

# **Green Grocery**

*Mini Project Report*

*Submitted by*

**Sreelakshmi P S**

**Reg. No.: AJC20MCA-I053**

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

**INTEGRATED MASTER OF COMPUTER APPLICATIONS**

**(INMCA)**

**APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**



**AMAL JYOTHI COLLEGE OF ENGINEERING AUTONOMOUS**

**KANJIRAPPALLY**

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

**2024-2025**

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



**CERTIFICATE**

This is to certify that the Mini Project report, “**GREEN GROCERY**” is the bona fide work of **SREELAKSHMI P S (Regno: AJC20MCA-I053)** in partial fulfillment of the requirements for the award of the Degree of Integrated Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2024-25.

**Ms. Jetty Benjamin**  
**Internal Guide**

**Mr. Binumon Joseph**  
**Coordinator**

**Rev. Fr. Dr. Rubin Thottupurathu Jose**  
**Head of the Department**

## **DECLARATION**

I hereby declare that the mini project report “**GREEN GROCERY**” is a bona fide work done at Amal Jyothi College of Engineering Autonomous, towards the partial fulfilment of the requirements for the award of the **Integrated Master of Computer Applications (INMCA)** from **APJ Abdul Kalam Technological University**, during the academic year **2024-2025**.

**Date:**  
**KANJIRAPPALLY**

**SREELAKSHMI P S**  
**Reg: AJC20MCA-I053**

## 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 Director (Administration) **Rev. Fr. Dr. Roy Abraham Pazhayaparampil** 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 **Mr. Binumon Joseph, Assistant Professor** for his 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. Jetty Benjamin, Assistant Professor** 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.

SREELAKSHMI P S

## **ABSTRACT**

This platform enables seamless sale and distribution of fresh fruits and vegetables, connecting customers with local farmers and ensuring quality through a machine learning-based detection system. Customers can browse products, filter options, add items to their cart, and securely complete purchases, with real-time order tracking and delivery updates. Farmers manage inventory and orders via a dedicated portal, while delivery personnel track routes and update delivery statuses to ensure timely fulfilment. The machine learning system uses computer vision to assess produce freshness and quality based on color, texture, and appearance, guaranteeing high standards for listed items. An admin portal supports platform-wide management, including user control, inventory and payment tracking, and product categorization, creating a reliable, quality-driven marketplace for both customers and suppliers.

# CONTENT

SL. NO	TOPIC	PAGE NO
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
<b>1.1</b>	<b>PROJECT OVERVIEW</b>	<b>2</b>
<b>1.2</b>	<b>PROJECT SPECIFICATION</b>	<b>3</b>
<b>2</b>	<b>SYSTEM STUDY</b>	<b>4</b>
<b>2.1</b>	<b>INTRODUCTION</b>	<b>5</b>
<b>2.2</b>	<b>LITERATURE REVIEW</b>	<b>5</b>
<b>2.3</b>	<b>PROPOSED SYSTEM</b>	<b>7</b>
<b>2.4</b>	<b>ADVANTAGES OF PROPOSED SYSTEM</b>	<b>8</b>
<b>3</b>	<b>REQUIREMENT ANALYSIS</b>	<b>9</b>
<b>3.1</b>	<b>FEASIBILITY STUDY</b>	<b>10</b>
<b>3.1.1</b>	<b>ECONOMICAL FEASIBILITY</b>	<b>10</b>
<b>3.1.2</b>	<b>TECHNICAL FEASIBILITY</b>	<b>10</b>
<b>3.1.3</b>	<b>BEHAVIORAL FEASIBILITY</b>	<b>11</b>
<b>3.1.4</b>	<b>FEASIBILITY STUDY QUESTIONNAIRE</b>	<b>11</b>
<b>3.2</b>	<b>SYSTEM SPECIFICATION</b>	<b>12</b>
<b>3.2.1</b>	<b>HARDWARE SPECIFICATION</b>	<b>12</b>
<b>3.2.2</b>	<b>SOFTWARE SPECIFICATION</b>	<b>12</b>
<b>3.3</b>	<b>SOFTWARE DESCRIPTION</b>	<b>13</b>
<b>3.3.1</b>	<b>DJANGO</b>	<b>13</b>
<b>3.3.2</b>	<b>SQLITE</b>	<b>13</b>
<b>4</b>	<b>SYSTEM DESIGN</b>	<b>14</b>
<b>4.1</b>	<b>INTRODUCTION</b>	<b>15</b>
<b>4.2</b>	<b>UML DIAGRAM</b>	<b>15</b>
<b>4.2.1</b>	<b>USE CASE DIAGRAM</b>	<b>16</b>
<b>4.2.2</b>	<b>SEQUENCE DIAGRAM</b>	<b>17</b>
<b>4.2.3</b>	<b>ACTIVITY DIAGRAM</b>	<b>21</b>
<b>4.2.4</b>	<b>CLASS DIAGRAM</b>	<b>25</b>
<b>4.2.5</b>	<b>OBJECT DIAGRAM</b>	<b>26</b>
<b>4.3</b>	<b>USER INTERFACE DESIGN USING FIGMA</b>	<b>27</b>
<b>4.4</b>	<b>DATABASE DESIGN</b>	<b>28</b>
<b>4.4.1</b>	<b>RDBMS</b>	<b>28</b>

<b>4.4.2</b>	<b>NORMALIZATION</b>	<b>28</b>
<b>4.4.3</b>	<b>SANITIZATION</b>	<b>30</b>
<b>4.4.4</b>	<b>INDEXING</b>	<b>30</b>
<b>4.5</b>	<b>TABLE DESIGN</b>	<b>31</b>
<b>5</b>	<b>SYSTEM TESTING</b>	<b>36</b>
<b>5.1</b>	<b>INTRODUCTION</b>	<b>37</b>
<b>5.2</b>	<b>TEST PLAN</b>	<b>37</b>
<b>5.2.1</b>	<b>UNIT TESTING</b>	<b>37</b>
<b>5.2.2</b>	<b>INTEGRATION TESTING</b>	<b>38</b>
<b>5.2.3</b>	<b>VALIDATION TESTING</b>	<b>38</b>
<b>5.2.4</b>	<b>USER ACCEPTANCE TESTING</b>	<b>38</b>
<b>5.2.5</b>	<b>AUTOMATION TESTING</b>	<b>39</b>
<b>5.2.6</b>	<b>SELENIUM TESTING</b>	<b>39</b>
<b>6</b>	<b>IMPLEMENTATION</b>	<b>50</b>
<b>6.1</b>	<b>INTRODUCTION</b>	<b>51</b>
<b>6.2</b>	<b>IMPLEMENTATION PROCEDURE</b>	<b>51</b>
<b>6.2.1</b>	<b>USER TRAINING</b>	<b>51</b>
<b>6.2.2</b>	<b>TRAINING ON APPLICATION SOFTWARE</b>	<b>52</b>
<b>6.2.3</b>	<b>SYSTEM MAINTENANCE</b>	<b>52</b>
<b>6.2.4</b>	<b>HOSTING</b>	<b>53</b>
<b>7</b>	<b>CONCLUSION &amp; FUTURE SCOPE</b>	<b>54</b>
<b>7.1</b>	<b>CONCLUSION</b>	<b>55</b>
<b>7.2</b>	<b>FUTURE SCOPE</b>	<b>55</b>
<b>8</b>	<b>BIBLIOGRAPHY</b>	<b>56</b>
<b>9</b>	<b>APPENDIX</b>	<b>58</b>
<b>9.1</b>	<b>SAMPLE CODE</b>	<b>59</b>
<b>9.2</b>	<b>SCREEN SHOTS</b>	<b>68</b>
<b>9.3</b>	<b>GIT LOG</b>	<b>71</b>

## List of Abbreviations

- **UML** - Unified Modeling Language
- **CSS** - Cascading Style Sheets
- **HTML** - Hypertext Markup Language
- **SQLite** - Structured Query Language (Lightweight Database)
- **IDE** - Integrated Development Environment
- **Python Django** - Python-based Web Framework



# **CHAPTER 1**

## **INTRODUCTION**

## 1.1 PROJECT OVERVIEW

Green Grocery online platform, a dedicated marketplace that specializes in the sale and distribution of fresh fruits and vegetables directly from local farmers to customers. Our website is designed to provide a seamless shopping experience for those who value quality produce and want to support sustainable farming practices. With a user-friendly interface, customers can easily browse and search for products based on type, seasonality, and pricing, making it simple to find the items they need. Each product listing includes detailed descriptions, vibrant images, and customer reviews to assist shoppers in making informed choices. We regularly offer promotions, discounts, and loyalty programs to reward our customers for their purchases, ensuring they get the best value for their money. Our platform supports various secure payment options, including credit cards, debit cards, and PayPal, allowing customers to shop with confidence. Additionally, we provide convenient services like home delivery and flexible scheduling, making it easier than ever to enjoy fresh produce without leaving the comfort of home. We strive to create an accessible and enjoyable shopping experience while promoting sustainable practices and supporting local farmers.

## 1.2 PROJECT SPECIFICATION

The proposed system is a comprehensive website designed for the sale and distribution of fresh fruits and vegetables. It automates the functionalities of a physical grocery store, providing a user-friendly experience that is available 24/7. Customers can conveniently order a wide variety of products directly from local farmers, enjoying competitive pricing and services such as home delivery from the comfort of their own homes. The platform also integrates essential features for inventory management, customer orders, and delivery scheduling.

### List of Modules

#### Customer Portal:

- User Registration and Login: Securely create accounts and log in to the platform.
- Product Browsing and Search Functionality: Explore a wide range of fruits and vegetables using filters.
- Shopping Cart and Checkout Process: Add items to the cart and complete the purchase securely.

- Order Tracking and Delivery: View order history and track delivery status.

**Supplier Portal:**

- Registration and Login: Farmers can create accounts and log in to manage their profiles.
- Inventory Management: List, update, and remove products from their inventory.
- Order Management: View and accept incoming orders from customers.
- Add Product: Easily update or remove products from their listings.

**Delivery Management Portal:**

- Login: Delivery personnel can access their accounts securely.
- View and Manage Delivery Assignments: Check assigned deliveries and their details.
- Delivery Tracking and Status Updates: Update the status of deliveries in real-time.

**Admin Portal:**

- User Management and Access Control: Manage customer and supplier accounts and control access levels.
- Track Stock Level: Monitor inventory levels across the platform.
- Price Chart and Comparison: Analyze product pricing for competitive analysis.
- Manage Payment Details: Oversee payment processing and transaction records.
- Add Categories, Subcategories, and Product Names: Maintain organized product listings.

## **CHAPTER 2**

### **SYSTEM STUDY**

## **2.1 INTRODUCTION**

A system study for the Green Grocery online platform involves evaluating various aspects of the website to ensure it provides a seamless shopping experience for its users. This study may include analyzing the website's user interface, navigation, security, privacy, payment and shipping options, customer service, website performance, and marketing strategies. By conducting a thorough analysis of these areas, we can identify opportunities to improve the platform's functionality and user experience, increase customer satisfaction, and ultimately drive more traffic and sales. This evaluation will help ensure that the Green Grocery website meets the needs of its users while also achieving the business goals of the platform, fostering a reliable connection between customers and local farmers.

