**Project Report**

**Househunt-Finding Your Perfect Rental Home**

**1. INTRODUCTION**

**1.1 Project Overview**

HouseHunt is a web-based platform that simplifies the process of renting properties by connecting renters and property owners in a secure and transparent environment. It enables users to browse verified listings, communicate with owners, and manage bookings online.

**1.2 Purpose**

The purpose of this project is to create a reliable and user-friendly system that addresses common issues in property rentals, such as unverified listings, lack of trust, and inefficient communication, while streamlining the entire rental workflow.

**2. IDEATION PHASE**

**2.1 Problem Statement**

Renters often face difficulties finding authentic rental listings and verifying property details. Property owners struggle to manage listings, respond to inquiries, and select trustworthy tenants. The current process lacks transparency and convenience.

**2.2 Empathy Map Canvas**

**Says:**  
"I need a place within my budget."  
"I want to know if this listing is real."

**Thinks:**

"Will the owner reply?"  
"Is this secure?"  
**Does:**

Searches on multiple sites, contacts owners, waits for responses.  
Reads reviews and checks social media for property credibility.

**Feels:**  
Frustrated, uncertain, hopeful.  
Overwhelmed by too much scattered information.

**2.3 Brainstorming**

Key ideas included:

* Verified property listings
* In-app secure messaging
* Owner approval workflows
* Digital lease agreements
* Booking status tracking
* Review and rating system

**3. REQUIREMENT ANALYSIS**

**3.1 Customer Journey Map**

* Renter signs up and verifies account.
* Searches and filters property listings.
* Views property details.
* Sends booking request.
* Communicates with owner.
* Confirms booking and signs lease.
* Leaves a review after stay.

**3.2 Solution Requirements**

* User authentication
* Property management features
* Booking workflow
* Admin approval controls
* Responsive user interface

**3.3 Data Flow Diagram**

User -> Frontend -> Backend API -> Database

|

<- Responses & Data

* User actions trigger REST API calls.
* Backend processes requests and updates MongoDB.
* Results are returned to frontend.

**3.4 Technology Stack**

* **Frontend:** React.js, Bootstrap, Ant Design
* **Backend:** Node.js, Express.js
* **Database:** MongoDB, Mongoose
* **Authentication:** JWT, bcrypt.js
* **Hosting:** Netlify/Vercel (frontend), Heroku/Render (backend)

**4. PROJECT DESIGN**

**4.1 Problem Solution Fit**

HouseHunt effectively addresses the challenges renters and owners face by combining verified listings, secure communication, and transparent booking in a single platform.

**4.2 Proposed Solution**

A MERN stack application where users can register, manage listings, search properties, request bookings, and sign digital agreements. Admins can monitor and approve owner accounts.

**4.3 Solution Architecture**

**Frontend:** React components for all views and user roles.  
**Backend:** Express.js REST APIs, authentication middleware.  
**Database:** MongoDB collections for users, properties, bookings, reviews.

**5. PROJECT PLANNING & SCHEDULING**

**5.1 Project Planning**

**Sprints Overview:**

* Sprint 1: Registration and Login
* Sprint 2: Property Management
* Sprint 3: Booking Features
* Sprint 4: Admin Panel and Review System

**Velocity:** ~8 story points per sprint.

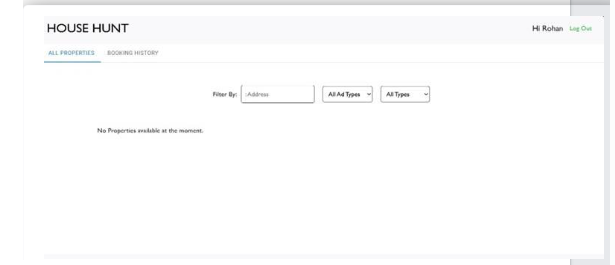
**6. FUNCTIONAL AND PERFORMANCE TESTING**

