

# **Online Ration Shop**

*Mini Project Report*

*Submitted by*

**Nithin Rajappan**

**Reg. No.: AJC22MCA-2070**

*In Partial fulfillment for the Award of the Degree of*

**MASTER OF COMPUTER APPLICATIONS**

**(MCA TWO YEAR)**

**APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**



**AMAL JYOTHI COLLEGE OF ENGINEERING**

**KANJIRAPPALLY**

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with 'A' grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

**2023-2024**

**DEPARTMENT OF COMPUTER APPLICATIONS**  
**AMAL JYOTHI COLLEGE OF ENGINEERING**  
**KANJIRAPPALLY**



**CERTIFICATE**

This is to certify that the Project report, “Online Ration Shop” is the bonafide work of **Nithin Rajappan (Regno: AJC22MCA-2070)** in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2023-24.

**Ms.Sruthimol Kurian**

**Internal Guide**

**Ms. Meera Rose Mathew**

**Coordinator**

**Rev. Fr. Dr. Rubin Thottupurathu Jose**

**Head of the Department**

## **DECLARATION**

I hereby declare that the project report “**Online Ration Shop**” is a bona-fide work of done at Amal Jyothi College of Engineering, towards the partial fulfillment of the requirements for the award of the Master of Computer Applications ( MCA) from APJ Abdul Kalam Technological University, during the academic year 2023-2024.

**Date:**

**Nithin Rajappan**

**Reg: AJC22MCA-2070**

**KANJIRAPPALLY**

## ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our Manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev.Fr.Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Ms. Meera Rose Mathew** for her valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Ms.Sruthimol Kurian** for her inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

**Nithin Rajappan**

## ABSTRACT

**Online-Ration Shop** The scheme of providing basic domestic commodities to poor families in developing countries like India is important aspect to meet fundamental requirement of people. If autonomous system is available for ration shop it is very helpful for the card holders and the employees of the shop in many ways. E-Ration shopping's a system developed to dispense the correct quantity of ration to the card holders depending on type of card and the number of members in the family, and also maintain the details of transactions in database.

The **Admin** Module encompasses critical functionalities to maintain control and ensure the smooth operation of the platform. This includes the ability to log in and out securely, manage shop owner accounts, and control access permissions. Admins also play a pivotal role in overseeing and regulating stock details, ensuring efficient inventory management.

For **Shop Owners**, the module facilitates straightforward registration and creation of shop details. Shop owners can actively manage user registrations and details, exercise control over stock by creating, editing, and managing inventory. Additionally, they have the flexibility to update essential information such as shop details, working hours, and contact details.

In the **User Module**, individuals seeking properties for rent can seamlessly register, update, and delete their profiles. The platform provides secure and convenient payment options for transactions, ensuring a smooth and trustworthy process. Users also have the functionality to submit ration requests and initiate purchases, enhancing their overall experience on the platform.

# CONTENT

SL. NO	TOPIC	PAGE NO
<b>1</b>	<b>INTRODUCTION</b>	
<b>1.1</b>	<b>PROJECT OVERVIEW</b>	
<b>1.2</b>	<b>PROJECT SPECIFICATION</b>	
<b>2</b>	<b>SYSTEM STUDY</b>	
<b>2.1</b>	<b>INTRODUCTION</b>	
<b>2.2</b>	<b>EXISTING SYSTEM</b>	
<b>2.3</b>	<b>DRAWBACKS OF EXISTING SYSTEM</b>	
<b>2.4</b>	<b>PROPOSED SYSTEM</b>	
<b>2.5</b>	<b>ADVANTAGES OF PROPOSED SYSTEM</b>	
<b>3</b>	<b>REQUIREMENT ANALYSIS</b>	
<b>3.1</b>	<b>FEASIBILITY STUDY</b>	
<b>3.1.1</b>	<b>ECONOMICAL FEASIBILITY</b>	
<b>3.1.2</b>	<b>TECHNICAL FEASIBILITY</b>	
<b>3.1.3</b>	<b>BEHAVIORAL FEASIBILITY</b>	
<b>3.1.4</b>	<b>FEASIBILITY STUDY QUESTIONNAIRE</b>	
<b>3.2</b>	<b>SYSTEM SPECIFICATION</b>	
<b>3.2.1</b>	<b>HARDWARE SPECIFICATION</b>	
<b>3.2.2</b>	<b>SOFTWARE SPECIFICATION</b>	
<b>3.3</b>	<b>SOFTWARE DESCRIPTION</b>	
<b>3.3.1</b>	<b>DJANGO</b>	
<b>3.3.2</b>	<b>SQLITE</b>	
<b>4</b>	<b>SYSTEM DESIGN</b>	
<b>4.1</b>	<b>INTRODUCTION</b>	
<b>4.2</b>	<b>UML DIAGRAM</b>	
<b>4.2.1</b>	<b>USE CASE DIAGRAM</b>	
<b>4.2.2</b>	<b>SEQUENCE DIAGRAM</b>	
<b>4.2.3</b>	<b>STATE CHART DIAGRAM</b>	
<b>4.2.4</b>	<b>ACTIVITY DIAGRAM</b>	
<b>4.2.5</b>	<b>CLASS DIAGRAM</b>	
<b>4.2.6</b>	<b>OBJECT DIAGRAM</b>	

<b>4.2.7</b>	<b>COMPONENT DIAGRAM</b>	
<b>4.2.8</b>	<b>DEPLOYMENT DIAGRAM</b>	
<b>4.2.9</b>	<b>COLLABORATION DIAGRAM</b>	
<b>4.3</b>	<b>USER INTERFACE DESIGN USING FIGMA</b>	
<b>4.4</b>	<b>DATABASE DESIGN</b>	
<b>5</b>	<b>SYSTEM TESTING</b>	
<b>5.1</b>	<b>INTRODUCTION</b>	
<b>5.2</b>	<b>TEST PLAN</b>	
<b>5.2.1</b>	<b>UNIT TESTING</b>	
<b>5.2.2</b>	<b>INTEGRATION TESTING</b>	
<b>5.2.3</b>	<b>VALIDATION TESTING</b>	
<b>5.2.4</b>	<b>USER ACCEPTANCE TESTING</b>	
<b>5.2.5</b>	<b>AUTOMATION TESTING</b>	
<b>5.2.6</b>	<b>SELENIUM TESTING</b>	
<b>6</b>	<b>IMPLEMENTATION</b>	
<b>6.1</b>	<b>INTRODUCTION</b>	
<b>6.2</b>	<b>IMPLEMENTATION PROCEDURE</b>	
<b>6.2.1</b>	<b>USER TRAINING</b>	
<b>6.2.2</b>	<b>TRAINING ON APPLICATION SOFTWARE</b>	
<b>6.2.3</b>	<b>SYSTEM MAINTENANCE</b>	
<b>7</b>	<b>CONCLUSION &amp; FUTURE SCOPE</b>	
<b>7.1</b>	<b>CONCLUSION</b>	
<b>7.2</b>	<b>FUTURE SCOPE</b>	
<b>8</b>	<b>BIBLIOGRAPHY</b>	
<b>9</b>	<b>APPENDIX</b>	
<b>9.1</b>	<b>SAMPLE CODE</b>	
<b>9.2</b>	<b>SCREEN SHOTS</b>	

## List of Abbreviation

IDE	Integrated Development Environment
HTML	Hyper Text Markup Language.
CSS	Cascading Style Sheet
SQLite	Relational Database Management System
UML	Unified Modeling Language
JS	Java Script



# **CHAPTER 1**

## **INTRODUCTION**

## 1.1 PROJECT OVERVIEW

The E-Ration Shop project seeks to transform the distribution of essential commodities for economically disadvantaged families, especially in countries like India. The current manual system faces challenges such as long queues, cumbersome record-keeping, and limited access for those with physical disabilities. To address these issues, the proposed E-Ration Shop introduces an autonomous and user-friendly system to improve the efficiency and convenience of ration distribution. Key drawbacks of the existing system include extended waiting times for cardholders, manual record-keeping prone to errors, and delays in updating personal information on ration cards. Lack of information about ration distribution days adds to the challenges, and physically challenged individuals find it hard to access ration shops. In response, the E-Ration Shop offers a 24/7 accessible platform to bypass long queues. Automated transaction recording ensures accuracy, managed by a robust SQL server. The admin module provides secure login/logout functionality and oversight of shop owner accounts, access permissions, and stock details. Shop owners benefit from an efficient registration process, user management capabilities, and control over inventory. They can update shop details, including working hours and contact information. Users enjoy streamlined profile management, secure payment options, and the ability to request and purchase ration products online. The overarching goal is to overcome the limitations of manual ration distribution by creating a transparent, accessible, and efficient platform. The E-Ration Shop aims to meet fundamental needs while minimizing administrative complexities and ensuring a more equitable distribution of essential resources.

---

## PROJECT SPECIFICATION

In the Proposed System, the user can access the system for ration at any time and it is easy for them to use. This system enrolls each transaction automatically in to the database and the database is created using Sql server. The main reason for creating the project is making this computerized and to remove the drawbacks of present way of issuing product base of on ration card and avoid the drawbacks present ration store issues.

### Admin Module:

- Login and Logout.
- Manage shop onwer accounts and access permissions.
- Manage Stock details

### Shop onwer Module:

- Register and create a the shop details
- Manage user registration and details
- Create, edit, and manage stocks.
- Shop details ,Working hours, Contact details

### User Module:

- Profile Registration ,update, delete ,login
- Payment option
- Ration request and buying option

### Technologies Used:

Frontend: HTML, CSS, JavaScript

Backend: Django.

Database: SQLite.

Payment Gateway Integration: Razor pay.

## **CHAPTER 2**

### **SYSTEM STUDY**

## **2.1 INTRODUCTION**

The Proposed System for the E-Ration Shop introduces a user-friendly platform, allowing users to access ration services conveniently at any time. The system automates transaction enrollment into a SQL server database, streamlining the process. The primary objective is to modernize and address the drawbacks of the current ration distribution system based on physical ration cards. This project aims to eliminate inefficiencies present in traditional ration stores. The Admin Module facilitates secure login/logout, overseeing shop owner accounts, access permissions, and stock details. Shop owners benefit from efficient registration, user management, and comprehensive control over inventory and shop details. Users experience a simplified registration process, secure payment options, and the ability to request and purchase ration products online. Overall, the E-Ration Shop endeavors to create a computerized, accessible, and efficient platform, rectifying existing issues and ensuring a more effective distribution of essential resources.

## **2.2 EXISTING SYSTEM**

This manual system is facing many problems such as • Card holders wasting time for collecting ration by standing in a queue for hours together. • Maintenance of records in the form of hard copy is difficult. • It takes long time to change any personal information in the ration card. • Sometimes people don't get to know about on which day ration is going to be provided by the shop. • For physically challenged people it's difficult for them to go to the ration shop. There is no online system to buy ration product and stock maintenance. Government will provide product to buy that waiting for queuing, bargaining and products are robbed by making wrong entries without knowledge of the ration card holder. So government meets money loss, time and cost.

### **2.2.1 NATURAL SYSTEM STUDIED**

The Online ration shop Management System functions as a natural system, embodying a dynamic ecosystem where technological components, user behaviors, and market forces seamlessly converge. The technological backbone, comprised of servers and databases, facilitates efficient property management and user interactions, forming the foundation of the system. Users, as integral participants in this ecosystem, exert influence through their behaviors and preferences, shaping the platform's response to evolving market dynamics and trends within the property rental industry. This holistic approach ensures a symbiotic relationship between technology, user engagement, and the ever-changing landscape of property rentals.

### **2.2.2 DESIGNED SYSTEM STUDIED**

2.3 The scheme of providing basic domestic commodities to poor families in developing countries like India is important aspect to meet fundamental requirement of people. If autonomous system is available for ration shop it is very helpful for the card holders and the employees of the shop in many ways. E-Ration shopping's a system developed to dispense the correct quantity of ration to the card holders depending on type of card and the number of members in the family, and also maintain the details of transactions in database.

### **2.4 DRAWBACKS OF EXISTING SYSTEM**

This manual system is facing many problems such as

- Card holders wasting time for collecting ration by standing in a queue for hours together.
- Maintenance of records in the form of hard copy is difficult.
- It takes long time to change any personal information in the ration card.
- Sometimes people don't get to know about on which day ration is going to be provided by the shop.
- For physically challenged people it's difficult for them to go to the ration shop.

There is no online system to buy ration product and stock maintenance. Government will provide product to buy that waiting for queuing, bargaining and products are robbed by making wrong entries without knowledge of the ration card holder. So government meets money loss, time and cost.

### **2.5 PROPOSED SYSTEM**

The In the Proposed System, the user can access the system for ration at any time and it is easy for them to use. This system enrolls each transaction automatically in to the database and the database is created using Sql server. The main reason for creating the project is making this computerized and to remove the drawbacks of present way of issuing product base of on ration card and avoid the drawbacks present ration store issues.

### **ADVANTAGES OF PROPOSED SYSTEM**

- ✧ The proposed online ration shop project brings forth several advantages that mark a significant leap from traditional manual systems. Firstly, it offers users the convenience of accessing the ration system at any time, providing a 24/7 accessible platform. The system's automation ensures that each transaction is seamlessly recorded into a secure SQL server database,

promoting accuracy and efficiency in managing ration distribution.

- ✧ This computerized approach addresses the drawbacks of the existing manual system, where issues like prolonged waiting times, cumbersome record-keeping, and limited accessibility for physically challenged individuals persist. By introducing an online platform, the project eliminates the need for cardholders to stand in long queues, streamlining the ration collection process.
- ✧ For administrators, the system provides a secure login/logout mechanism and efficient management of shop owner accounts, access permissions, and stock details. Shop owners benefit from simplified registration processes, comprehensive control over inventory, and the ability to manage user registrations and details. The system allows them to update shop-specific information, including working hours and contact details.
- ✧ Users, on the other hand, experience a user-friendly interface for profile registration, updates, and deletion. Secure payment options enhance the online transaction process, and users can seamlessly request and purchase ration products through the platform.
- ✧ In essence, the proposed online ration shop project embraces technology to enhance accessibility, efficiency, and transparency in the distribution of essential commodities, providing a more user-centric and modernized approach to address the challenges inherent in traditional ration store systems.

## **CHAPTER 3**

### **REQUIREMENT ANALYSIS**



### **3.1 FEASIBILITY STUDY**

Feasibility is the degree to which a project can be carried out successfully. A feasibility study is conducted to assess the solution's viability, which establishes whether it is viable and implementable in the program. The feasibility study considers details like the availability of resources, software development costs, the advantages of the software to the business once it is built, and the costs associated with maintaining it. The outcome of the feasibility study should be a report recommending whether the requirements engineering, and system development process should be continued. A system is of no real value to a corporation if it does not serve its goals. Even though this may seem obvious, many organizations create systems that do not support their goals, either because they lack a clear statement of these goals, because they fail to specify the system's business requirements, or because other organizational or political factors have an impact on the procurement of the system.

#### **3.1.1 ECONOMIC FEASIBILITY**

The economic feasibility of your online property rentals project is critical to assess its financial viability and potential profitability. Conducting a thorough evaluation of various financial aspects will help determine if the project is economically sound. Begin with a comprehensive cost-benefit analysis, comparing the total expenses of development, operation, and maintenance against the expected benefits and revenue generation from book sales. Estimate the initial development costs, ongoing operational expenses, and potential revenue from property sales to calculate the projected return on investment (ROI) and the break-even point. Analyze the profit margin per book sale to ensure sustainability and competitiveness. Additionally, conduct market research to understand the demand, competition, and growth potential in the book industry. By conducting a comprehensive economic feasibility analysis, you can make informed decisions and assess the project's financial potential for long-term success.