## **2.2 LITERATURE REVIEW**

The proposed online platform for purchasing fresh fruits and vegetables directly from local farmers addresses critical limitations associated with traditional grocery shopping, particularly for customers seeking convenience, variety, and ease of access to fresh produce. Traditional grocery shopping systems rely on in-store visits, which can often be inconvenient and time-consuming, particularly for individuals managing busy schedules or lacking proximity to grocery stores. Studies indicate that these in-store experiences frequently lead to customer dissatisfaction due to a range of factors, including limited product availability, fixed store hours, long checkout lines, and often inadequate inventory tracking systems. Manual inventory management in traditional stores, in particular, presents significant limitations, as it is susceptible to human error and can lead to discrepancies in stock levels, ultimately impacting customer satisfaction and store efficiency. In an age where consumers increasingly seek seamless and efficient shopping experiences, these challenges underscore a demand for innovative grocery solutions that enhance access, availability, and the overall shopping experience [1, 3, 6].

The proposed system utilizes a user-friendly online interface paired with real-time inventory management and efficient order tracking capabilities, which collectively provide a streamlined and responsive shopping experience for consumers. Digital grocery platforms have shown, in prior research, to significantly improve customer satisfaction through features such as 24/7 availability, secure payment methods, and flexible delivery or pickup options that adapt to consumers' busy lifestyles [2, 4]. Additionally, the platform leverages a direct-to-consumer model, strengthening the connection between local farmers and

consumers by reducing intermediaries in the supply chain and, as a result, promoting fresher produce options. This approach not only ensures that customers receive fresher, higher-quality produce but also supports local farmers by creating direct economic opportunities and eliminating the need for multiple third-party distributors. By creating this direct bridge between farmers and consumers, the proposed platform allows consumers transparency regarding the sources of their products, building trust and contributing to more informed and ethically conscious purchasing decisions. By addressing these limitations associated with traditional grocery models, the proposed platform fills a significant gap in the grocery shopping experience for both consumers and farmers, offering a model that is efficient, accessible, and socially beneficial [7].

Furthermore, the emphasis on quality assurance within this online platform is particularly supported by advancements in automated quality detection systems, which are becoming increasingly relevant in grocery environments. Traditional quality assessment methods in grocery stores are generally labor-intensive, reliant on manual inspections, and vulnerable to inconsistencies, leading to issues with quality assurance and consumer dissatisfaction. However, recent technological advances have introduced image processing and machine learning methodologies that can provide accurate, objective, and consistent quality assessments of produce. These technologies allow for the implementation of automated systems that analyze critical factors like color, texture, and shape of fruits and vegetables to ensure optimal freshness and quality. For example, image processing techniques enable systems to assess ripeness based on color changes, texture analysis allows for the detection of wrinkles or soggyiness indicative of spoilage, and shape analysis helps identify irregularities that may suggest damage or contamination. Studies have demonstrated the effectiveness of such systems, showing that they can substantially reduce the margin for error compared to traditional manual inspections, thereby providing customers with higher-quality produce and reducing overall food waste [5, 8, 12].

Incorporating automated quality assessment tools into the proposed online grocery platform not only addresses quality control issues but also enhances the customer experience by ensuring consistent access to high-quality, fresh produce. Research indicates that integrating these automated quality detection systems in grocery environments has additional benefits beyond consumer satisfaction, as it minimizes the need for labor-intensive manual inspections and thus lowers operational costs for grocery providers. Moreover, these technologies are equipped with real-time feedback mechanisms that help grocery stores and

consumers identify produce freshness immediately. This can be particularly advantageous in preventing food wastage, as products that do not meet quality standards can be swiftly removed from the distribution chain. By combining machine learning with IoT (Internet of Things) sensors, future quality detection systems can offer even more refined quality control capabilities, incorporating factors like humidity and temperature data to make precise judgments about produce shelf life and quality. By reducing waste, enhancing produce quality, and streamlining the shopping experience, these automated systems align well with the goals of the proposed online platform, contributing to a more sustainable and efficient supply chain [9, 10, 15].

Overall, the proposed online platform's innovative approach to grocery shopping reflects broader trends in digital transformation within the retail industry. By leveraging an online interface, direct-to-consumer supply chains, and automated quality control systems, the platform not only modernizes the grocery shopping experience but also bridges existing gaps within the traditional grocery model. These advancements create a mutually beneficial ecosystem for consumers and local farmers, whereby consumers gain access to fresher and higher-quality produce, and farmers benefit from more streamlined market access and increased economic support. This alignment of consumer needs, technological capabilities, and sustainable practices represents a shift toward a more adaptable, responsive, and customer-centered grocery model. Additionally, the system's focus on reducing food waste, maintaining quality, and supporting local agriculture ensures a lasting positive impact on the broader food supply chain. Thus, by effectively addressing the limitations of conventional grocery shopping, the proposed online platform promises to enhance customer satisfaction and provide a more resilient, efficient, and sustainable solution for the future of grocery shopping [12].

### **2.3 PROPOSED SYSTEM**

The proposed system is an online platform for buying fresh fruits and vegetables directly from local farmers, designed to enhance the grocery shopping experience. It features a user-friendly interface for customers to easily register, browse, and search for products from the comfort of their homes. Customers can add items to their shopping cart, modify quantities, and securely complete payments using various options. The platform provides real-time inventory management to ensure product availability and allows order tracking and delivery scheduling for convenience. Farmers can manage their profiles and list produce with pricing details, while delivery personnel can access assigned deliveries. The admin portal enables

user management, stock tracking, and order oversight. Overall, this system offers improved accessibility, enhanced customer satisfaction, and streamlined operations, making it a significant advancement over traditional grocery shopping methods.

## 2.4 ADVANTAGES OF PROPOSED SYSTEM

- **Convenience:** Customers can shop for fresh fruits and vegetables from the comfort of their homes, eliminating the need to travel to physical stores.
- **24/7 Availability:** The platform is accessible around the clock, allowing users to browse and make purchases at any time that suits them.
- **User-Friendly Interface:** An intuitive design simplifies navigation, enabling customers to easily search for products and complete transactions.
- **Real-Time Inventory Management:** Customers can view up-to-date product availability, reducing the chances of stockouts and ensuring they can select items with confidence.
- **Order Tracking:** Users can track their orders in real-time, providing transparency and enhancing customer satisfaction.
- **Flexible Payment Options:** The system supports multiple payment methods, making it easier for customers to complete transactions securely.
- **Feedback Mechanism:** Customers can provide feedback on products and services, facilitating continuous improvement and responsiveness to user needs.
- **Support for Farmers:** Local farmers can efficiently manage their profiles and product listings, enabling them to reach a broader audience and sell directly to consumers.



## **CHAPTER 3**

### **REQUIREMENT ANALYSIS**

### **3.1 FEASIBILITY STUDY**

A feasibility study evaluates whether a project can be practically performed, considering factors like resource availability, cost estimation, benefits, and maintenance expenses. Green Grocery is an online platform designed to streamline the sale and distribution of fresh fruits and vegetables directly from local farmers to customers, featuring inventory management, customer orders, delivery scheduling, nutritional information, product quality assurance through image recognition, and a chatbot for user assistance. The study aims to determine if the software meets organizational needs, can be built with current technology within budget and on time, and integrates with existing systems. The platform automates processes, improves communication, provides real-time updates, and enhances inventory and sales management. Green Grocery will improve business efficiency, enhance customer experience, increase revenue, and provide valuable data insights. While compatible with existing systems, it may require new technologies like cloud computing, necessitating training and support. The platform handles essential functions such as order processing, delivery tracking, inventory management, secure payments, and communication, while disregarding non-essential or outdated systems. In conclusion, Green Grocery presents significant benefits and is a feasible and valuable investment for the organization.

#### **3.1.1 Economical Feasibility**

The online grocery store is economically feasible due to lower setup and operational costs compared to physical stores. Using Django and SQLite reduces development expenses, and automation minimizes labor needs. With efficient scaling and machine learning-based quality detection, the system enhances customer satisfaction, offering a cost-effective solution with potential for high returns.

#### **3.1.2 Technical Feasibility**

Creating an online grocery store is technically feasible due to advancements in web development technologies and e-commerce platforms, which have made it easier and more affordable to establish a digital storefront. These platforms offer a variety of templates, themes, and plugins that facilitate the development of an attractive and functional website. . Numerous secure payment gateways can be integrated to process transactions efficiently.

The proposed grocery store website can be developed using technologies such as python

Django and a SQLite server for the backend, ensuring robust data management. The frontend can utilize HTML, CSS, JavaScript, jQuery, Bootstrap, and AJAX, resulting in a highly responsive and user-friendly interface. The development can be accomplished on a standard system equipped with an Intel i3 core processor, 4GB of RAM, and an SSD, making it technically feasible to complete the project effectively.

### **3.1.3 Behavioral Feasibility**

The behavioral feasibility of an online grocery store hinges on the ability to effectively cater to the needs and preferences of the target audience while distinguishing the business from existing competitors in the marketplace. By focusing on delivering an exceptional shopping experience, the store can attract busy individuals and families who prioritize convenience. Leveraging high-quality images and detailed product descriptions can help build trust and ensure customers feel confident in their purchases. Additionally, implementing user-friendly navigation and personalized recommendations can further enhance customer satisfaction. To stand out in a crowded market, the business could consider integrating features such as a loyalty program, exclusive access to seasonal local produce, and subscription services that offer regular deliveries of essentials. Providing flexible return policies and excellent customer service can also contribute to fostering loyalty and encouraging repeat business. In this way, the online grocery store can create a compelling value proposition that not only meets but exceeds customer expectations, paving the way for long-term success.

### **3.1.4 Feasibility Study Questionnaire**

**1. How do you currently sell your products?**

**Ans:** I sell my products at the local farmer's market and to some local grocery stores.

**2. How do you decide the pricing for your products?**

**Ans:** I set prices based on the current market prices and the quality of the products

**3. How do you ensure the quality and freshness of your products?**

**Ans:** Keeping the products fresh, harvest them carefully and store them in cool places and monitor their condition closely

**4. Do you sell your products directly to consumers or through intermediaries?**

**Ans:** Mostly directly to the consumers, but some products is sold through local grocery stores.

**5. How do you handle customer complaints?**

**Ans:** I often offer replacements or refunds if necessary.

**6. How do you decide what to plant each season?**

**Ans:** I decide based on the demand from customers and the climate conditions.

**7. How do you handle payment transactions?**

**Ans:** Through cash and sometimes use mobile payment apps.

**8. Do you offer any special deals or discounts to you customers?**

**Ans:** Yes, sometimes give discount for bulk purchases or to regular customers.

**9. Have you ever used any pesticides on your crops?**

**Ans:** Yes, I have used pesticides, but I try to use them as little as possible.

**10. How do you know when it's the best time to pick your fruits/vegetables?**

**Ans:** Look at their color, size and how ripe they are. We also taste them to make sure they're ready to harvest.

### **3.1 SYSTEM SPECIFICATION**

#### **3.2.1 Hardware Specification**

Processor - Intel Core i3

RAM - 4 G B

Hard disk - S S D

#### **3.2.2 Software Specification**

Front End - HTML, CSS, JavaScript

Back End - Python Django

Database - SQLite

Client on PC - Windows 10

Technologies used - HTML, CSS, JavaScript, jQuery, AJAX, Python Django, SQLite

## **3.3 SOFTWARE DESCRIPTION**

### **3.3.1 Python Django**

Django, a high-level Python web framework, is designed for rapid development and clean, pragmatic design. As of its latest version, Django provides a robust set of features for building web applications that can easily interact with databases, manage user authentication, and generate dynamic content. With its "batteries-included" philosophy, Django includes everything needed to create secure and scalable web applications, such as an ORM (Object-Relational Mapping) for database interactions, a customizable admin interface, and built-in security features that help protect against common vulnerabilities like SQL injection and cross-site scripting (XSS).

