey becomes frustrating due to information overload and lack of guidance.

**Our Goal:**

To build a **secure, scalable, and intuitive web application** that addresses these problems and makes house hunting and property listing **simpler, safer, and smarter** for everyone involved.

| **Problem Statement (PS)** | **I am (Customer)** | **I’m trying to** | **But** | **Because** | **Which makes me feel** |
| --- | --- | --- | --- | --- | --- |
| **PS-1** | A college student looking for a room near my campus | Find an affordable rental with decent amenities | All listings are scattered across different platforms with no verification | There is no centralized, trustworthy source for verified student housing | Frustrated, overwhelmed, and anxious about being scammed |
| **PS-2** | A property owner with a vacant flat | List my property online to reach potential tenants | The process is complicated and full of spam inquiries | Most platforms have poor filters and no easy-to-use dashboard | Helpless and unsure if I will find genuine tenants |
| **PS-3** | A working professional relocating to a new city | Shortlist houses that match my budget and location preference | I have to call each owner individually to check availability | Current platforms lack real-time updates and modern user experience | Annoyed and drained during an already stressful time |
| **PS-4** | A tenant who used personal contacts to find houses before | Use digital tools this time to make it faster and more reliable | I don’t trust many of the listings or apps I find | There is a lack of verified listings and user reviews | Skeptical and hesitant to try online rental services |

**Key Takeaways from Customer Problem Statements**

* Tenants are looking for **reliability, convenience, and verified listings**.
* Property owners want **ease of listing, control, and fewer fake inquiries**.
* Both parties need a **centralized, trustworthy platform** with real-time updates and good UX.
* Emotional pain points include **frustration, skepticism, stress, and decision fatigue**.

## 2.2 Empathy Map Canvas

**Empathy Map Canvas:**

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user’s behaviours and attitudes.

It is a useful tool to helps teams better understand their users.

Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user’s perspective along with his or her goals and challenges.

An **Empathy Map** is a collaborative visualization tool used to articulate what we know about a particular user type. It helps teams understand and align around the user’s perspective by breaking down their behaviors, thoughts, feelings, and needs. It is often used during the **ideation and discovery phase** of product development to build **user-centered solutions**.

Empathy maps help answer questions such as:

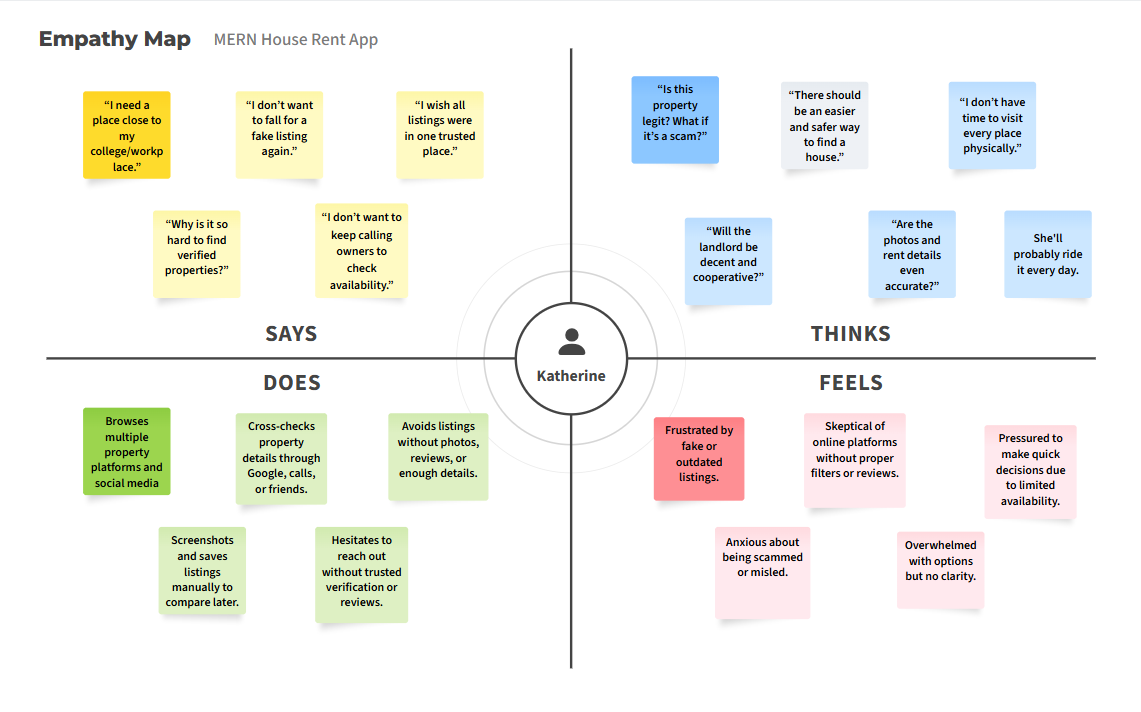
* What is the user experiencing?
* What are they thinking and feeling?
* What are their challenges and motivations?
* How can we best solve their problems?

### ****Application to HOUSE HUNT (House Rent App)****

In the context of your **house rental platform**, the empathy map becomes a **crucial UX tool** to understand the **tenant** (e.g., college students, working professionals) or **property owner** and their journey in renting or listing a property.

## ****Why is the Empathy Map Important for HOUSE HUNT?****

* Helps your **development team build features** that address real user concerns
* Aids in creating a **more intuitive and user-friendly UI/UX**
* Guides your **content and communication strategy** (e.g., what messages users respond to)
* Ensures your solution is **empathetic, practical, and user-centric**, not just technically sound



## ****User Needs (for Tenants)****

These are the essential desires or features that the user expects from a rental platform.

* Verified and trustworthy rental listings
* Filters for location, price, room type, etc.
* Real-time availability and updates
* Photos and accurate descriptions of properties
* Easy communication with landlords (chat/call)
* Mobile-friendly and easy-to-use interface
* Option to bookmark or save listings
* Reviews or ratings from previous tenants
* Transparent pricing (no hidden charges)

## ****Pain Points (for Tenants)****

These represent the challenges or frustrations that users commonly face with current platforms.

* Fake or scam property listings
* Outdated or inaccurate property details
* No centralized source of verified properties
* Poor user interface and complicated navigation
* Lack of direct communication with landlords
* No review system for landlords or properties
* Wasting time calling or visiting unavailable listings
* Security concerns during transactions or site visits

## 2.3 Brainstorming

**Brainstorm & Idea Prioritization :**

The ideation phase lays the foundation for the success of the HOUSE HUNT project. It involves gathering the right team, identifying a real-world problem to solve, listing down ideas to address it, and prioritizing features to deliver a functional and user-friendly solution within a limited timeframe.

**Goal of the Project**

Create a **House Rent Application (HOUSE HUNT)** using the **MERN Stack** (MongoDB, Express.js, React.js, Node.js) that allows users to:

* Search for rental properties based on filters (location, price, amenities, etc.)
* View detailed property information
* Post their property for rent (for owners)
* Contact property owners or agents
* Manage their account (sign up, login, update profile)

**Step-1: Team Gathering, Collaboration and Select the Problem Statement**

**Objective:**

To bring together individuals with diverse skill sets, align on a shared vision, and finalize a real-world problem to address using the MERN stack.

**Actions Taken:**

**Team Formation**: A balanced team was formed consisting of frontend developers, backend developers, and database designers—all passionate about solving real-life problems through web development.

**Skill Mapping**:

* Frontend: React.js, Bootstrap, HTML/CSS
* Backend: Node.js, Express.js
* Database: MongoDB

**Collaboration Tools Used**:

* Communication: WhatsApp, Google Meet
* Project Planning: Trello, Google Docs

**Problem Identification:**

After several rounds of discussion and idea pitching, the team unanimously agreed on solving a common issue: the difficulty in finding and renting houses conveniently and transparently.