#### **3.1.2 TECHNICAL FEASIBILITY**

The technical feasibility of your online property rentals project is crucial to determine whether it can be practically implemented from a technological standpoint. Evaluating various aspects is essential for its success. Firstly, ensure that your organization has the necessary technology infrastructure or can acquire it to support the platform, including web hosting, databases, and server capacity. Check for skilled software developers who can build and maintain the system effectively. Integration with third-party services like payment gateways and shipping providers must be

seamless. The architecture should be scalable to accommodate increasing traffic and data as the platform grows. Security measures to protect user data and transactions must be robust. Ensure the platform is mobile-friendly for users on various devices, and its performance is fast and responsive. Compliance with data protection and privacy regulations is essential. Lastly, evaluate the financial resources required for development, hosting, maintenance, and support. A comprehensive assessment of technical feasibility will help identify potential challenges and enable you to make informed decisions to create a successful online bookstore platform

### **3.1.3 BEHAVIORAL FEASIBILITY**

The Behavioral feasibility of your online property rentals project is essential to assess whether the proposed system can be effectively integrated into the organization's existing operations. Evaluating various aspects will help determine the project's viability and success. Firstly, ensure that the necessary resources, including human capital, technology, and infrastructure, are available or can be obtained within the project's time frame and budget. Consider the skill set of the organization's staff and identify the need for additional training or hiring. Evaluate how well the new system will integrate with existing processes, such as inventory management and order processing. Plan for change management to address potential resistance to organizational changes resulting from the system implementation. Ensure that third-party services or suppliers can support operational requirements. Verify compliance with legal and regulatory aspects related to online sales and data protection. Gather user feedback to gauge acceptance and identify areas for improvement. Lastly, analyse operational costs to ensure alignment with the organization's budget. A thorough assessment of operational feasibility will help ensure the successful integration of the online bookstore platform into the organization's existing operations.

### **3.1.4 FEASIBILITY STUDY QUESTIONNAIRE**

#### **1. Project Overview?**

The "Property Rentals" platform is an innovative digital solution designed to streamline and enhance the property rental experience for both landlords and tenants. Our platform is dedicated to providing a seamless and user-friendly interface, allowing users to effortlessly navigate, discover, and engage with available rental properties from the comfort of their homes. Property Rentals offers a diverse range of rental options, including residential properties, commercial spaces, and land for various purposes, ensuring a comprehensive selection to meet the diverse needs of our users. With a commitment to user satisfaction and leveraging advanced technology, Property Rentals aims to establish a digital sanctuary for property seekers, offering a hassle-free and enjoyable journey to

find and secure their ideal rental spaces. Whether searching for a home, a business location, or a plot of land for specific purposes, Property Rentals endeavors to simplify the entire rental process.

## **.2.To what extend the system is proposed for?**

The proposed Property Rentals platform is a comprehensive online system designed to serve a broad audience in the property rental domain. Offering diverse rental spaces such as residential, commercial, and land, the platform ensures accessibility for tenants and property owners alike. With a user-friendly interface, transparent property information, and secure transaction processes, it aims to provide a seamless rental experience. Community engagement features like ratings, reviews, wishlists, and periodic discounts enhance user interaction. The platform aspires to revolutionize the property rental landscape, making it accessible and enjoyable for a wide range of users, including tenants seeking their ideal spaces and property owners with varying rental offerings.

## **3.Specify the Viewers/Public which is to be involved in the System?**

The Property Rentals platform involves individuals seeking rental spaces, property owners looking to list their properties, students searching for accommodation, business owners scouting commercial spaces, and anyone in need of land for various purposes. Tenants explore diverse rental options, while property owners showcase their available spaces. Students find suitable accommodations, and businesses discover commercial properties. The platform serves a broad audience, making it inclusive for individuals with varied rental needs. Whether someone is seeking a home, a storefront, or land for a specific purpose, the Property Rentals system aims to provide a user-friendly and comprehensive solution for all involved in the property rental process.

## **4.List the Modules included in your System?**

Here is a concise list of modules included in The Property Rentals system:

1. User Registration and Login
2. Property Listing and Management
3. Ration Details
4. Sending Ration Requests
5. Payment Gateway Integration
6. User Account Management
7. Admin Panel

Each module plays a crucial role in providing users with a seamless and enjoyable online Property Rentals experience while efficiently managing the platform's operations and enhancing user satisfaction.

## **5. Identify the users in your project?**

The Property Rentals involves three types of users: Property Owner, Tenant and admin.

## **6. Who owns the system?**

Administrator

## **7. System is related to which firm/industry/organization?**

The system described, a Property Rentals platform, is related to the real estate and property rental industry. It caters to individuals and businesses seeking rental spaces, as well as property owners looking to list their properties for rent. The platform is designed to streamline the process of discovering, booking, and managing rental properties, covering various categories such as residential, commercial, and land. Targeted towards tenants, property owners, students, businesses, and anyone in need of rental spaces, the system aims to provide a user-friendly and comprehensive solution in the dynamic realm of property rentals.

## **8. Details of person that you have contacted for data collection.**

Name:

Position:

## **9. Questionnaire to collect details about the project?**

1. What motivated the development of the E-Ration Shop project? - The development of the E-Ration Shop project was motivated by the need to improve the efficiency of the ration distribution system and enhance the overall experience for cardholders.
2. Can you explain the key features and functionalities of the platform in more detail? - Certainly, the platform offers features like automated ration quantity calculation, user-friendly registration and login, convenient browsing of available products, and secure transactions. It also includes administrative functions for managing shop owners, stock, and support.
3. How does the system determine the correct quantity of ration for cardholders? - The system calculates the ration quantity based on the type of card and the number of members in the cardholder's family. It ensures that the distribution aligns with the entitlements specified for each card type.
4. What technologies are being used for the frontend and backend development of the project? - The project utilizes modern web development technologies, with React for the frontend and Node.js for the backend.
5. How do you plan to ensure cardholder convenience and accessibility through the system? - Cardholders can access the system anytime, eliminating the need to wait in long queues. The user interface is designed to be user-friendly and accessible, making it easy for cardholders to register, browse, and place orders.

6. What administrative functions are available for managing shop owners, stock, and support? - Admins can manage shop owner accounts, access permissions, stock details, and provide support as needed. They play a crucial role in ensuring the system's smooth operation.
7. Can you describe the user registration and login process? - Users register by providing necessary details, and the login process requires a username and password for authentication.
8. How are transactions recorded and stored in the database? - Transactions are automatically recorded and stored in the database, creating a digital record of all ration distribution activities for future reference and analysis.
9. How does the system support shop owners in managing their stock? - Shop owners have the ability to manage their stock, add new items, update existing ones, and remove items that are no longer available. This feature ensures that the stock is always up-to-date and accurate.
10. What measures are in place to ensure data security and privacy for cardholders' information? - The system implements robust security measures, including encryption and access controls, to protect cardholders' information and ensure data privacy. Regular security audits are conducted to identify and address vulnerabilities.

## SYSTEM SPECIFICATION

### 3.2.1 Hardware Specification

Processor	AMD Ryzen3 7000
RAM	8GB
Hard disk	512 SSD

### 3.2.2 Software Specification

Front End	HTML5, CSS, Bootstrap
Back End	Django, SQLite
Database	SQLite
Client on PC	Windows 7 and above.
Technologies used	Django, HTML5, Bootstrap, JS,

### **3.3 SOFTWARE DESCRIPTION**

#### **3.3.1 DJANGO**

Django, a leading open-source web framework, epitomizes the pinnacle of modern web development with its efficiency and versatility. Built on Python, Django follows the Model-View-Controller architecture, prioritizing simplicity and flexibility. Noteworthy features include its Object-Relational Mapping system, streamlined URL routing, and a user-friendly template engine. The built-in admin interface simplifies data management, while robust security measures and middleware support enhance application integrity. Django scales effortlessly, accommodating growing datasets and traffic demands. Its REST Framework extension further extends its utility to API development. Supported by an active community and extensive documentation, Django stands as a powerful toolkit, seamlessly shaping the development of dynamic and secure web applications for diverse purposes, from content management systems to Restful API. In essence, Django empowers developers with a comprehensive and adaptable framework for crafting sophisticated and salable web solutions.

#### **3.3.2SQLite**

SQLite, a lightweight and server less relational database management system, is celebrated for its simplicity, portability, and efficiency. Operating as a self-contained, single-file database, SQLite requires minimal setup and eliminates the need for a separate server process. This design makes it a preferred choice for embedded systems, mobile applications, and small to medium-scale projects. With zero configuration and a server less architecture, SQLite streamlines the database management process. It supports ACID transactions, ensuring data integrity and reliability, even in the face of system interruptions. The dynamic typing feature allows flexible data storage, accommodating various data types within the same column.

Noteworthy is SQLite's cross-platform compatibility, making it versatile across different operating systems and environments. Its wide adoption is evident in applications ranging from mobile apps to web browsers, where it is utilized for local data storage. SQLite stands out as a go-to solution for projects requiring a lightweight, self-contained, and easily deployable database system, embodying simplicity without compromising on functionality and reliability.

## **CHAPTER 4**

### **SYSTEM DESIGN**

## 4.1 INTRODUCTION

Any designed system or product's development process begins with the design phase. An efficient system depends on a well-executed design, which is a creative process. It entails utilizing a variety of techniques and concepts to completely specify a procedure or system for it to be implemented. Regardless of the development paradigm used in software engineering, the design phase is essential. It seeks to produce the architectural detail needed to design a system or product and acts as the process's technical backbone.

To maximize every aspect of efficiency, performance, and accuracy, this program underwent a comprehensive design phase. A user-oriented document is converted into a document for programmers or database workers throughout the design phase.

## 4.2 UML DIAGRAM

Software engineering uses the Unified Modeling Language (UML), a standardized visual language, to model, develop, and document software systems. UML diagrams provide a concise and organized approach to represent different facets of a software system, serving as a common communication tool for developers, stakeholders, and designers. There are many different varieties of UML diagrams, including class diagrams, sequence diagrams, use case diagrams, and more. Each is designed to communicate a particular piece of knowledge about the design, operation, and interactions of the system. UML diagrams are essential to the software development process because they help with the visualization, analysis, and design of complex systems, which in turn makes the process more effective and efficient.

- Use case diagram
- Sequence diagram
- State diagram
- Activity diagram
- Class diagram
- Object diagram
- Component diagram
- Deployment diagram



### 4.2.1 USE CASE DIAGRAM

A use case diagram is a visual representation of the interactions between system components. An approach for identifying, outlining, and organizing system requirements is called a use case. The word "system" here refers to a thing that is being created or run, like a website for mail-order product sales and services. UML (Unified Modeling Language), a standard language for the modelling of real-world objects and systems, uses use case diagrams. The planning of general requirements, the validation of a hardware design, the testing and debugging of a software product in development, the creation of an online help reference, or the completion of a job focused on customer support are all examples of system objectives. For instance, use cases in a product sales context can involve customer service, item ordering, catalogue updating, and payment processing. There are four elements in a use case diagram.

- The boundary, which defines the system of interest in relation to the world around it.
- The actors, usually individuals involved with the system defined according to their roles.
- The use cases, which are the specific roles are played by the actors within and around the system.
- The relationships between and among the actors and the use cases. Use case diagrams are drawn to capture the functional requirements of a system. After identifying the above items, we must use the following guidelines to draw an efficient use case diagram.
- The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements.
- Use notes whenever required to clarify some important points.

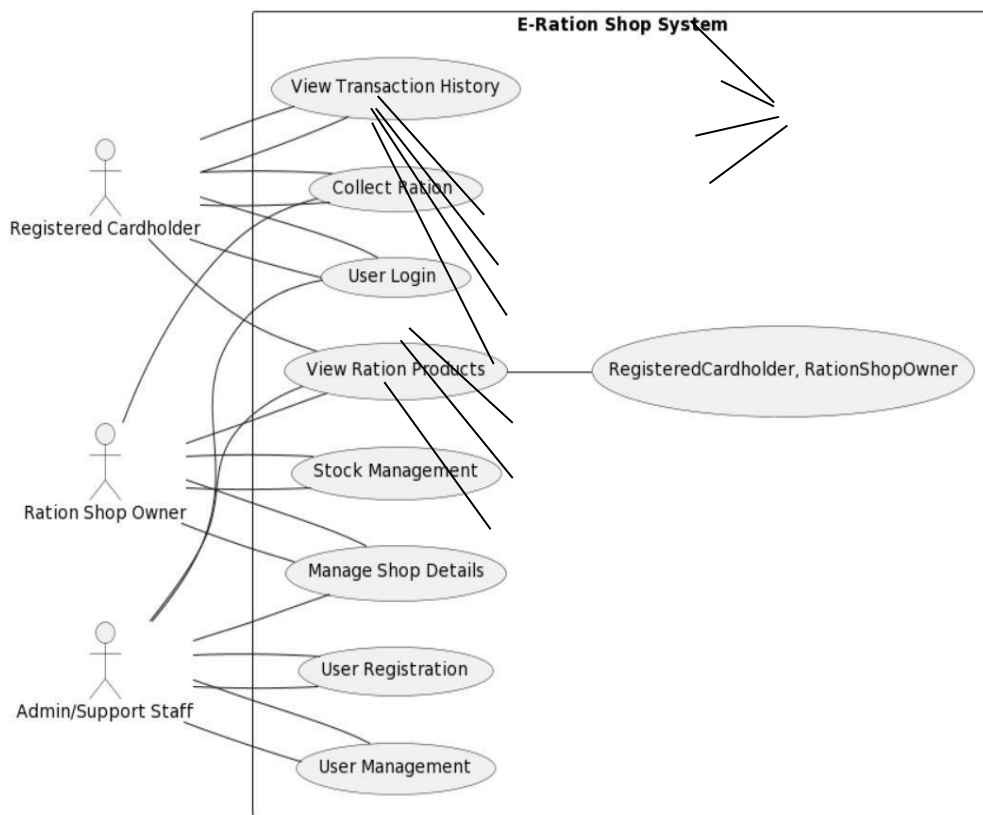
1

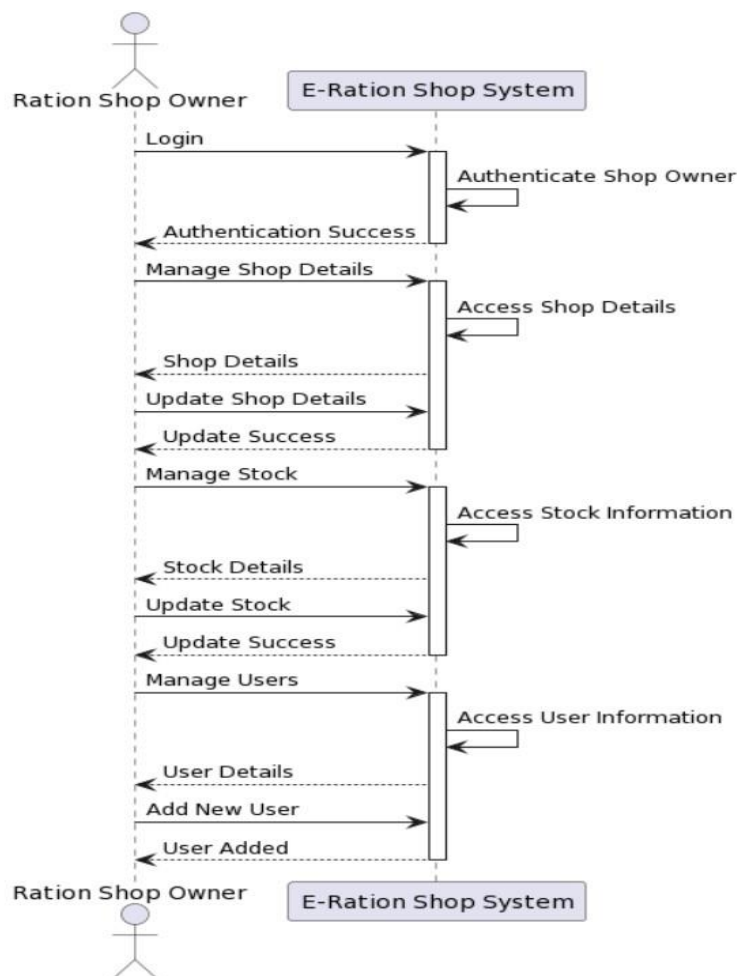
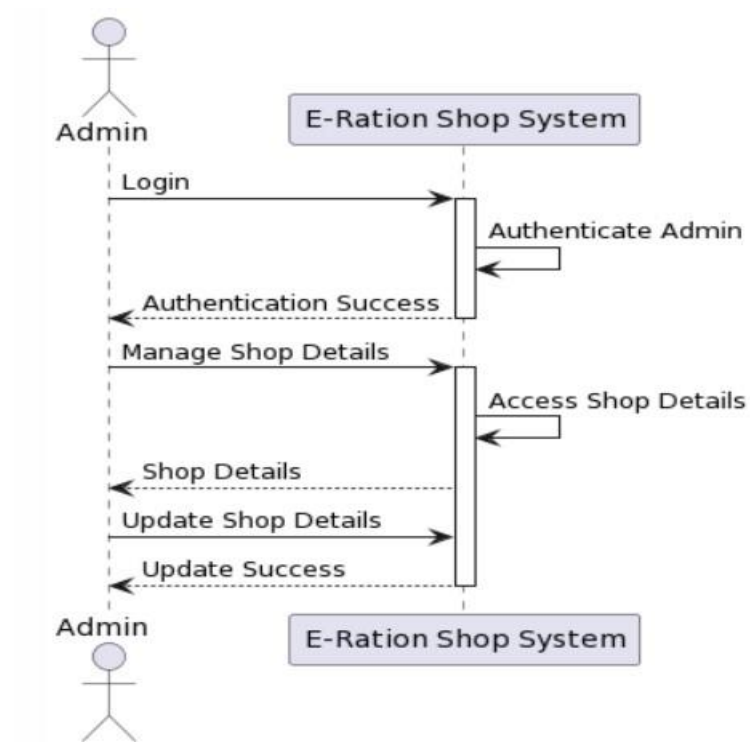
Figure 1: Use Case diagram

### 4.2.2 SEQUENCE DIAGRAM

A sort of Unified Modeling Language (UML) diagram called a sequence diagram is used in software engineering to depict the communications and interactions that take place across time between various system components or objects. It gives a live view of the interactions between objects as they work together to complete a certain job or situation.