Django is known for its ease of use, which makes it suitable for both beginners and experienced developers. Its clear and concise syntax allows developers to build complex applications quickly. The framework supports the Model-View-Template (MVT) architectural pattern, which helps organize code effectively and enhances maintainability. Django is also highly extensible, with a wide range of third-party packages available to extend its functionality.

### **3.3.2 SQLite**

SQLite is a popular open-source relational database management system (RDBMS) known for its simplicity and efficiency. It is a self-contained, serverless database engine that is embedded within applications, making it lightweight and easy to set up. SQLite is particularly well-suited for small to medium-sized applications and is often used for development and testing due to its ease of use and minimal configuration requirements.

One of the key features of SQLite is its ability to store data in a single file, which simplifies data management and deployment. It supports standard SQL syntax, making it accessible to developers familiar with SQL. SQLite is reliable and performs well for read-heavy workloads, and it can handle multiple concurrent read operations efficiently.

Additionally, SQLite is compatible with various programming languages, including Python, making it an ideal choice for applications built with Django. Its built-in data integrity features, such as ACID compliance (Atomicity, Consistency, Isolation, Durability), ensure that transactions are processed reliably. Overall, SQLite is a powerful and versatile database solution that is widely used for storing and managing data in web applications.

## **CHAPTER 4**

### **SYSTEM DESIGN**

## **4.1 INTRODUCTION**

System design is the process of creating a blueprint for a system to meet specified requirements. In software engineering, it involves defining the system's modules, components, their relationships, and interfaces. Effective system design ensures scalability, maintainability, and efficiency by making systems modular, flexible, and easy to use and maintain. It lays the foundation for structured development and helps meet user and stakeholder needs.

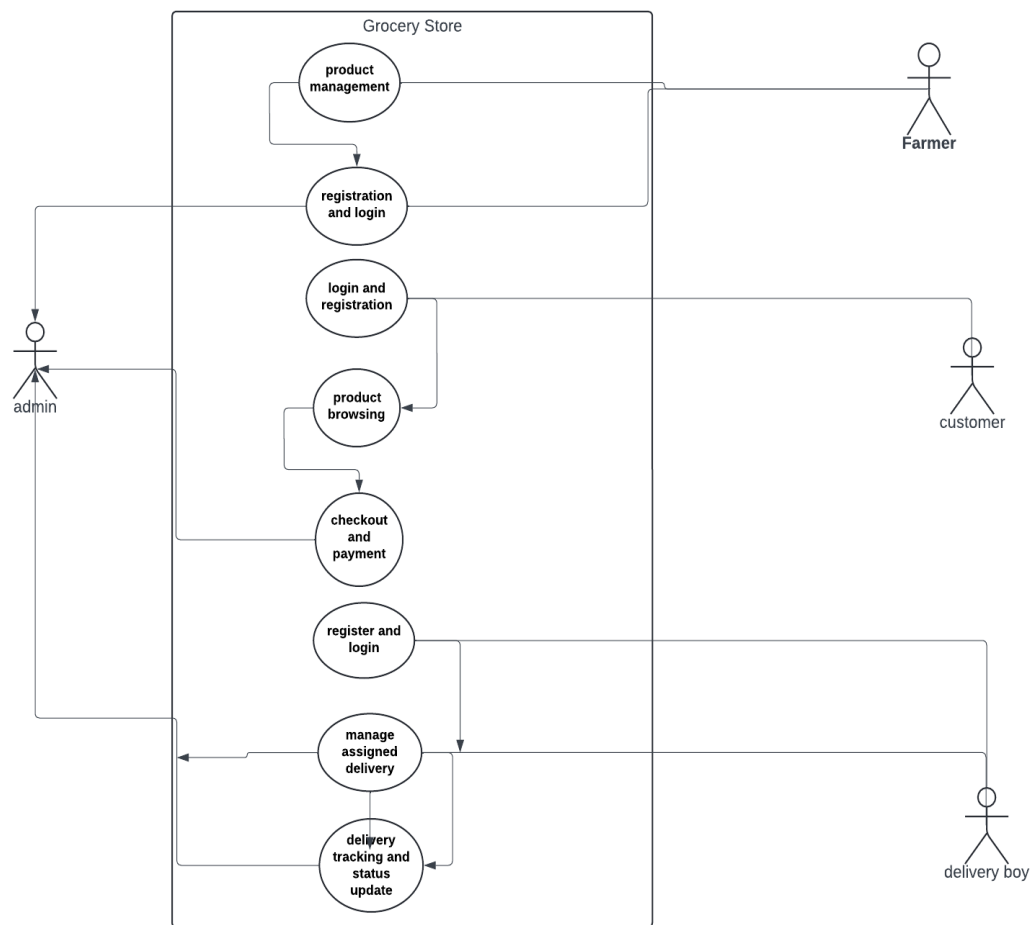
## **4.2 UML DIAGRAM**

Unified Modelling Language (UML) diagrams visually represent software systems, helping developers understand and communicate system structure and behaviour. Managed by the Object Management Group (OMG), UML is a standardized language used throughout software development. It provides graphical notations to create diagrams for different system aspects, from planning to deployment. UML ensures clear communication among developers and maintains consistency across interconnected diagrams, reflecting system-wide changes effectively includes mainly following diagrams:

- Class Diagram
- Object Diagram
- Use Case Diagram
- Sequence Diagram
- Activity Diagram
- State chart Diagram
- Deployment Diagram
- Component Diagram
- Collaboration Diagram

### **4.2.1 USE CASE DIAGRAM**

A use case diagram is a UML diagram that illustrates the interactions between actors (users or systems) and a system through various use cases. Each use case, shown as an oval, represents a functional requirement, while actors, depicted as stick figures, connect to use cases to show their interaction with the system. Use case diagrams help communicate system functionality to stakeholders, ensure it meets user requirements, and provide a basis for testing. Alongside other UML diagrams, like class, sequence, and activity diagrams, use case diagrams offer a complete view of a system's design and functionality.



**Figure 1: Use Case Diagram**

### 4.2.1 SEQUENCE DIAGRAM

A sequence diagram is a UML tool used to visualize interactions between objects or components in a system, focusing on the order of message exchanges and their timing. Objects are represented as vertical lifelines, while messages are shown as horizontal arrows indicating flow and direction. Sequence diagrams help model complex scenarios, ensuring the system behaves as expected. They are valuable in software design, development, testing, and debugging by offering a clear view of event sequences, making error identification and performance optimization easier.