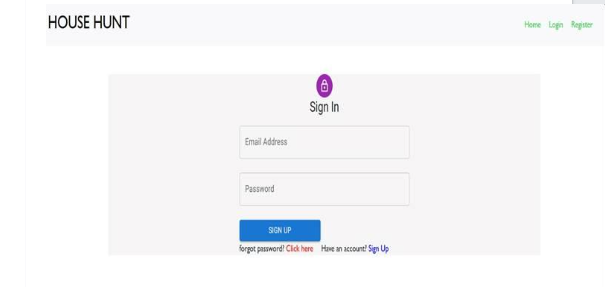
**6.1 Performance Testing**

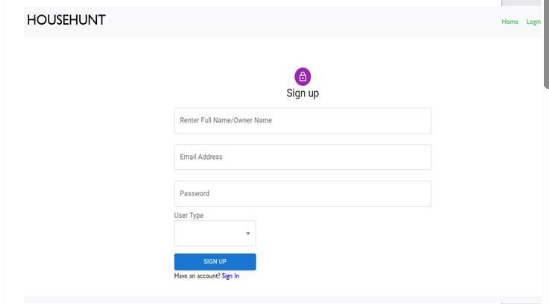
* Load testing APIs using Postman.
* Verifying response times under simulated multiple user requests.

**7. RESULTS**

**7.1 Output Screenshots**







**8. ADVANTAGES & DISADVANTAGES**

**Advantages:**

* Transparent and verified listings
* Secure communication and booking
* User-friendly design

**Disadvantages:**

* Dependent on stable internet connection
* Limited payment options (future enhancement)

**9. CONCLUSION**

HouseHunt successfully provides a modern, reliable platform for renters and property owners to manage the entire rental process online, improving trust, convenience, and efficiency.

**10. FUTURE SCOPE**

* Integrate online payments
* Enable SMS notifications
* Add AI-powered property recommendations
* Support multilingual interfaces
* Mobile app development

**11. APPENDIX**

**Source Code:**

const express = require("express");

const dotenv = require("dotenv");

const cors = require("cors");

const connectionofDb = require("./config/connect.js");

const path = require("path");

const app = express();

dotenv.config();

connectionofDb();

const PORT = process.env.PORT || 8001;

app.use(express.json());

app.use(cors({

  origin:'http://localhost:3000',

  methods:['GET','POST']

}));

app.use("/uploads", express.static(path.join(\_\_dirname, "uploads")));

app.use('/api/user', require('./routes/userRoutes.js'))

app.use('/api/admin', require('./routes/adminRoutes'))

app.use('/api/owner', require('./routes/ownerRoutes'))

app.listen(PORT, () => {

  console.log(`Server is running on port ${PORT}`);

});

{

  "name": "backend",

  "version": "1.0.0",

  "description": "",

  "main": "index.js",

  "scripts": {

    "start": "nodemon index",

    "test": "echo \"Error: no test specified\" && exit 1"

  },

  "keywords": [],

  "author": "",

  "license": "ISC",

  "dependencies": {

    "bcryptjs": "^2.4.3",

    "cors": "^2.8.5",

    "dotenv": "^16.5.0",

    "express": "^4.21.2",

    "jsonwebtoken": "^9.0.1",

    "mongoose": "^7.8.7",

    "multer": "^1.4.5-lts.1",

    "nodemon": "^3.0.1"

  }

}

 <div className="App">

        <Router>

          <div className="content">

            <Routes>

              <Route path="/" element={<Home />} />

              <Route path="/login" element={<Login />} />

              <Route path="/register" element={<Register />} />

              <Route path="/forgotpassword" element={<ForgotPassword />} />

              {userLoggedIn ? (

                <>

                  <Route path="/adminhome" element={<AdminHome />} />

                  <Route path="/ownerhome" element={<OwnerHome />} />

                  <Route path="/renterhome" element={<RenterHome />} />

                </>

              ) : (

                <Route path="/login" element={<Login />} />

              )}

            </Routes>

          </div>

import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>

);

MONGO\_DB = 'mongodb://127.0.0.1:27017/'

JWT\_KEY = '12345'

**GitHub Repository:**

[**https://github.com/manasachowdary451/HouseHunt**](https://github.com/manasachowdary451/HouseHunt)

**Project Demo Link:**

<https://drive.google.com/file/d/1CUBvdkgJ-kEdZNgYJLyFLtNA_EN5iJXV/view?usp=drive_link>