Objects are shown as lifelines in a sequence diagram, and vertical lines indicate their existence over time. The flow of calls or messages between objects is shown by arrows and lines between lifelines. These diagrams are very helpful for demonstrating the interactions within a particular use case, assisting in determining the sequence of operations and the functions that each object performs.

Sequence diagrams are essential for understanding the behavior of a system and for ensuring that the software functions as intended, as they offer a clear and detailed depiction of how different components or objects collaborate to achieve a particular functionality.



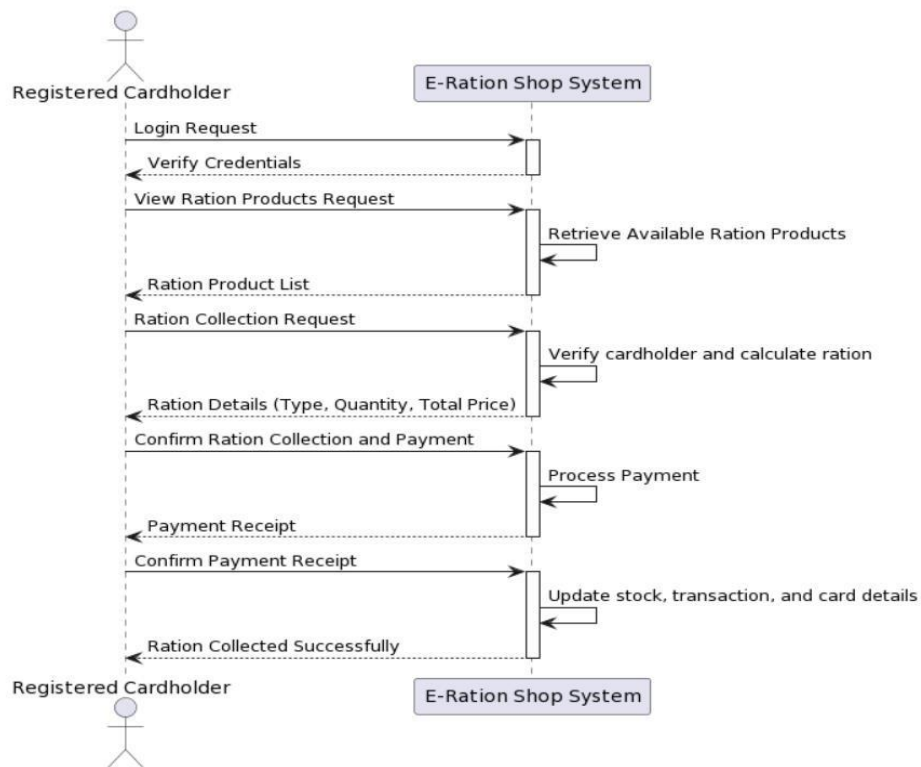


Figure 2: Sequence diagram

### 4.2.3 STATE CHART DIAGRAM

A state diagram, also known as a state machine diagram or state chart diagram, is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modeling Language (UML). In this context, a state defines a stage in the evolution or behavior of an object, which is a specific entity in a program or the unit of code representing that entity. A state diagram resembles a flowchart in nature; however, a flowchart shows the processes within a system that alters the state of an object rather than the actual state changes themselves. The first step to creating a state chart diagram is identifying the initial and final states of a system. Then, all the possible existing states are placed in relation to the beginning and the end. Lastly, all of the events that trigger state changes are labeled as transition elements.

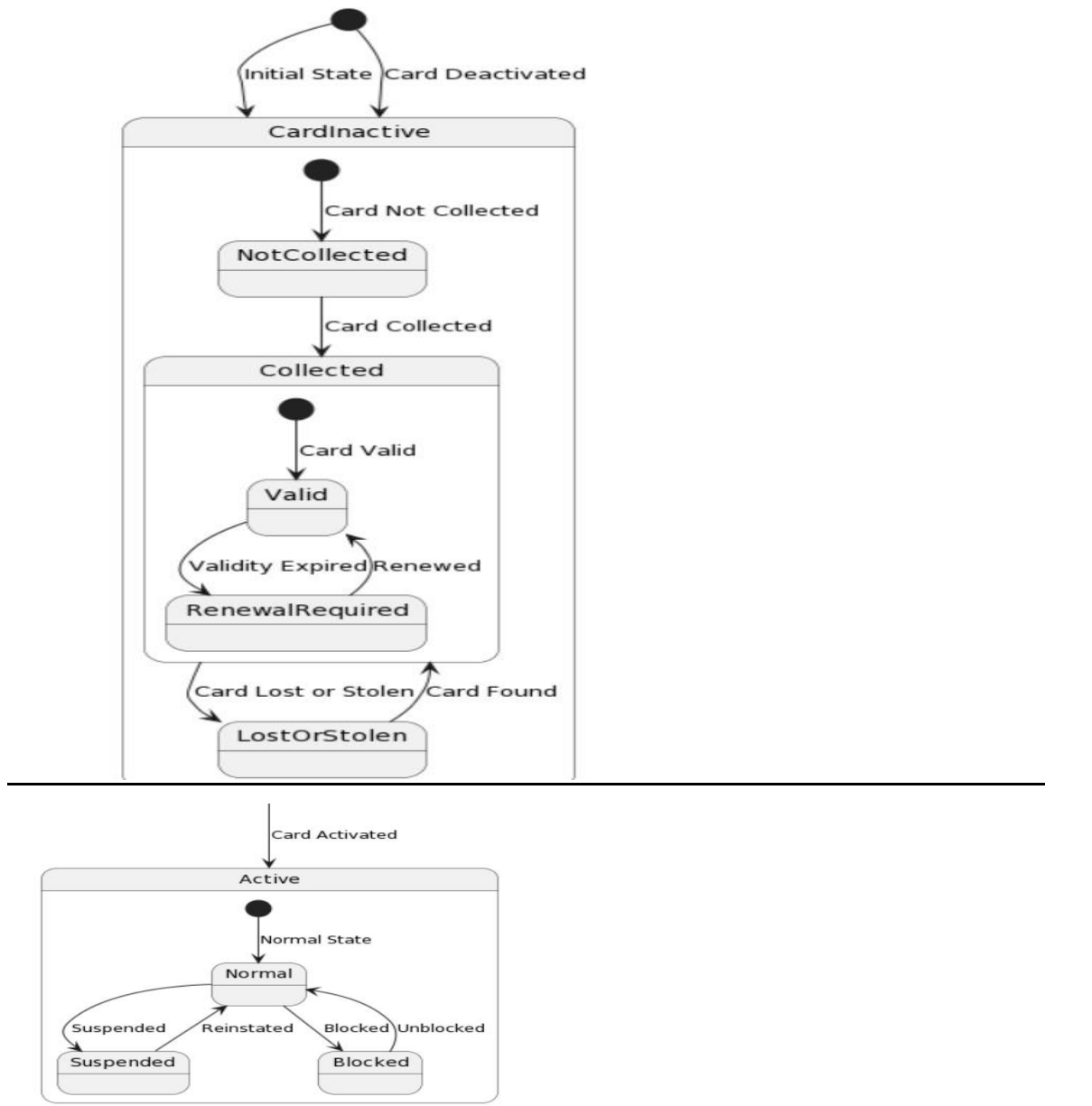


Figure 3: State Chart diagram

#### 4.2.4 ACTIVITY DIAGRAM

An activity diagram is essentially a flowchart that shows how one activity leads to another. The action might be referred to as a system operation. One operation leads to the next in the control flow. This flow may be parallel, contemporaneous, or branched. Activity diagrams use many features, such as fork, join, etc., to cope with all types of flow control. An activity diagram is a behavioral diagram i.e., it depicts the behavior of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the

activity is being executed.

Activity diagram is another important behavioral diagram in UML diagram to describe dynamic aspects of the system. Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity.

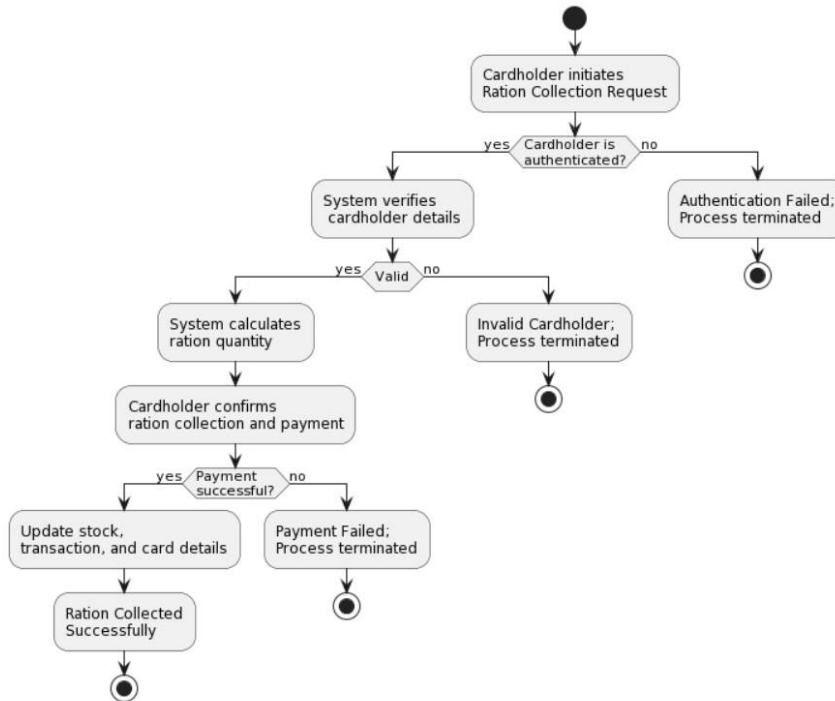


Figure 4: Activity diagram

### 4.2.5 CLASS DIAGRAM

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling, translating the models into programming code. Class diagrams can also be used for data modeling. Class diagrams are the blueprints of your system or subsystem. You can use class diagrams to model the objects that make up the system, to display the relationships between the objects, and to describe what those objects do and the services that they provide. Class diagrams are useful in many stages of system design. In the analysis stage, a class diagram can help you to understand the requirements of your problem domain and to identify its components. In an object-oriented software project, the class diagrams that you create during the early stages of the project contain classes that often translate into actual software classes and objects when you write code. Later, you can refine your earlier analysis and conceptual models into class diagrams that show the specific parts of your system, user interfaces, logical implementations, and so on. Your class diagrams then become a snapshot that describes exactly how your system works, the relationships between system components at many levels, and how you plan to implement those components.

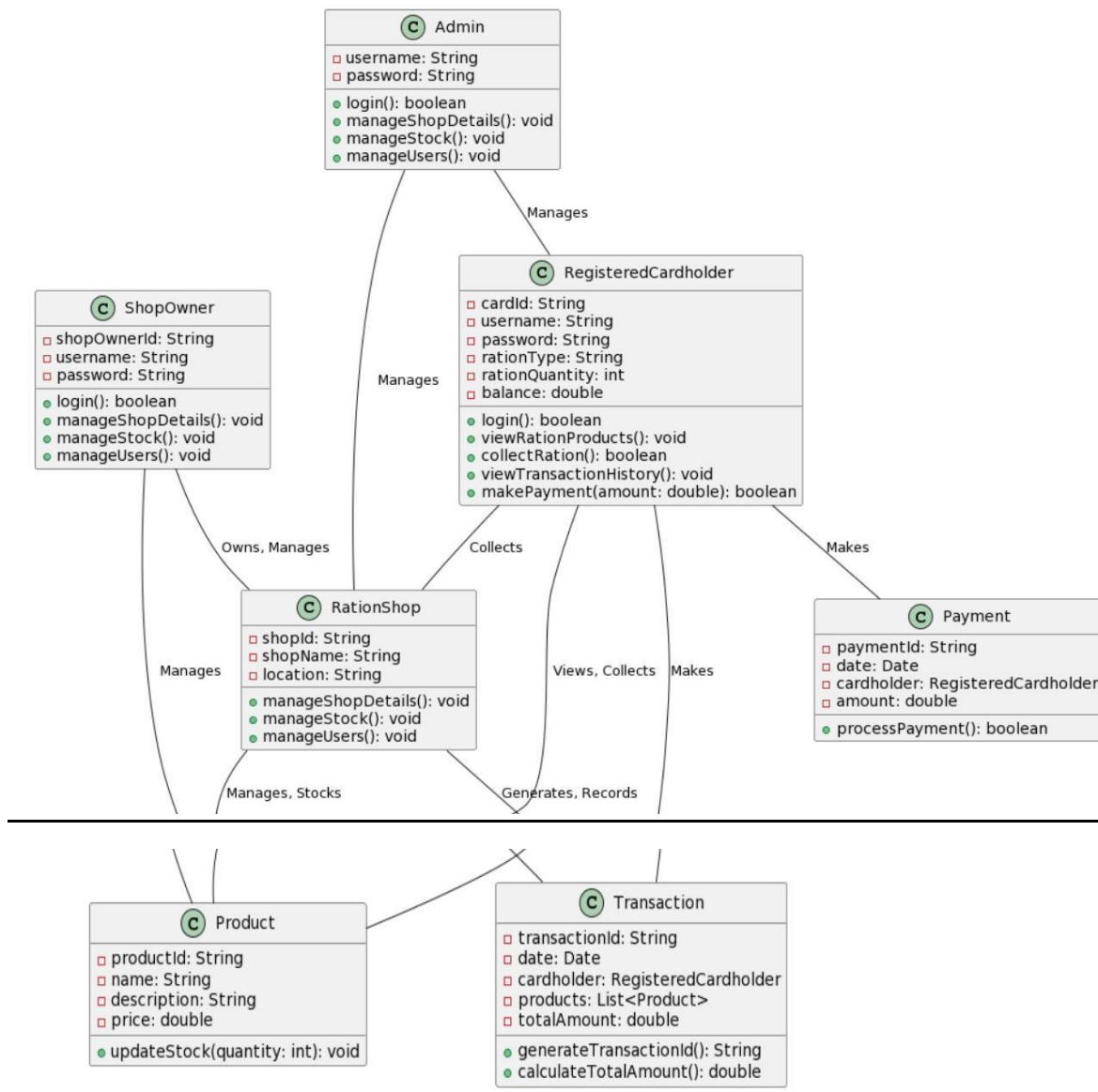


Figure 5: Class diagram

#### 4.2.6 OBJECT DIAGRAM

Since class diagrams are the source of object diagrams, class diagrams are a prerequisite for object diagrams. An instance of a class diagram is represented by an object diagram. Class and object diagrams both use the same fundamental ideas. The static view of a system is also represented by object diagrams, but this static view represents a momentary snapshot of the system. To represent a group of items and their connections as an instance, object diagrams are employed