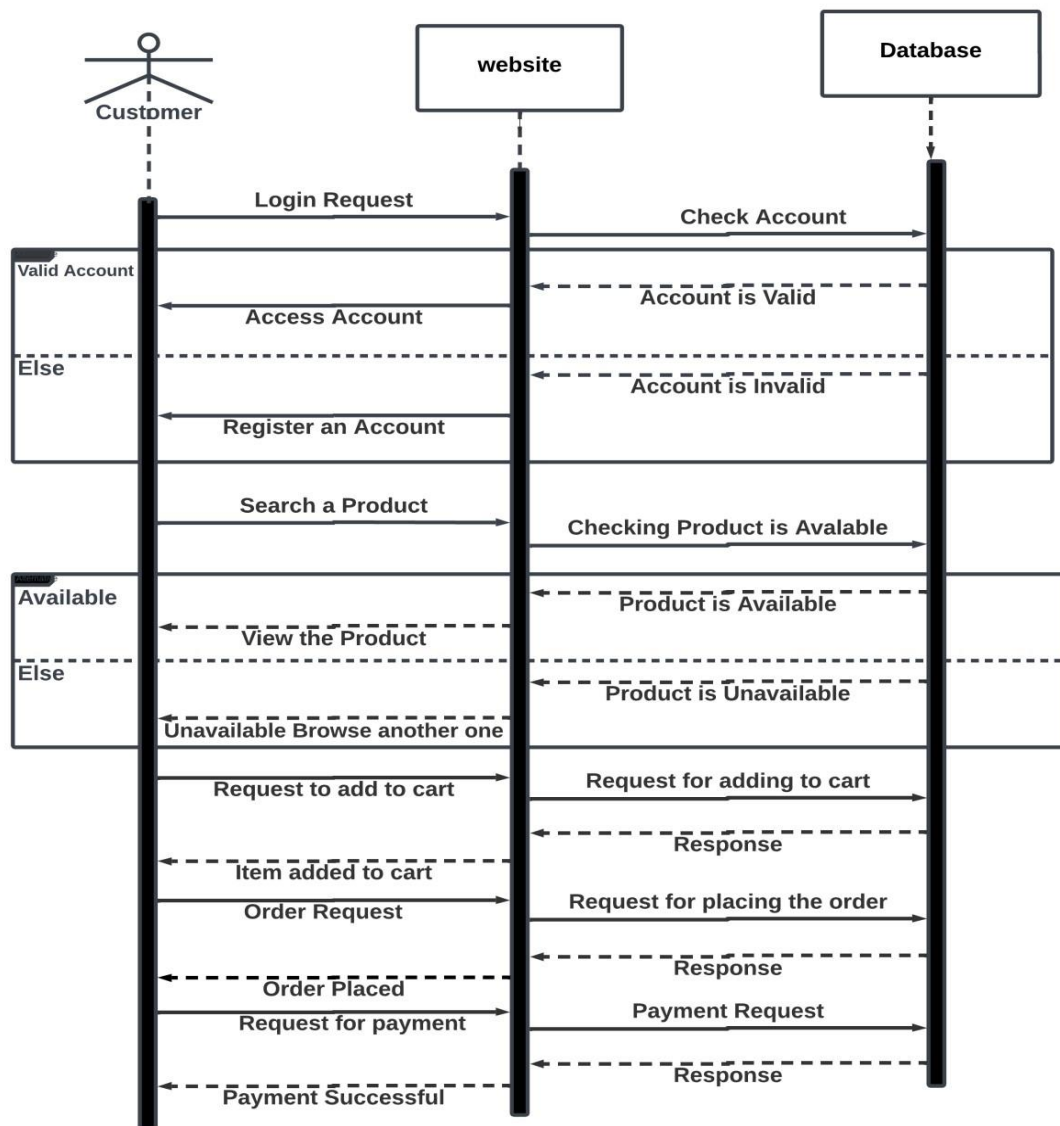


Figure 2: Sequence diagram of Customer

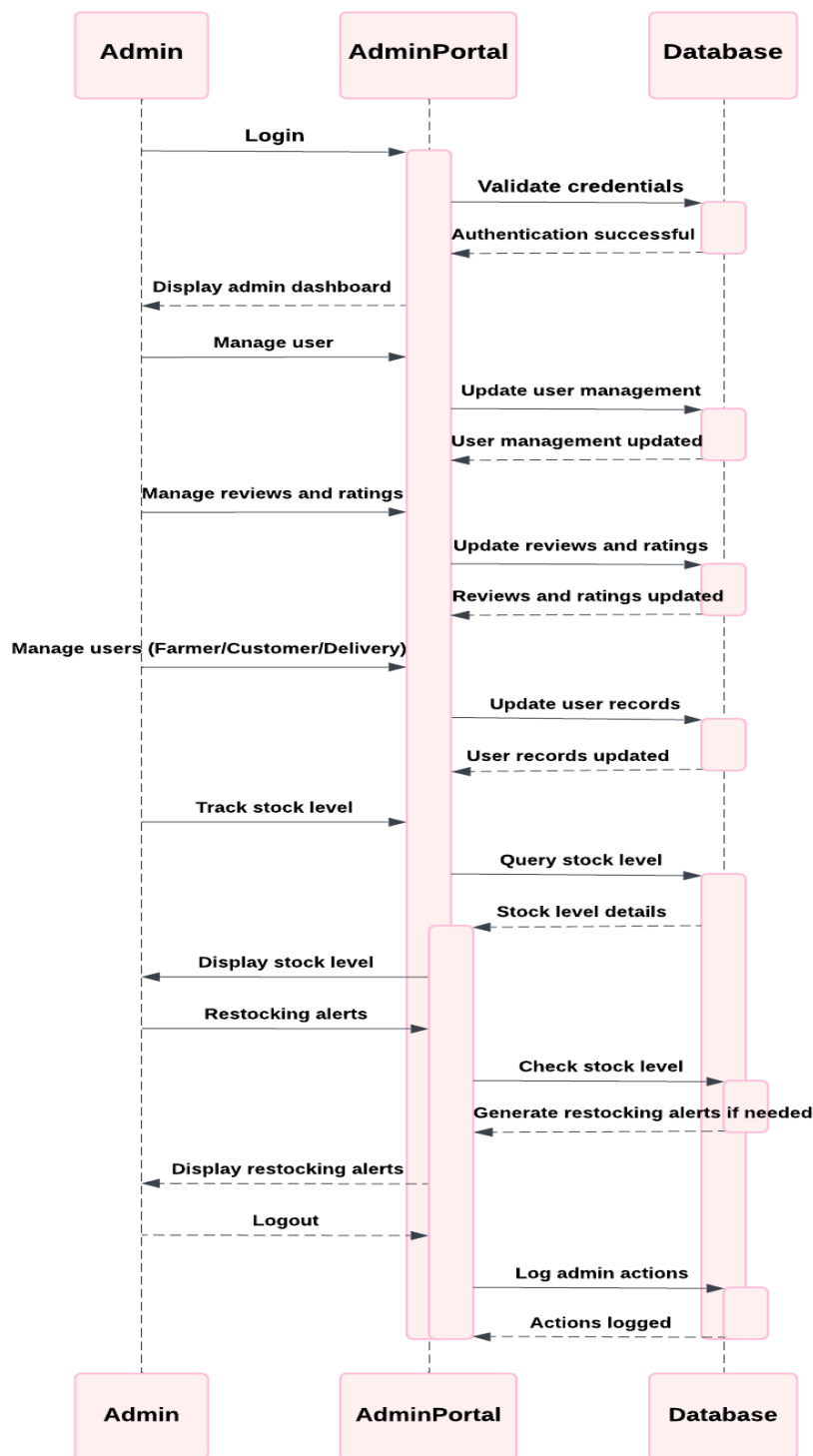


Figure 3: Sequence diagram of Admin

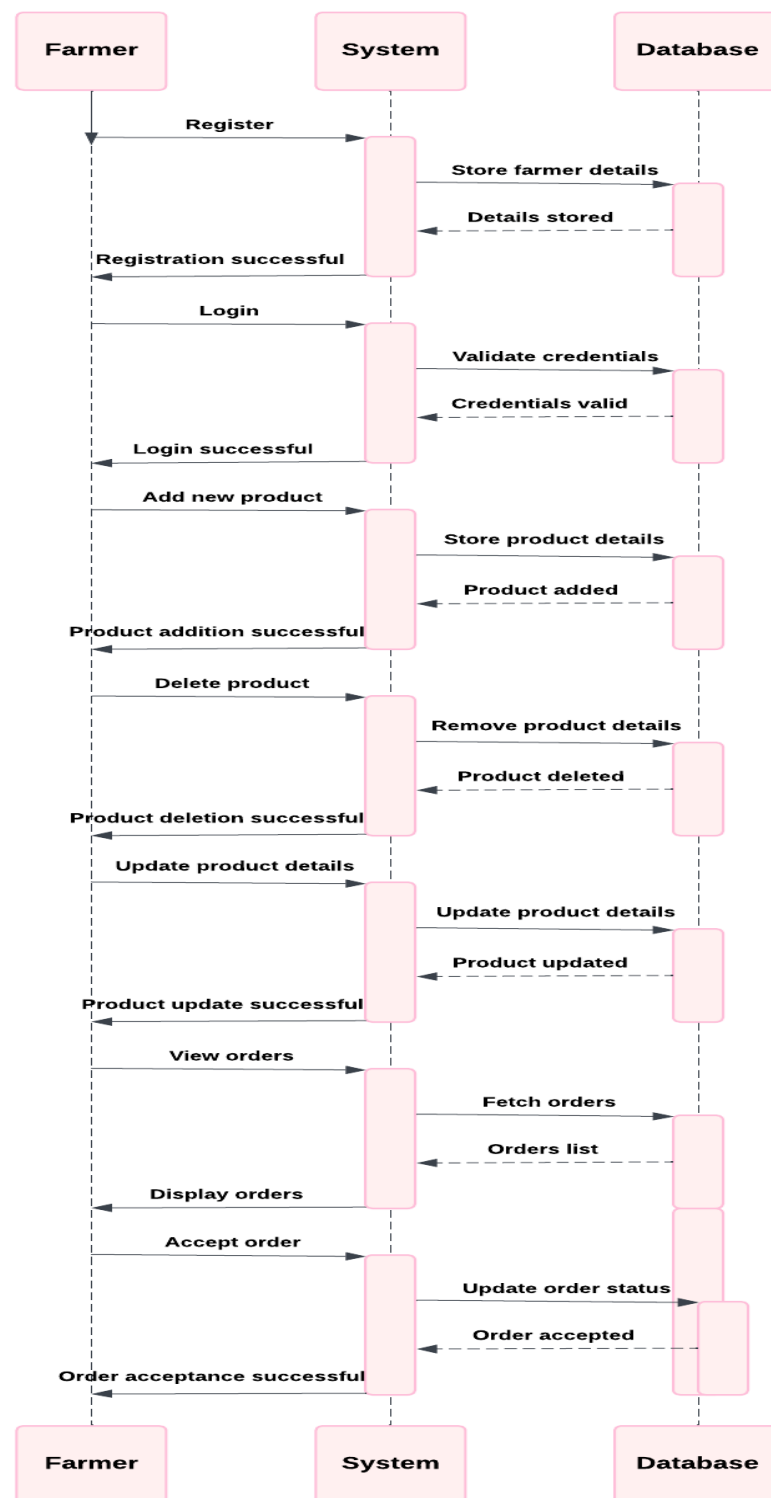
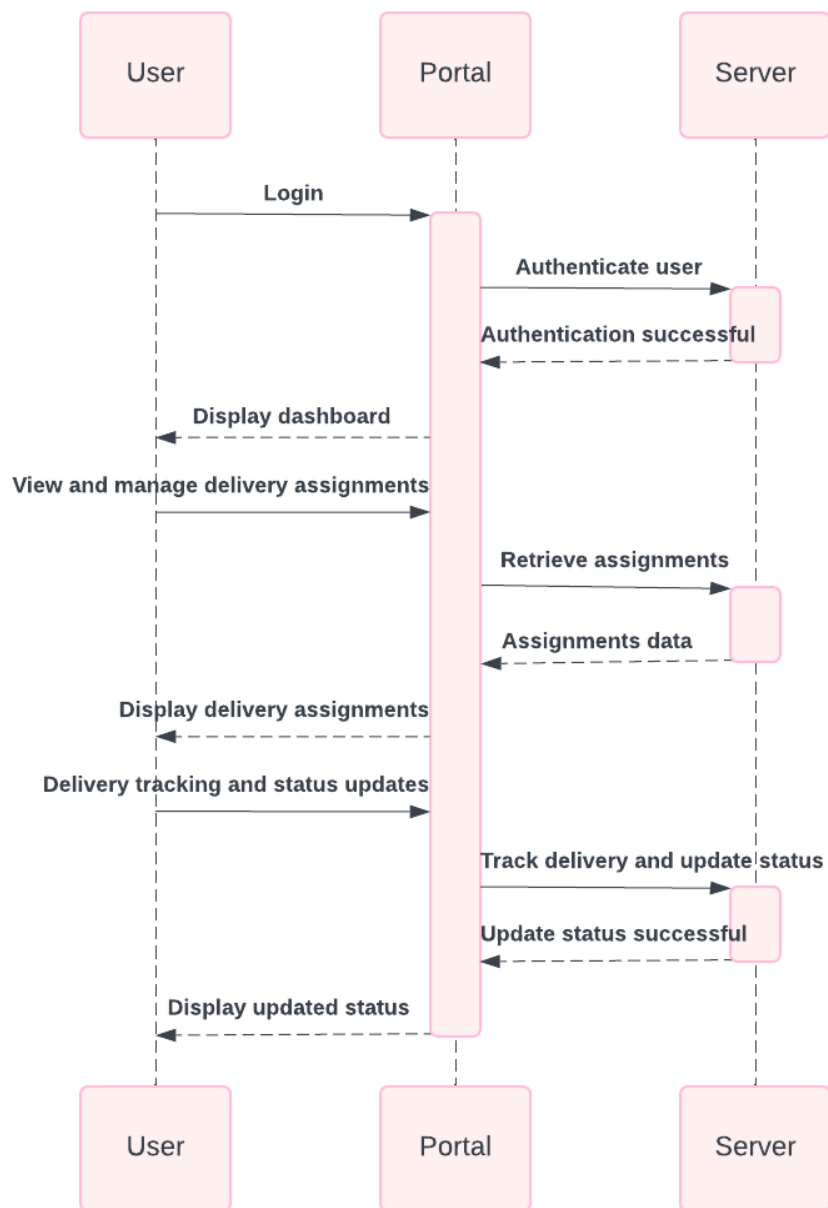


Figure 4: Sequence diagram of Farmer



**Figure 5: Sequence diagram of Delivery Boy**

### 4.2.1 Activity Diagram

An activity diagram is a UML diagram that visualizes the flow of control or workflow in a process or use case. Activities are shown as rounded rectangles connected by arrows indicating their sequence. Additional elements, such as decision points (diamonds) and forks/joins for parallel processing, help depict complex workflows. Activity diagrams are useful for modeling and optimizing business processes, software systems, and workflows by providing a clear view of the steps involved and their order.