**1. Team Formation and Role Distribution**

The foundation of any successful project lies in assembling a well-balanced and dedicated team. For our project titled **HOUSE HUNT – A House Rent Application Using MERN Stack**, we began by identifying team members with strong enthusiasm for web development, practical problem-solving, and interest in building a real-world application from scratch.

Our team comprises four committed members. Each member was assigned a specific role based on their individual skill set, domain knowledge, and preferences. This structured approach ensured a clear division of responsibilities, accountability, and efficient collaboration throughout the development lifecycle.

| **Team Member** | **Assigned Role** | **Key Responsibilities** |
| --- | --- | --- |
| Member 1 | **Project Lead & Full Stack Developer** | Coordinating project timelines, managing GitHub repository, integrating frontend and backend, deployment. |
| Member 2 | **Frontend Developer** | Designing responsive user interfaces using React.js, managing UI/UX flow, integrating API data, using Tailwind CSS for styling |
| Member 3 | **Backend Developer** | Developing RESTful APIs using Node.js and Express.js, implementing authentication and authorization (JWT), business logic |
| Member 4 | **Database Manager & QA Tester** | Designing MongoDB database schemas, managing data models, writing queries, performing manual testing and reporting bugs |

We adopted a collaborative model that allowed flexibility and learning opportunities. Members often cross-collaborated across modules to support one another, strengthening teamwork and overall output.

**2. Collaboration Tools and Workflow**

To ensure seamless communication and progress tracking, especially in a hybrid working environment, we utilized a range of collaboration tools. These helped us stay aligned with project goals, deadlines, and updates:

**Git & GitHub**: For version control, collaborative code reviews, branching strategies, and pull requests.

**Google Meet / Whatsapp** : For regular virtual meetings, daily/weekly stand-ups, and technical discussions.

**Google Drive / Docs**: For storing and sharing important project-related documents, reports, and references.

We followed a simplified agile approach with iterative development and continuous integration. Weekly check-ins helped us identify blockers early, track progress, and maintain momentum.

**3. Brainstorming and Problem Selection**

Our initial goal was to identify a real-world problem that could be solved effectively using the MERN stack. We conducted several brainstorming sessions where each team member proposed different ideas from areas such as:

* Healthcare management systems
* Online education platforms
* E-commerce stores for small businesses
* Rental property listing and management platforms
* Local job portals and community service apps

After evaluating these ideas based on feasibility, scope, relevance, and innovation, we collectively agreed upon the **Online House Rental Management System** as the most promising and impactful choice.

**4. Why We Chose This Problem**

The decision to work on a house rent application was driven by the following key factors:

**High Practical Relevance**: With increasing urban migration and digitization, there is a growing demand for online rental platforms, especially among students and working professionals relocating to new cities.

**Market Gap**: Most existing rental portals are either paid, limited in scope, or difficult to use for individual landlords. Our platform aims to bridge this gap by offering a simple, user-friendly, and accessible solution for both tenants and property owners.

**Technology Fit**: The problem is ideal for implementing CRUD operations, user authentication, RESTful API design, and MongoDB-based data management — perfectly aligning with the MERN stack.

**Scalability and Innovation Potential**: This system can be expanded in the future with features such as:

* Online payment integration (e.g., Razorpay, Stripe)
* Google Maps API for property location tracking
* Messaging/chat system between tenants and owners
* Admin analytics and property verification dashboard
* Mobile app version using React Native

**5. Final Problem Statement**

**"To design and develop a responsive, secure, and scalable web application using the MERN stack that enables house owners to list rental properties and allows tenants to search, filter, view, and inquire about properties, thereby simplifying the rental process and minimizing dependency on third-party agents."**

This problem not only allowed us to apply our technical knowledge in full-stack development but also inspired us to build something that could be expanded and deployed in a real-world setting.

**Step-2: Brainstorm, Idea Listing and Grouping**

Once the team was formed and the problem statement was finalized, the next crucial step in the project development process was to brainstorm different ideas related to the solution, list those ideas collaboratively, and group them into meaningful categories. This step helped us visualize the complete scope of the application, prioritize features, and streamline the development process.

**1. Purpose of the Brainstorming Session**

The goal of the brainstorming session was to:

* Understand what functionalities users would expect from a house rental platform.
* Identify all possible features that can enhance user experience and platform efficiency.
* Break down the problem into smaller manageable modules for better planning and development.
* Align everyone's vision regarding the project scope and deliverables.

**2. Brainstorming Methodology**

We conducted a series of brainstorming sessions over virtual meetings using tools like:

* **Miro** for real-time collaborative whiteboarding and mind mapping.
* **Google Docs** for live note-taking and idea capturing.
* **Trello** for organizing features into categories and prioritizing tasks.

Each team member contributed ideas based on:

* Personal experiences
* Existing popular rental apps (like NoBroker, Housing.com, MagicBricks, etc.)
* User expectations and pain points

We used a **Round-Robin format** where every team member was encouraged to present one or more ideas at a time. No idea was considered irrelevant or dismissed immediately — everything was listed for review.

**3. Idea Listing**

Below is the comprehensive list of ideas/features that emerged during the brainstorming phase:

**Core Features:**

* Property listing by owners
* Tenant registration/login
* Property search with filters (location, price, number of rooms)
* Property detail view with photos, amenities, and descriptions
* Owner contact or inquiry form
* User dashboard for both tenants and owners
* Property status (available/booked)

**Extended Features:**

* User reviews and ratings on properties
* Save/favorite properties for later
* Admin panel to manage listings and users
* Push notifications (e.g., new listings in user’s area)
* Email confirmation on inquiry or registration
* OTP-based login or multi-factor authentication

**Future Enhancement Ideas:**

* Online rent agreement generation
* Integrated payment gateway (Razorpay, Stripe)
* Chat system between tenant and owner
* Google Maps integration for viewing location
* Recommendation system based on user search history
* Mobile app version (React Native or Flutter)

**4. Grouping of Ideas**

After listing all possible ideas, we organized them into logical **feature groups** for better clarity and to ease project execution. This step was critical for defining the Minimum Viable Product (MVP) and planning development sprints.

| **Category** | **Features Grouped** |
| --- | --- |
| **User Management** | Registration, Login, Logout, Dashboard, Profile Management |
| **Property Listings** | Add Property, View Property, Edit/Delete Listings, Add Photos, Property Status |
| **Search & Filters** | Search by location, price range, bedrooms, availability |
| **Inquiries & Contact** | Inquiry form, Contact owner, Email confirmation, Inquiry history |
| **Admin Operations** | User monitoring, Property moderation, Feedback & review management |
| **Security** | JWT authentication, Password encryption, Role-based access |
| **Advanced Features** | Favorites, Reviews, Ratings, Notifications, Chat system (Future), Payment gateway |

**5. Key Takeaways from This Phase**

We successfully transformed a broad problem statement into well-defined, actionable modules.

Prioritized **core functionalities** that are essential for launch, and documented **advanced features** for future expansion.

Ensured everyone in the team had a shared understanding of what needs to be built and how we’ll divide it across upcoming development phases.

Established the foundation for **Requirement Analysis**, **System Architecture Design**, and **Sprint Planning** in the next phases.

**Step-3: Idea Prioritization**

After brainstorming and organizing a comprehensive list of ideas and features in Step 2, the next logical step in our development process was **prioritizing these ideas**. This stage was essential to identify which features to develop first (Minimum Viable Product – MVP), which ones to add later (Post-MVP or V2), and which ones to consider as long-term enhancements or stretch goals.

* Effective idea prioritization ensured that:
* The project remained manageable within the given timeline and resources.
* Core user needs were addressed from the start.
* The development process followed a clear, goal-driven roadmap.