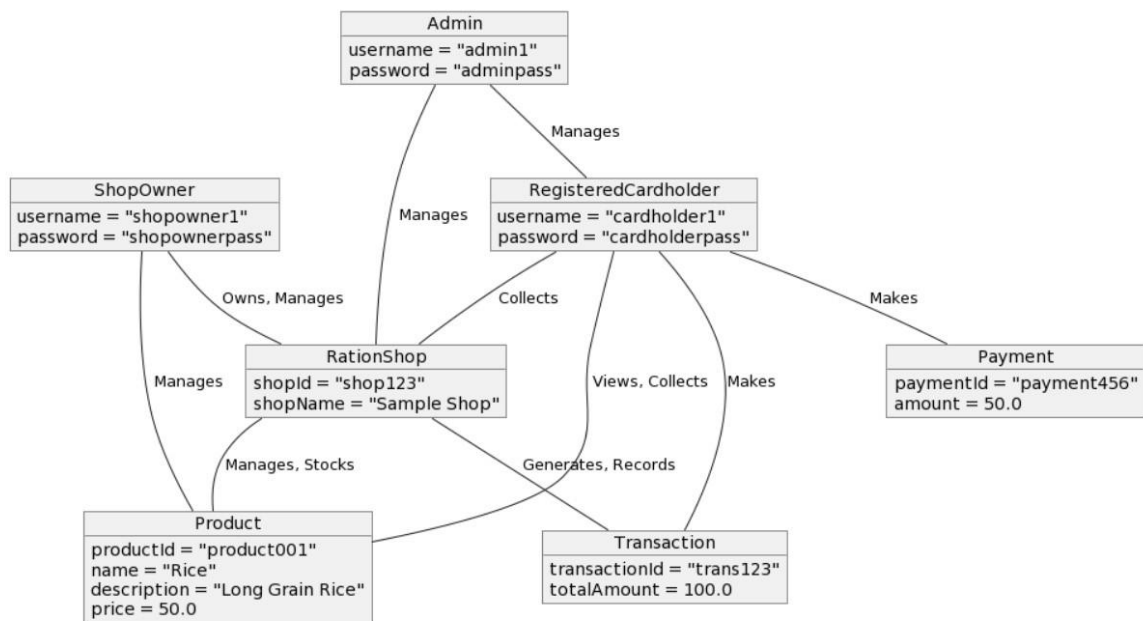


Figure 6: Object diagram

### 4.2.7 COMPONENT DIAGRAM

UML diagrams are used to represent and describe the behavior of object-oriented systems, assisting in visualization, explanation, and thorough documentation. Particularly in class diagrams, where they record the static behavior of a system, these diagrams play a vital role. On the other hand, graphics can impair the efficiency of real multitasking systems. To maintain coordination, each part within the broader process has a distinct duty and only communicates with other essential components.

A component diagram depicts how components are wired together to form larger components or software systems. They are used to illustrate the structure of arbitrarily complex systems.

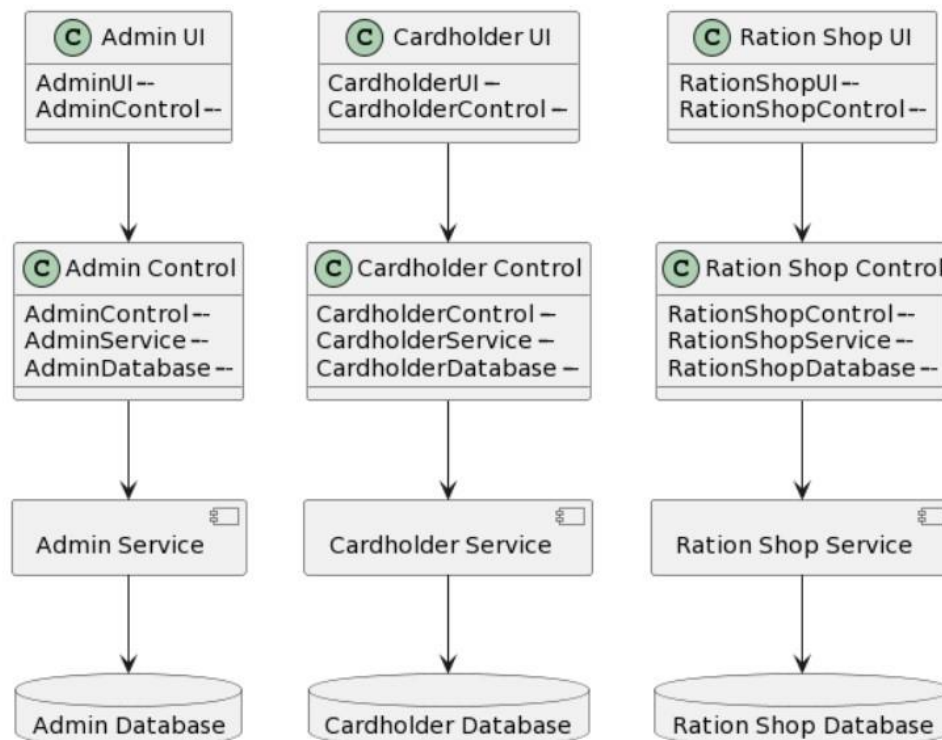


Figure 7: Component diagram

#### 4.2.8 DEPLOYMENT DIAGRAM

The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships. It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software will be executed as a node. Since it involves many nodes, the relationship is shown by utilizing communication paths. In contrast to other UML diagram types, which primarily depict the logical components of a system. The deployment diagram does not focus on the logical components of the system, but it put its attention on the hardware topology.

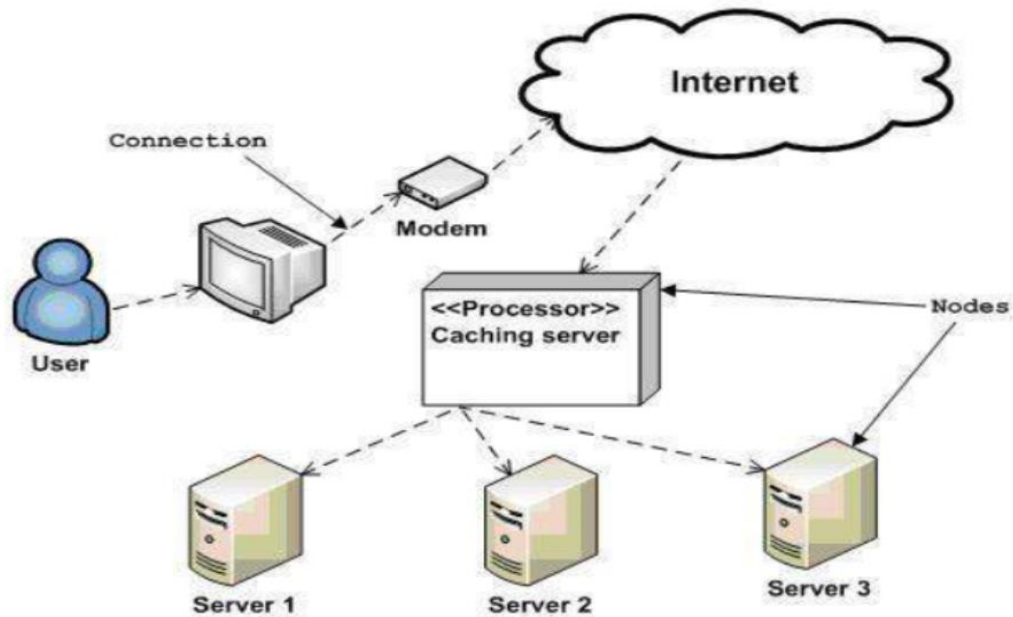
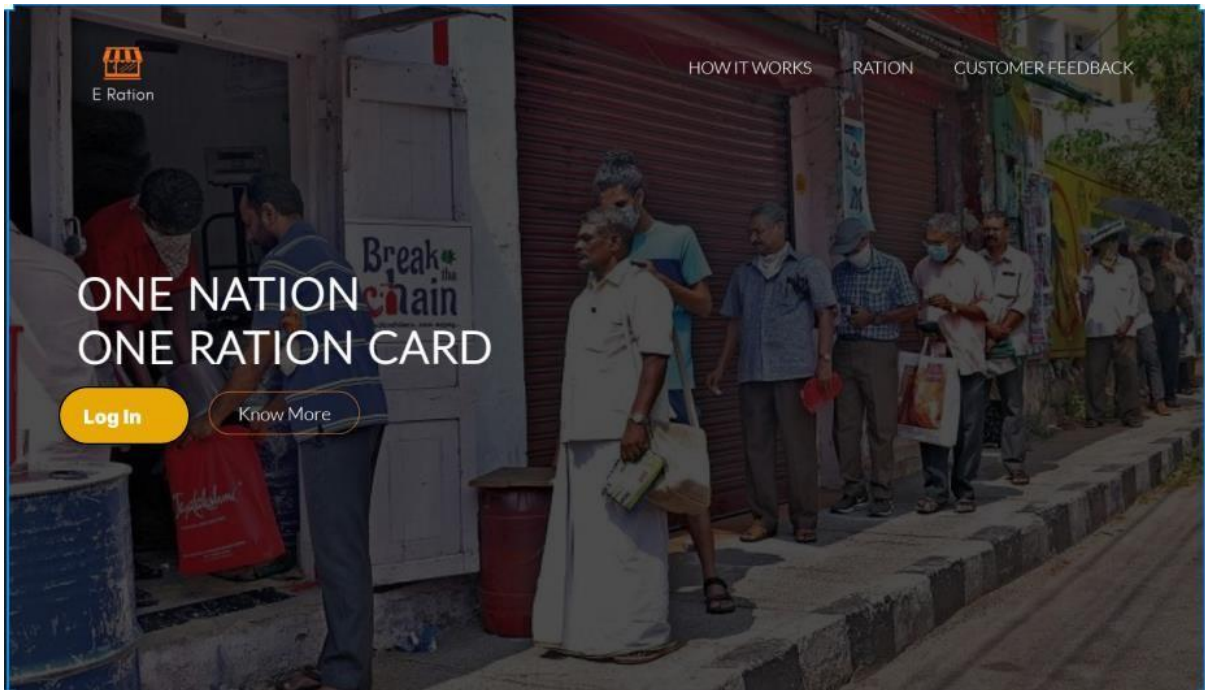


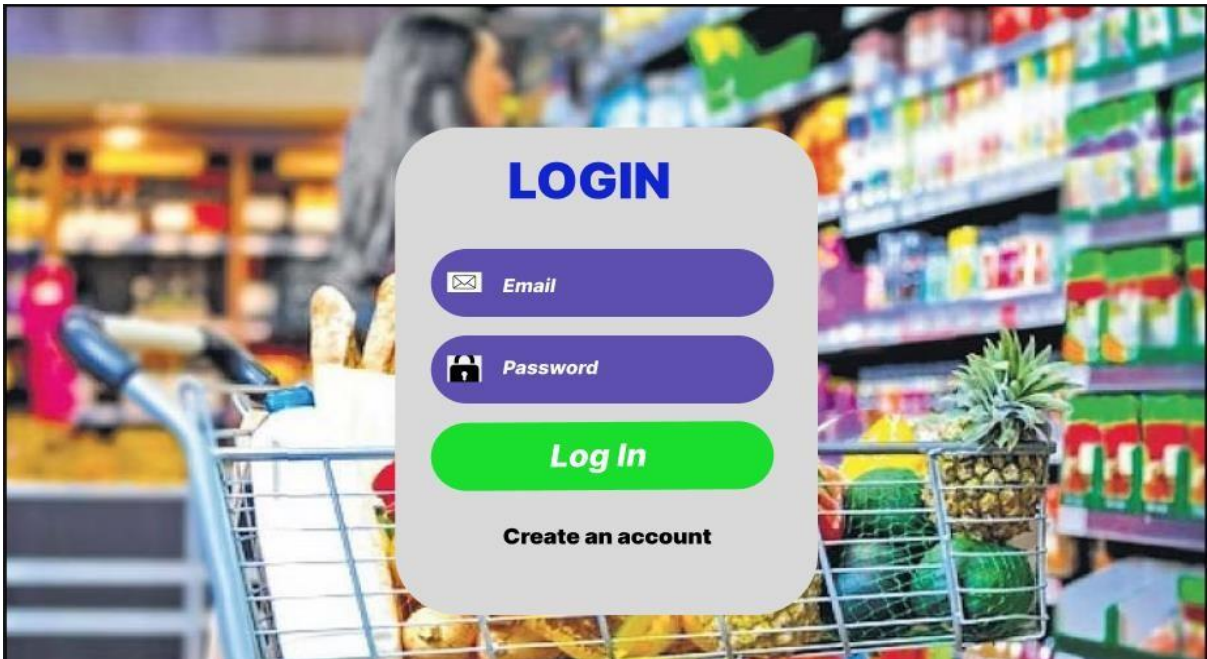
Figure 8: Deployment diagram

## 4.3 USER INTERFACE DESIGN USING FIGMA

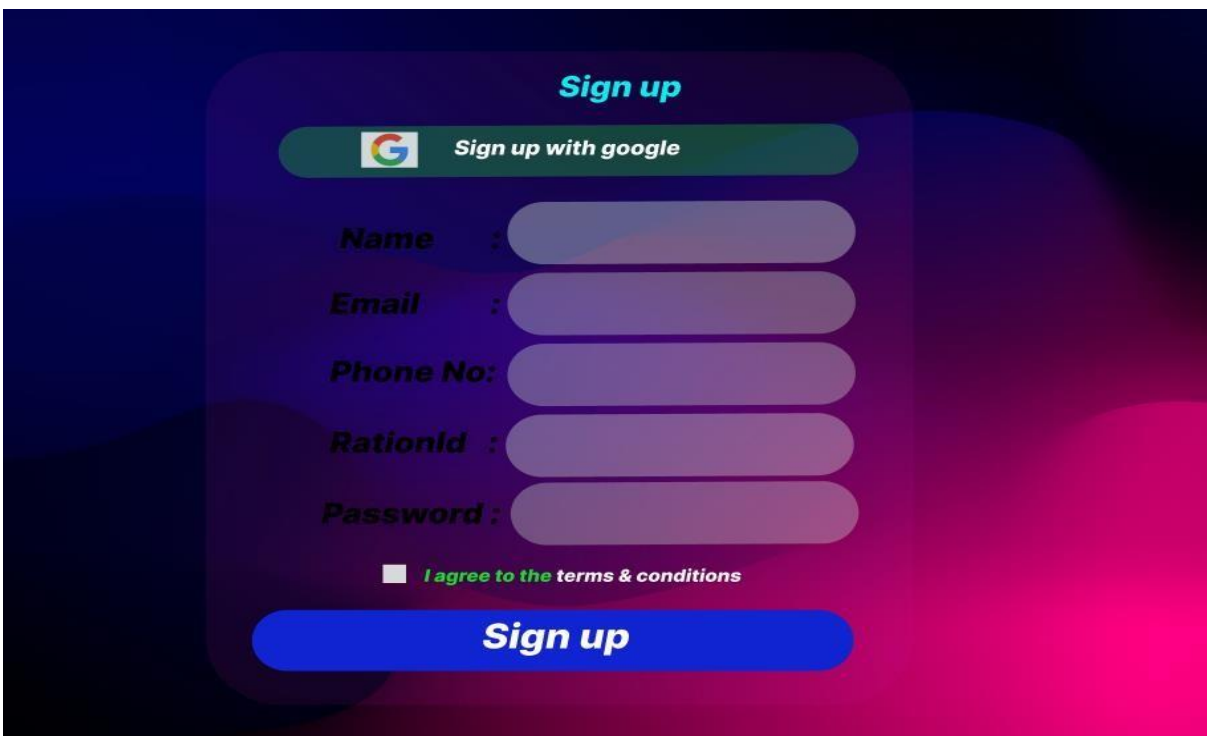
### Home Page



## Login Form



## Registration Form



## 4.3 DATABASE DESIGN

A database is a collection of data that has been organized to make it simple to manage and update. Information security could be one of the main aims of any database. The database design process is divided into two steps. The goal of the first step is to gather user requirements to create a database that as clearly as possible satisfies user needs. It is known as information-level design and is carried out without the aid of any DBMS. An information-level design is changed to a specific DBMS design that will be used to develop the system in the following stage. The physical-level design phase is where the characteristics of the specific DBMS are considered.