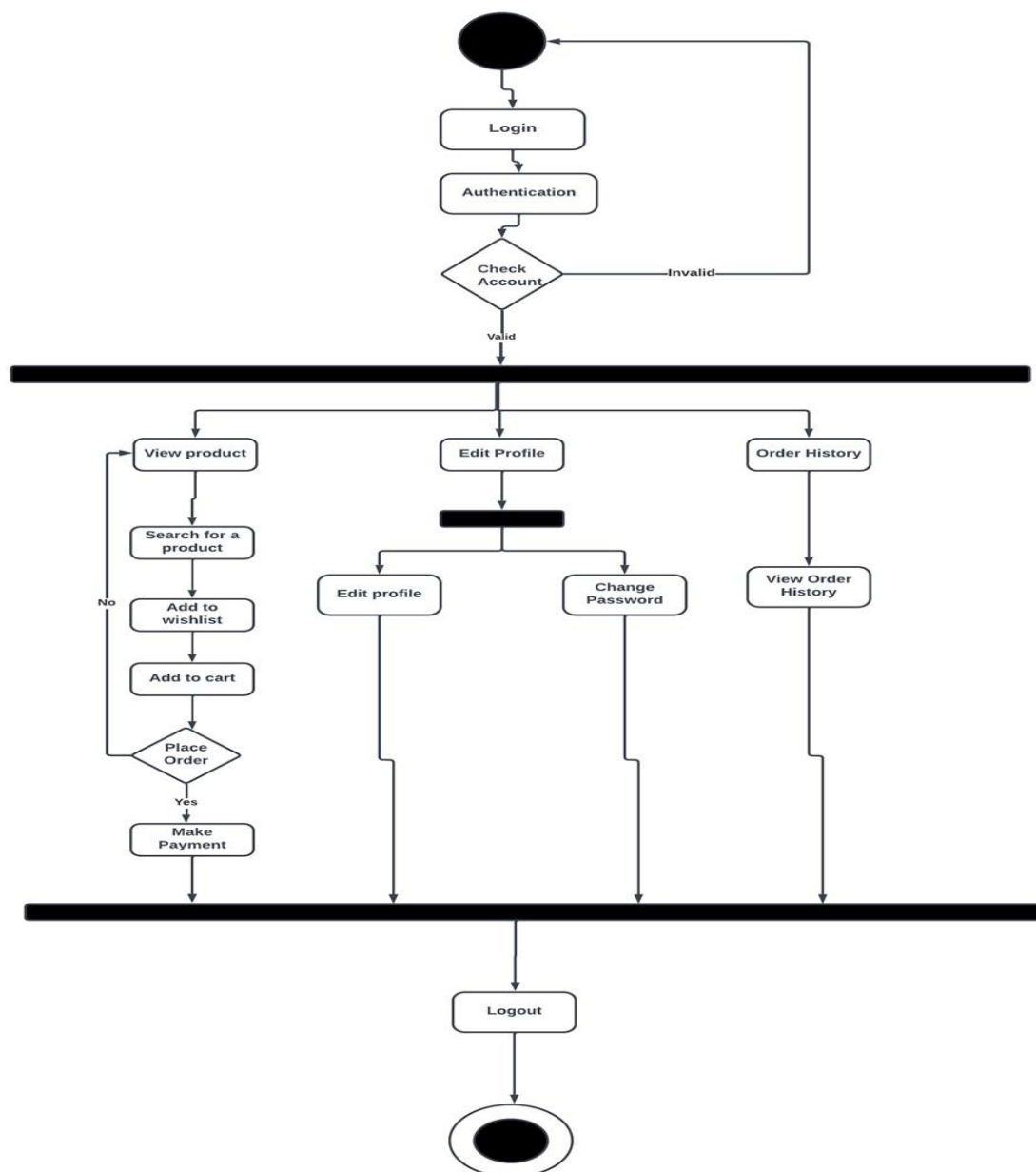
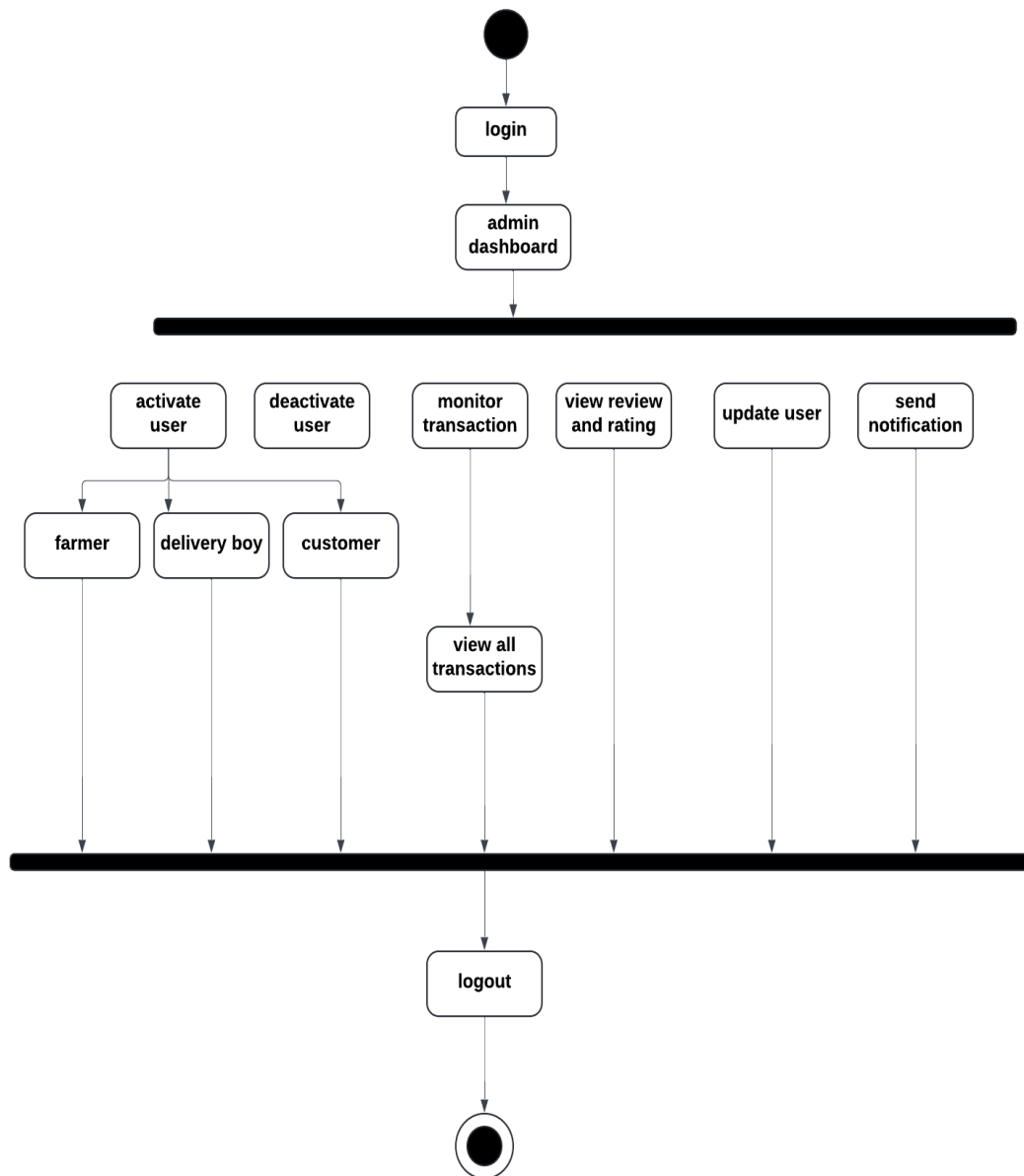
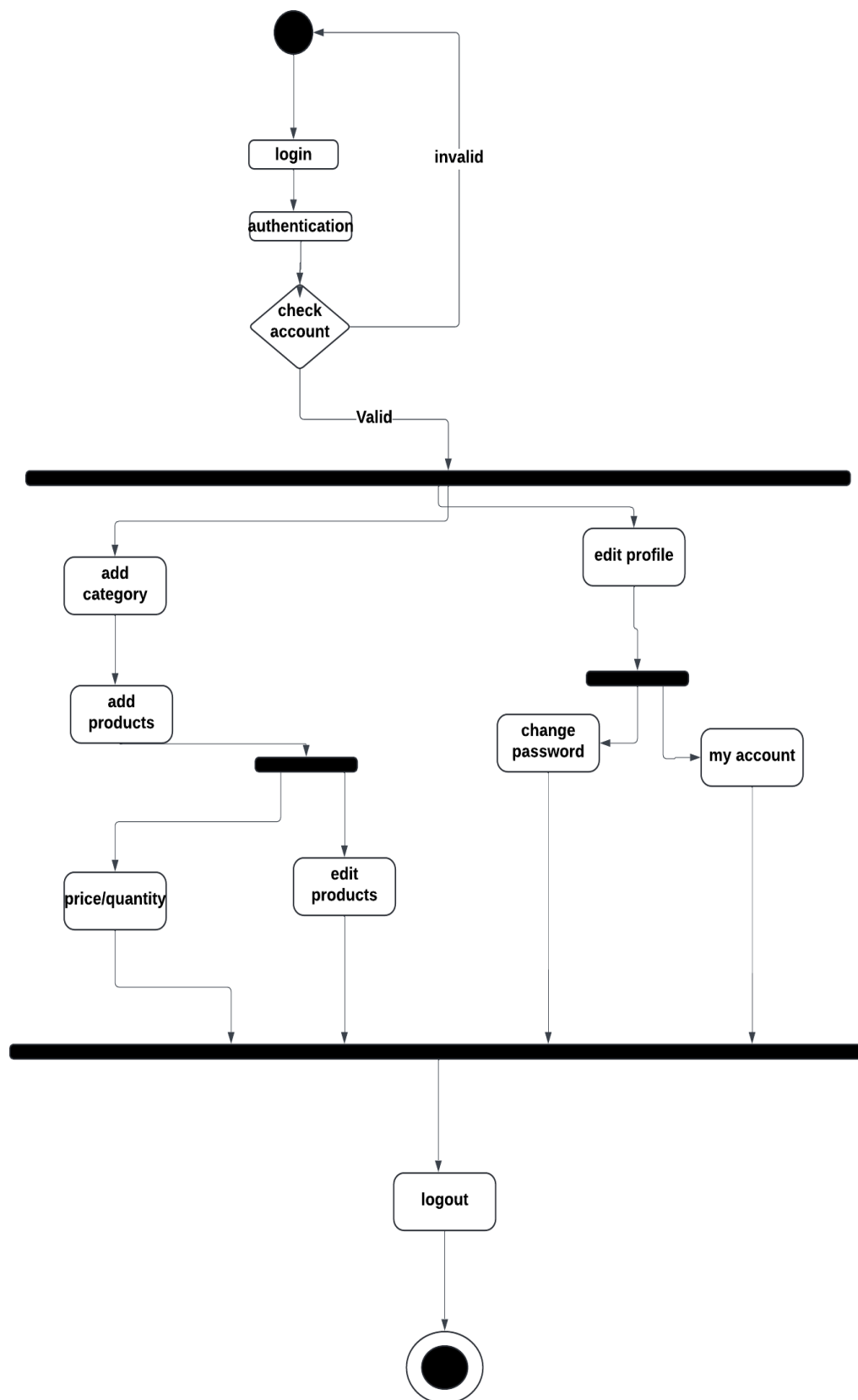
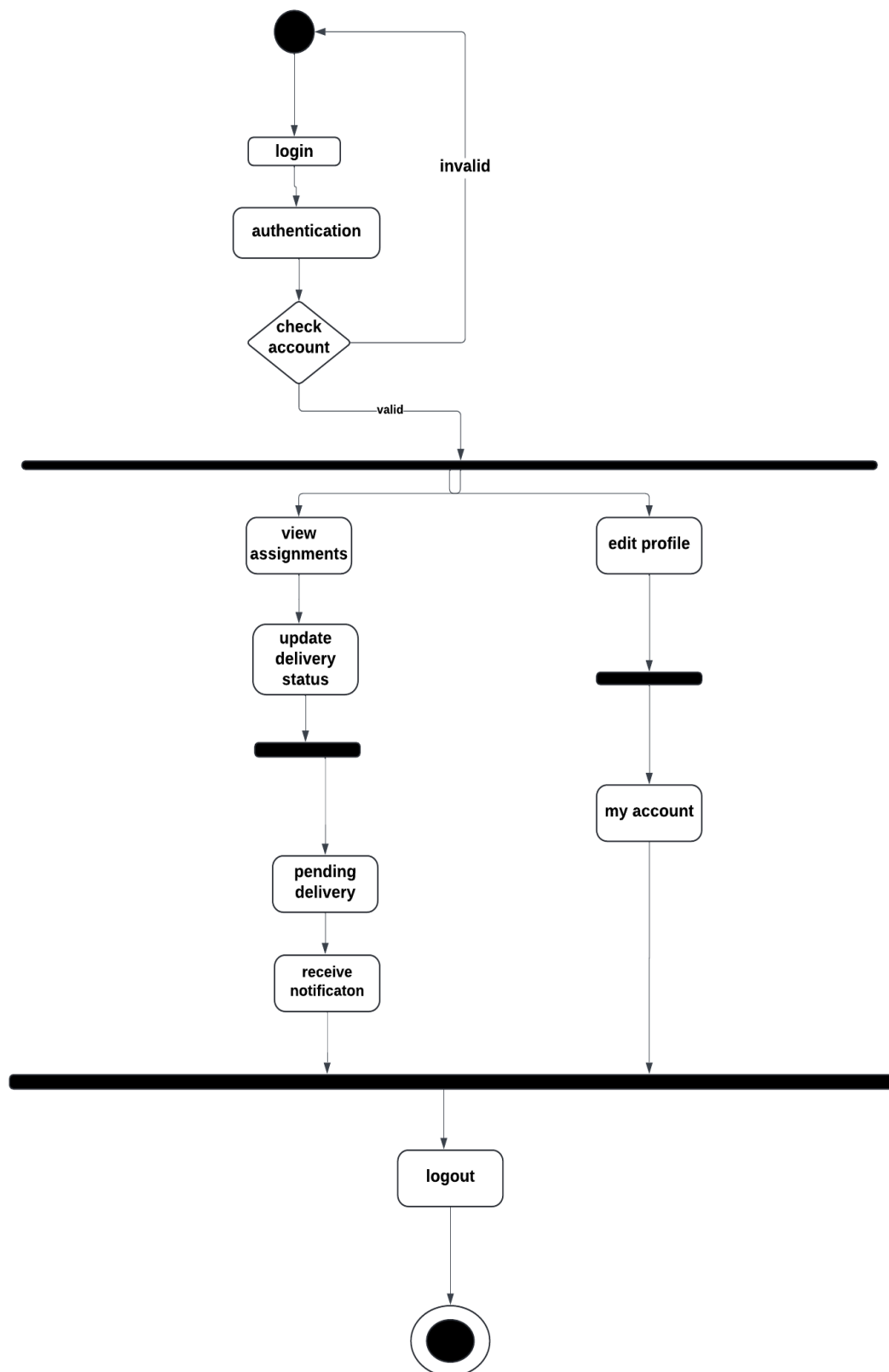