**1. Prioritization Approach**

To determine what features should be prioritized for development, we adopted a combination of two popular techniques:

**a. MoSCoW Method**

We classified each idea into four categories:

**Must Have** – Critical features required for the app to function.

**Should Have** – Important features that enhance user experience but are not critical for MVP.

**Could Have** – Nice-to-have features that can be added if time/resources allow.

**Won’t Have (for now)** – Features we decided to postpone or not include in this version.

**b. Value vs. Effort Matrix**

Each feature was analyzed based on:

**Value** to the end-users (usability, necessity, impact)

**Effort** required to implement (time, complexity, team skill)

This allowed us to balance our work between quick wins, high-impact features, and manageable complexities.

**2. Prioritized Feature List (with Justifications)**

| **Feature** | **Priority** | **Justification** |
| --- | --- | --- |
| User Registration and Login | Must Have | Basic entry point for all users; essential for access control |
| JWT Authentication | Must Have | Secures APIs and protects user data |
| Owner Dashboard | Must Have | Allows property owners to add/manage listings |
| Add/View/Edit/Delete Property | Must Have | Core CRUD functionality; enables main feature of the platform |
| Property Detail Page | Must Have | Users must see full details before making inquiries |
| Search and Filters | Must Have | Helps users find relevant listings quickly |
| Inquiry/Contact Form | Must Have | Enables tenant-owner communication |
| Email Confirmation for Inquiry | Should Have | Adds professionalism and ensures communication reliability |
| Favorites (Save Property) | Should Have | Enhances user convenience |
| Admin Dashboard | Should Have | Helps in content moderation and managing abuse |
| Reviews and Ratings | Could Have | Adds value but can be added post-MVP |
| Chat System | Could Have | Requires WebSockets or Firebase; good for future versions |
| OTP Login / Multi-Factor Auth | Could Have | Adds extra security; useful in real-world scenarios |
| Online Rent Agreement | Won’t Have | Requires legal template integration; out of scope for current version |
| Payment Gateway Integration | Won’t Have | Complex to implement and test within current timeline |

**3. Final MVP Feature Set**

Based on the prioritization, the following features were **locked in as MVP (Minimum Viable Product)**:

* User Registration & Login (Tenant/Owner)
* JWT Authentication
* Owner Property Listing (Add/Edit/Delete)
* Property Search with Filters
* Property Details Page
* Inquiry Form (with optional email notification)
* Tenant Dashboard to view inquiries and saved properties

This MVP would allow us to deliver a fully functional product that solves the primary pain points of both property owners and tenants.

**4. Post-MVP Planning**

We also created a **Phase 2 backlog** to record "Should Have" and "Could Have" features, which could be implemented after the initial launch if time permits. This list was added to Trello for tracking and sprint planning purposes.

These features include:

* Admin Panel
* Save/Favorite Properties
* Google Maps Integration
* Property Reviews
* Chat Functionality

**5. Outcome of the Prioritization Phase**

We defined a **clear roadmap** that separates core from auxiliary features.

The team could now focus on the most **impactful and feasible deliverables**.

Helped manage scope and timeline, avoiding overengineering or feature creep.

Set the stage for **requirement analysis, UI wireframing, and architecture design**.

# 3. REQUIREMENT ANALYSIS

## 3.1 Customer Journey map

## 3.2 Solution Requirement

**Functional Requirements:**

Following are the functional requirements of the proposed solution.

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | User Registration | Registration via Email Registration via Gmail Registration via Phone Number |
| FR-2 | User Login | Login via Email Login via OTP Remember Me Functionality |
| FR-3 | Property Listings | Landlords can post property Add photos, descriptions, rent amount, location Edit/Delete listings |
| FR-4 | Property Search | Search by location, price, type Advanced filters (furnished, number of rooms, pet-friendly) |
| FR-5 | Booking & Scheduling Visits | Tenants can request a visit Landlords approve visit slots Calendar integration |
| FR-6 | Favorites | Save listings to favorites View/manage saved properties |
| FR-7 | Reviews & Ratings | Tenants can review landlords Landlords can rate tenants |
| FR-8 | Notifications | Email/Push notifications for booking status New listings alerts |
| FR-9 | Admin Panel | Manage users Approve/reject listings View reports |
| FR-10 | Chat System | Direct chat between tenant and landlord File/image sharing support |

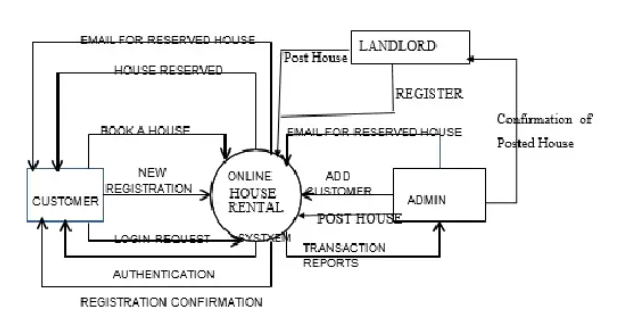
**Non-functional Requirements:**

Following are the non-functional requirements of the proposed solution.

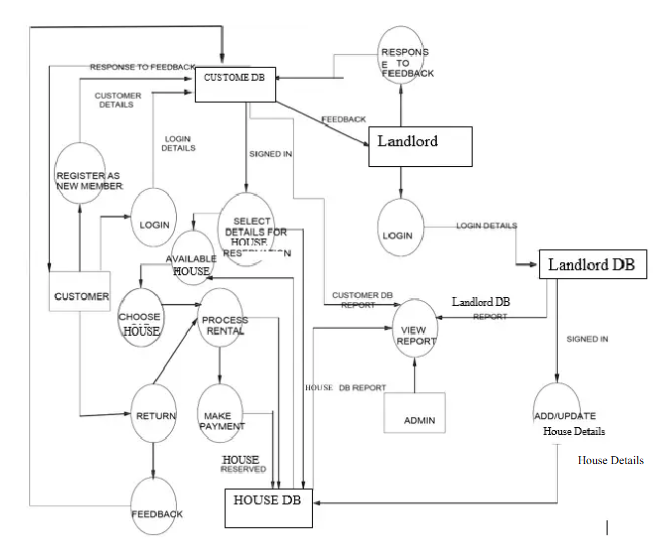
|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | Usability | The UI should be clean, intuitive, and mobile-responsive |
| NFR-2 | Security | Data encryption, secure authentication, and role-based access control |
| NFR-3 | Reliability | Ensure data accuracy and uptime through robust backend and monitoring |
| NFR-4 | Performance | Fast load times (<2 seconds), optimized image loading |
| NFR-5 | Availability | 99.9% uptime, with auto-scaling on traffic surge |
| NFR-6 | Scalability | Support for thousands of concurrent users and listings |