### 4.4.1 RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)

A Relational Database Management System (RDBMS) is a critical software application for organizing and managing data in a structured manner. It stores data in tables with rows and columns, where each row represents a record, and each column is a specific attribute or field. RDBMS systems like MySQL, Oracle, or Microsoft SQL Server ensure data integrity, enforce relationships between tables, and allow for efficient data retrieval and manipulation through Structured Query Language (SQL). These systems are widely used in various applications, such as e-commerce websites, financial systems, and inventory management, due to their robustness, scalability, and ability to handle complex data structures, making them essential for modern data-driven environments.

A Relational Database Management System (RDBMS) is a cornerstone of modern data management. It structures data into tables, where each row represents a unique entry, and each column corresponds to a specific attribute, ensuring a logical and organized storage system. RDBMS systems employ complex algorithms for data retrieval and manipulation, guaranteeing data consistency and adherence to predefined relationships between tables. Popular RDBMS software, including PostgreSQL, MySQL, and Microsoft SQL Server, provides robust features for transactions, data security, and scalability. These systems are integral to a myriad of applications, from healthcare records and customer databases to inventory control and financial platforms, supporting the efficient storage, retrieval, and analysis of vast datasets, making them essential tools in today's data-driven world.

### 4.4.2 NORMALIZATION

Normalization is a database design technique used in relational database management systems (RDBMS) to eliminate data redundancy and improve data integrity. It involves organizing data in a way that reduces data duplication and ensures that relationships between tables are defined and maintained.

The primary goals of normalization are:

**Minimizing Data Redundancy:** By breaking down data into separate tables and ensuring that each piece

of data is stored only once, normalization helps reduce the chances of data inconsistencies or errors.

**Ensuring Data Integrity:** Normalization enforces the rules of referential integrity, which means that relationships between tables are well-defined and maintained. This ensures that data remains accurate and consistent.

Normalization typically involves dividing a database into multiple related tables and using primary keys and foreign keys to establish relationships between these tables. There are several normal forms, from First Normal Form (1NF) to Fifth Normal Form (5NF), each with specific rules and requirements. The level of normalization achieved depends on the specific needs of the database and the trade-off between data redundancy and query performance.

By applying normalization principles, designers can create efficient and reliable database structures that support data integrity and ease data maintenance and manipulation.

There are several normal forms (NF) that define specific rules and requirements for achieving progressively higher levels of normalization. Here are the most common normal forms, from First Normal Form (1NF) to Fifth Normal Form (5NF):

**First Normal Form (1NF):**

- Each table has a primary key.
- All columns contain atomic (indivisible) values.
- There are no repeating groups or arrays in columns.

**Second Normal Form (2NF):**

- The table is in 1NF.
- All non-key attributes are fully functionally dependent on the entire primary key. In other words, all non-key attributes must be dependent on the entire primary key, not just part of it.

**Third Normal Form (3NF):**

- The table is in 2NF.
- There is no transitive dependency, meaning that non-key attributes are not dependent on other non-key attributes.

**Boyce-Codd Normal Form (BCNF):**

- A stricter version of 3NF.
- It enforces that every non-trivial functional dependency involves a super key.

**Fourth Normal Form (4NF):**

- Addresses multi-valued dependencies.
- Eliminates any multi-valued dependencies within the data.

**Fifth Normal Form (5NF):**

- Also known as Project-Join Normal Form (PJ/NF).
- Handles cases where data can be derived by joining multiple tables.

#### 4.4.3 SANITIZATION

In Python Django, sanitization refers to the process of cleaning and validating data to ensure that it is safe and free from malicious content before it is used or stored in a database. Sanitization is a crucial security practice to prevent various forms of attacks, such as cross-site scripting (XSS) and SQL injection, which can compromise the security and integrity of a web application.

#### 4.4.4 INDEXING

Indexing in Django is a fundamental database optimization technique. It involves creating data structures, or indexes, on specific fields to expedite data retrieval. These indexes act as signposts that enable the database to swiftly locate and fetch relevant data, especially in tables with substantial amounts of information. Django offers automatic index creation for primary keys, unique fields, and foreign keys. Additionally, developers can define custom indexes for fields frequently used in filtering, sorting, or searching. The choice of proper indexing plays a pivotal role in improving query performance, resulting in more responsive and scalable web applications. It's essential to consider application-specific query patterns and create indexes accordingly, as well as to understand the capabilities and limitations of the chosen database backend. Effective indexing is a cornerstone of efficient database operations in Django, contributing to enhanced application performance.

### 4.5 TABLE DESIGN

#### 1.Tbl\_customuser

Primary key: **user\_id**

No:	Field name	Datatype	Key Constraints	Description of the field
1	<b>User_id</b>	IntegerField(20)	PK	Users_id
2	username	Charfield(20)	Not Null	Users Username
3	Password	Charfield(20)	Not Null	Password of user
4	Email	EmailField(20)	Unique	Email_id of user
5	role	CharField(20)	Not Null	Role of the registered user

**2.Tbl \_userprofile(propertyowner)**Primary key: **UserProfile\_id**Foreign key: **user\_id** references table **Tbl\_customuser**

No:	Field name	Datatype	Key Constraints	Description of the field
1	<b>UserProfile_id</b>	IntegerField(20)	PK	Unique id for Userprofile
2	user_id	IntegerField(20)	FK	Unique id for User
3	fullname	Charfield(20)	Null	fullname of user
4	date_of_birth	date	Null	d.o.b of user
5	gender	Charfield (20)	Null	gender of user
6	phone	IntegerField(15)	Null	Phone number of user
7	Photo_id	file	Null	Photo of the user
8	address	CharField(20)	Null	Address of the user

**3.Tbl \_profile(tenant)**Primary key: **profile\_id**Foreign key: **user\_id** references table **Tbl\_customuser**

No:	Field name	Datatype	Key Constraints	Description of the field
1	<b>Profile_id</b>	IntegerField(20)	PK	Unique id for Userprofile
2	user_id	IntegerField(20)	FK	Unique id for User
3	fullname	Charfield(20)	Null	fullname of user
4	date_of_birth	date	Null	d.o.b of user
5	gender	Charfield (20)	Null	gender of user
6	phone	IntegerField(15)	Null	Phone number of user
7	Photo_id	file	Null	Photo of the user
8	address	CharField(20)	Null	Address of the user



**4. Tbl \_property**

Primary key: **property\_id**

Foreign key: **userprofile\_id** references table **Tbl \_userprofile**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	<b>Property_id</b>	IntegerField(10)	PK	Unique id for Property
2	Property_type	CharField	Null	Type of Property
3	address	Charfield(20)	Not Null	Property address
4	Monthly_rent	Decimalfield(20)	Not Null	Monthly rent of property
5	Security_deposit	Decimalfield (20)	Not Null	Security deposit of property
6	Lease_duration	Charfield (2000)	Not Null	Property lease duration
7	Availability_date	DateField	Not Null	Date the property is available
8	Approval_status	CharField(20)	Not Null	Admin Property Approval
9	Userprofile_id	IntegerField(10)	FK	Property owner id

**5. Tbl \_Propertyimage**

Primary key: **pimage\_id**

Foreign key: **property\_id** references table **Tbl \_property**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	<b>pimage_id</b>	IntegerField(10)	PK	Unique id for images
2	Category	CharField(200)	Null	Property image category
3	image	ImageField()	Not Null	Category wise image
4	Property_id	IntegerField(10)	FK	Id of the property

**6. Tbl \_Amenity**

Primary key: **Amenity\_id**

Foreign key: **property\_id** references table **Tbl \_property**

No.	Field name	Datatype(Size)	Key Constraints	Description of the field
1	<b>Amenity_id</b>	IntegerField(10)	PK	Property amenityid
2	Sqfootage	Positiveintegerfield(10)	Not null	Property squarefoot
3	Bedrooms	Positiveintegerfield(10)	Not null	Noofbedrooms
4	Bathrooms	Positiveintegerfield(10)	Not null	Noofbathrooms
5	Stories	Positiveintegerfield(10)	Not null	Property storiesno
6	Acresofland	Decimalfield(10)	Not null	Acresof property type land
7	Rentspace	CharField(200)	Not null	Full or partial
8	noofrooms	Positiveintegerfield(10)	Not null	Number of rooms
9	parkingspace	TextField(10)	Not null	Yes or no for commercial property
10	purpose	TextField(10)	Not null	land property type purpose
11	acresorcent	Decimalfield(10)	Not null	Acres/cent the property in
12	Property_id	IntegerField(10)	FK	Property id

### 7.Tbl \_propertydocument

Primary key: **doc\_id**

Foreign key: **property\_id** references table **Tbl \_property**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	<b>doc_id</b>	IntegerField(10)	PK	Unique id for property document
2	Document_type	CharField(10)	NotNull	Property document type
3	document	FileField(10)	NotNull	Property document
4	Property_id	IntegerField(10)	NotNull	Property id

**8.Tbl \_rentalrequest**

Primary key: **request\_id**

Foreign key: **property\_id** references table **Tbl \_property**

Foreign key: **profile\_id** references table **Tbl \_profile**

No:	Field name	Datatype	Key Constraints	Description of the field
1	<b>request_id</b>	IntegerField(20)	PK	Unique id for each rental request
2	Property_id	IntegerField(20)	FK	Property id
3	Profile_id	IntegerField(20)	FK	Tenant who had made rental request
4	Status	CharField(20)	NotNull	Pending/accepted/rejected
5	Created_at	DateTimeField	NotNull	Time rental request has created

**9.Tbl \_leaseagreement**

Primary key: **agreement\_id**

Foreign key: **property\_id** references table **Tbl \_property**

No:	Field name	Datatype	Key Constraints	Description of the field
1	<b>agreement_id</b>	IntegerField(20)	PK	Unique id for each agreement
2	Property_id	IntegerField(20)	FK	Property id
3	Start_date	DateField	NotNull	rentalstartdate
4	End_date	DateField	NotNull	Rentalenddate
5	Rent_amount	DecimalField(20)	NotNull	Total rentamount each month
6	Security_amount	DecimalField	Not Null	Initial security amount

## **CHAPTER 5**

### **SYSTEM TESTING**

## 5.1 INTRODUCTION

Software testing is a way to check if the computer program works like it's supposed to. We use testing to make sure the software does what it is supposed to do. Validation means checking or testing things like software to make sure they meet the requirements and standards they are supposed to follow. Software testing is a way to check if a program works well. It goes along with other methods like checking and walking through the program. Validation means making sure that what the user wanted is what they got. There are several rules that can serve as testing objectives.

They are:

Testing is a process of executing a program with the intent of finding an error.

- A good test case is one that has high possibility of finding an undiscovered error.
- A successful test is one that uncovers an undiscovered error.

If a test works well and follows its goals, it can find mistakes in the software. The test showed that the computer program is working like it's supposed to and is doing well.

There are three ways to test program.

- For correctness
- For implementation efficiency
- For computational complexity

## 5.2 TEST PLAN

A test plan suggests several required steps that need be taken to complete various testing methodologies. The activity that is to be taken is outlined in the test plan. A computer program, its documentation, and associated data structures are all created by software developers. It is always the responsibility of the software developers to test each of the program's separate components to make sure it fulfills the purpose for which it was intended. To solve the inherent issues with allowing the builder evaluate what they have developed, there is an independent test group (ITG). Testing's precise goals should be laid forth in quantifiable language. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test.

The levels of testing include:

- Unit testing
- Integration Testing
- Data validation Testing & Output Testing

### **5.2.1 UNIT TESTING**

Unit testing concentrates verification efforts on the software component or module, which is the smallest unit of software design. The component level design description is used as a guide when testing crucial control paths to find faults inside the module's perimeter. The level of test complexity and the untested area determined for unit testing. Unit testing is white-box focused, and numerous components may be tested simultaneously. To guarantee that data enters and exits the software unit under test properly, the modular interface is tested. To make sure that data temporarily stored retains its integrity during each step of an algorithm's execution, the local data structure is inspected. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested.

Before starting any other test, tests of data flow over a module interface are necessary. All other tests are irrelevant if data cannot enter and depart the system properly. An important duty during the unit test is the selective examination of execution pathways. Error circumstances must be foreseen in good design, and error handling paths must be put up to cleanly reroute or halt work when an error does arise. The final step of unit testing is boundary testing. Software frequently fails at its limits.

In the Sell-Soft System, unit testing was carried out by treating each module as a distinct entity and subjecting them to a variety of test inputs. The internal logic of the modules had some issues, which were fixed. Each module is tested and run separately after coding. All unused code was eliminated, and it was confirmed that every module was functional and produced the desired outcome

### **5.2.2 INTEGRATION TESTING**

Integration testing is a methodical approach for creating the program's structure while also carrying out tests to find interface issues. The goal is to construct a program structure that has been determined by design using unit tested components. The program is tested. Correction is challenging since the size of the overall program makes it challenging to isolate the causes. As soon as these mistakes are fixed, new ones arise, and the process repeats itself in an apparently unending cycle. All the modules were integrated after unit testing was completed in the system to check for an interface inconsistency. A distinctive program structure also developed when discrepancies in program structures.

### **5.2.3 VALIDATION TESTING OR SYSTEM TESTING**

The testing process comes to an end here. This involved testing the entire system in its entirety, including all forms, code, modules, and class modules. Popular names for this type of testing include system tests and black box testing. The functional requirements of the software are the main emphasis of the black box testing approach. That example, using Black Box testing, a software engineer can create sets of input conditions that will fully test every program requirement. The following sorts of problems are targeted by black box testing: erroneous or missing functions, interface errors, data structure or external data access errors, performance errors, initialization errors, and termination errors.

### **5.2.4 OUTPUT TESTING OR USER ACCEPTANCE TESTING**

The system considered is tested for user acceptance; here it should satisfy the firm's need. The software should keep in touch with perspective system; user at the time of developing and making changes whenever required.

This done with respect to the following points:

- Input Screen Designs,
- Output Screen Designs,

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future.

### **5.2.5 AUTOMATION TESTING**

Automation Testing is a software testing technique that performs using special automated testing software tools to execute a test case suite. Essentially, it's a test to double-check that the equipment or software does exactly what it was designed to do. It tests for bugs, defects, and any other issues that can arise with product development. Although some types of testing, such as regression or functional testing can be done manually, there are greater benefits of doing it automatically. Automation testing can be run at any time of the day. It uses scripted sequences to examine the software. It then reports on what has been found, and this information can be compared with earlier test runs. Automation developers generally write in the following programming languages: C#, JavaScript, and Ruby.

