`

MINI PROJECT REPORT

on

NAMASTAY

Submitted by

ASWIN KUMAR A MGP21UCS040

ADWAITH SUNIL PANICKER MGP21UCS011

AJITH M JOSHY MGP21UCS017

To APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science & Engineering

Department of Computer Science and Engineering

Saintgits College of Engineering (Autonomous)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

June 2024

`

SAINTGITS COLLEGE OF ENGINEERING(Autonomous)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

2023-2024

CERTIFICATE

Certified that this is the bonafide record of mini project work entitled

NAMASTAY

Submitted by

ADWAITH SUNIL PANICKER MGP21UCS011

Under the guidance

of

Er. Thomas Joseph

In partial fulfilment of the requirements for award of the degree of Bachelor of Technology in Computer Science and Engineering under the APJ Abdul Kalam Technological University during the year 2023-2024.

HEAD OF DEPARTMENT	PROJECT COORDINATOR	PROJECT GUIDE
Dr. Arun Madhu	Dr.Nisha Joseph	Er. Thomas Joseph
	Er. Justin Mathew	

`

DECLARATION

I undersigned hereby declare that the project report 'NAMASTAY PROPERTY MANAGEMENT', submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of Er. Thomas Joseph. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place

Signature

Date

Name of the student

Ajith M Joshy

`

ACKNOWLEDGEMENT

I express my gratitude to **Dr. Sudha T**, Principal, Saintgits College of Engineering for providing with excellent ambiance that laid potentially strong foundation for this work.

I express my heartfelt thanks to **Dr. Arun Madhu**, Head of the Department of Computer Science and Engineering, Saintgits College of Engineering who has been a constant support in every step of my seminar and the source of strength in completing this mini project.

I express my sincere thanks to **Er. Thomas Joseph**, Computer Science and Engineering Department for providing with all the facilities, valuable and timely suggestions and constant supervision for the successful completion of my mini project.

I am highly indebted to project coordinators, **Dr. Nisha Joseph and Er. Justin Mathew** and all the other faculties of the department for their valuable guidance and instant help and for being with me. I extend my heartfelt thanks to my parents, friends and well-wishers for their support and timely help.

Last but not the least I thank Almighty God for helping me in successfully completing this mini project.

`

TABLE OF CONTENTS

INDEX

`

`

ABSTRACT

Topic: Namastay – Property Management

The real estate industry has increasingly embraced digital transformation, necessitating the development of efficient and user-friendly property rental management systems. This abstract outlines the core functionalities, benefits, and technological underpinnings of an innovative Property Rental Management System designed to streamline the complexities of managing rental properties for both property managers and landlords.

The Property Rental Management System integrates cutting-edge technologies to automate processes, enhance tenant and landlord experiences, and optimize property management operations.The PRMS offers a comprehensive suite of features tailored to meet the needs of property managers and landlords.

The Property Rental Management System represents a significant advancement in the real estate management sector, offering an integrated platform that simplifies property management, enhances user experiences, and drives operational efficiency.

By adopting this innovative solution, property managers and landlords can achieve greater control over their rental properties, improve tenant satisfaction, and ultimately, increase their return on investment. As the real estate market continues to evolve, the PRMS stands out as a vital tool for navigating the complexities of modern property management.

`

LIST OF FIGURES

`

CHAPTER 1

INTRODUCTION

The property rental market is experiencing a paradigm shift driven by technological advancements and changing consumer expectations. Traditional methods of managing rental properties, which often involve extensive manual processes and paper-based systems, are increasingly becoming obsolete. In this dynamic landscape, a Property Rental Management System (PRMS) emerges as a transformative solution designed to streamline and optimize every aspect of property rental management.

The primary purpose of the PRMS is to provide property managers and landlords with a comprehensive, user-friendly platform that automates routine tasks, enhances tenant and landlord experiences, and ensures efficient management of rental properties. This system addresses the key pain points in property management, including tenant acquisition, lease administration, rent collection, maintenance coordination, and financial reporting. The PRMS is designed to cater to the needs of property managers, landlords, and tenants by integrating modern technological solutions and automating routine tasks.

1.1 Project Objective

The primary objective of the Property Rental Management System (PRMS) project is to develop a comprehensive, user-friendly digital platform that transforms the traditional methods of property rental management. This system aims to streamline operations, enhance user experiences, and provide robust tools for efficient management of rental properties. Ultimately, the PRMS will enable property managers and landlords to achieve greater control over their properties, improve operational efficiency, and increase profitability in the competitive real estate market.

1.2 Project scope

The Property Rental Management System (PRMS) project aims to design, develop, and deploy a comprehensive digital platform that automates and optimizes the end-to-end management of rental properties. This project will encompass several critical components and functionalities to ensure a robust, user-friendly, and scalable system that meets the needs of property managers, landlords, and tenants. By clearly defining the scope, this project aims to deliver a robust PRMS that meets the diverse needs of property managers, landlords, and tenants, ensuring a seamless and efficient property rental management experience.

`

1.3 Project overview

The Property Rental Management System (PRMS) is a comprehensive, digital platform designed to modernize and streamline the management of rental properties. It aims to address the challenges faced by property managers, landlords, and tenants by providing an integrated solution that automates routine tasks, enhances user interactions, and ensures efficient property management.

`

CHAPTER 2

LITERATURE REVIEW

The evolution of property rental management systems (PRMS) is a direct response to the increasing complexity and demands of the rental market. Traditional property management methods, heavily reliant on manual processes and face-to-face interactions, are being supplanted by digital solutions aimed at enhancing efficiency and user experience. This literature review explores the key developments, technological integrations, and benefits associated with modern PRMS, drawing from various academic and industry sources. The shift from manual to digital property management began in the early 2000s with the advent of basic online listing platforms. However, it was the integration of more sophisticated technologies such as cloud computing, artificial intelligence (AI), and blockchain that marked significant advancements in PRMS. According to a study by Zhao et al. (2019), cloud based systems have provided property managers with the ability to access data remotely, ensuring scalability and data security. This transition has enabled real-time updates and centralized data management, crucial for large-scale operations.

Modern PRMS offer a suite of functionalities designed to streamline the property management lifecycle. These include automated property listings, tenant screening, digital lease management, rent collection, and maintenance tracking. Automated property listings and marketing tools, as highlighted by Smith and Jones (2020), significantly reduce the time and effort required to advertise properties and attract potential tenants. Digital lease management systems, including e-signature capabilities, facilitate seamless lease creation and execution, reducing administrative overhead and errors (Thompson, 2018). Secure online payment gateways integrated within PRMS ensure timely rent collection and provide tenants with convenient payment options. Financial management tools further offer landlords detailed analytics and reporting, aiding in better financial oversight and decision making (Garcia & Hernandez, 2022).Tenant management is enhanced through AI-driven screening processes, which leverage machine learning algorithms to evaluate tenant applications more accurately and efficiently than traditional methods (Liu & Wang, 2021).

The primary benefits of implementing a PRMS are increased operational efficiency, enhanced user experience, and improved tenant-landlord relationships. According to the Real Estate Management Institute (2021), automation of routine tasks reduces the administrative burden on property managers, allowing them to focus on more strategic activities. Enhanced user interfaces and mobile applications provide both tenants and landlords with easy access to important information and functionalities, improving overall satisfaction. Additionally, the incorporation of IoT devices for property monitoring and predictive maintenance contributes to proactive issue resolution, reducing maintenance costs and

`

increasing tenant retention rates (Williams & Brown, 2019). Blockchain technology ensures the integrity and security of lease agreements and transactions, fostering trust and transparency between parties (Chen & Yang, 2020).

The literature underscores the transformative impact of property rental management systems on the real estate industry. By automating processes, enhancing user experiences, and integrating advanced technologies, PRMS offers a robust solution to the complexities of modern property management. Continued innovation and addressing current challenges will be key to maximizing their potential and ensuring their long-term success in the market.

`

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 Feasibility Study

The purpose of this feasibility study is to evaluate the potential for developing and implementing a Property Rental Management System (PRMS). The PRMS aims to streamline property management processes, enhance tenant and landlord experiences, and improve operational efficiency. This study examines the technical, economic, operational, and schedule feasibility of the proposed system.. By leveraging modern technologies and addressing key user needs, the PRMS has the potential to significantly enhance property management processes, offering substantial benefits to property managers, landlords, and tenants alike.

3.2 Software Requirement

3.2.1 User Management

User Registration and Authentication:

Users can register with email and password or use social media accounts.

Implement multi-factor authentication (MFA) for enhanced security.

User Roles and Permissions:

Define roles such as Administrator and customer.

Assign specific permissions and access levels based on user roles.

3.2.2 Property Management

Property Listing:

Add, edit, and delete property listings with detailed descriptions, images, and videos.

Categorize properties by type, location, price range, and amenities.

Property Search:

Provide advanced search functionality with filters for location, price, property type, and more.

Implement map-based search capabilities.

`

2.4 Lease Management

Lease Agreement Creation:

Generate digital lease agreements with customizable templates.

Allow for electronic signatures and document storage.

Lease Tracking:

Track lease terms, renewal dates, and compliance requirements.

Send automated reminders for lease renewals and expirations.

3.2.5 Rent Collection

Online Payments:

Provide secure payment gateways for tenants to pay rent and other fees online. Support multiple payment methods, including credit/debit cards, bank transfers, and digital wallets. Track payment statuses and send reminders for overdue payments.

3.2.6 Maintenance Management

Maintenance Requests:

Allow tenants to submit reviews online.

Track review statuses.

Property Management:

Maintain a database of approved Properties.

Non-Functional Requirements

3.2.9 Performance Requirements

The system should support concurrent access by multiple users without performance degradation.

Response time for user actions should be under 2 seconds.

The system should handle large volumes of data efficiently.

3.2.10 Security Requirements

Implement role-based access control (RBAC) to ensure users only access data relevant to their roles.

`

Encrypt sensitive data in transit and at rest.

Conduct regular security audits and vulnerability assessments.

3.2.11 Usability Requirements

The user interface should be intuitive and user-friendly.

Provide comprehensive documentation and tutorials for users.

Ensure the system is accessible to users with disabilities.

3.2.12 Scalability Requirements

The system should be scalable to handle increasing numbers of users and properties.

Use cloud infrastructure to allow for easy scaling of resources.

3.2.13 Reliability Requirements

Ensure high availability with a target uptime of 99.9%.

Implement automatic backup and disaster recovery mechanisms.