## 3.3 Data Flow Diagram

* **Data Flow Diagrams:**

****

**Level 0 DFD of Online House Rental System**

****

**Level 1 DFD of Online House Rental System**

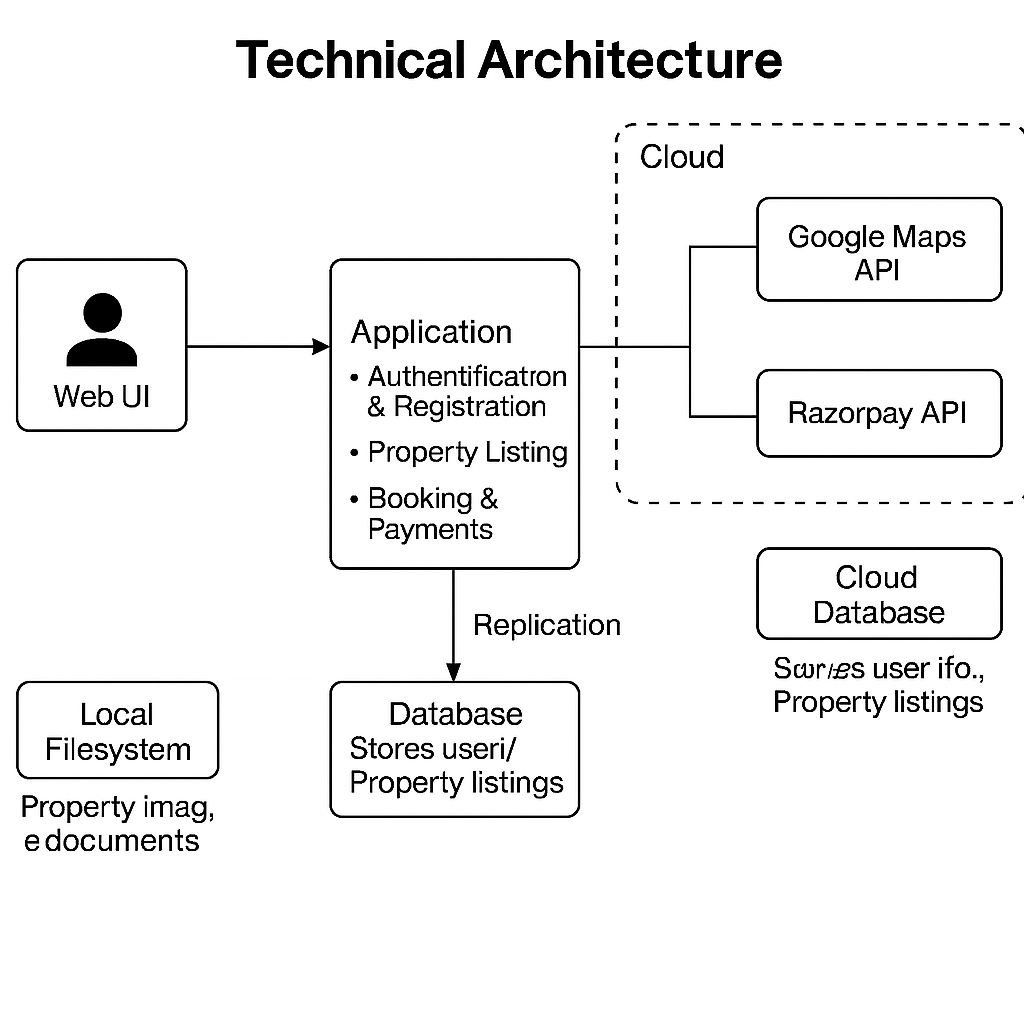
**User Stories**

Use the below template to list all the user stories for the product.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Type** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Acceptance Criteria** | **Priority** | **Release** |
| Customer (Web/App) | Registration | USN-1 | As a user, I can register with name, email, password, and phone number | I receive confirmation email and can log in | High | Sprint-1 |
|  |  | USN-2 | As a user, I can sign up using Google or Facebook | I can log in via social accounts | Medium | Sprint-2 |
|  | Login | USN-3 | As a user, I can log in using email and password | Dashboard access after successful login | High | Sprint-1 |
|  | Property Search | USN-4 | As a user, I can search properties by location, rent range, and type | List of matching properties is shown | High | Sprint-1 |
|  | Booking Request | USN-5 | As a user, I can send a request to book a selected property | Owner receives notification and I get booking status | High | Sprint-2 |
|  | Payments | USN-6 | As a user, I can pay booking amount via UPI, card, or net banking | Payment success message and receipt shown | Medium | Sprint-3 |
| Property Owner | Property Listing | USN-7 | As an owner, I can list a property with rent, address, photos, and availability | Property visible to users after admin approval | High | Sprint-2 |
|  | Booking Approval | USN-8 | As an owner, I can approve or reject booking requests | Customer gets notified of approval/rejection | High | Sprint-3 |
| Admin | User Management | USN-9 | As an admin, I can approve new user and owner registrations | Users can access platform only after approval | High | Sprint-1 |
|  | Listing Moderation | USN-10 | As an admin, I can review and approve all property listings before they go live | Only approved listings appear on the site | High | Sprint-2 |
|  | Booking Dispute Handling | USN-11 | As an admin, I can manage and resolve complaints from users or owners | Dispute resolved, both parties informed | Medium | Sprint-3 |

## 3.4 Technology Stack

* **Technical Architecture:**



**Table-1: Components & Technologies:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
| 1 | User Interface | Web UI and responsive Mobile Web | HTML, CSS, JavaScript, React.js |
| 2 | Application Logic-1 | User authentication & registration | Node.js / Express.js |
| 3 | Application Logic-2 | Property search, filtering, and listing | Node.js / Express.js |
| 4 | Application Logic-3 | Booking and payment module | Razorpay API + Node.js |
| 5 | Database | Stores user info, property listings, bookings | MongoDB (NoSQL) |
| 6 | Cloud Database | Cloud-hosted version for scaling | MongoDB Atlas |
| 7 | File Storage | Property images and documents | AWS S3 / Local Filesystem |
| 8 | External API-1 | Geolocation for property address mapping | Google Maps API |
| 9 | External API-2 | Payment gateway integration | Razorpay API |
| 10 | Machine Learning Model | Recommendation of listings based on user interest (optional feature) | Collaborative Filtering Model (Python) |
| 11 | Infrastructure | Cloud deployment of entire application | AWS EC2 / Docker / Kubernetes |

**Table-2: Application Characteristics:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Characteristics** | **Description** | **Technology** |
| 1 | Open-Source Frameworks | React.js, Node.js, Express.js | JavaScript (ME\_N\_ Stack) |
| 2 | Security Implementations | SHA-256 hashing for passwords, JWT tokens, HTTPS, Role-based access control | Bcrypt, JWT, HTTPS, Helmet.js |
| 3 | Scalable Architecture | Supports horizontal scaling, containerized microservices | Docker, Kubernetes, 3-tier architecture |
| 4 | Availability | Load balanced architecture, auto-scaling group | AWS ELB, Auto Scaling |
| 5 | Performance | CDN for static assets, caching for repeated search queries | Cloudflare CDN, Redis for caching |

**References:**

**<https://c4model.com/>**

**<https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/>**

**<https://www.ibm.com/cloud/architecture>**

**<https://aws.amazon.com/architecture>**

**<https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d>**

# 4. PROJECT DESIGN

## 4.1 Problem Solution Fit

**Problem – Solution Fit Template:**

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer’s problem. It helps entrepreneurs, marketers and corporate innovators identify behavioral patterns and recognize what would work and why

**Purpose:**

* Solve complex problems in a way that fits the state of your customers.
* Succeed faster and increase your solution adoption by tapping into existing mediums and channels of behavior.
* Sharpen your communication and marketing strategy with the right triggers and messaging.
* Increase touch-points with your company by finding the right problem-behavior fit and building trust by solving frequent annoyances, or urgent or costly problems.
* **Understand the existing situation in order to improve it for your target group.**

**Template:**

Calendar

Description automatically generated

| **Section** | **Details (Based on HOUSE HUNT project)** |
| --- | --- |
| **1. Customer Segment(s) (CS)** | - Working professionals looking for rental homes - College students - Landlords wanting to list properties - Families shifting to new cities |
| **2. Jobs-To-Be-Done / Problems (J&P)** | - Difficulty in finding verified rental houses quickly - Lack of trust in listings - No easy way to communicate with landlords - Inadequate filters to search desired homes |
| **3. Triggers (TR)** | - Urgent job relocation - Semester beginning for students - Eviction or lease expiry - Seeing a friend use a house rental app |
| **4. Emotions: Before / After (EM)** | - **Before:** Stressed, unsure, fearful of scams, frustrated with house hunting - **After:** Confident, informed, secure, in control |
| **5. Available Solutions (AS)** | - Facebook housing groups, WhatsApp groups, OLX, 99acres, Housing.com - Brokers/agents (offline) - Pros: Wide reach - Cons: Unverified, not real-time, commission-based |
| **6. Customer Constraints (CC)** | - No time for physical house hunting - Limited tech skills (for some) - Poor internet in remote areas - Budget constraints - Device availability |
| **7. Behaviour (BE)** | - Searching listings on social media - Asking friends and family - Visiting areas physically - Posting in online groups |
| **8. Channels of Behaviour (CH)** | **8.1 Online** - Social media (Facebook groups) - Property portals - University/office forums  **8.2 Offline** - Agents - Word of mouth - Local newspapers |
| **9. Problem Root Cause (RC)** | - Unorganized rental ecosystem - Lack of centralized trusted rental platform - Information not verified or outdated - No bridge between landlords and tenants |
| **10. Your Solution (SL)** | - HOUSE HUNT: A MERN stack-based web app that enables verified property listings, secure registration, direct communication, filters, and real-time status updates |