### 5.2.6 SELENIUM TESTING

Selenium is an open-source automated testing framework used to verify web applications across different browsers and platforms. Selenium allows for the creation of test scripts in various programming languages such as Java, C#, and Python. Jason Huggins, an engineer at Thought Works, developed Selenium in 2004 while working on a web application that required frequent testing. He created a JavaScript program called "JavaScriptTestRunner" to automate browser actions and improve testing efficiency. Selenium has since evolved and continues to be developed by a team of contributors. In addition to Selenium, another popular tool used for automated testing is Cucumber. Cucumber is an open-source software testing framework that supports behavior-driven development (BDD). It allows for the creation of executable specifications in a human-readable format called Gherkin. One of the advantages of using Cucumber is its ability to bridge the gap between business stakeholders and technical teams. By using a common language, Cucumber facilitates effective communication and collaboration during the testing process. It promotes a shared understanding of the requirements and helps ensure that the developed software meets the intended business goals. Cucumber can be integrated with Selenium to combine the benefits of both tools. Selenium is used for interacting with web browsers and automating browser actions, while Cucumber provides a structured framework for organizing and executing tests. This combination allows for the creation of end-to-end tests that verify the behavior of web applications across different browsers and platforms, using a business-readable and maintainable format.

#### Test Case 1

##### Code

Feature: Test login Functionality

Scenario: Check login is successful with valid credentials

- Given browser is open
- And user is on login page
- When user enters username and password
- And user clicks on login
- Then user is navigated to the owner page

```
package stepdefinition;  
import io.cucumber.java.en.Given;  
import io.cucumber.java.en.And;  
import io.cucumber.java.en.When;  
import io.cucumber.java.en.Then;
```



```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import static org.junit.Assert.assertEquals;

public class loginsteps {
    private WebDriver driver;
    boolean testPassed = true; // Initialize a flag to check test status

    @Given("browser is open")
    public void browserIsOpen() {
        // Set up WebDriver
        System.setProperty("webdriver.chrome.driver", "C:\\chromedriver-
win64\\chromedriver.exe");
        driver = new ChromeDriver();
    }

    @And("user is on login page")
    public void userIsOnLoginPage() {
        driver.get("http://127.0.0.1:8000/login/");
    }

    @When("user enters username and password")
    public void userEntersUsernameAndPassword() {
        WebElement usernameInput = driver.findElement(By.id("username"));
        WebElement passwordInput = driver.findElement(By.id("password"));

        usernameInput.sendKeys("rijulrojo");
        passwordInput.sendKeys("rR@12345");
    }

    @And("user clicks on login")
    public void userClicksOnLogin() {
        WebElement loginButton = driver.findElement(By.id("login")); // Replace "login-button"
        loginButton.click();
    }
}
```

```

@Then("user is navigated to the owner page")
public void navigateToHomePage() {
    try {
        // Add a delay to ensure the page loads completely
        // Thread.sleep(2000);

        // Check if the test is passed
        if (driver.getCurrentUrl().contains("http://127.0.0.1:8000/ownerpage")) {
            System.out.println("Test Passed: User is on the home page");
        } else {
            System.out.println("Test Failed: User is not on the home page");
            testPassed = false;
        }

        // Assert the test status using JUnit's Assert class
        assertEquals(true, testPassed);
    } finally {
        // Close the driver after all assertions are made
        if (driver != null) {
            driver.quit();
        }
    }
}

```

## Screenshot

```

Scenario: Check login is successful with valid credentials # src/test/resources/Features/login.feature:3
  Given browser is open # stepdefinition.loginsteps.browserIsOpen()
  And user is on login page # stepdefinition.loginsteps.userIsOnLoginPage()
  When user enters username and password # stepdefinition.loginsteps.userEntersUsernameAndPassword()
  And user clicks on login # stepdefinition.loginsteps.userClicksOnLogin()
Test Passed: User is on the home page
  Then user is navigated to the owner page # stepdefinition.loginsteps.navigateToHomePage()

1 Scenarios (1 passed)
5 Steps (5 passed)
0m3.693s

```

## Test Report

Test Case 1					
Project Name: Property Rentals					
Login Test Case					
Test Case ID: Test_1			Test Designed By:Nithin Rajappan		
Test Priority (Low/Medium/High): High			Test Designed Date: 04-12-2023		
Module Name: Login Screen			Test Executed By : Ms.Sruthimol Kurian		
Test Title : User Login			Test Execution Date: 04-12-2023		
Description: Verify login with valid username and password					
Pre-Condition :					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation to Login Page		Login Page should be displayed	Login Page be displayed	Pass
2	Provide valid Username	Email-id: shyam@gmail.com	User should be able to login	User is logged in and navigated to corresponding Owner Page	Pass
3	Provide Valid Password	Password: shyam			
4	Click on Login button				
Post-Condition: User is validated with database and successfully login into account. The Account session details are logged in database					

## Test Case 2:

### Code

import time

```
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By

class LoginTest(unittest.TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()

    def test_successful_login(self):
        self.driver.get('http://127.0.0.1:8000/login')
        username_input = self.driver.find_element(By.ID, 'username')
        password_input = self.driver.find_element(By.ID, 'password')

        # Replace 'valid_username' and 'valid_password' with actual credentials
        username_input.send_keys('rijulrojio')
        password_input.send_keys('rR@12345')

        login_button = self.driver.find_element(By.ID, 'login')
        login_button.click()

        time.sleep(2) # Adjust the time as needed
        self.assertIn('http://127.0.0.1:8000/ownerpage', self.driver.current_url.lower())

        # Click on the "Dashboard" link
        dashboard_link = self.driver.find_element(By.LINK_TEXT, 'Dashboard')
        dashboard_link.click()

        # Check if redirected to the owner's dashboard
        self.assertIn('http://127.0.0.1:8000/ownerpg', self.driver.current_url.lower())

        # Click on the "Add Properties" link
        add_properties_link = self.driver.find_element(By.LINK_TEXT, 'Add
Properties')
        add_properties_link.click()
```

```
property_type_dropdown = self.driver.find_element(By.ID, 'property_type')
address_input = self.driver.find_element(By.ID, 'address')
monthly_rent_input = self.driver.find_element(By.ID, 'monthly_rent')
security_deposit_input = self.driver.find_element(By.ID, 'security_deposit')
lease_duration_dropdown = self.driver.find_element(By.ID, 'lease_duration')
availability_date_input = self.driver.find_element(By.ID, 'availability_date')
property_type_dropdown.send_keys('Apartment')
address_input.send_keys('123 Main St')
monthly_rent_input.send_keys('1500')
security_deposit_input.send_keys('2000')
lease_duration_dropdown.send_keys('1 year')
availability_date_input.send_keys('01-02-2022')

# Submit the form
submit_button = self.driver.find_element(By.CSS_SELECTOR, '#add-
property-form button[type="submit"]')
submit_button.click()

time.sleep(2) # Adjust the time as needed

# Now you can add assertions to check if the property was added successfully
# For example, check for success messages or check if the property appears in
the list

def tearDown(self):
    self.driver.quit()

if __name__ == "__main__":
    unittest.main()
```

## Screenshot

```
PS D:\Mini Project\Mini_Project\Mini Project Code\property_rental_platform> python manage.py test
Found 1 test(s).
System check identified no issues (0 silenced).

DevTools listening on ws://127.0.0.1:13814/devtools/browser/07510994-f948-48d6-ab72-c4c5189366de
[2056:16640:1205/091552.887:ERROR:device_event_log_impl.cc(192)] [09:15:52.892] USB: usb_service_win.cc:415 Could not read
device interface GUIDs: The system cannot find the file specified. (0x2)
.
-----
Ran 1 test in 14.564s

OK
```

## Test report

Test Case 2					
Project Name: Property Rentals					
Test Case ID: Test_2			Test Designed By: :Nithin Rajappan		
Test Priority (Low/Medium/High): High			Test Designed Date: 04-12-2023		
Module Name: AddItem			Test Executed By : Ms.Sruthimol Kurian		
Test Title:Adding Properties			Test Execution Date: 04-12-2023		
Description: admin adding new Product					
Pre-Condition : User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation to Login Page		Login Page should be displayed	Login Page be displayed	Pass
2	Provide valid Username	Email-id: shyam@gmail.com	User should be able to login	User is logged in and navigated to corresponding Owner Page	Pass
3	Provide Valid Password	Password: shyam			
4	Click on Login button				
5	Click on Dashboard button			Admin dashboard is displayed	Pass
6	Click on Add Item	Fill the Item details		Item is added succesfully	Pass

<b>Post-Condition:</b> Item is Added Successfully
---

**Test Case 3:****Code**

```
import time
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By

class LoginTest(unittest.TestCase):

    def setUp(self):
        self.driver = webdriver.Chrome()

    def test_successful_login(self):
        self.driver.get('http://127.0.0.1:8000/login')
        username_input = self.driver.find_element(By.ID, 'username')
        password_input = self.driver.find_element(By.ID, 'password')

        # Replace 'valid_username' and 'valid_password' with actual credentials
        username_input.send_keys('jubil')
        password_input.send_keys('jJ@12345')

        login_button = self.driver.find_element(By.ID, 'login')
        login_button.click()

        time.sleep(2) # Adjust the time as needed

        # Check if redirected to the tenantpage
        self.assertIn('http://127.0.0.1:8000/tenantpage', self.driver.current_url.lower())

        # Click on the "Dashboard" link
        dashboard_link = self.driver.find_element(By.LINK_TEXT, 'Dashboard')
```

```
dashboard_link.click()

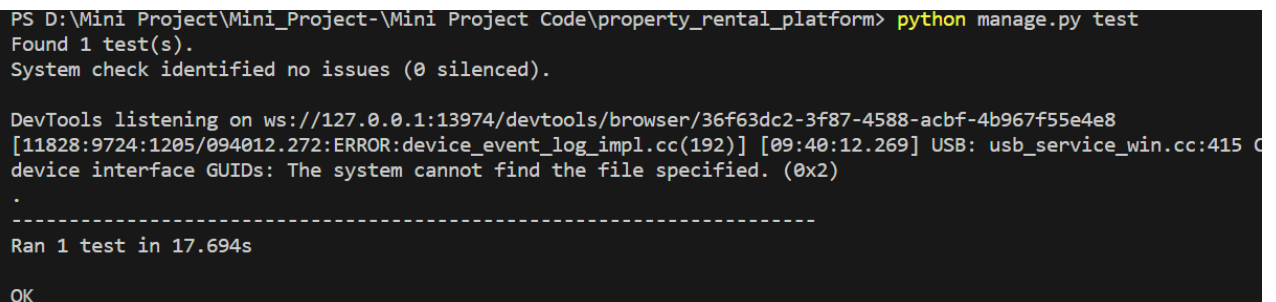
time.sleep(2) # Adjust the time as needed
self.assertIn('http://127.0.0.1:8000/tenantpg', self.driver.current_url.lower())
property_type_dropdown = self.driver.find_element(By.ID, 'property_type')
min_rent_input = self.driver.find_element(By.NAME, 'min_rent')
max_rent_input = self.driver.find_element(By.NAME, 'max_rent')
search_button = self.driver.find_element(By.CSS_SELECTOR, '#property-
search-form button[type="submit"]')

# Replace these values with the desired search criteria
property_type_dropdown.send_keys('Apartment')
min_rent_input.send_keys('1000')
max_rent_input.send_keys('2000')
search_button.click()

time.sleep(2) # Adjust the time as needed
def tearDown(self):
    self.driver.quit()

if __name__ == "__main__":
    unittest.main()
```

## Screenshot



```
PS D:\Mini Project\Mini_Project-Code\property_rental_platform> python manage.py test
Found 1 test(s).
System check identified no issues (0 silenced).

DevTools listening on ws://127.0.0.1:13974/devtools/browser/36f63dc2-3f87-4588-acbf-4b967f55e4e8
[11828:9724:1205/094012.272:ERROR:device_event_log_impl.cc(192)] [09:40:12.269] USB: usb_service_win.cc:415 C
device interface GUIDs: The system cannot find the file specified. (0x2)
.
-----
Ran 1 test in 17.694s

OK
```



## **CHAPTER 6**

### **IMPLEMENTATION**

## 6.1 INTRODUCTION

Implementation is the stage of the project where the theoretical design is turned into a working system. It can be the most crucial stage in achieving a successful new system gaining the users confidence that the new system will work and will be effective and accurate. It is primarily concerned with user training and documentation. Conversion usually takes place about the same time the user is being trained or later. Implementation simply means convening a new system design into operation, which is the process of converting a new revised system design into an operational one.

At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned or controlled, it can create chaos and confusion. Implementation includes all those activities that take place to convert from the existing system to the new system. The new system may be a totally new, replacing an existing manual or automated system or it may be a modification to an existing system. Proper implementation is essential to provide a reliable system to meet organization requirements. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after thorough testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required to implement the three main aspects: education and training, system testing and changeover. The implementation state involves the following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover.

## 6.2 IMPLEMENTATION PROCEDURES

The implementation procedures for this project involve a systematic and phased approach to bring the online crime reporting portal to fruition. This complex system, designed to cater to the needs of various stakeholders, requires careful planning and execution.

To commence the implementation, the project team will initiate a detailed requirements analysis phase. This stage involves comprehensive discussions with law enforcement officials, control room staff, prison wardens, and potential end-users to determine their specific needs and expectations.

The results of this analysis will inform the development of a detailed system specification and design.

The development phase will follow, involving the creation of the portal's architecture, databases, and user interfaces. It's during this stage that the functionalities outlined in the project's scope, including user management, crime reporting, and communication features, will be integrated into the system. The portal will be designed with a user-friendly interface to ensure ease of use for all categories of users.

Simultaneously, a dedicated team will work on the incorporation of advanced technologies, such as machine learning capabilities for crime trend analysis and predictive features. These technologies will be implemented carefully to ensure the portal's robustness and security.

Once the development phase is complete, rigorous testing will be conducted to identify and rectify any bugs or issues. The portal will undergo extensive quality assurance and user acceptance testing to ensure that it meets all the necessary requirements and is free of vulnerabilities.

Deployment of the system will be carried out in a controlled manner, ensuring minimal disruption to the existing processes, and allowing for gradual adaptation by the involved stakeholders. Comprehensive training sessions will be conducted to familiarize law enforcement personnel, control room staff, prison wardens, and registered users with the portal's functionality and features. Post-deployment, the project team will continue to provide support and maintenance, addressing any issues that may arise and making necessary improvements as the system matures. Regular updates and security measures will be implemented to safeguard user data and maintain the portal's efficiency.

### **6.2.1 USER TRAINING**

User training is designed to prepare the user for testing and converting the system. To achieve the objective and benefits expected from computer-based system, it is essential for the people who will be involved to be confident of their role in the new system. As system becomes more complex, the need for training is more important. By user training the user comes to know how to enter data, respond to error messages, interrogate the database, and call up routine that will produce reports and perform other necessary functions

### **6.2.2 TRAINING ON THE APPLICATION SOFTWARE**

After providing the necessary basic training on computer awareness the user will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the date

entered. It should then cover information needed by the specific user/ group to use the system or part of the system while imparting the training of the program on the application. This training may be different across different user groups and across different levels of hierarchy.

### **6.2.3 SYSTEM MAINTENANCE**

Maintenance is the enigma of system development. The maintenance phase of the software cycle is the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is for it to make adaptable to the changes in the system environment. Software maintenance is of course, far more than “Finding Mistakes”.

## **CHAPTER 7**

### **CONCLUSION AND FUTURE SCOPE**

## **7.1 CONCLUSION**

The Property Renting Platform endeavors to redefine how property owners and tenants interact, creating a digital haven for property rentals. Through features such as advanced analytics for Admins, efficient property listing management for Owners, and detailed property search for Tenants, the platform seeks to address the evolving challenges in the fast-paced urban environment. By prioritizing user satisfaction, inclusivity, and the utilization of advanced technologies, the Property Renting Platform aims to establish itself as a cornerstone in the landscape of online property rental platforms. The proposed system's strengths lie in its user-friendly interfaces, advanced functionalities, and a commitment to providing a comprehensive solution for the diverse needs of property owners and tenants alike.