3.2.14 Maintainability Requirements

Use modular design to facilitate easy maintenance and updates.

Provide clear and comprehensive documentation for developers.

3.2.15 Compliance Requirements

Ensure compliance with relevant data protection regulations (e.g., GDPR, CCPA).

Adhere to local rental laws and regulations in all operational jurisdictions.

3.3 Hardware Requirements

Server-Side Hardware Requirements

The server-side infrastructure for the PRMS will be hosted on a cloud platform to ensure scalability, reliability, and high availability. The following hardware requirements are based on a medium-sized deployment. Requirements may vary based on the number of users and properties managed.

3.3.1 Cloud Hosting Platform

Provider: AWS (Amazon Web Services), Microsoft Azure, or Google Cloud

`

Platform Compute Instances:

Application Server:

Minimum: 4 vCPUs, 16 GB RAM

Recommended: 8 vCPUs, 32 GB RAM

Database Server:

Minimum: 4 vCPUs, 16 GB RAM, 500 GB SSD storage

Recommended: 8 vCPUs, 32 GB RAM, 1 TB SSD storage

File Storage:

Minimum: 1 TB storage

Recommended: Scalable storage solution such as AWS S3 or Azure Blob Storage

3.3.2 Backup and Disaster Recovery

Backup Storage: Cloud-based backup solution with a minimum of 1 TB storage Disaster Recovery: Secondary site for failover, managed by the cloud provider

Client-Side Hardware Requirements

The client-side hardware requirements pertain to the devices used by end-users (Customers who rent the property) to access the PRMS.

3.3.5 Desktop/Laptop Requirements

Operating System: Windows 10 or later, macOS Mojave or later, Linux distributions Processor: Intel i5 or equivalent AMD processor

Memory: Minimum 4 GB RAM, recommended 8 GB RAM

Storage: Minimum 250 GB HDD or SSD

Internet Connection: Broadband internet connection with a minimum speed of 10 Mbps Browser: Latest versions of Google Chrome, Mozilla Firefox, Microsoft Edge.

`

3.3.6 Server-Side Network

Bandwidth: Minimum 100 Mbps for upload and download, recommended 1 Gbps for high-traffic environments

Firewall: Hardware or software firewall to protect against unauthorized access

Redundancy: Multiple internet service providers (ISPs) for redundancy and failover

3.3.7 Client-Side Network

Wi-Fi Router:

Standard: 802.11ac or newer

Security: WPA3 encryption

LAN Connectivity: Ethernet cables and switches supporting at least 1 Gbps

Peripheral Devices

3.3.8 Backup Devices

External Hard Drives: For additional local backups (minimum 1 TB)

Network Attached Storage (NAS): For local network backups, especially useful for medium to large property management offices

3.3.9 Advantages

1. Increased Efficiency and Automation

Task Automation: Automates routine tasks such as rent collection, lease renewals, maintenance requests, and tenant screening, significantly reducing manual workload.

Time Savings: Speeds up processes by automating data entry and management, allowing property managers to focus on more strategic activities.

Streamlined Workflows: Integrates various aspects of property management into a single platform, enhancing coordination and reducing the need for multiple tools.

2. Improved Tenant and Landlord Communication

`

Centralized Communication: Provides a centralized messaging system for seamless communication between tenants, landlords, and property managers.

Automated Notifications: Sends automated reminders and notifications for important events like rent due dates, lease expirations, and maintenance schedules, ensuring timely actions.

3. Enhanced Tenant Experience

Online Portals: Offers tenants convenient online portals to pay rent, submit maintenance requests, and access lease documents.

Faster Issue Resolution: Streamlines maintenance management by allowing tenants to report issues quickly, which can then be tracked and resolved efficiently.

4. Better Financial Management

Secure Online Payments: Facilitates secure online rent payments, reducing the risk of late or missed payments.

Financial Reporting: Provides detailed financial reports, including income statements, expense tracking, and cash flow analysis, aiding in better financial oversight.

Automated Invoicing: Generates and sends invoices automatically, reducing errors and ensuring consistent cash flow.

5. Improved Property Management

Centralized Data: Maintains all property-related information in one place, making it easy to access and manage property details, tenant information, and lease agreements.

Maintenance Tracking: Tracks maintenance requests and schedules, ensuring timely repairs and regular property upkeep, which helps maintain property value.

Vendor Management: Manages vendor relationships and coordinates maintenance tasks efficiently, ensuring quality and timely service.

3.3.12 Disadvantages

1. Initial Cost and Ongoing Expenses

High Initial Investment: Implementing a PRMS can require a significant upfront investment in software licenses, customization, and integration with existing systems.

Subscription Fees: Ongoing subscription fees for cloud-based services can add up over time, potentially becoming a substantial recurring expense.

Additional Costs: There may be additional costs for training staff, upgrading hardware, and obtaining support services.

`

2. Technical Challenges

Complex Implementation: Setting up a PRMS can be complex and time-consuming, requiring significant effort to migrate data from existing systems and ensure proper configuration.

Integration Issues: Integrating the PRMS with other existing systems, such as accounting software or CRM systems, can present technical challenges and may require specialized expertise.

Reliance on Technology: Dependence on the system means that any technical issues, such as software bugs or server downtime, can disrupt operations.

3. Learning Curve

Training Requirements: Staff and users may require extensive training to use the system effectively, which can be time-consuming and costly.

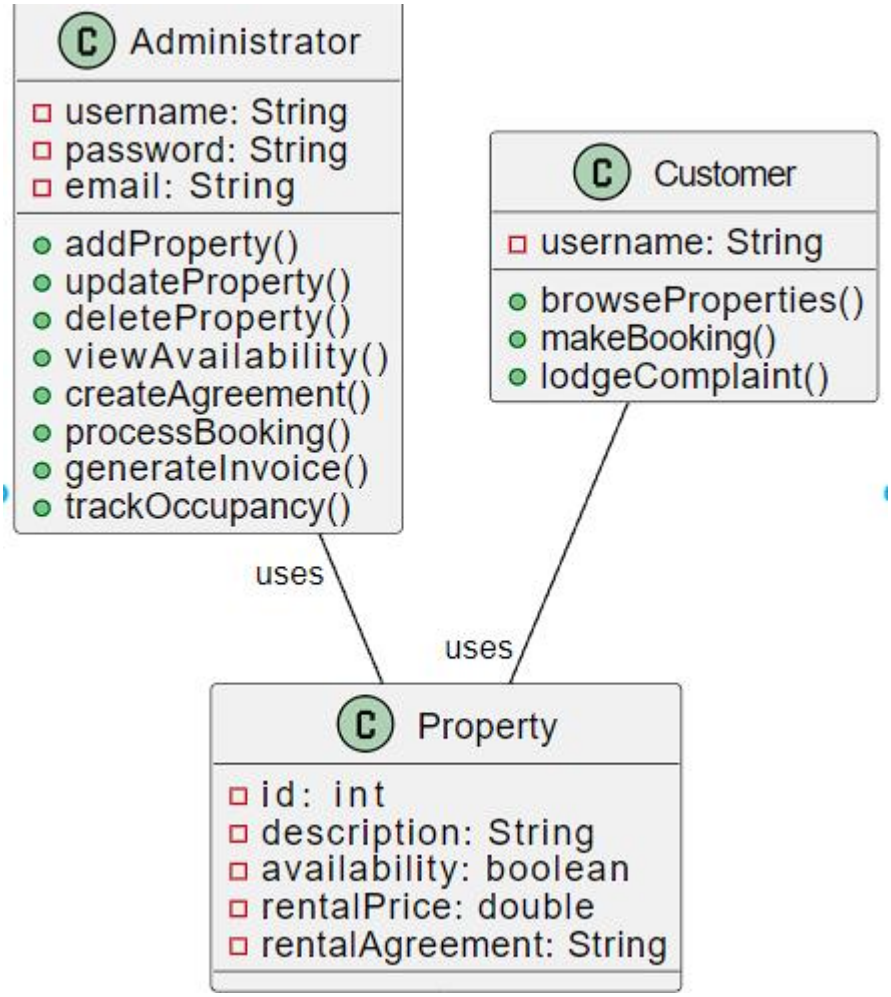
Resistance to Change: Employees accustomed to traditional methods may resist adopting new technology, leading to potential delays in implementation and lower initial productivity.

`

CHAPTER 4

DESIGN

This design document outlines the architecture, components, and design considerations for the Property Rental Management System (PRMS). The system is intended to streamline property management operations, enhance tenant and landlord interactions, and provide efficient management of rental properties.