References:

1. [https://www.ideahackers](https://www.ideahackers.network/problem-solution-fit-canvas/).network/problem-solution-fit-canvas/

## <https://medium.com/@epicantus/problem-solution-fit-canvas-aa3dd59cb4fe>

## 4.2 Proposed Solution

**Proposed Solution : House Hunt**

| **S.No.** | **Parameter** | **Description** |
| --- | --- | --- |
| 1. | **Problem Statement** | Finding affordable rental houses, verifying property details, and securely connecting landlords with tenants is often time-consuming, unreliable, and unorganized. |
| 2. | **Idea / Solution Description** | HOUSE HUNT is a MERN-based web application that simplifies house renting by offering verified listings, tenant-landlord chat, filter-based search, and user-friendly registration and login systems. It integrates secure authentication and real-time notifications to improve user experience. |
| 3. | **Novelty / Uniqueness** | Unlike traditional listing platforms, HOUSE HUNT incorporates real-time availability status, admin verification of listings, in-app messaging, and smart filters for precise house searching. |
| 4. | **Social Impact / Customer Satisfaction** | The platform ensures safety, transparency, and convenience for both landlords and tenants. It reduces fraud, saves time, and increases trust—especially beneficial for students and working professionals relocating to new cities. |
| 5. | **Business Model (Revenue Model)** | Revenue can be generated through featured property listings, subscription plans for premium users, advertisements, and partnerships with moving services and real estate agents. |
| 6. | **Scalability of the Solution** | The solution is highly scalable due to its modular MERN architecture. It can be extended to support mobile platforms, multiple cities, commercial rentals, and third-party integrations. |

## 4.3 Solution Architecture

**Solution Architecture:**

### **Solution Architecture Overview**

**Solution Architecture** is a strategic process that ensures your technology choices align effectively with business needs. For the HOUSE HUNT project, this approach helps deliver a scalable, secure, and user-friendly rental management system.

#### **Purpose of Solution Architecture in HOUSE HUNT**

* **Identify Best Tech Solution:** Leverage the MERN stack (MongoDB, Express.js, React.js, Node.js) to build a responsive, real-time web application tailored to house rentals.
* **Communicate with Stakeholders:** Visualize and communicate how users interact with the system, what data is processed, and how the backend supports these operations.
* **Define Features & Phases:** Clearly structure the development timeline through sprints—starting from registration, login, and listing to admin panel and search filters.
* **Deliver Specifications:** Provide technical documentation including APIs, data models, authentication mechanisms, and UI/UX workflows.

### **Key Components of the Solution Architecture**

| **Component** | **Description** |
| --- | --- |
| **Frontend (React.js)** | Delivers a dynamic and responsive UI for tenants and landlords. |
| **Backend (Node.js + Express.js)** | Handles API routing, business logic, and user authentication. |
| **Database (MongoDB)** | Stores user data, property listings, bookings, chat history, etc. |
| **Authentication** | JWT-based secure login and signup for users (email, Gmail, Facebook). |
| **Admin Panel** | Enables admin to verify listings and manage reported properties. |
| **Search & Filters** | Allows users to find homes based on price, location, and preferences. |
| **Chat Feature** | Real-time chat between tenants and landlords (optional WebSocket integration). |
| **Deployment** | Hosted on cloud platforms (like Vercel/Netlify for frontend, and Render/Heroku for backend). |

### **Development Phases**

1. **Sprint 1:** Registration, login, email verification.
2. **Sprint 2:** Property listing, homepage UI, integration with MongoDB.
3. **Sprint 3:** Advanced search filters, admin panel setup.
4. **Sprint 4:** Final integration, testing, chat feature (if feasible), and deployment.

**Example - Solution Architecture Diagram:**

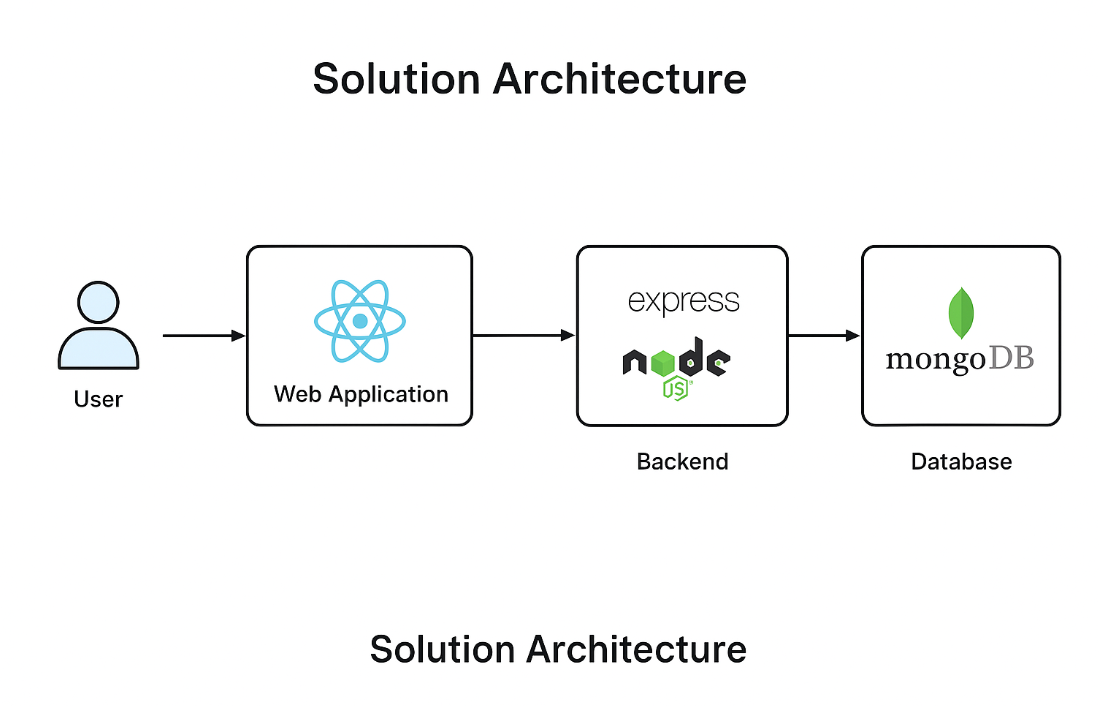
****

Figure 1: Architecture and data flow of the voice patient diary sample application

**Reference: <https://aws.amazon.com/blogs/industries/voice-applications-in-clinical-research-powered-by-ai-on-aws-part-1-architecture-and-design-considerations/>**