Figure 6: Activity Diagram of Customer



**Figure 7: Activity Diagram of Admin**

**Figure 8: Activity Diagram of Farmer**

**Figure 9: Activity Diagram of Delivery Boy**



### 4.2.2 Class Diagram

A class diagram is a type of UML diagram that shows the static structure of a software system by representing the classes, their attributes, methods, and relationships with other classes. It is used to illustrate the object-oriented design of a system by showing the classes and their relationships with each other.

In a class diagram, classes are represented as rectangles with the class name inside. The class attributes (properties or data members) are listed under the class name, and the class methods (functions or operations) are listed below the attributes. Relationships between classes are represented by lines between the classes, with different types of lines representing different types of relationships.

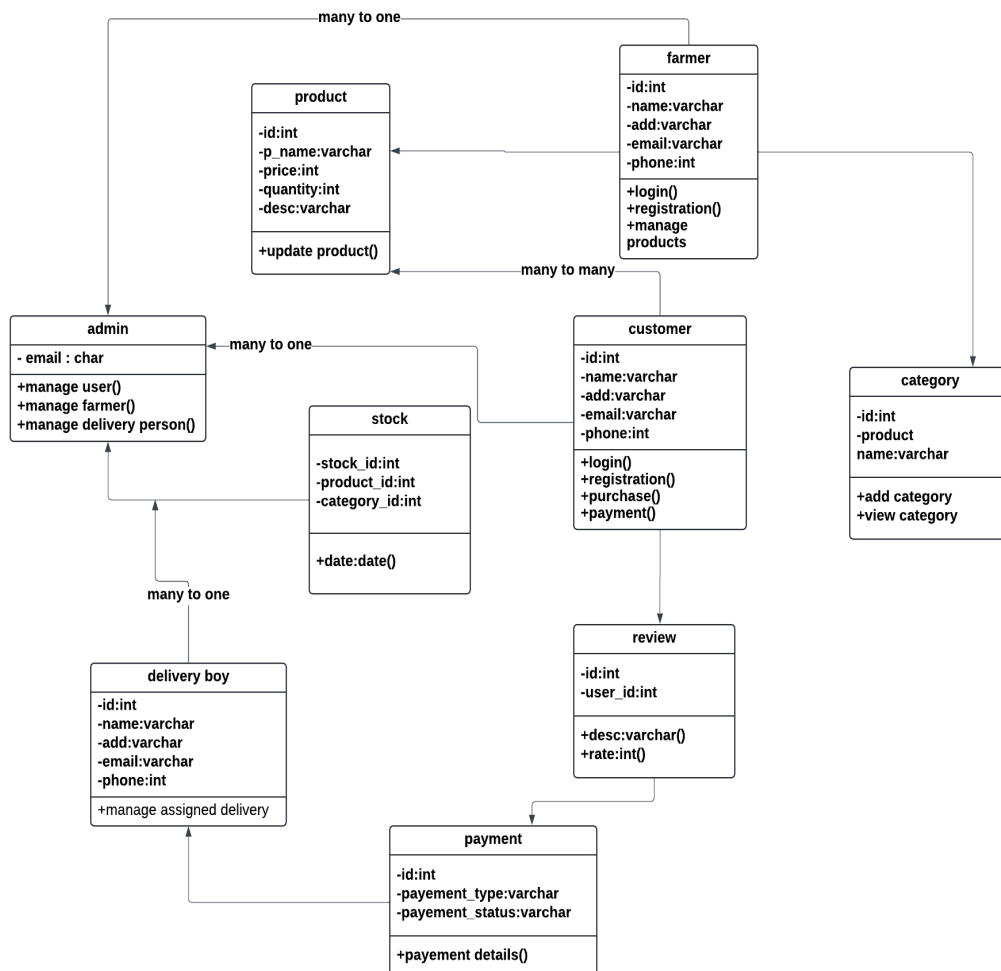


Figure 10: Class Diagram

### 4.2.3 Object Diagram

An object diagram is a UML diagram that represents a snapshot of objects and their relationships in a system at a specific point in time. It shows objects as rectangles with their attributes and values inside, and the relationships between them are depicted using lines. These diagrams are useful for understanding object interactions in specific scenarios or use cases, helping to visualize the organization of objects and their relationships. Object diagrams enhance communication and collaboration among teams by offering a clear view of the system's behavior.