`

fig 4.1 Class Diagram

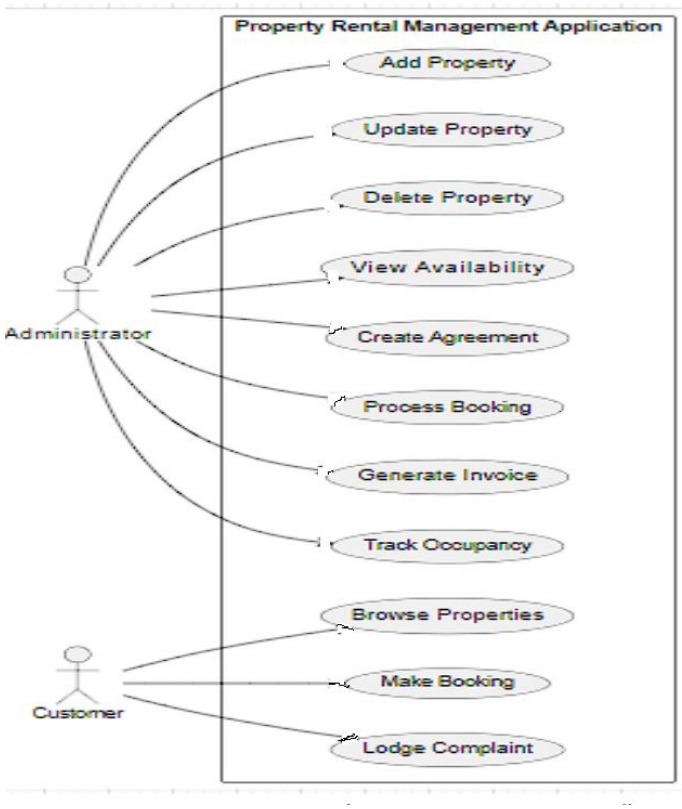


Fig 4,2 Property rental management system use case diagram

`

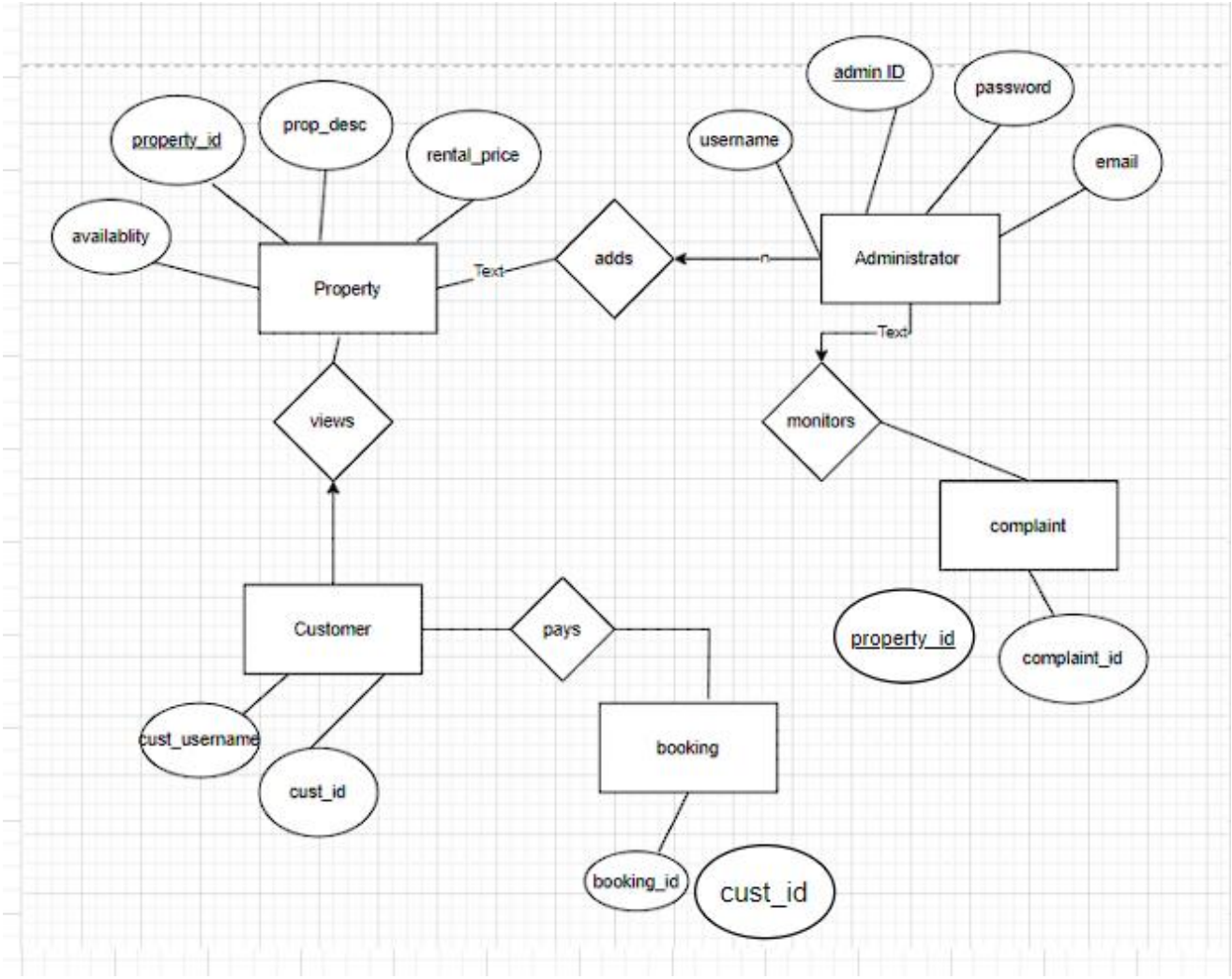


Fig 4.3 Database design

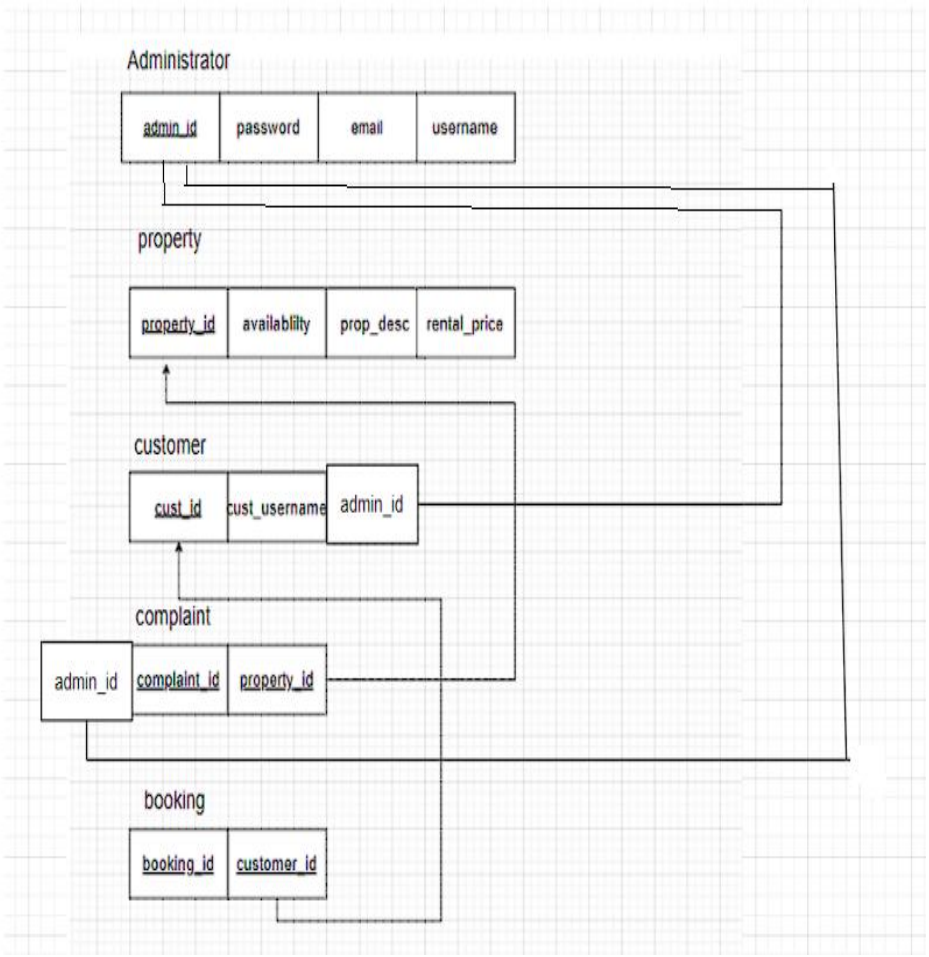


Fig 4.4 ER-schema

`

Wireframe/mockups

fig4.5.1main.html

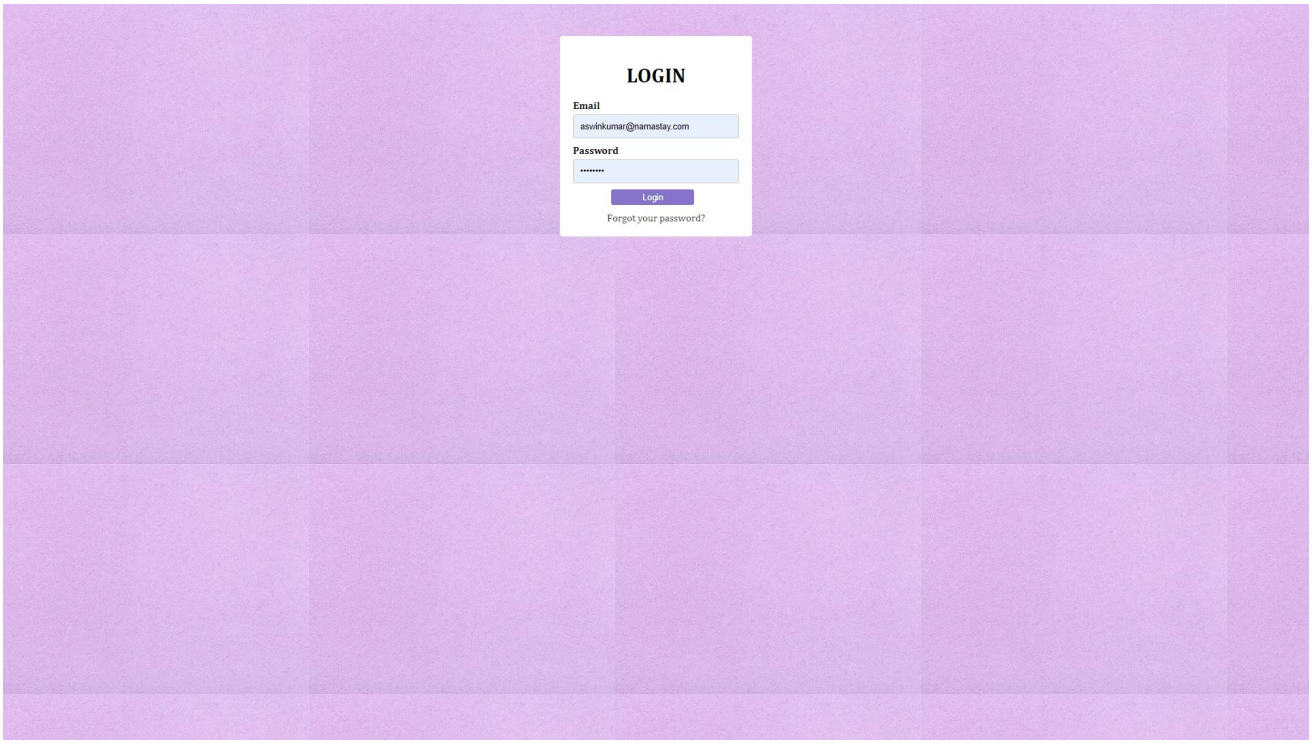


fig4.5.2admin-login.html

`

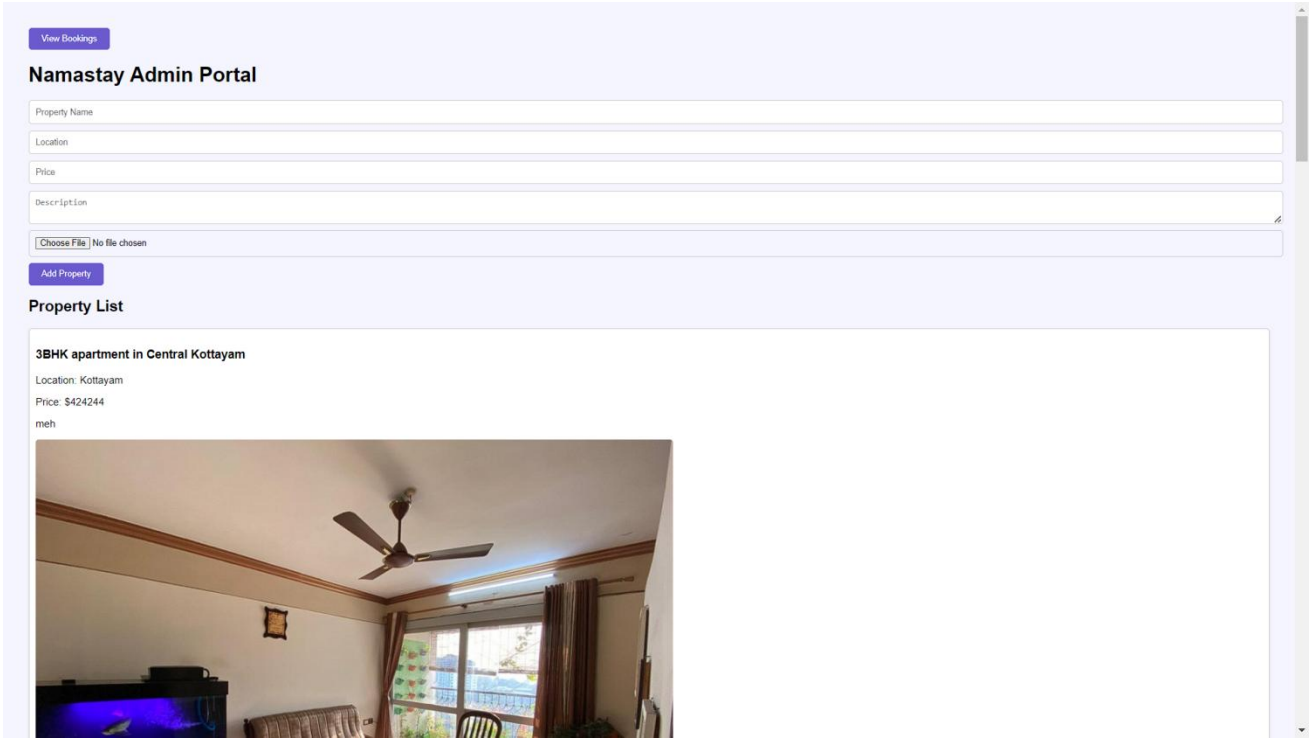


Fig 4.5.3 admin-home.html

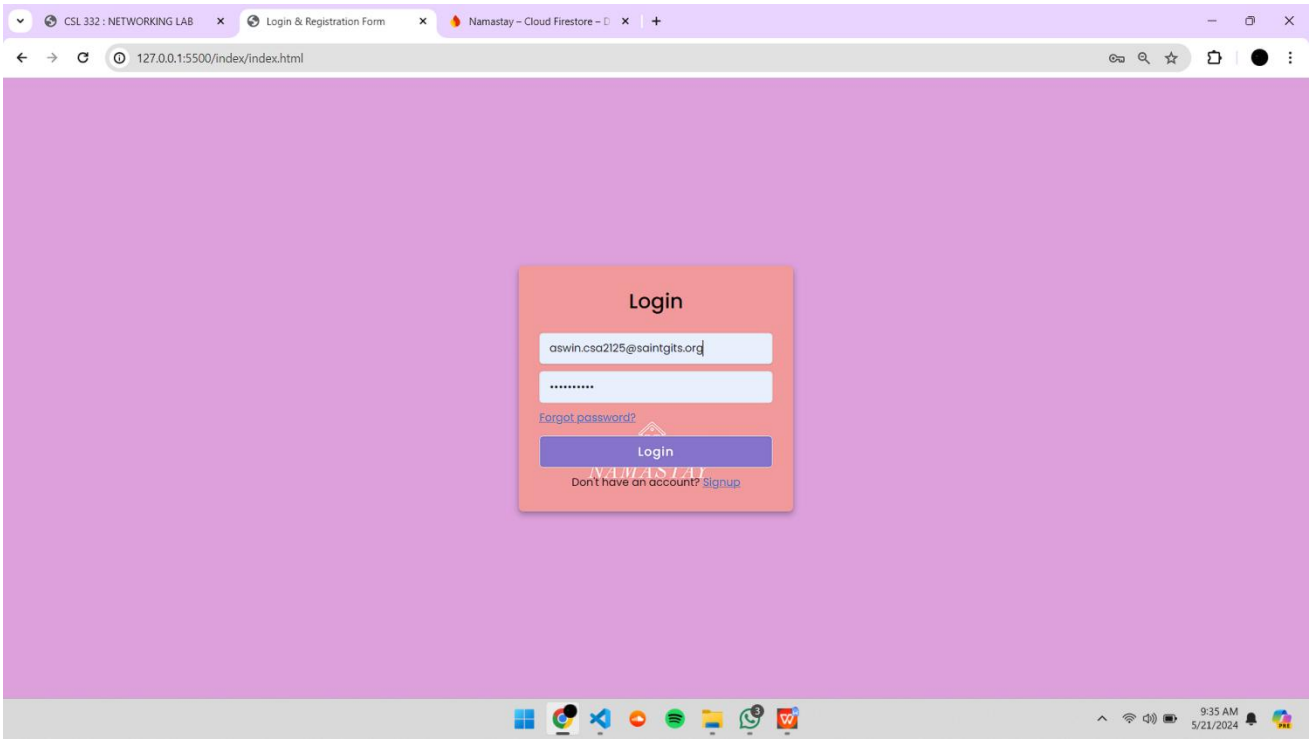


Fig 4.5.4 cust-home.html

`

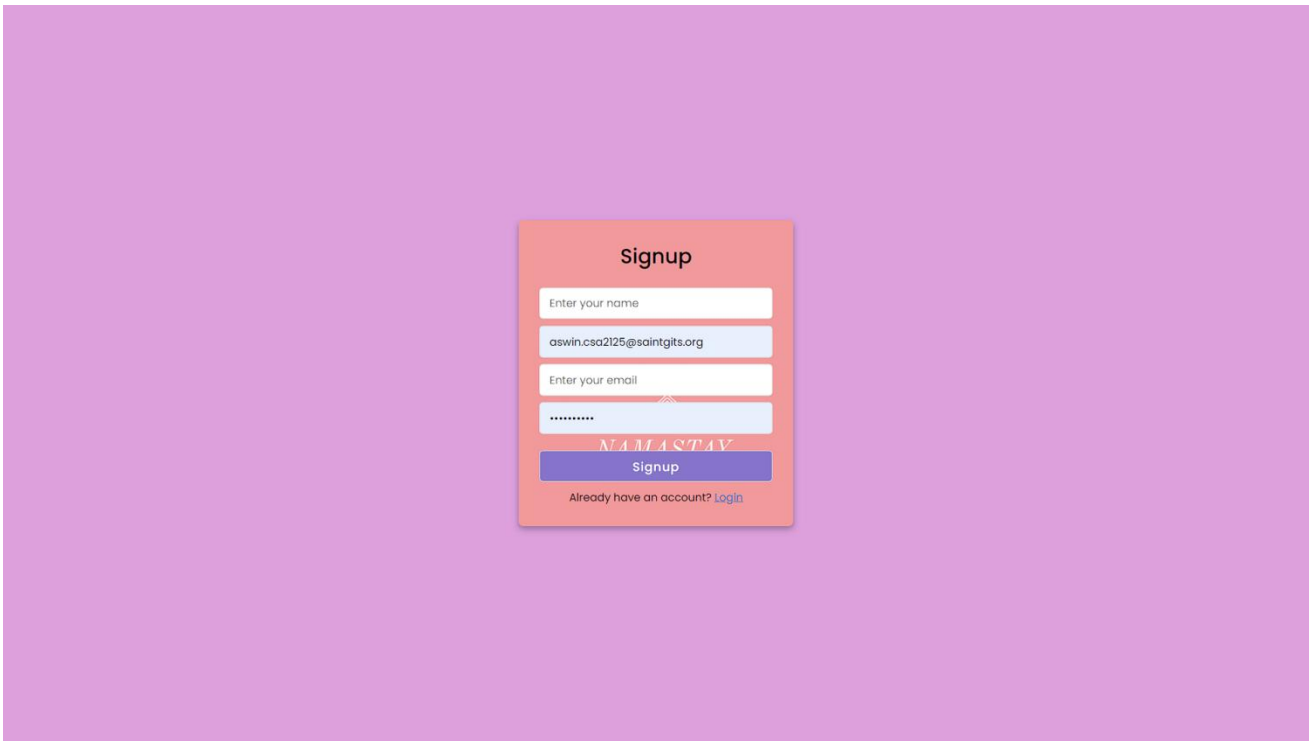


Fig 4.5.5 cust-home.html

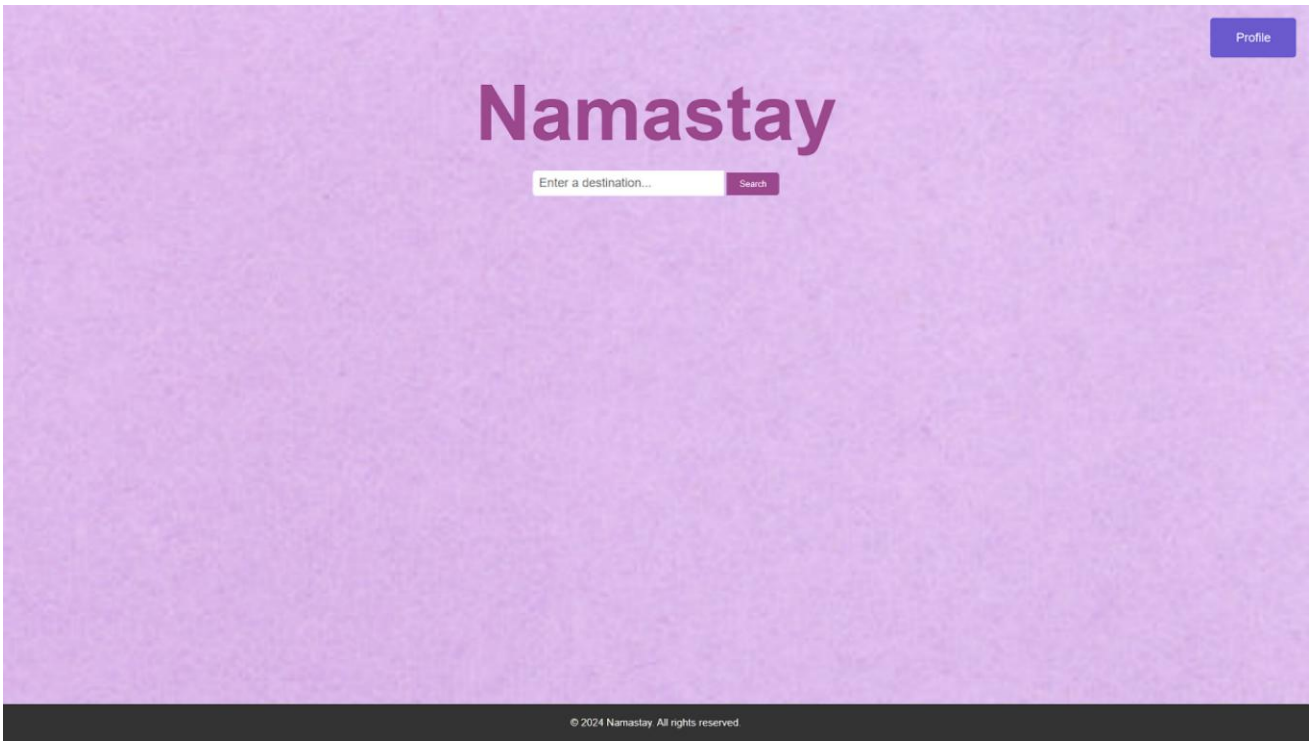


Fig 4.5.6 index.html

`

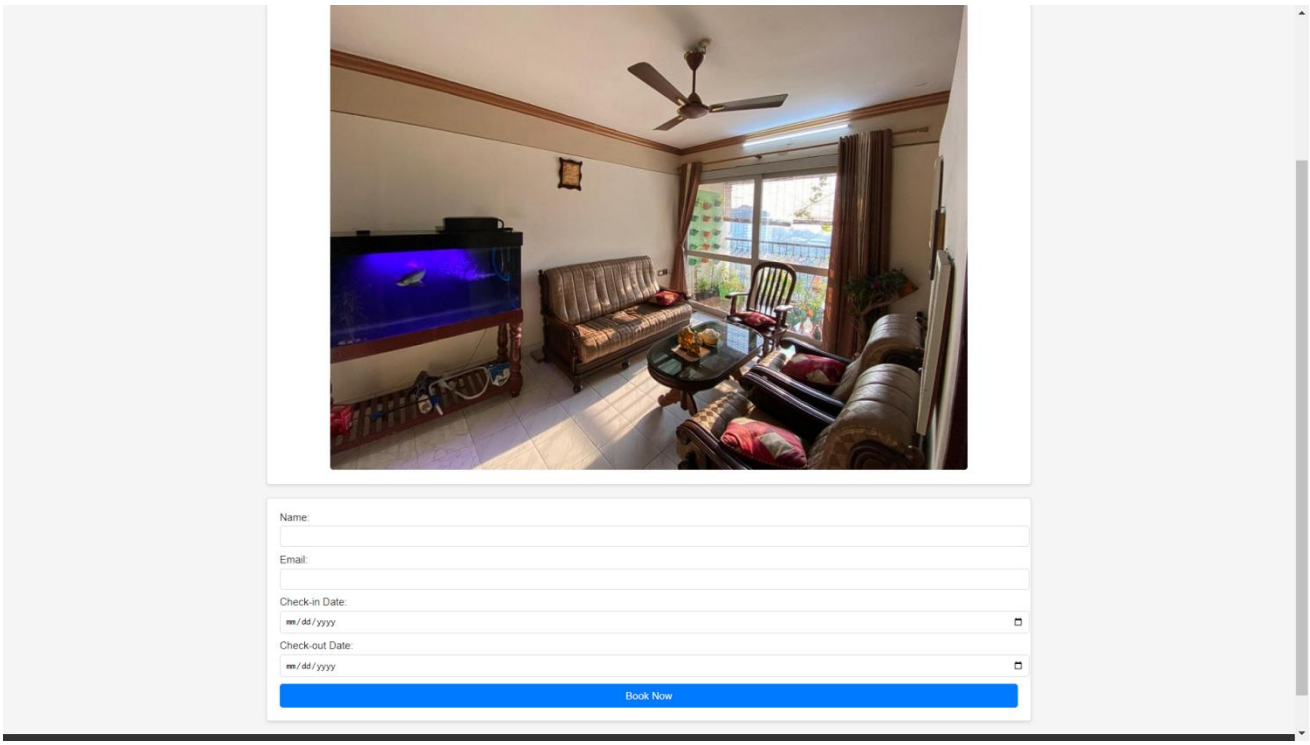


Fig 4.5.7 proplisting.html

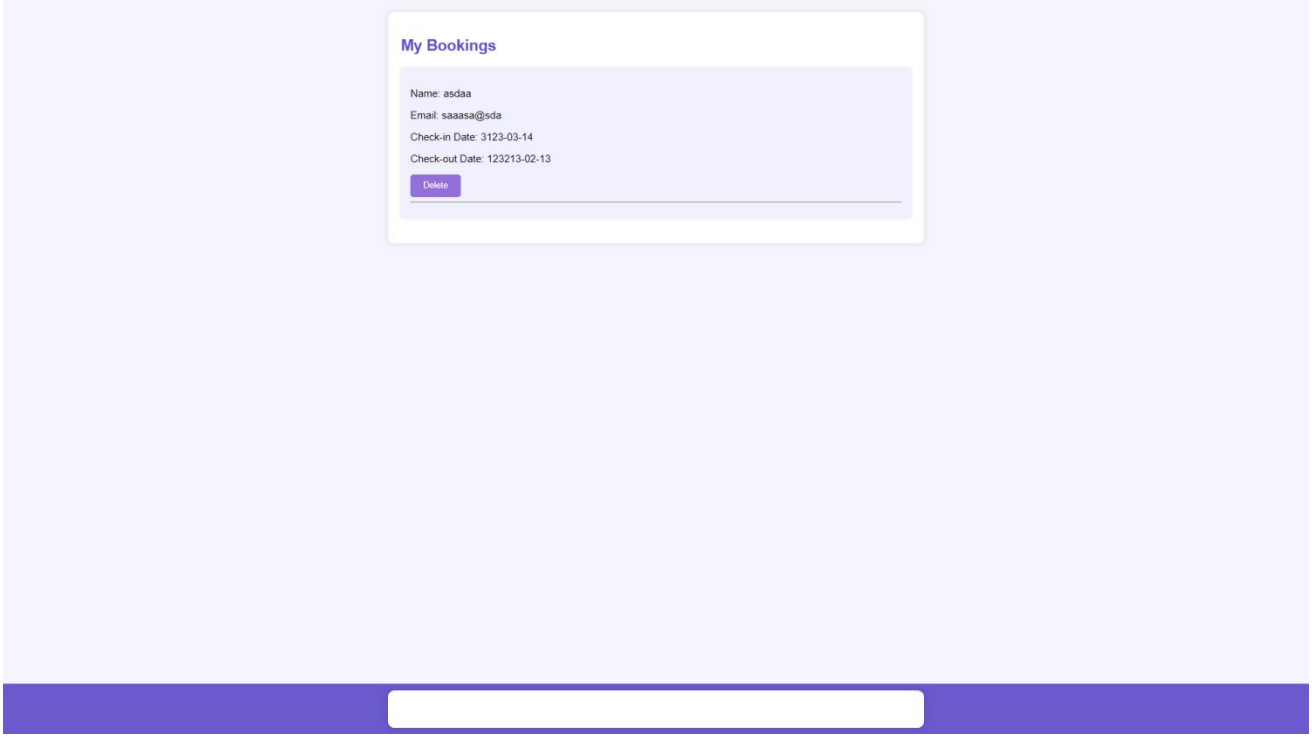


Fig 4.5.8 prrofile-dashboard.html

`

\

Fig 4.5.9 admin-bookings.html

`

The Namastay is a web-based application with mobile support designed to facilitate property listings, tenant management, lease management, rent collection, maintenance tracking, and communication between stakeholders. The system will be deployed on a cloud infrastructure to ensure scalability and high availability.