# 5. PROJECT PLANNING & SCHEDULING

## 5.1 Project Planning

**Product Backlog, Sprint Schedule, and Estimation (4 Marks)**

Use the below template to create product backlog and sprint schedule

| **Sprint** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Story Points** | **Priority** | **Team Members** |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-1 | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password. | 2 | High | Anshika |
| Sprint-1 | Registration | USN-2 | As a user, I will receive a confirmation email once I have registered for the application. | 1 | High | Anshika |
| Sprint-1 | Registration | USN-4 | As a user, I can register for the application through Gmail. | 2 | Medium | Anshika |
| Sprint-1 | Login | USN-5 | As a user, I can log into the application by entering email & password. | 1 | High | Anukul |
| Sprint-2 | Registration (Social Media) | USN-3 | As a user, I can register for the application through Facebook. | 2 | Low | Anukul |
| Sprint-2 | Dashboard | USN-6 | As a user, I can view my profile dashboard after logging in. | 3 | High | Anukul |
| Sprint-2 | Dashboard | USN-7 | As a user, I can see recent property listings on the dashboard. | 2 | Medium | Viraj |
| Sprint-3 | Property Listings | USN-8 | As a user, I can filter property listings by price, location, and amenities. | 3 | High | Viraj |
| Sprint-3 | Property Listings | USN-9 | As a user, I can view details, images, and availability of a property. | 2 | High | Yash |
| Sprint-4 | Booking | USN-10 | As a user, I can request to book a property and schedule a visit. | 3 | High | Yash |
| Sprint-4 | Chat | USN-11 | As a user, I can chat with the property owner for further queries. | 3 | Medium | Yash |

**Project Tracker, Velocity & Burndown Chart: (4 Marks)**

| **Sprint** | **Total Story Points** | **Duration** | **Sprint Start Date** | **Sprint End Date (Planned)** | **Story Points Completed (as on Planned End Date)** | **Sprint Release Date (Actual)** |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-1 | 10 | 5 Days | 2 Apr 2025 | 6 Apr 2025 | 10 | 6 Apr 2025 |
| Sprint-2 | 12 | 6 Days | 7 Apr 2025 | 12 Apr 2025 | 12 | 12 Apr 2025 |

**Velocity Calculation**

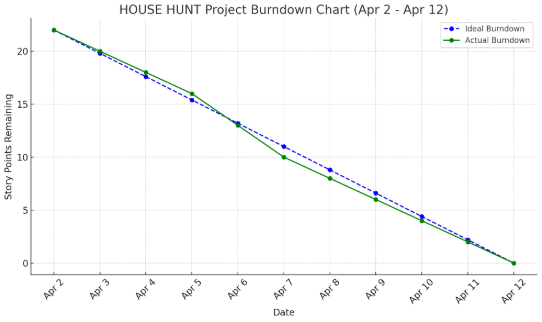
* **Sprint-1 Velocity:** 10 story points / 5 days = **2 points per day**
* **Sprint-2 Velocity:** 12 story points / 6 days = **2 points per day**

**Average Velocity:** **2 story points/day**

**Burndown Chart:**

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile[software development](https://www.visual-paradigm.com/scrum/what-is-agile-software-development/) methodologies such as [Scrum](https://www.visual-paradigm.com/scrum/scrum-in-3-minutes/). However, burn down charts can be applied to any project containing measurable progress over time.

**Burndown Chart** for the HOUSE HUNT project. It visually shows both the **ideal progress** (dashed blue line) and your **actual progress** (green line) from April 2 to April 12. The team is on track to complete all story points by April 12.



**Reference:**

**<https://www.atlassian.com/agile/project-management>**

**<https://www.atlassian.com/agile/tutorials/how-to-do-scrum-with-jira-software>**

**<https://www.atlassian.com/agile/tutorials/epics>**

**<https://www.atlassian.com/agile/tutorials/sprints>**

**<https://www.atlassian.com/agile/project-management/estimation>**

**<https://www.atlassian.com/agile/tutorials/burndown-charts>**

# 6. FUNCTIONAL AND PERFORMANCE TESTING

## 6.1 Performance Testing

Functional testing is a type of software testing that validates the software system against the functional requirements or specifications. The purpose of functional testing in the **HOUSE HUNT** project is to ensure that every feature of the application performs exactly as expected from the user’s perspective.

#### ****Key Objectives:****

* To verify that all functionalities of the application are working as intended.
* To test the user interface, APIs, database, security, and client/server communications.
* To ensure that user flows such as registration, login, property listing, and search yield correct results.

#### ****Test Scenarios Covered:****

* **User Registration:** Verifying if new users can register with valid data and appropriate error messages are displayed for invalid inputs.
* **Login Functionality:** Testing successful login with correct credentials and appropriate feedback for incorrect attempts.
* **Email Confirmation:** Ensuring that confirmation emails are sent and verified before account activation.
* **Property Listing:** Validating that landlords can post new rental listings with all necessary details.
* **Search & Filters:** Testing the ability to filter and search properties based on city, price, availability, and house type.
* **Contacting Owner:** Checking if the user can directly contact the property owner through the platform.
* **User Dashboard:** Verifying the display of user-specific data and options such as saved listings or posted properties.
* **Logout:** Ensuring secure and smooth logout operation.

#### ****Tools Used for Functional Testing:****

* Manual Testing
* Postman (for testing API responses)
* Selenium (optional automation, if applicable)

Performance testing is conducted to determine how the application behaves under load and stress. The aim is to measure the responsiveness, stability, scalability, and resource usage of **HOUSE HUNT** under expected and peak traffic conditions.

#### ****Key Objectives:****

* To ensure the system can handle multiple users accessing or modifying data simultaneously.
* To identify bottlenecks and performance issues in various modules like search, listing uploads, and dashboard loading.
* To measure the response time, throughput, and server behavior under different conditions.

#### ****Types of Performance Testing Performed:****

* **Load Testing:** Checked how the system handles a normal expected load (e.g., 100 simultaneous users searching properties).
* **Stress Testing:** Evaluated how the system behaves under peak load (e.g., 1000 concurrent requests during festive/academic admission periods).
* **Spike Testing:** Tested performance impact due to sudden surges in user activity.
* **Endurance Testing:** Checked for memory leaks and system stability during extended use.

#### ****Key Metrics Evaluated:****

* **Response Time:** Time taken to return results for user queries.
* **Throughput:** Number of requests handled per second.
* **Server Resource Usage:** CPU, memory, and database usage trends under load.
* **Error Rate:** Frequency of failed transactions or dropped requests.

#### ****Tools Used for Performance Testing:****

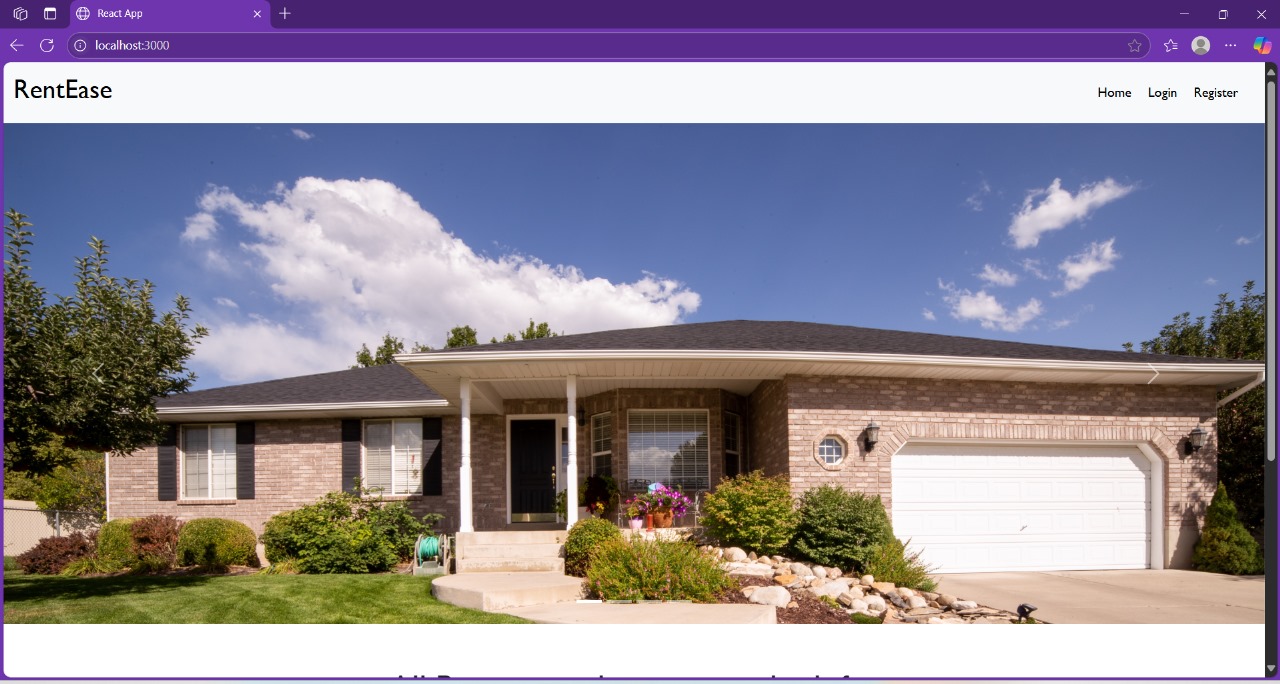
* JMeter (for simulating multiple users)
* Google Chrome DevTools (for client-side performance and load times)
* Lighthouse (to evaluate page performance and best practices)