## **7.2 FUTURE SCOPE**

For the Property Rentals platform, the future scope is equally promising, marked by avenues for innovation and growth. By integrating advanced technologies such as machine learning and data analytics, the platform can refine its property recommendation system, offering personalized suggestions aligned with user preferences and behaviors. Expanding community engagement features to include forums, virtual property tours, and user-generated content can foster a sense of collaboration among property owners and tenants. Global expansion is a viable prospect, with the platform accommodating multilingual support to cater to diverse markets and communities. Strategic partnerships with real estate agencies, property developers, and local authorities can enhance the platform's reach and credibility. Additionally, exploring the potential for integrating augmented reality (AR) or virtual reality (VR) for immersive property viewing experiences can further set the platform apart in the competitive landscape of property rentals. Overall, the future of the Property Rentals platform holds exciting possibilities for technological advancements, community building, and extended market reach.

## **CHAPTER 8**

## **BIBLIOGRAPHY**

**REFERENCES:**

- Harvard Business Review - Unlocking Value through Questions:
- AlsoAsked - People Also Ask keyword research tool: A tool to find questions people also ask related to a specific topic.
- Google Support - How to search on Google: Tips on effective Google searches, including finding quick answers.
- Qualtrics - Multiple Choice Question: Information about the multiple-choice question type for surveys.
- MyGreatLearning - 180+ SQL Interview Questions and Answers in 2023: A compilation of SQL interview questions for job seekers

**WEBSITES:**

- <https://www.grafiati.com/>
- <https://annas-archive.org/>



## **CHAPTER 9**

### **APPENDIX**

## 9.1 Sample Code

index.html

```
{% load static %}
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Property Rentals - Find Your Dream Home</title>
    <style>
        /* Reset some default styles */
        body, h1, h2, p, ul, li {
            margin: 0;
            padding: 0;
        }

        /* CSS for Header */
        body {
            font-family: Arial, sans-serif;
            background-size: cover;
            background-repeat: no-repeat;
            background-attachment: fixed;
            background-color: #f0f0f0; /* Light gray background */
        }

        header {
            background-color: #64a4a4; /* Dark blue header background */
            color: #ffffff; /* White text color */
            text-align: center;
            padding: 20px;
        }

        header h1 {
            font-size: 37px; /* Larger font size */
```

```
        margin-bottom: 10px;
        text-align: left;
        font-family: Arial, sans-serif;
    }

    a {
        text-decoration: none;
        color: #dbdada; /* Red link color */
    }

    header nav ul {
        list-style-type: none;
        padding: 0;
    }

    header nav li {
        display: inline;
        margin-right: 20px; /* Increased margin between navigation items */
    }

    header nav a {
        text-decoration: none;
        color: #ffffff;
        font-weight: bold;
        background-color: #81465b; /* Blue button background color */
        padding: 10px 20px;
        border-radius: 5px;
        transition: background-color 0.3s;
    }

    header nav a:hover {
        background-color: #fbc58f; /* Orange color on hover */
    }
```

```
/* CSS for Main Content */
main {
    padding: 20px;
    background-color: #f0f0f0;
    border-radius: 10px;
    margin: 20px;
    background-image: url('{% static "images/prop2.jpg" %}');
    background-size: cover;
    background-position: center;
    color: #333; /* Dark text color */
}

/* CSS for Search Form */
.search-form {
    padding: 20px;
    background-color: rgba(255, 255, 255, 0.8); /* Semi-transparent white
background */
    border-radius: 10px;
    margin: 20px 0;
}

.search-form h3 {
    font-size: 24px; /* Larger font size for the form title */
    color: #333;
    margin-bottom: 20px;
}

.search-form label {
    display: block;
    margin-bottom: 10px;
    font-weight: bold;
}

.search-form input[type="text"],
```

```
.search-form select,
.search-form button {
    width: 60%;
    padding: 10px;
    margin-bottom: 15px; /* Increased margin */
    border: 1px solid #ccc;
    border-radius: 5px;
}

.search-form button {
    background-color: #f2a298; /* Red search button background color */
    color: #fff; /* White text color */
    font-weight: bold;
    transition: background-color 0.3s;
}

.search-form button:hover {
    background-color: #e0d4d3; /* Darker red on hover */
}

/* CSS for Property Listings */
.property-listings {
    padding: 20px;
    background-color: rgba(255, 255, 255, 0.8);
    border-radius: 10px;
    margin: 20px 0;
    display: flex;
    flex-wrap: wrap;
    gap: 20px;
}

.property {
    flex: 1 1 calc(33.33% - 20px);
    border: 1px solid #ccc;
```

```
padding: 10px;
border-radius: 5px;
transition: transform 0.3s;
background-color: #fff;
}

.property:hover {
    transform: scale(1.05);
    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.2); /* Add a subtle shadow on hover */
}

.property img {
    max-width: 80%;
    height: auto;
}

/* CSS for Site Description Section */
.site-description {
    padding: 20px;
    background-color: #fff;
    border-radius: 10px;
    margin: 20px 0;
    text-align: center;
    animation: fadeInUp 1s ease;
}

/* Footer Styles */
footer {
    background-color: #333; /* Dark background color */
    color: #fff;
    text-align: center;
    padding: 10px;
    position: bottom;
```

```
        bottom: 0;
        width: 100%;
    }

</style>
<link rel="shortcut icon" href="{% static 'images/Key.png' %}" type="">

</head>
<body >

    <!-- Updated Header -->

<header>
    <div class="logo">
        <h1 img src="{% static 'images/logo.png' %}"> PROPERTY RENTALS </h1>

    </div>
    <nav>
        <ul>
            <li><a href="{% url 'signup' %}">Sign up</a></li>
            <li><a href="{% url 'login' %}">Log in</a></li>
        </ul>
    </nav>
</header>

<!-- Updated Main Content -->
<main>
    <h2>Welcome to Property Rentals</h2>
    <p>Find your dream rental property today.</p>

    <!-- Updated Search Form -->
    <div class="search-form">
        <h3>Search for Properties</h3>
        <form>
```

```
<label for="location">Location:</label>
<input type="text" id="location" name="location" placeholder="e.g., New
York City">
<label for="property-type">Property Type:</label>
<select id="property-type" name="property-type">
  <option value="apartment">Apartment</option>
  <option value="house">House</option>
  <option value="condo">Condo</option>
</select>
<button type="submit">Search</button>
</form>
</div>
```

```
<!-- Updated Property Listings Section -->
<section class="property-listings">
  <h3></h3>
  <div class="property">
    
    <p></p>
  </div>
  <div class="property">
    
    <p></p>
  </div>
  <div class="property">
    
    <p></p>
  </div>
  <div class="property">
    
    <p></p>
  </div>
  <!-- Add more property listings as needed -->
</section>
```



```
<!-- Updated Site Description Section -->
<section class="site-description">
</section>
</main>

<!-- Updated Footer -->
<!-- Updated Footer -->
<footer>
  <div class="footer-content">
    <div class="footer-section about">
      <h4>About Us</h4>
      <p>We are your premier source for rental properties. Our mission is to help you
find the perfect place to call home.</p>
    </div>
    <div class="footer-section contact">
      <h4>Contact Us</h4>
      <p>Email: info@propertyrentals.com</p>
      <p>Phone: +1-123-456-7890</p>
    </div>
    <div class="footer-section links">

    </div>
    <div class="footer-section social">

    </div>
  </div>
</div>
<div class="footer-bottom">
  <p>&copy; 2023 Property Rentals</p>
</div>
</footer>

</body>
</html>
```

**PropertyOwnerDashboard.html**

```
{% load static %}
```

```
<!DOCTYPE html>
```

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

```
<head>
```

```
    <title>Property Management</title>
```

```
    <!-- Add Bootstrap CSS link for international styling -->
```

```
    <link rel="stylesheet"
```

```
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
```

```
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-  
awesome/5.15.3/css/all.min.css">
```

```
<style>
```

```
    /* Custom CSS */
```

```
    body {
```

```
        background-color: #f2f2f2;
```

```
        font-family: Arial, sans-serif;
```

```
        margin: 0;
```

```
        padding: 0;
```

```
    }
```

```
    h1 {
```

```
        background-color: #7f9600;
```

```
        color: #fff;
```

```
        text-align: center;
```

```
        padding: 10px;
```

```
        font-size: 24px;
```

```
    }
```

```
    .container {
```

```
        max-width: 1200px;
```

```
        margin: 0 auto;
```

```
padding: 20px;
background-color: lightblue;
border-radius: 10px;
box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

h2 {
font-size: 24px;
margin-bottom: 20px;
color: #009688;
}

.row {
display: flex;
flex-wrap: wrap;
margin: 0 -15px;
}

.col-md-4 {
flex: 0 0 33.333333%;
max-width: 33.333333%;
padding: 0 15px;
}

.card {
margin-bottom: 20px;
background-color: cornsilk;
transition: background-color 0.3s, box-shadow 0.3s;
}

.card:hover {
background-color: antiquewhite;
box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
}
```

```
.card-body {  
    height: 800px; /* Set your desired fixed height */  
    overflow: hidden; /* Hide overflow content if it exceeds the fixed height */  
}  
  
.card-title {  
    font-size: 20px;  
    color: #009688;  
}  
  
.card-text {  
    font-size: 16px;  
    color: #333;  
}  
  
.form-group {  
    margin-bottom: 20px;  
}  
  
.form-control {  
    width: 100%;  
    padding: 10px;  
    border: 1px solid #ccc;  
    border-radius: 5px;  
}  
  
/* Notification Panel Styles */  
.notification-panel {  
    position: fixed;  
    top: 20px;  
    right: 20px;  
    max-width: 300px;  
    background-color: #fff;
```

```
border: 1px solid #ddd;
border-radius: 5px;
padding: 10px;
box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
display: none;
}
```

```
/* Notification Bell Button Styles */
```

```
.notification-btn {
    position: fixed;
    top: 20px;
    right: 20px;
    font-size: 24px;
    color: rgb(90, 0, 150);
    cursor: pointer;
    transition: color 0.3s;
}
```

```
.notification-btn:hover {
    color: #790000;
}
```

```
.btn {
    background-color: rgb(150, 0, 0);
    color: #fff;
    font-weight: bold;
    cursor: pointer;
    padding: 10px 20px;
    border-radius: 5px;
    transition: background-color 0.3s;
}
```

```
.btn:hover {
    background-color: #790000;
}
```

```
    }

    .modal-dialog {
        max-width: 100%;
    }

    .image-placeholder {
        background-color: #ddd;
        height: 200px;
    }

    a {
        text-decoration: none;
        color: blueviolet;
    }

    a:hover {
        color: blueviolet;
    }
</style>
<link rel="shortcut icon" href="{% static 'images/Key.png' %}" type="">
</head>

<body>
    <h1>Manage Your Properties</h1>

    <h6><a href="{% url 'ownerpg' %}">Back to Home</a></h6>

    <!-- Notification Bell Button -->
    <div class="notification-btn" id="notificationBtn"
onclick="toggleNotificationPanel()">
        <i class="fas fa-bell"></i>
    </div>

    <!-- Notification Panel -->
```

```

<div class="notification-panel" id="notificationPanel">
    <!-- Your notifications go here -->
    <p>Notification 1</p>
    <p>Notification 2</p>
    <!-- Add more notifications as needed -->
</div>

<div class="container">
    <div class="user-name">Property Owner: {{ request.session.username }}</div>

    <!-- Property Listing Section -->
    <section>
        <h2>Your Properties</h2>
        <div class="row">
            {% for property in properties %}
            <div class="col-md-4">
                <div class="card mb-4">
                    <div class="card-body">
                        <p class="card-title">{{ property.property_type }}</p>
                        <p class="card-text"><strong>Address:</strong> {{
property.address }}</p>
                        <p class="card-text"><strong>Monthly Rent:</strong> Rs{{
property.monthly_rent }}</p>
                        <p class="card-text"><strong>Security Amount:</strong>Rs {{
property.security_deposit }}</p>
                        <p class="card-text"><strong>Availability Date:</strong> {{
property.availability_date }}</p>

                        <!-- Property Images Carousel/Slider -->
                        <div id="propertyImageCarousel{{ property.id }}"
class="carousel slide" data-ride="carousel">
                            <ol class="carousel-indicators">
                                {% for image in property.propertyimage_set.all %}
                                    <li data-target="#propertyImageCarousel{{ property.id

```

```

}}" data-slide-to="{{ forloop.counter0 }}" {% if forloop.first %}class="active"{% endif
%}></li>

        {% endfor %}
    </ol>
    <div class="carousel-inner">
        {% for image in property.propertyimage_set.all %}
        <div class="carousel-item {% if forloop.first
%}active{% endif %}">

            
        </div>
        {% empty %}
        <div class="carousel-item active">
            <div class="image-placeholder"></div>
        </div>
        {% endfor %}
    </div>
    <a class="carousel-control-prev"
href="#propertyImageCarousel{{ property.id }}" role="button"
data-slide="prev">
        <span class="carousel-control-prev-icon" aria-
hidden="true"></span>

        <span class="sr-only">Previous</span>
    </a>
    <a class="carousel-control-next"
href="#propertyImageCarousel{{ property.id }}" role="button"
data-slide="next">
        <span class="carousel-control-next-icon" aria-
hidden="true"></span>

        <span class="sr-only">Next</span>
    </a>
</div>

<p class="card-text"><a href="{% url 'propimgup' property.id

```



```

%}">Upload Property Images</a></p>
    <p class="card-text"><a href="{% url 'propdocs' property.id
%}">Upload Property Documents</a></p>
    <!-- Inside the loop for properties -->
    <p class="card-text"><strong>Approval Status:</strong> {{
property.approval_status }}</p>
    <!-- Display Rental Requests for the particular property -->
    <div class="card-text">
        <strong>Rental Requests:</strong> {{
property.rentalrequest_set.count }}
        <ul class="list-unstyled">
            {% for request in property.rentalrequest_set.all %}
                <li>
                    <strong>Tenant:</strong> {{
request.tenant.full_name }} |
                    <strong>Contact no:</strong>{{
request.tenant.phone_number }} |
                    <strong>Created at:</strong> {{
request.created_at }}
                    {% if request.status == 'Accepted' %}
                        <button class="btn btn-success"
disabled>Accepted</button>
                    <!-- Show an icon to create a lease
agreement -->
                        <a href="{% url 'lease' property.id %}">
                            <i class="fas fa-file-contract"></i>
                        </a>
                    {% else %}
                        <form method="post" action="{% url
'accept_rental_request' request.id %}">
                            {% csrf_token %}
                            <button type="submit" class="btn btn-
primary">Accept Request</button>
                        </form>

```

```

        {% endif %}
    </li>
{% endfor %}
</ul>
</div>
<!-- Add other details as needed -->

<form method="post" onsubmit="return confirm('Are you sure
you want to delete this property?');">
    {% csrf_token %}
    <input type="hidden" name="property_id_to_delete"
value="{{ property.id }}">
    <button type="submit" class="btn btn-danger">Delete
Property</button>
</form>
</div>
</div>
</div>
{% endfor %}
</div>
</section>
</div>

<!-- Modals for Property Images -->
{% for property in properties %}
    <div class="modal fade" id="propertyImageModal{{ property.id }}" tabindex="-1"
role="dialog"
    aria-labelledby="propertyImageModalLabel{{ property.id }}" aria-hidden="true">
        <div class="modal-dialog modal-lg">
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title" id="propertyImageModalLabel{{ property.id
}}">Property Images</h5>
                    <button type="button" class="close" data-dismiss="modal" aria-

```

```

label="Close">
    <span aria-hidden="true">&times;</span>
</button>
</div>
<div class="modal-body">
    <div id="propertyImageCarousel{{ property.id }}" class="carousel
slide" data-ride="carousel">
        <ol class="carousel-indicators">
            {% for image in property.propertyimage_set.all %}
            <li data-target="#propertyImageCarousel{{ property.id }}"
data-slide-to="{{ forloop.counter0 }}"
                {% if forloop.first %}class="active"{% endif %}></li>
            {% endfor %}
        </ol>
        <div class="carousel-inner">
            {% for image in property.propertyimage_set.all %}
            <div class="carousel-item {% if forloop.first %}active{% endif
%}">
                
            </div>
            {% empty %}
            <div class="carousel-item active">
                <div class="image-placeholder"></div>
            </div>
            {% endfor %}
        </div>
        <a class="carousel-control-prev" href="#propertyImageCarousel{{
property.id }}" role="button"
            data-slide="prev">
            <span class="carousel-control-prev-icon" aria-
hidden="true"></span>
            <span class="sr-only">Previous</span>
        </a>