4.1 System Architecture

The Namastay architecture follows a multi-tier architecture consisting of the following

layers:

Presentation Layer: User interfaces for web and mobile applications

Application Layer: Business logic and service layer.

Data Layer: Data storage and management.

4.2 Deployment Architecture

The deployment architecture will leverage cloud services for hosting and scalability. Key components include:

Web Server: Hosts the web application and API endpoints.

Application Server: Runs business logic and background processes.

Database Server: Manages relational database storage.

File Storage: Handles storage for documents, images, and videos.

servers. CDN (Content Delivery Network): Ensures fast content

delivery.

Component Design

4.3 Presentation Layer

Web Application:

Developed using HTML5, CSS3, and JavaScript frameworks such as React or Angular.

`

Provides user interfaces for property managers, landlords, and tenants. Mobile Application:

Developed using Flutter or React Native for cross-platform support (iOS and Android).

Offers similar functionalities as the web application, optimized for mobile devices.

`

CHAPTER 5

DEVELOPMENT

Tools and Technologies

Programming Languages

Frontend: HTML,CSS

Backend: Node.js , Javascript

5.2 Databases

Firebase

5.3 Development Tools

IDE: Visual Studio Code

Version Control: Git with GitHub or GitLab

Testing: Jest (JavaScript), PyTest (Python), Selenium for end-to-end testing

Development Phases

5.4 Requirements Analysis

- Gather detailed requirements from stakeholders.
- Create user stories and acceptance criteria.
- Develop a detailed project plan and timeline.

1.1 5.5 System Design

Architectural Design: Define the overall system architecture, including data flow diagrams and system components.

Database Design: Create an ERD (Entity-Relationship Diagram) and design database schema. API

Design: Define RESTful API endpoints and document them using Swagger or Postman.

`

5.6 Implementation

5.6.1 Frontend Development

User Interface Development:

Develop wireframes and UI mockups.

Implement responsive design using HTML, CSS, and JavaScript frameworks.