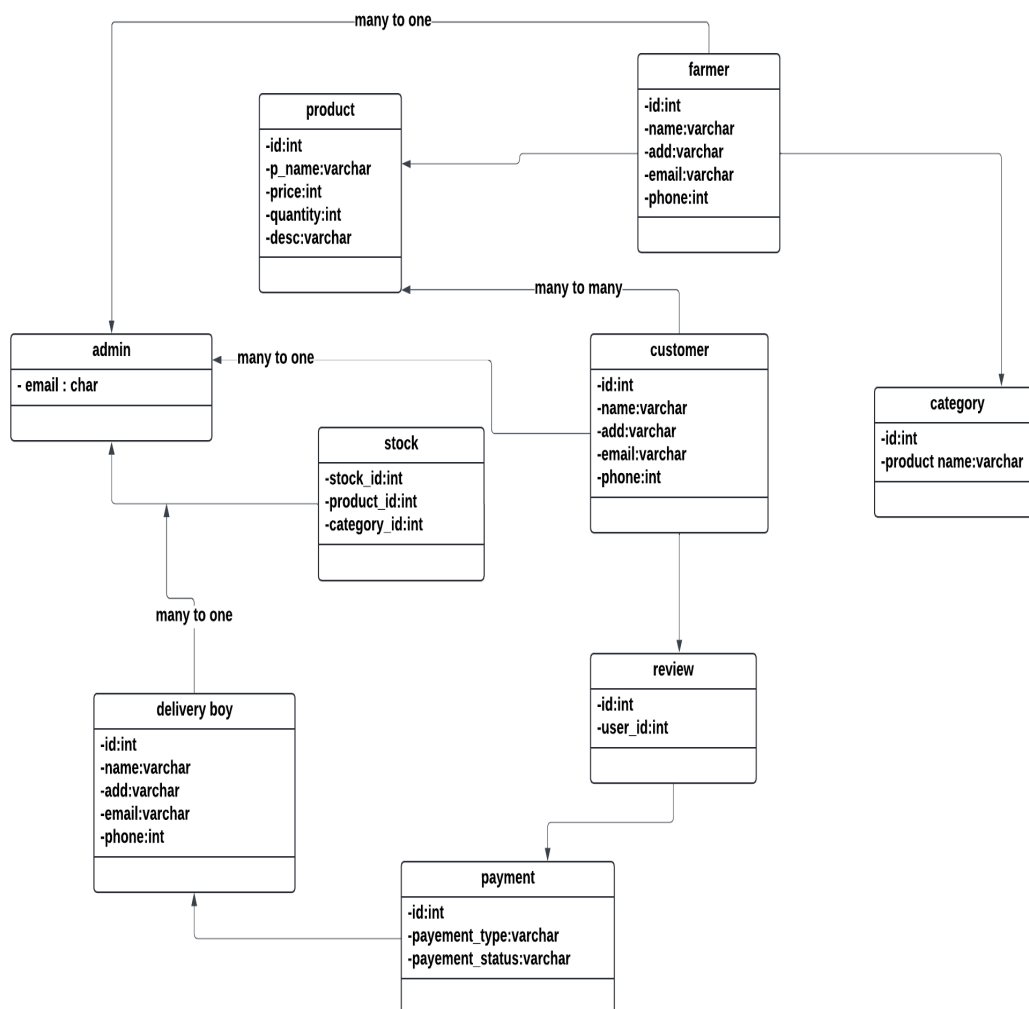
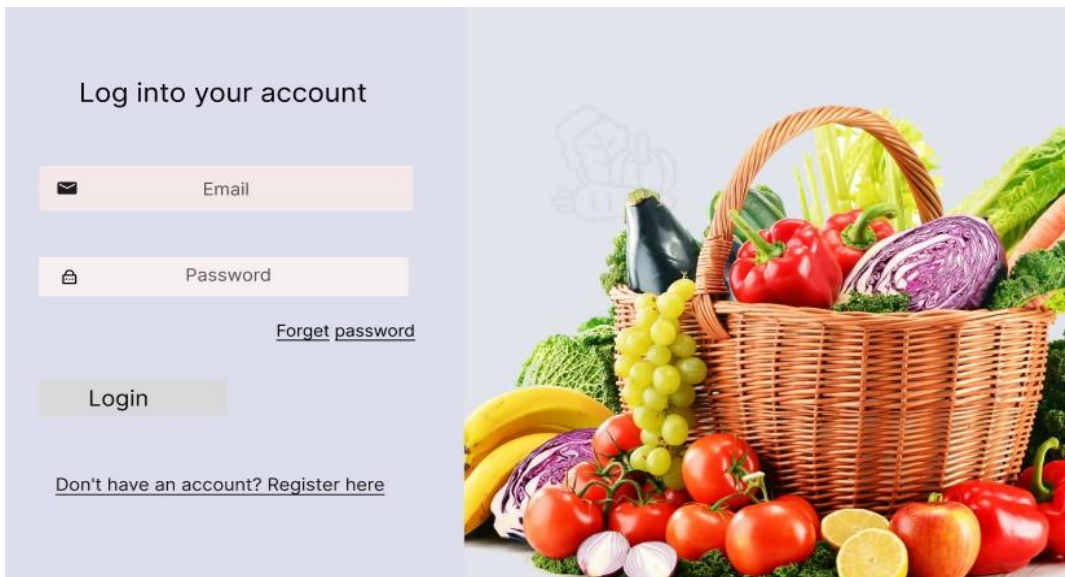


Figure 11: Object diagram

## 4.3 USER INTERFACE DESIGN USING FIGMA

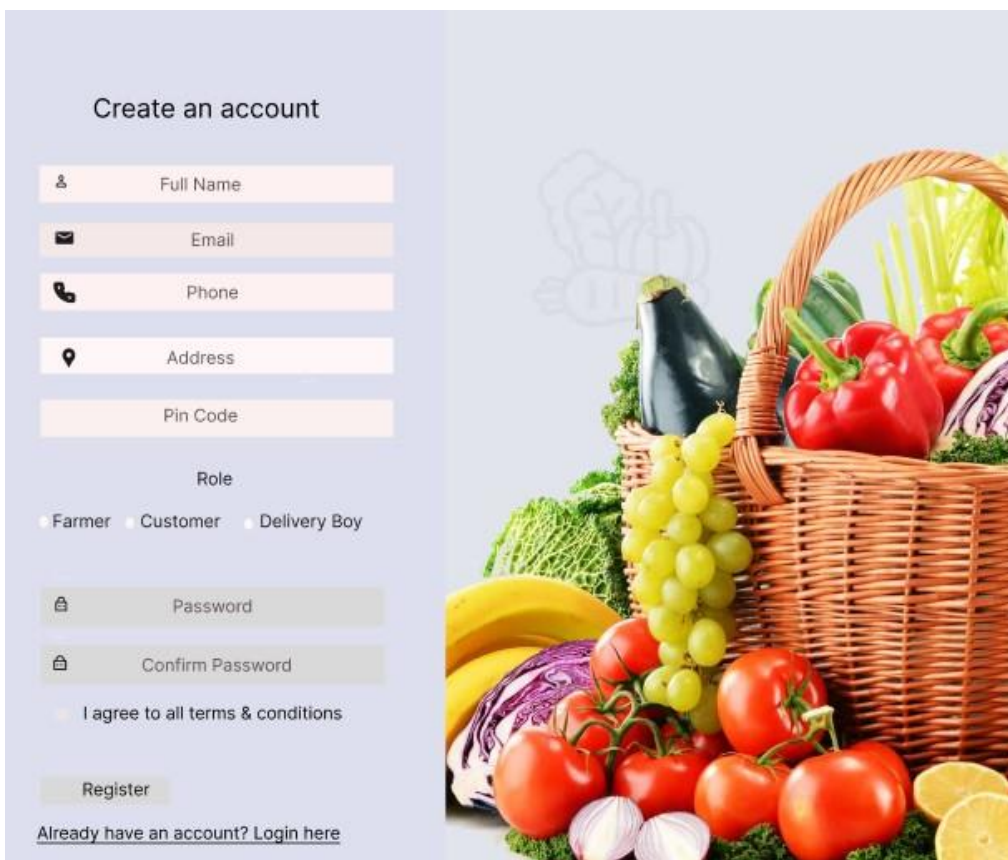
### Form Name: User Login



The image displays a user login interface on the left and a basket of fresh vegetables on the right. The login form is titled "Log into your account" and includes input fields for "Email" (with an envelope icon) and "Password" (with a lock icon). Below the password field is a link for "Forgot password". A "Login" button is positioned below the inputs. At the bottom of the form is a link: "Don't have an account? Register here". The vegetable basket on the right contains a variety of items including tomatoes, grapes, eggplants, and leafy greens.

Figure 12: User Login

### Form Name: Registration Form



The image displays a user registration interface on the left and a basket of fresh vegetables on the right. The registration form is titled "Create an account" and includes input fields for "Full Name" (with a person icon), "Email" (with an envelope icon), "Phone" (with a phone icon), "Address" (with a location pin icon), and "Pin Code". Below these fields is a "Role" section with three radio button options: "Farmer", "Customer", and "Delivery Boy". Further down are "Password" and "Confirm Password" fields (both with lock icons), followed by a checkbox for "I agree to all terms & conditions". A "Register" button is located below the checkbox. At the bottom of the form is a link: "Already have an account? Login here". The vegetable basket on the right is identical to the one in Figure 12, containing tomatoes, grapes, eggplants, and leafy greens.

Figure 13: User Registration Form

## **4.4 DATABASE DESIGN**

Database design is the process of creating a structured data model that defines entities, attributes, and relationships within a database. It aims to ensure the database is efficient, scalable, and easy to maintain. Key steps include defining the schema, selecting data types, and establishing table relationships while minimizing redundancy through normalization. A well-designed database improves data retrieval and management, ensures security, and controls access. Poor design can lead to performance issues and data inconsistencies, making careful planning essential to meet organizational needs effectively.

### **4.4.1 Relational Database Management System (RDBMS)**

A Relational Database Management System (RDBMS) organizes data into structured tables and allows querying and manipulation using SQL. It is widely used in businesses, organizations, and governments to manage large volumes of data. The main advantage of an RDBMS is its ability to ensure data consistency and accuracy by enforcing constraints and relationships between tables. Additionally, it can scale efficiently as data grows, making it suitable for managing increasing data demands in expanding organizations.

### **4.4.2 Normalization**

Normalization is the process of structuring data in a relational database to minimize redundancy and dependency by dividing large tables into smaller, related ones. This improves data consistency, reduces errors, and enhances retrieval. There are various normal forms, including 1NF, 2NF, and 3NF, each with specific criteria for reducing redundancy. While normalization optimizes storage and performance, over-normalizing can negatively impact efficiency, so it's important to balance it with practical data access needs.

#### **First Normal Form**

First Normal Form (1NF) ensures that data in a table is organized in a way that each field contains atomic values, with no repeating groups or arrays, and each row is uniquely identifiable by a primary key. This eliminates redundancy and inconsistencies, enabling efficient querying, updating, and maintenance of the database. To achieve 1NF, tables may need to be decomposed into smaller ones, and relationships are established using primary and foreign keys. Ensuring 1NF is crucial for maintaining data integrity and optimizing database performance.

## **Second Normal Form**

Second Normal Form (2NF) is a concept in relational database design that builds upon the principles of First Normal Form (1NF) by eliminating data redundancy and dependency issues in the database. According to 2NF, a table is in its second normal form if it is in 1NF and every non-key attribute is functionally dependent on the table's primary key. In other words, 2NF ensures that each table column is uniquely determined by the primary key, and there is no partial dependency between columns.

To achieve 2NF, one must identify and separate the columns that depend on a subset of the primary key and group them into a new table with a new primary key. This process is known as decomposition, and it helps eliminate data redundancy and inconsistency issues by breaking down tables into smaller, more manageable units. By decomposing the table into smaller units, it is possible to achieve higher levels of normalization, leading to more efficient and reliable database operations.

## **Third Normal Form**

Third Normal Form (3NF) enhances relational database design by eliminating data redundancy and transitive dependency issues, building on the principles of First Normal Form (1NF) and Second Normal Form (2NF). A table is in 3NF if it is already in 2NF and all non-key attributes depend solely on the primary key, not on other non-key attributes. Achieving 3NF involves identifying and removing transitive dependencies, often by decomposing the table into smaller units based on functional dependencies and establishing relationships through foreign keys. This process improves data integrity, consistency, and accuracy, while simplifying database design for more efficient operations and easier maintenance. Overall, 3NF is essential for creating a reliable and well-structured database schema.