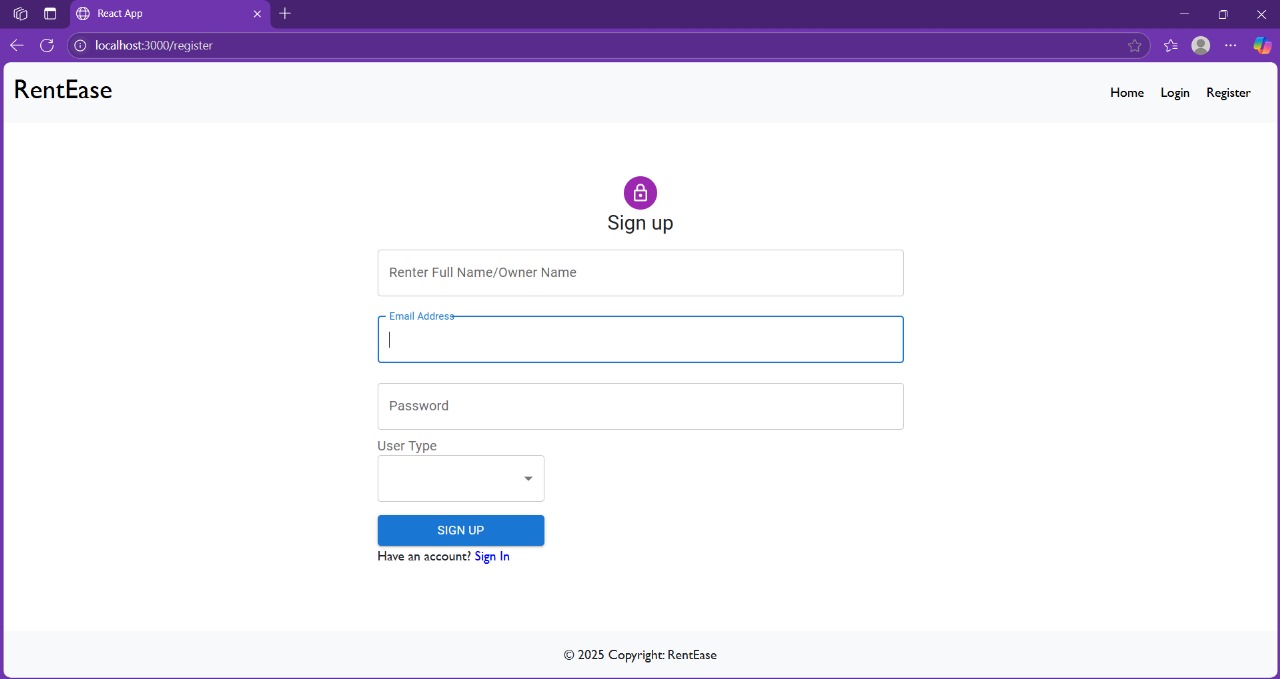
### ****Summary:****

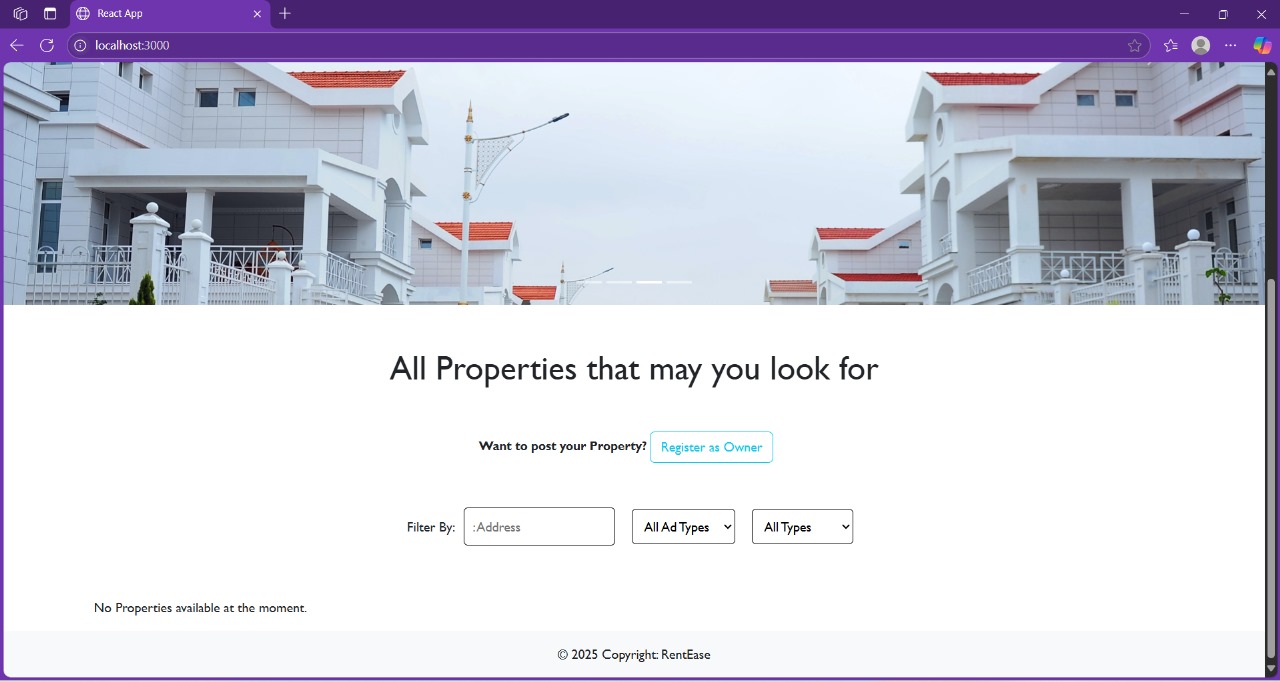
The **HOUSE HUNT** application passed all functional tests for its core features and demonstrated stable performance under normal load conditions. Future scalability testing will be conducted as more users are onboarded to the platform. This phase of testing ensures the product is both **functionally accurate** and **performance-optimized** for real-world use.