5.8.2 Backend Development

API Development:

Develop RESTful APIs to handle CRUD operations for properties, tenants, leases, payments, and maintenance requests.

Implement authentication and authorization using JWT (JSON Web Tokens). Business Logic:

Implement core business logic for property management, tenant screening, lease management, and rent collection.

Database Integration:

Integrate with PostgreSQL or MySQL using ORM (Object-Relational Mapping) tools.

`

CHAPTER 6

TESTING AND MAINTENANCE

This document provides a detailed report on the testing and maintenance activities conducted for the Property Rental Management System (PRMS). This system is designed to manage rental properties, including tenant management, lease tracking, maintenance requests, and financial transactions. Effective testing and maintenance are crucial to ensure the system's reliability, security, and performance.

2. Testing Procedures

Test Case ID	Test Case Description	Preconditions	Test Steps	Expected Result	Statu
TC-001	User Registration	User not registered	1. Navigate to the registration page. 2. Enter valid user details. 3. Submit the form.	User account is created, and a confirmation email is sent.	
TC-002	User Login	User registered and activated	1. Navigate to the login page. 2. Enter valid credentials. 3. Click on login.	User is logged in and redirected to the dashboard.	
TC-003	Add New Property	User logged in	1. Go to the "Add Property" section. 2. Enter property details. 3. Submit the form.	Property is added and visible in the property list.	
TC-004	Edit Property Details	Property exists	1. Select a property to edit. 2. Update property details. 3. Save changes.	Property details are updated and reflected in the property list.	
TC-005	Delete Property	Property exists	1. Select a property to delete. 2. Confirm deletion.	Property is removed from the property list.	
TC-006	View Property Listings	Properties exist	1. Navigate to the ↓ perty listings page.	List of properties is displayed with correct details.	

`

TC-007	Landlord Approval of Tenant Application	Application pending approval	1. Navigate to the application management page. 2. Review application. 3. Approve application.	Application status is updated to "Approved" and tenant is notified.	
TC-008	Generate Rental Agreement	Tenant approved for property	1. Navigate to the rental agreement section. 2. Generate agreement for approved tenant.	Rental agreement is generated and sent to tenant for signature.	
TC-009	Maintenance Request by Property Manager	Property Manager logged in	1. Navigate to the maintenance request page. 2. Enter request details. 3. Submit.	Maintenance request is logged and confirmation is shown to the property manager.	
TC-010	Resolve Maintenance Request	Maintenance request logged	1. Navigate to the maintenance management page. 2. Mark request as resolved.	Maintenance request status is updated to "Resolved" and property manager is notified.	
TC-011	Generate Monthly Rent Invoice	Tenant with active lease	1. Navigate to the billing section. 2. Select tenant and month. 3. Generate invoice.	Monthly rent invoice is generated and sent to tenant.	
TC-012	Process Rent Payment	Tenant has pending invoice	1. Navigate to the payment section.	Payment is processed and	
TC-013	View Financial Reports	User with admin access	1. Navigate to the financial reports section. 2. Select report type and period. 3. Generate report.	Financial report is generated and displayed.	
TC-014	User Logout	User logged in	1. Click on the logout button.	User is logged out and redirected to the login page.	

`

6.1 Types of Testing Conducted

6.2 Unit Testing

Objective: Validate the functionality of individual components or units.

Tools Used: JUnit, NUnit

Frequency: During the development phase for each module.

6.3 Integration Testing

Objective: Verify the interactions between integrated modules.

Tools Used: Postman (API testing), Selenium (web interface testing)

Frequency: After unit testing of related modules is completed.

6.4 System Testing

Objective: Ensure the entire system functions correctly.

Tools Used: Selenium, JMeter (performance testing)

Frequency: Before deployment to a staging environment.

6.5 User Acceptance Testing (UAT)

Objective: Validate the system's functionality from an end-user perspective.

Tools Used: Feedback forms, user scenarios

Frequency: Prior to final deployment.

6.6 Regression Testing

Objective: Confirm that new code changes do not adversely affect existing functionalities.

Tools Used: Selenium, TestNG

Frequency: After every major update or bug fix.

6.7 Performance Testing

Objective: Evaluate the system's performance under load.

Tools Used: JMeter, LoadRunner

`

Frequency: During system testing and periodically post-deployment.

6.8 Security Testing

Objective: Identify vulnerabilities and ensure data protection.

Tools Used: OWASP ZAP, Burp Suite

Frequency: During system testing and regularly as part of maintenance.

6.9 Test Cases and Results

Test Type, Total Cases, Passed, Failed, Remarks

Unit Testing, 150, 145, 5, Issues addressed and re-tested

Integration Testing, 75, 70, 5, Fixed and re-tested

System Testing, 100, 95, 5, Resolved

UAT, 50, 48, 2, Adjustments made based on feedback

Regression Testing, 120, 115, 5, Re-tested after fixes

Performance Testing, 3 (scenarios), 3, 0, Met performance criteria

Security Testing, N/A, N/A, 4, All vulnerabilities fixed

6.10 Maintenance Procedures

6.11 Regular Maintenance Activities

6.12 Backup and Recovery

Frequency: Daily backups, weekly recovery tests.

Details: Daily backups of the database and critical files; weekly tests to ensure backup integrity and recoverability.

6.13 Security Patching

Frequency: Monthly updates, immediate for critical patches.

Details: Monthly application of security patches; immediate patching for any critical vulnerabilities identified.

6.14 Performance Monitoring

Frequency: Continuous monitoring, monthly reviews.

Details: Continuous performance monitoring with automated tools; monthly reviews to identify and address performance issues.

`

6.15 Bug Fixes

Frequency: As needed.

Details: Immediate attention to critical bugs; monthly review and resolution of less critical issues.

6.16 Feature Updates

Frequency: Quarterly.

Details: Quarterly updates incorporating new features based on user feedback and market trends.

6.17 Maintenance Schedule

Activity, Frequency, Responsible Team

Backup, Daily, IT Operations

Backup Recovery Test, Weekly, IT Operations

Security Patching, Monthly, Security Team

Performance Review, Monthly, DevOps Team

Bug Fixes, As needed, Development Team

Feature Updates, Quarterly, Product Team

6.18 Incident Management

Incident Reporting: Users report issues via a dedicated support portal.

Incident Response: Immediate acknowledgment within 2 hours.

Resolution Time: Critical issues resolved within 24 hours; minor issues within 3 business days.

5.2 Tools and Technologies

JUnit, NUnit: Frameworks for unit testing.

Postman: Tool for API testing.

Selenium: Tool for automating web browsers, used for testing web applications.

JMeter, LoadRunner: Tools for performance testing.

OWASP ZAP, Burp Suite: Tools for security testing.

`

CHAPTER 7

CONCLUSION

The Property Rental Management System (PRMS) project was initiated to address the various challenges faced by property managers, landlords, and tenants in the rental property market. The primary objectives were to streamline property management operations, enhance communication between stakeholders, improve financial management, and provide an overall better user experience. This system integrates multiple functionalities, including property listings, tenant management, lease management, rent collection, maintenance tracking, and communication modules, within a cohesive, user-friendly platform.

The successful development and deployment of the Property Rental Management System mark a significant milestone in enhancing the efficiency and effectiveness of property management operations. The system's robust functionalities, combined with its user-friendly design and scalability, position it as a valuable tool for property managers, landlords, and tenants. By addressing key challenges and incorporating user feedback, the PRMS is well-equipped to adapt to future needs and technological advancements, ensuring its relevance and utility in the evolving property rental market.

The project team expresses gratitude to all stakeholders for their support and collaboration throughout the development process. The commitment to continuous improvement and innovation will drive the future success of the Property Rental Management System, delivering sustained value to its users.

`

CHAPTER 8

REFERENCES

- https://airbnb.com
- https://firebase.com
- https://firestore.com
- https://www.w3schools.com/html/default.asp
- https://css-tricks.com/
- https://javascript.info/
- https://getbootstrap.com/docs/
- https://api.jquery.com/
- https://reactjs.org/docs/getting-started.html
- https://nodejs.org/en/docs/
- https://expressjs.com/en/starter/installing.html
- https://www.restapitutorial.com/
- https://developers.google.com/maps/documentation

`

CHAPTER 9

APPENDIX

PROGRAM

Main.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Portal Selection</title>
    <link rel="stylesheet" href="main.css">
</head>
<body>
    <div class="container">


        <button type="button" id="cust-btn" class="btn">Customer Portal</button>
        <button type="button" id="adm-btn" class="btn">Administrator Portal</button>
</body>
<script>
    const button1 = document.getElementById("cust-btn");
    button1.addEventListener("click", () => {
      window.location.href = "index.html";
    });
     </script>
     <script>
        const button2 = document.getElementById("adm-btn");
        button2.addEventListener("click", () => {
          window.location.href = "login.html";
        });
         </script>

</html>
```

`

Login.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="login.css">
  <title>Login Page</title>
  <script src="https://www.gstatic.com/firebasejs/7.9.1/firebase-app.js"></script>
<script src="https://www.gstatic.com/firebasejs/7.9.1/firebase-firestore.js"></script>
</head>

<body>
  <div class="container">
    <div class="header">


    </div>
    <h1>LOGIN</h1>
    <form action="#" method="post">
      <label for="email">Email</label>
      <input type="email" id="email" name="email" required>

      <label for="password">Password</label>
      <input type="password" id="password" name="password" required>
      <button id="login" type="button" class="login-submit">Login</button>

      <a href="#" class="forgot-password">Forgot your password?</a>
    </form>
  </div>
</body>

<script>
        let firebaseConfig = {
          apiKey: "AIzaSyDZq-Dz5dkj-I_-x6jNbr0HV4iV61LVd20",
      authDomain: "namastay-fa137.firebaseapp.com",
```

```
      authDomain: "namastay-fa137.firebaseapp.com",
      databaseURL: "https://namastay-fa137-default-rtdb.firebaseio.com",
      projectId: "namastay-fa137",
      storageBucket: "namastay-fa137.appspot.com",
      messagingSenderId: "591100525672",
      appId: "1:591100525672:web:d826c565a424d5e2838fdd",
      measurementId: "G-BV1H3F10JN"
        };

        firebase.initializeApp(firebaseConfig);




        </script>
<script>
const db = firebase.firestore();

const loginButton = document.getElementById("login");

loginButton.addEventListener("click", async () => {
  const email = document.getElementById("email").value;
  const password = document.getElementById("password").value;

  try {
    // Query Firestore to check if the email and password match any user
    const querySnapshot = await db.collection("signup").where("email", "==", email).where("password", "==", password).get();

    if (querySnapshot.empty) {
      // If no matching user found, display an error message
      const errorMessage = document.createElement("p");
      errorMessage.textContent = "Incorrect email or password. Please try again.";
```

```
      errorMessage.style.color = "red";
      document.querySelector("form").appendChild(errorMessage);
    } else {
      // If a matching user found, redirect to the homepage or another page
      window.location.href = "admin-home.html";
    }
  } catch (error) {
    console.error("Error fetching user data:", error);
    // Handle errors here, such as displaying a generic error message
  }
});
</script>




</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="admin-login.css">
  <title>Login Page</title>
</head>
<body>
  <div class="container">
    <div class="header">
      <a href="/home-page/homepage.html">
      <button class="back-home">Back to Home</button>
    </a>
    </div>
    <h1>ADMIN-LOGIN</h1>
    <form action="#" method="post">
      <label for="email">Email</label>
      <input type="email" id="email" name="email" required>

      <label for="password">Password</label>
      <input type="password" id="password" name="password" required>

      <button type="submit">Login
        <a href="/main/property.html"></a>
      </button>

    </form>
  </div>
</body>
</html>
```