```



</html>

### TenantDashboard.html

{% load static %}

<!DOCTYPE html>

<html>

<head>

    <title>Tenant Dashboard</title>

    <link rel="stylesheet"

href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.2.0/css/all.min.css">

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

    <script

src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>

    <script

src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

    <style>

        body {

            font-family: Arial, sans-serif;

            background-color: #f2f2f2;

            margin: 0;

            padding: 0;

        }

        header {

            background-color: #bbcecc;

            color: #fff;

            text-align: center;

            padding: 5px;

        }

```
header h1 {  
    font-size: 28px;  
}  
  
.dashboard {  
    max-width: 1200px;  
    margin: 0 auto;  
    padding: 20px;  
    background-color: #fff;  
    border-radius: 10px;  
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);  
}  
  
.dashboard h2 {  
    font-size: 24px;  
    margin-bottom: 10px;  
    color: #009688;  
}  
  
.section {  
    margin: 20px 0;  
    padding: 20px;  
    background-color: #f9f9f9;  
    border: 1px solid #ddd;  
    border-radius: 5px;  
}  
  
.section h3 {  
    font-size: 20px;  
    margin-bottom: 10px;  
    color: #009688;  
}  
  
.property-list {
```

```
    display: flex;
    flex-wrap: wrap;
    list-style-type: none;
    padding: 0;
}

.property-card {
    border: 1px solid #ddd;
    border-radius: 5px;
    padding: 15px;
    margin: 10px;
    background-color: cornflowerblue;
    flex: 1;
    max-width: calc(33.33% - 20px); /* Three columns with margin */
    min-height: 300px;
}

.property-card img {
    max-width: 100%;
    border-radius: 5px;
    margin: 10px 0; /* Add margin to images */
}

.property-card .property-details {
    padding: 10px;
}

.property-card h4 {
    font-size: 20px;
    margin: 0;
    color: #009688;
}

.property-card p {
```

```
        font-size: 16px;
        margin: 0;
    }

    .search-form {
        background-color: #fff;
        padding: 20px;
        border: 1px solid #ddd;
        border-radius: 5px;
        margin-top: 20px;
    }

    .search-form label {
        font-weight: bold;
    }

    .search-form input[type="text"],
    .search-form select {
        width: 100%;
        padding: 10px;
        border: 1px solid #ccc;
        border-radius: 5px;
        margin-top: 5px;
    }

    .search-button {
        background-color: #009688;
        color: #fff;
        font-weight: bold;
        cursor: pointer;
        padding: 10px 20px;
        border-radius: 5px;
        transition: background-color 0.3s;
    }
```



```
.search-button:hover {  
    background-color: #00796B;  
}  
  
.large-message-box {  
    display: none;  
    position: fixed;  
    top: 50%;  
    left: 50%;  
    transform: translate(-50%, -50%);  
    background-color: #fff;  
    border: 1px solid #ccc;  
    border-radius: 5px;  
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2);  
    padding: 20px;  
    z-index: 9999;  
}  
  
/* CSS for Message Box Content */  
.message-content {  
    font-size: 18px;  
    text-align: center;  
}  
  
/* CSS for Close Button */  
.close-button {  
    position: absolute;  
    top: 10px;  
    right: 10px;  
    cursor: pointer;  
    color: #333;  
}
```

---

**/\* Additional CSS styles... \*/**

**.filter-title {**  
    **font-weight: bold;**  
    **margin-top: 15px;**  
    **color: #009688;**  
**}**

**.filter-checkbox {**  
    **margin-top: 10px;**  
**}**

**/\* New styles for the larger profile picture \*/**

**.profile-picture-large {**  
    **position: absolute;**  
    **right: 10px;**  
    **top: 10px;**  
    **width: 90px; /\* Adjust the size as needed \*/**  
    **height: 90px; /\* Adjust the size as needed \*/**  
    **border-radius: 50%;**  
    **overflow: hidden;**  
    **background-color: #fff; /\* Add background color for the circular container \*/**  
**}**

**/\* Style for the circular image \*/**

**.profile-image {**  
    **width: 100%;**  
    **height: 100%;**  
    **object-fit: cover; /\* Maintain aspect ratio and cover the circular container \*/**  
    **border-radius: 50%;**  
**}**

**/\* Improved navigation link style \*/**

**header nav a {**

```
    text-decoration: none;
    color: #fff;
    font-weight: bold;
    background-color: #00796B;
    padding: 10px 20px;
    border-radius: 5px;
    transition: background-color 0.3s;
}

header nav a:hover {
    background-color: #005950;
}

footer {
    background-color: #333;
    color: #fff;
    text-align: center;
    padding: 10px;
}

/* Your existing CSS... */

/* Adjust styles for the carousel */
.property-card .carousel {
    width: 100%;
}

.property-card .carousel .carousel-inner {
    max-height: 200px; /* Adjust the maximum height as needed */
}

.property-card .carousel .carousel-inner img {
    max-width: 100%;
    height: auto;
}
```

```
    }

    /* Style for the "Rent" button */
    /* Style for the "Rent" button */
    .rent-button {
        background-color: #009688;
        color: #fff;
        font-weight: bold;
        border: none;
        border-radius: 5px;
        padding: 10px 20px;
        cursor: pointer;
        transition: background-color 0.3s;
        margin-top: 10px; /* Add margin to push the button down */
    }

    .rent-button:hover {
        background-color: #00796B;
    }

    /* Adjust the margin between the "More Info" and "Rent" buttons */
    .property-details a {
        margin-right: 10px;
    }

    /* Style for the "Pay Now" button */
    .btn-primary {
        background-color: blue;
        color: #fff;
        font-weight: bold;
        border: none;
        border-radius: 5px;
        padding: 10px 20px;
        cursor: pointer;
        transition: background-color 0.3s;
```

```
margin-top: 10px; /* Add margin to push the button down */
}
```

```
.btn-primary:hover {
    background-color: #005950;
}
```

```
.popup-message {
    display: none;
    position: absolute;
    background-color: #f1f1f1;
    color: #333;
    padding: 10px;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2);
    z-index: 1;
}
```

```
</style>
```

```
<link rel="shortcut icon" href="{% static 'images/Key.png' %}" type="image/x-icon">
</head>
```

```
<body>
```

```
{% if messages %}
```

```
<div class="large-message-box">
```

```
<span class="close-button" onclick="closeMessageBox()">&times;</span>
```

```
<div class="message-content">
```

```
{% for message in messages %}
```

```
{{ message|safe }}<br>
```

```
{% endfor %}
```

```
</div>
```

```
</div>
```

```
{% endif %}
```

```
<header>
  <nav>
    <a href="{% url 'logout' %}" style="color: black; float: left; margin-left:
10px;">Log out</a>
    <a href="{% url 'tenantpage' %}" style="color: black; float:left; margin-right:
10px;">Back</a>
  </nav>
  <h1>Tenant Dashboard</h1>
  <div class="profile-picture-large">
    
  </div>
</header>

<div class="dashboard">
  <h2>Welcome, { { request.session.username } }</h2>

  <!-- Property search and filter form -->
  <div class="search-form">
    <h3>Search for Properties</h3>
    <form id="property-search-form" action="{% url 'tenantpg' %}"
method="get">
      <label for="property_type">Property Type:</label>
      <select name="property_type" id="property_type">
        <option value="">Any</option>
        <option value="apartment">Apartment</option>
        <option value="house">House</option>
        <option value="office">Office</option>
      </select>
      <label for="min_rent">Enter the range for rent:</label>
      <input type="number" name="min_rent" placeholder=" Monthly Rent">
      <input type="number" name="max_rent" placeholder=" Monthly Rent">
```

<!-- Add checkboxes and input fields for other filters (stories, acrescent, rent space, number of rooms, parking space, purpose) -->

```

        <button class="search-button" type="submit">Search</button>
    </form>
</div>

<!-- Property listing based on search results -->
<div class="section">
    <h3>Available Properties</h3>
    <ul class="property-list">
        {% for property in properties %}
            <li class="property-card">
                <!-- Add a Bootstrap carousel for property images -->
                <div class="carousel slide" data-ride="carousel">
                    <div class="carousel-inner">
                        {% for image in property.propertyimage_set.all %}
                            <div class="carousel-item {% if forloop.first
%}active{% % endif %}">
                                
                            </div>
                        {% endfor %}
                    </div>
                </div>
                <div class="property-details">
                    <h4>{{ property.property_name }}</h4>
                    <p><strong>Property type:</strong> {{ property.property_type
}}</p><br>
                    <p><strong>Address:</strong> {{ property.address }}</p><br>
                    <p><strong>Monthly Rent:</strong> Rs{{
property.monthly_rent }}</p><br>
                    <p><strong>Security Amount:</strong> Rs {{
property.security_deposit }}</p><br>

```

```

        <p><strong>Availability Date:</strong> {{
property.availability_date }}</p><br>
        <a href="{% url 'property_detail' property.id %}" class="btn
btn-info">
            <i class="fas fa-info-circle"></i> More Info
        </a>
        <a href="{% url 'ownerinfo' property.id %}" class="btn btn-
info" style="margin-right: 10px;">
            <i class="fas fa-info-circle"></i> Owner Info
        </a>
        <!-- View Documents Button -->
        <a href="{% url 'admviewdocs' property.id %}" class="btn
btn-secondary" style="margin-top: 10px; margin-right: 10px;">
            <i class="fas fa-file-alt"></i> View Documents
        </a>
        {% if not property.is_rental_request_accepted %}
        <div class="popup-message">Rental request already sent</div>
        {% endif %}
        <!-- Inside the property card in tenantpg.html -->
        {% if property.is_rental_request_accepted %}
        <form method="post" action="{% url 'payment' %}">

            {% csrf_token %}
            <button type="submit" class="btn btn-primary">
                Pay Now
            </button>

        </form>
        {% else %}
        <form method="post" action="{% url 'submit_rental_request'
property.id %}">

            {% csrf_token %}
            <button type="submit" class="btn btn-success rent-

```



```
button">
    <i class="fas fa-shopping-cart"></i> Rent Now
</button>
</form>
{% endif %}
</li>
{% endfor %}

<!-- Display a message if no results are found -->
{% if properties.count == 0 %}
    <p>No properties found matching your search criteria.</p>
{% endif %}
</ul>
</div>

<!-- Upcoming rent payments -->
<div class="section">
    <h3>Upcoming Rent Payments</h3>
    <p>Rent Payment: 1,200 (Due on 15th of the month)</p>
</div>
</div>

<footer>
    <p>&copy; 2023 Property Owner Dashboard</p>
</footer>

<!-- ... Your existing HTML code ... -->

<!-- ... Your existing HTML code ... -->

<script>
    // Function to show the large message box
    function showMessageBox() {
        const messageBox = document.querySelector(".large-message-box");
```

```
messageBox.style.display = "block";

// Automatically hide the message box after 5 seconds (5000 milliseconds)
setTimeout(() => {
    messageBox.style.display = "none";
}, 2000); // Adjust the time (in milliseconds) as needed
}

// Call the showMessageBox function when the page loads
window.addEventListener("load", showMessageBox);

document.addEventListener("DOMContentLoaded", function () {
    // Get all property cards
    var propertyCards = document.querySelectorAll(".property-card");

    // Iterate through each property card
    propertyCards.forEach(function (card) {
        // Check if the card has a popup message
        var popupMessage = card.querySelector(".popup-message");

        if (popupMessage) {
            // Show the popup if rental request not accepted
            if (!card.classList.contains("rental-request-accepted")) {
                popupMessage.style.display = "block";
            }

            // Hide the popup when clicking outside of it
            document.addEventListener("click", function (event) {
                if (!popupMessage.contains(event.target)) {
                    popupMessage.style.display = "none";
                }
            });
        }
    });
});
```

```
// Check if the "Pay Now" button was clicked before
var payNowButton = card.querySelector(".btn-primary");

// Use sessionStorage to persist the information
var paymentStatus = sessionStorage.getItem("paymentStatus");

if (paymentStatus === "paid") {
    // If payment is done, hide "Pay Now" button permanently
    payNowButton.style.display = "none";

    // Show a new button in its place
    var newButton = document.createElement("button");
    newButton.className = "btn btn-success";
    newButton.textContent = "Rented";
    card.appendChild(newButton);
} else {
    // If payment is not done, add click event listener
    payNowButton.addEventListener("click", function () {
        // Hide "Pay Now" button permanently
        payNowButton.style.display = "none";

        // Set payment status to "paid" in sessionStorage
        sessionStorage.setItem("paymentStatus", "paid");
    });
}

});

});

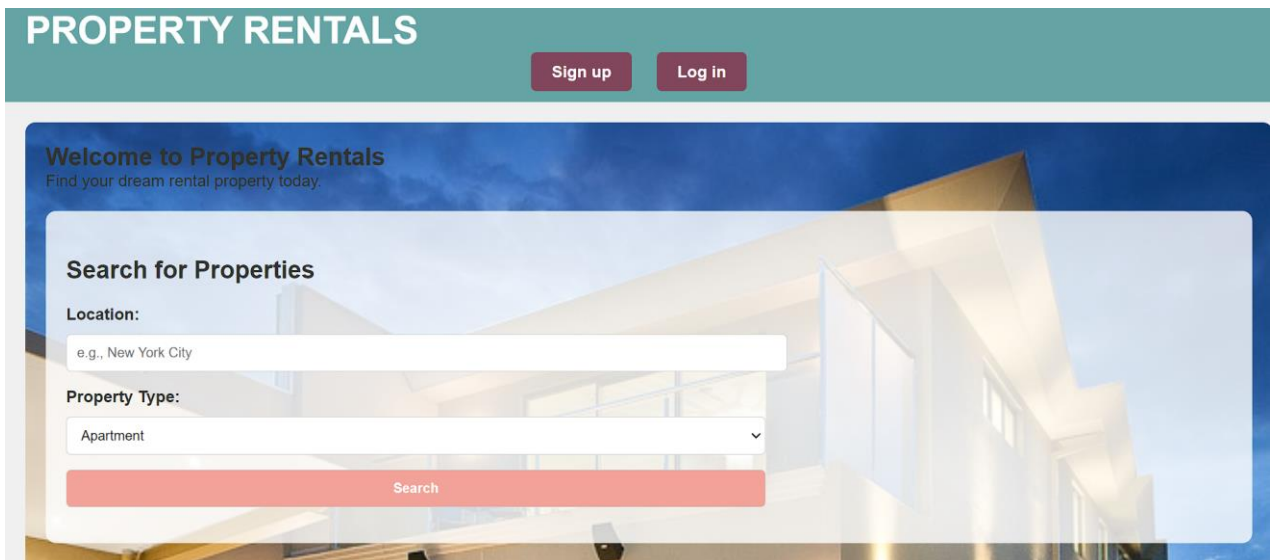
</script>

</body>
```

</html>

## 9.1 Screen Shots

### Homepage



The screenshot shows the homepage of a website titled "PROPERTY RENTALS". The header is a teal bar with the title in white. To the right of the title are two buttons: "Sign up" and "Log in". Below the header is a large banner image of a modern building at night. Overlaid on the banner is a white search box. The search box has a title "Search for Properties". It contains two input fields: "Location:" with a placeholder "e.g., New York City" and "Property Type:" with a dropdown menu showing "Apartment". Below these fields is a red "Search" button.

**PROPERTY RENTALS**

[Sign up](#) [Log in](#)

**Welcome to Property Rentals**  
Find your dream rental property today.

**Search for Properties**

**Location:**  
e.g., New York City

**Property Type:**  
Apartment

**Search**