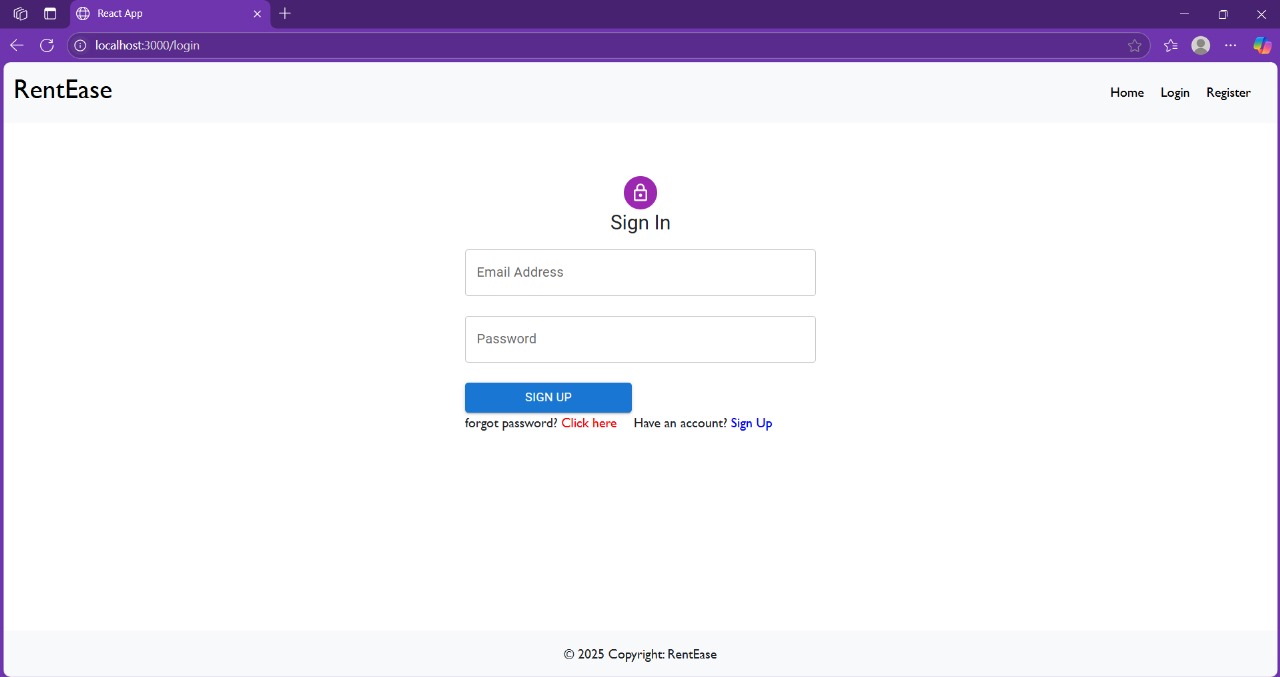
# 7. RESULTS

## 7.1 Output Screenshots











# 8. ADVANTAGES & DISADVANTAGES

### ****Advantages:****

* **Broker-Free Communication:** Direct interaction between landlords and tenants removes the need for brokers, saving both time and money.
* **Verified Listings:** Every property listed is verified, reducing the chances of scams and ensuring safety for renters.
* **Real-Time Search & Filters:** Users can filter properties based on availability, budget, location, and property type, offering a highly personalized experience.
* **User-Friendly Interface:** The application is designed to be intuitive and easy to use for all age groups, even those with minimal tech experience.
* **Time-Efficient:** The digital platform simplifies the process of finding a rental property, which traditionally is time-consuming and stressful.
* **Cost Savings:** By eliminating third-party commissions and reducing paperwork, users save significantly.
* **Accessibility:** The platform is accessible from anywhere, allowing users to search or list properties even before relocating.
* **Secure Data Handling:** The system ensures the privacy and security of user data through authentication and authorization features.

### ****Disadvantages:****

* **Internet Dependency:** Users must have an active internet connection to access the platform, which may be a limitation in rural or low-connectivity areas.
* **Initial Trust Barrier:** Some users may still prefer brokers due to a lack of trust in online platforms, especially for high-value transactions like renting homes.
* **Technical Glitches:** As with any digital platform, bugs or downtime can temporarily affect usability and user experience.
* **Limited Personalization:** While the app offers filtering, it may lack the human touch and negotiation flexibility that brokers sometimes offer.
* **Verification Challenges:** Ensuring all listings are genuinely verified requires continuous monitoring, which can be resource-intensive.

# 9. CONCLUSION

### ****Advantages:****

* **Broker-Free Communication:** Direct interaction between landlords and tenants removes the need for brokers, saving both time and money.
* **Verified Listings:** Every property listed is verified, reducing the chances of scams and ensuring safety for renters.
* **Real-Time Search & Filters:** Users can filter properties based on availability, budget, location, and property type, offering a highly personalized experience.
* **User-Friendly Interface:** The application is designed to be intuitive and easy to use for all age groups, even those with minimal tech experience.
* **Time-Efficient:** The digital platform simplifies the process of finding a rental property, which traditionally is time-consuming and stressful.
* **Cost Savings:** By eliminating third-party commissions and reducing paperwork, users save significantly.
* **Accessibility:** The platform is accessible from anywhere, allowing users to search or list properties even before relocating.
* **Secure Data Handling:** The system ensures the privacy and security of user data through authentication and authorization features.

### ****Disadvantages:****

* **Internet Dependency:** Users must have an active internet connection to access the platform, which may be a limitation in rural or low-connectivity areas.
* **Initial Trust Barrier:** Some users may still prefer brokers due to a lack of trust in online platforms, especially for high-value transactions like renting homes.
* **Technical Glitches:** As with any digital platform, bugs or downtime can temporarily affect usability and user experience.
* **Limited Personalization:** While the app offers filtering, it may lack the human touch and negotiation flexibility that brokers sometimes offer.
* **Verification Challenges:** Ensuring all listings are genuinely verified requires continuous monitoring, which can be resource-intensive.

# 10. FUTURE SCOPE

The **HOUSE HUNT** platform has laid a strong foundation for transforming the house rental experience. While the current version fulfills essential functionalities like user registration, login, property listing, and search filtering, there are numerous opportunities for future enhancements that can significantly elevate user experience, scalability, and commercial viability. The following outlines the potential future scope of the project:

### ****1. Integration of AI and Machine Learning****

* **Smart Recommendations:** Use AI algorithms to suggest properties based on user preferences, search history, and behavioral patterns.
* **Dynamic Pricing Analysis:** Implement ML models to predict and suggest optimal rental pricing based on market trends and location analytics.

### ****2. In-App Chat and Communication System****

* Enable real-time communication between landlords and tenants through secure, in-app messaging to enhance interaction and response time.

### ****3. Online Rental Agreement and E-signature****

* Integrate a legal document management system that allows tenants and landlords to sign digital rental agreements through secure e-signature services.

### ****4. Advanced Verification and Fraud Detection****

* Implement KYC (Know Your Customer) verification methods, such as document uploads and biometric checks, to ensure authenticity of users and listings.
* Use anomaly detection systems to flag and prevent fraudulent activities.

### ****5. In-app Payment Gateway Integration****

* Add secure payment gateways to handle advance deposits, rent payments, and service charges within the application.

### ****6. Mobile Application Development****

* Expand the platform to Android and iOS for a seamless experience on mobile devices, thereby increasing reach and accessibility.

### ****7. Location-based Services****

* Integrate map services like Google Maps or Mapbox to provide users with geolocation-based property suggestions and route planning.

### ****8. Review and Rating System****

* Introduce user-generated reviews and ratings for both landlords and tenants to increase transparency and trustworthiness of listings.

### ****9. Multilingual Support****

* Add support for multiple languages to make the platform more inclusive for users from different regions and linguistic backgrounds.

### ****10. Cloud Hosting and Deployment****

* Migrate to scalable cloud services (like AWS, Azure, or Google Cloud) to handle large-scale traffic and ensure high availability.

### ****11. Subscription-based Business Model****

* Introduce premium features through a subscription model such as highlighted listings, detailed analytics, and faster verification for landlords.

# 11. APPENDIX

## GitHub **Link**

**Github Link : https://github.com/Thanuja-Gogada12/HOUSE-HUNT**