`

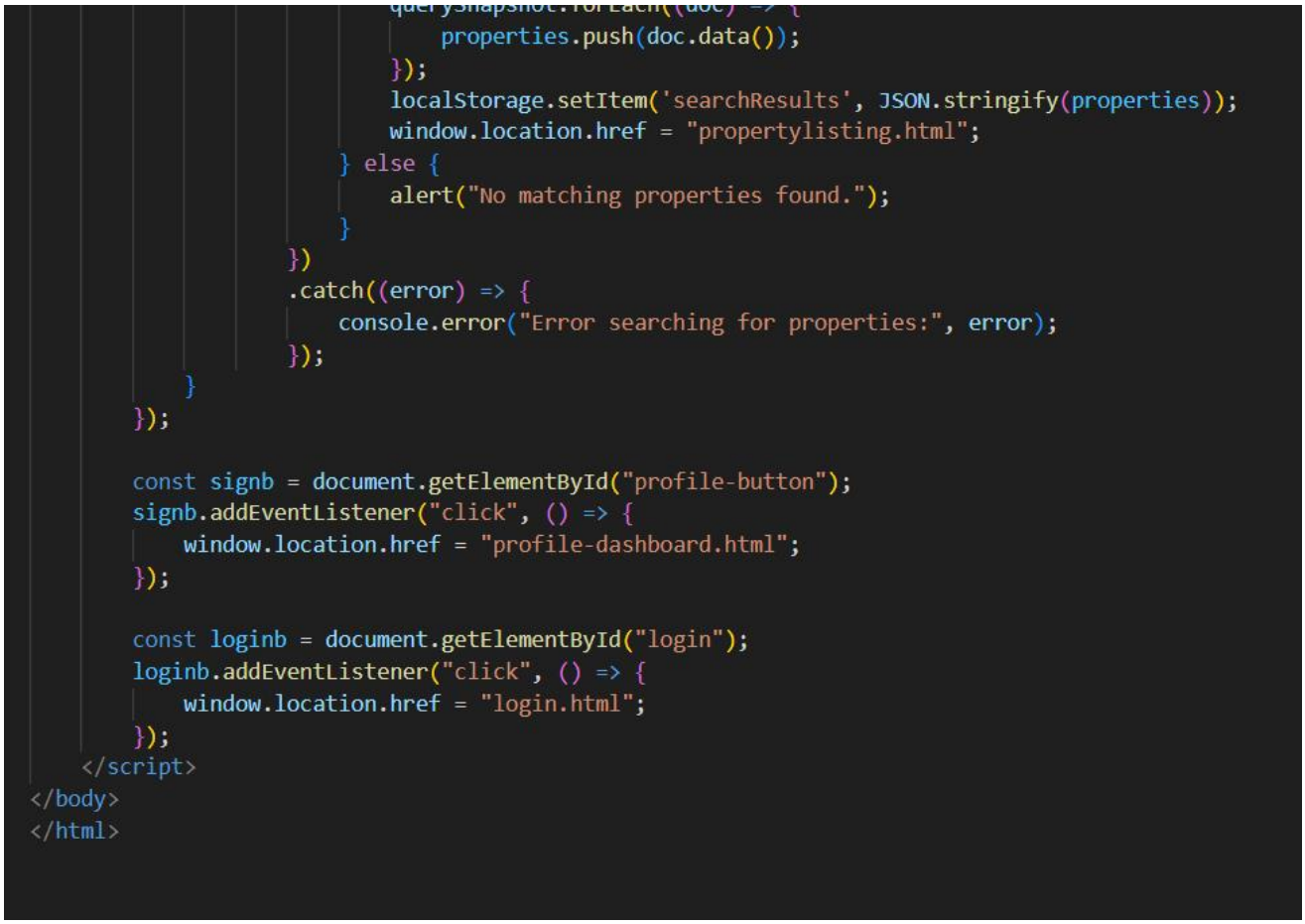
loggedhomepage.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Namastay</title>
    <link rel="stylesheet" href="loggedhomepage.css">
    <script src="https://www.gstatic.com/firebasejs/7.9.1/firebase-app.js"></script>
    <script src="https://www.gstatic.com/firebasejs/7.9.1/firebase-auth.js"></script>
    <script src="https://www.gstatic.com/firebasejs/7.9.1/firebase-firestore.js"></script>
</head>
<body>

    <button id="profile-button">Profile</button> <!-- Profile button -->

    <section id="hero">
        <div class="container">
            <h2>Namastay</h2>
            <form id="search-form">
                <input type="text" id="search-input" placeholder="Enter a destination..." required>
                <button class="search-submit" type="button" id="search-button">Search</button>
            </form>
        </div>
    </section>
    <section id="listings">
        <div class="container">
            <!-- Listings will be dynamically generated here -->
        </div>
    </section>
    <footer>
        <div class="container">
            <p>&copy; 2024 Namastay. All rights reserved.</p>
        </div>
    </footer>
```

`

```
<script>
    // Initialize Firebase
    const firebaseConfig = {
        apiKey: "AIzaSyDZq-Dz5dkj-I_-x6jNbr0HV4iV61LVd20",
        authDomain: "namastay-fa137.firebaseapp.com",
        databaseURL: "https://namastay-fa137-default-rtdb.firebaseio.com",
        projectId: "namastay-fa137",
        storageBucket: "namastay-fa137.appspot.com",
        messagingSenderId: "591100525672",
        appId: "1:591100525672:web:d826c565a424d5e2838fdd",
        measurementId: "G-BV1H3F10JN"
    };

    firebase.initializeApp(firebaseConfig);

    const searchButton = document.getElementById("search-button");
    const searchInput = document.getElementById("search-input");

    searchButton.addEventListener("click", () => {
        const searchTerm = searchInput.value.trim();

        if (searchTerm === "") {
            alert("Please enter a search term before clicking the search button.");
        } else {
            const db = firebase.firestore();
            const propertiesRef = db.collection("properties");

            propertiesRef.where("location", "==", searchTerm)
                .get()
                .then((querySnapshot) => {
                    if (!querySnapshot.empty) {
                        const properties = [];
                        querySnapshot.forEach((doc) => {
```

`



```
                    properties.push(doc.data());
                });
                localStorage.setItem('searchResults', JSON.stringify(properties));
                window.location.href = "propertylisting.html";
            } else {
                alert("No matching properties found.");
            }
        })
        .catch((error) => {
            console.error("Error searching for properties:", error);
        });
    }
});

const signb = document.getElementById("profile-button");
signb.addEventListener("click", () => {
    window.location.href = "profile-dashboard.html";
});

const loginb = document.getElementById("login");
loginb.addEventListener("click", () => {
    window.location.href = "login.html";
});
</script>
</body>
</html>
```

`

Admin-home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script src="https://www.gstatic.com/firebasejs/8.6.1/firebase.js"></script>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Real Estate Admin Portal</title>
    <link rel="stylesheet" href="admin-home.css">
</head>
<body>
    <div class="header">
        <a href="admin-bookings.html">
            <button class="view-bookings">View Bookings</button>
        </a>
    </div>
    <h1>Namastay Admin Portal</h1>
    <form id="property-form">
        <input type="text" id="name" placeholder="Property Name" required>
        <input type="text" id="location" placeholder="Location" required>
        <input type="number" id="price" placeholder="Price" required>
        <textarea id="description" placeholder="Description" required></textarea>
        <input type="file" id="image" accept="image/*" required>
        <button type="submit">Add Property</button>
    </form>
    <h2>Property List</h2>
    <div id="property-list"></div>

</body>
</html>

<script>
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyDZq-Dz5dkj-I_-x6jNbr0HV4iV61LVd20",
  authDomain: "namastay-fa137.firebaseapp.com",
  databaseURL: "https://namastay-fa137-default-rtdb.firebaseio.com",
```

`

```
  projectId: "namastay-fa137",
  storageBucket: "namastay-fa137.appspot.com",
  messagingSenderId: "591100525672",
  appId: "1:591100525672:web:d826c565a424d5e2838fdd",
  measurementId: "G-BV1H3F10JN"
};


firebase.initializeApp(firebaseConfig);
const db = firebase.firestore();
const storage = firebase.storage();

// Reference to the form and property list
const propertyForm = document.getElementById('property-form');
const propertyList = document.getElementById('property-list');

// Submit form and add property to Firestore
propertyForm.addEventListener('submit', (e) => {
    e.preventDefault();

    const name = document.getElementById('name').value;
    const location = document.getElementById('location').value;
    const price = document.getElementById('price').value;
    const description = document.getElementById('description').value;
    const image = document.getElementById('image').files[0];

    // Create a storage reference
    const storageRef = storage.ref('properties/' + image.name);

    // Upload the image to Firebase Storage
    const uploadTask = storageRef.put(image);

    uploadTask.on('state_changed',
        (snapshot) => {
```

`

```
            // Optional: Handle the progress of the upload
            let progress = (snapshot.bytesTransferred / snapshot.totalBytes) * 100;
            console.log('Upload is ' + progress + '% done');
        },
        (error) => {
            console.error('Error uploading image: ', error);
        },
        () => {
            // Get the image download URL and add the property details to Firestore
            uploadTask.snapshot.ref.getDownloadURL().then((downloadURL) => {
                db.collection('properties').add({
                    name: name,
                    location: location,
                    price: price,
                    description: description,
                    imageUrl: downloadURL
                }).then(() => {
                    console.log('Property added successfully');
                    propertyForm.reset();
                    loadProperties(); // Refresh the list
                }).catch((error) => {
                    console.error('Error adding property: ', error);
                });
            });
        }
    );
});

// Load properties from Firestore and display them
function loadProperties() {
    propertyList.innerHTML = ''; // Clear the list
```

`

```
    db.collection('properties').get().then((querySnapshot) => {
        querySnapshot.forEach((doc) => {
            const property = doc.data();
            const propertyDiv = document.createElement('div');
            propertyDiv.classList.add('property');
            propertyDiv.innerHTML = `
                <h3>${property.name}</h3>
                <p>Location: ${property.location}</p>
                <p>Price: $${property.price}</p>
                <p>${property.description}</p>
                <img src="${property.imageUrl}" alt="${property.name}">
            `;
            propertyList.appendChild(propertyDiv);
        });
    });
}

// Initial load of properties
loadProperties();

</script>
```

`

Admin-bookings.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Admin Bookings</title>
    <link rel="stylesheet" href="admin-bookings.css">
</head>
<body>
    <h1>Admin Bookings</h1>
    <div id="bookings-list"></div>

    <script src="https://www.gstatic.com/firebasejs/8.6.1/firebase.js"></script>
    <script>
        const firebaseConfig = {
            apiKey: "AIzaSyDZq-Dz5dkj-I_-x6jNbr0HV4iV61LVd20",
            authDomain: "namastay-fa137.firebaseapp.com",
            databaseURL: "https://namastay-fa137-default-rtdb.firebaseio.com",
            projectId: "namastay-fa137",
            storageBucket: "namastay-fa137.appspot.com",
            messagingSenderId: "591100525672",
            appId: "1:591100525672:web:d826c565a424d5e2838fdd",
            measurementId: "G-BV1H3F10JN"
        };

        firebase.initializeApp(firebaseConfig);
        const db = firebase.firestore();

        const bookingsList = document.getElementById('bookings-list');

        // Function to load and display bookings from Firestore
        function loadBookings() {
            db.collection('bookings').get().then((querySnapshot) => {
                querySnapshot.forEach((doc) => {
                    const booking = doc.data();
                    const bookingDiv = document.createElement('div');
                    bookingDiv.classList.add('booking');
```

```
                    bookingDiv.innerHTML = `
                        <h3>Booking Details</h3>
                        <p>Name: ${booking.name}</p>
                        <p>Email: ${booking.email}</p>
                        <p>Check-in Date: ${booking.checkIn}</p>
                        <p>Check-out Date: ${booking.checkOut}</p>
                        <button class="delete-booking" data-id="${doc.id}">Delete Booking</button>
                    `;
                    bookingsList.appendChild(bookingDiv);
                });
            });
        }

        // Call the function to load bookings when the page loads
        loadBookings();
    </script>
</body>
</html>
```

`

Index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Login & Registration Form</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <input type="checkbox" id="check">
    <div class="login form">
      <header>Login</header>
      <form action="#">
        <input type="text" id='inUsr' placeholder="Enter your email">
        <input type="password" id="inPass" placeholder="Enter your password">
        <span class="signup">
          <a href="#" id="forgotLabel">Forgot password?</a>
        </span>
        <input type="button" class="button loginbtn" value="Login">
      </form>
      <div class="signup">
        <span class="signup">Don't have an account?
          <a href="#" id="signupLabel">Signup</a>
        </span>
      </div>
    </div>
    <div class="registration form">
      <header>Signup</header>
      <form action="#">
        <input type="text" id="name" placeholder="Enter your name">
        <input type="text" id='username' placeholder="Enter your username">
        <input type="text" id='email' placeholder="Enter your email">
        <input type="password" id='password' placeholder="Create a password">
        <input type="button" class="button signupbtn" value="Signup">
```

`

```
      </form>
      <div class="signup">
        <span class="signup">Already have an account?
          <a href="#" id="loginLabel">Login</a>
        </span>
      </div>
    </div>
    <div class="forgot form" >
      <header>Forgot Password</header>
      <form action="#">
        <input type="text" id='forgotinp' placeholder="Enter your email">
        <input type="button" class="button forgotbtn" value="Submit">
      </form>
      <div class="signup">
        <span class="signup">Don't have an account?
          <a href="#" id="signupLabel">Signup</a>
        </span>
      </div>
      <div class="signup">
        <span class="signup">Already have an account?
          <a href="#" id="loginLabel">Login</a>
        </span>
      </div>
    </div>
  </div>
  <script src="https://www.gstatic.com/firebasejs/8.10.1/firebase-app.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.10.1/firebase-auth.js"></script>
<script src="https://www.gstatic.com/firebasejs/8.10.1/firebase-firestore.js"></script>
<script src="page0.js" type="module"></script>
</body>
</html>
```

`

Propertylisting.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Property Listing</title>
    <link rel="stylesheet" href="propertylisting.css">
</head>
<body>
    <section id="property-listings">
        <div class="container">
            <h2>Property Listings</h2>
            <div id="properties"></div>
        </div>
    </section>
    <footer>
        <div class="container">
            <p>&copy; 2024 Namastay. All rights reserved.</p>
        </div>
    </footer>

    <script>
        document.addEventListener("DOMContentLoaded", () => {
            const propertiesContainer = document.getElementById("properties");
            const properties = JSON.parse(localStorage.getItem('searchResults'));

            if (properties && properties.length > 0) {
                properties.forEach(property => {
                    const propertyDiv = document.createElement("div");
                    propertyDiv.classList.add("property");
                    propertyDiv.innerHTML = `
                        <h3>${property.name}</h3>
                        <div class="property-details">
                            <p>Location: ${property.location}</p>
                            <p>Price: $${property.price}</p>
                            <p>${property.description}</p>
                        </div>
                            </div>
                            <img src="${property.imageUrl}" alt="${property.name}">
                        `;
                        propertyDiv.addEventListener("click", () => {
                            localStorage.setItem('selectedProperty', JSON.stringify(property));
                            window.location.href = "booking.html";
                        });
                        propertiesContainer.appendChild(propertyDiv);
                    });
                } else {
                    propertiesContainer.innerHTML = "<p>No properties found.</p>";
                }
            });
        </script>
    </body>
    </html>
```

`

Profile-dashboard.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Profile Dashboard</title>
    <link rel="stylesheet" href="profile-dashboard.css">
    <script src="https://www.gstatic.com/firebasejs/7.9.1/firebase-app.js"></script>
    <script src="https://www.gstatic.com/firebasejs/7.9.1/firebase-firestore.js"></script>
</head>
<body>
    <section id="dashboard">
        <div class="container">
            <h2>My Bookings</h2>
            <ul id="booking-list"></ul>
        </div>
    </section>
    <footer>
        <div class="container">
            <p>&copy; 2024 Namastay. All rights reserved.</p>
        </div>
    </footer>

    <script>
        // Initialize Firebase
        const firebaseConfig = {
            apiKey: "AIzaSyDZq-Dz5dkj-I_-x6jNbr0HV4iV61LVd20",
            authDomain: "namastay-fa137.firebaseapp.com",
            databaseURL: "https://namastay-fa137-default-rtdb.firebaseio.com",
            projectId: "namastay-fa137",
            storageBucket: "namastay-fa137.appspot.com",
            messagingSenderId: "591100525672",
            appId: "1:591100525672:web:d826c565a424d5e2838fdd",
            measurementId: "G-BV1H3F10JN"
        };
```

`

```
    firebase.initializeApp(firebaseConfig);
    const db = firebase.firestore();

    const bookingList = document.getElementById("booking-list");

    // Function to delete a booking
    function deleteBooking(bookingId) {
        // Remove booking from UI
        const bookingItem = document.getElementById(bookingId);
        bookingItem.remove();

        // Remove booking from Firestore
        db.collection("bookings").doc(bookingId).delete()
            .then(() => {
                console.log("Booking successfully deleted!");
            })
            .catch((error) => {
                console.error("Error deleting booking: ", error);
            });
    }

    // Retrieve booking details from Firestore
    db.collection("bookings").get().then((querySnapshot) => {
        querySnapshot.forEach((doc) => {
            const bookingData = doc.data();
            const bookingId = doc.id;
            const bookingItem = document.createElement("li");
            bookingItem.setAttribute("id", bookingId);
            bookingItem.innerHTML = `
                <p>Name: ${bookingData.name}</p>
                <p>Email: ${bookingData.email}</p>
                <p>Check-in Date: ${bookingData.checkInDate}</p>
                <p>Check-out Date: ${bookingData.checkOutDate}</p>
                <button onclick="deleteBooking('${bookingId}')">Delete</button
                <hr>
            `;
            bookingList.appendChild(bookingItem);
        });
    }).catch((error) => {
        console.error("Error retrieving bookings: ", error);
    });
  </script>
</body>
</html>
```

`

Firebase.js

```
import { initializeApp } from "https://www.gstatic.com/firebasejs/10.12.0/firebase-app.js";
import { getAnalytics } from "https://www.gstatic.com/firebasejs/10.12.0/firebase-analytics.js";
import { getFirestore, getDocs, collection } from "https://www.gstatic.com/firebasejs/9.6.10/firebase-firestore.js";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyDZq-Dz5dkj-I_-x6jNbr0HV4iV61LVd20",
  authDomain: "namastay-fa137.firebaseapp.com",
  databaseURL: "https://namastay-fa137-default-rtdb.firebaseio.com",
  projectId: "namastay-fa137",
  storageBucket: "namastay-fa137.appspot.com",
  messagingSenderId: "591100525672",
  appId: "1:591100525672:web:d826c565a424d5e2838fdd",
  measurementId: "G-BV1H3F10JN"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const db=getFirestore(app)
const analytics = getAnalytics(app);
```

config.js

```
JS config.js ×

index > JS config.js > ...
1  const firebaseConfig = {
2    apiKey: "AIzaSyDM23VHHRQdUlJ7AzjkmQdnuyp_4v87FPI",
3    authDomain: "namastay-fa764.firebaseapp.com",
4    projectId: "namastay-fa764",
5    storageBucket: "namastay-fa764.appspot.com",
6    messagingSenderId: "433490487384",
7    appId: "1:433490487384:web:871d0392fa3ed7de16422d",
8    measurementId: "G-660G84CS0L"
9  };
10 export {firebaseConfig};
```