## **Boyce Codd Normal Form**

Boyce-Codd Normal Form (BCNF) is an advanced database normalization technique that ensures tables are well-structured and free from anomalies. It is a stricter version of Third Normal Form (3NF) and is achieved by addressing functional dependencies that violate BCNF principles. A table is considered to be in BCNF if every determinant in the table is a candidate key, meaning there are no non-trivial dependencies between attributes, and there are no non-trivial functional dependencies between attributes that are not part of any candidate key. If a table fails to meet these criteria, it can be decomposed into smaller tables that satisfy BCNF. This decomposition helps eliminate redundancies and anomalies;

## **Fourth Normal Form**

Fourth Normal Form (4NF) is an advanced level of database normalization that addresses multi-valued dependencies. Dependency arises when non-key attributes are not fully

dependent on the primary key but instead depend on combinations of other non-key attributes.

To achieve 4NF, tables containing multi-valued dependencies are decomposed into smaller tables, each representing a single multi-valued dependency. While 4NF may not be necessary for every database, and decomposition might not always be feasible due to performance concerns, it is beneficial in scenarios where multi-valued dependencies exist, enhancing data quality and consistency.

#### **4.4.3 Sanitization**

Sanitization, in the context of ensuring that no residual data can be recovered, is known as secure data wiping or secure data erasure. This process involves permanently deleting data from a storage device to prevent recovery through advanced forensic techniques.

Various methods, such as overwriting data with random patterns multiple times or physically destroying the storage device, are used depending on the data's sensitivity and the required security level. Secure data wiping is crucial when dealing with sensitive information, especially during hardware disposal, device ownership transfers, or business closures. Inadequate data sanitization can lead to unintentional exposure of sensitive information, resulting in significant legal, financial, or reputational risks.

#### **4.4.4 Indexing**

In SQL, indexing is the process of creating data structures that allow for faster retrieval of data from a database. Indexes are essentially lists of keys that point to specific locations within a table, making it possible to quickly locate and retrieve data that matches specific search criteria.

Creating indexes in SQL can significantly improve query performance, especially when working with large databases. By creating indexes on frequently accessed columns, such as primary keys or foreign keys, SQL can quickly locate the data that matches specific search criteria, without having to scan the entire table.

It's important to note that indexing can also have a negative impact on performance if done incorrectly. Over-indexing, or creating too many indexes, can slow down the insertion and updating of data, as well as increase the size of the database. Therefore, it's important to carefully consider the columns to index and the type of index to use to ensure that the indexing process does not adversely affect overall database performance.

## 4.5 TABLE DESIGN

### 1. Tbl\_ Registration table

Primary Key: user\_id

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	user_id	Int	PRIMARY KEY	Unique user ID
2	username	varchar (50)	UNIQUE, NOT NULL	Username
3	password	varchar (50)	NOT NULL	Password
4	email	varchar (50)	UNIQUE, NOT NULL	Email
5	address	Text		Customer address
6	role	varchar (20)	CHECK(role IN ('customer', 'supplier', 'admin', 'delivery')), NOT NULL	User,role(customer, supplier,admin, delivery)

### 2. Tbl\_ Customer login table

Primary Key: customer\_id

Foreign Key: user\_id references Registration(user\_id)

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	customer_id	Int	PRIMARY KEY	Primary key
2	User_id	varchar (50)	FOREIGN KEY REFERENCES	Foreign key referencing the registration table

### 3. Tbl\_ Supplier table

Primary Key: supplier\_id

Foreign Key: user\_id references Registration(user\_id)

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Supplier_id	Int	PRIMARY KEY	Primary key
2	User_id	varchar (50)	FOREIGN KEY REFERENCES	Foreign key referencing the registration table

**4. Tbl\_ Delivery person table**

Primary Key: delivery\_person\_id

Foreign Key: user\_id references Registration(user\_id)

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Deliveryboy_id	Int	PRIMARY KEY	Primary key
2	User_id	varchar (50)	FOREIGN KEY REFERENCES	Foreign key referencing the registration table

**5. Tbl\_ Category table**

Primary Key: category\_id

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	category_Id	int	PRIMARY KEY	Primary Key
2	Category name	Varchar	NOT NULL	Name of category

**6. Tbl\_ ProductCategory Table**

Primary Key: Subcategorycategory\_id

Foreign key: Cayegory\_id

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	sub_Id	int	PRIMARY KEY	Primary Key
2	category_Id	int	FOREIGN KEY REFERENCES	Foreign key from category table
3	subcategory name	varchar	NOT NULL	Name of sub category
4	product_id	int	FOREIGN KEY REFERENCES	Foreign key from product table



**7. Tbl\_ Product\_Farmer**

Primary Key: product\_id

Foreign Keys: category\_id references Category(category\_id), supplier\_id references

Supplier(supplier\_id)

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	product_id	int	PRIMARY KEY	Unique product ID
2	sub_Id	int	OREIGN KEY REFERENCES	Foreign key from subcategory table
3	Category_id	int	FOREIGN KEY REFERENCES	Foreign key from category table
4	Product name	varchar(100)	NOT NULL	Product name
5	description	text		Product description
6	image	varchar		Product image
7	price	int	NOT NULL	Product price
8	stock_level	int	DEFAULT	Current stock level
9	supplier_id	int	FOREIGN KEY REFERENCES	Foreign key referencing the Suppliers table

**8. Tbl\_ Product table**

Primary Key: product\_id

Foreign Keys: category\_id references Category(category\_id), supplier\_id references

Supplier(supplier\_id)

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	product_id	int	FOREIGN KEY REFERENCES	Foreign key from the product table
2	Category_id	int	FOREIGN KEY REFERENCES	Foreign key from the category table
3	Product name	varchar(100)	NOT NULL	Name of the product

**9. Tbl\_ Order table**

Primary Key: order\_id

Foreign Keys: customer\_id references Customer(customer\_id), product\_id references

Product(product\_id)

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	order_id	int	PRIMARY KEY	Unique order ID
2	customer_id	int	FOREIGN KEY REFERENCES	Foreign key referencing the Customers table
3	order_date	datetime	NOT NULL	Order date and time
4	total_amount	int	NOT NULL	Total order amount
5	status	varchar(20)	Default	Order status
6	product_id	int	FOREIGN KEY REFERENCES	Foreign key referencing the Products table
7	quantity	int	NOT NULL	Quantity of the product in the order

**10. Tbl\_ Price chart table**

Primary Key: price\_chart\_id

Foreign Keys: category\_id references category(category\_id), product\_id references

Productcategory(product\_id)

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Price_chart_id	int	PRIMARY KEY	Unique order ID
2	category_id	int	FOREIGN KEY REFERENCES	Foreign key referencing the Category table
3	product_id	int	FOREIGN KEY REFERENCES	Foreignkey referencing the product_category table
4	quantity	int	NOT NULL	quantity
5	price	int	NOT NULL	price
6	date	int	NOT NULL	date

**11. Tbl\_ Delivery assignment table**

Primary Key: delivery\_id

Foreign Keys: order\_id references Order(order\_id), delivery\_person\_id references

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	delivery_id	int	PRIMARY KEY	Unique delivery ID
2	order_id	int	FOREIGN KEY REFERENCES	Foreign key referencing the Orders table
3	delivery_date	datetime	NOT NULL	Scheduled delivery date and time
4	status	varchar(20)		Delivery status
5	delivery_person_id	int	FOREIGN KEY REFERENCES	Foreign key referencing the Delivery Persons table

**12. Tbl\_ Payment table**

Primary Key: payment\_id

Foreign Key: order\_id references Order(order\_id)

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	payment_id	int	PRIMARY KEY	Unique payment ID
2	order_id	int	FOREIGN KEY REFERENCES	Foreign key referencing the Orders table
3	payment_method	varchar(20)	NOT NULL	Payment method

**13. Tbl\_ Wishlist**

Primary key:wishlist\_id

Foreign key: product\_id references Product(product\_id), user\_id references Registration(user\_id)

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Wishlist_id	int	PRIMARY KEY	Unique id
2	User_id	int	FOREIGN KEY REFERENCES	Foreign key of user table
3	Product_id	int	FOREIGN KEY REFERENCES	Foreign key of product table

## **CHAPTER 5**

## **SYSTEM TESTING**

## 5.1 INTRODUCTION

System testing is a type of software testing that focuses on evaluating the overall performance and functionality of a software system. It involves testing the complete system, including all its components and subsystems, to ensure that it meets the specified requirements and performs as expected.

The goal of system testing is to identify defects and errors that may have been missed during unit testing and integration testing. It also verifies that the system meets the functional, performance, and security requirements specified in the project documentation

## 5.2 TEST PLAN

A test plan in system testing typically involves testing the entire software system as a whole, including all its features and functions. The objective of system testing is to verify that the software meets its specified requirements and is functioning as expected in its intended environment. The test plan typically includes details such as the testing environment, the test strategy, the test objectives, the test schedule, and the test team responsibilities. It may also include the testing techniques, tools, and metrics to be used. The test plan should be reviewed and approved by all stakeholders, including the project manager, development team, and testing team, to ensure that everyone is on the same page regarding the testing approach.

The test plan should also provide information on the different levels of testing. These typically include:

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing

### 5.2.1 Unit Testing

Unit testing is a type of testing that is focused on verifying the functionality of individual code units or modules. A well-designed test plan for unit testing should outline the specific objectives of unit testing, the approach to be taken, and the resources that will be required to carry out the testing. Unit testing is typically carried out by the development team and is an essential part of the software development lifecycle.

Unit testing is an important part of the overall testing process, as it can help to identify defects early in the software development lifecycle when they are easier and less expensive

to fix. By creating a detailed test plan for unit testing, the development team can ensure that all code units have been thoroughly tested and that the software meets the desired quality standards. This can ultimately help to improve the reliability, performance, and maintainability of the software

### **5.2.2 Integration Testing**

Integration testing is a type of testing that focuses on testing the interaction between different modules or components of the software system to ensure that they are working together as expected. In system testing, integration testing is typically performed after unit testing, and it is focused on testing the entire system as a whole. The goal of integration testing is to detect and resolve any issues that arise from the interaction between different modules before the system is released to users.

Integration testing can help to identify defects that may not have been caught during unit testing, as it focuses on the interaction between modules rather than on individual code units. By testing the software system as a whole, integration testing can help to ensure that the system meets the desired quality standards, performs as expected, and meets the needs of its users.

### **5.2.3 Validation Testing or System Testing**

Validation testing, also known as system testing, is a critical part of software testing that is conducted to evaluate the entire system or application under test. The goal of this type of testing is to validate that the software system meets its intended requirements and works as expected.

Validation testing involves a series of tests designed to check the software system's functionality, reliability, usability, performance, and security. It is typically performed after integration testing and prior to user acceptance testing. The objective of system testing is to ensure that the software system meets the customer's expectations and requirements before it is released to production.

### **5.2.4 Output Testing or User Acceptance Testing**

Output testing, commonly known as user acceptance testing (UAT), is a crucial phase in the software development life cycle that ensures the software system meets end-user needs and expectations. The primary goal of UAT is to validate that the software is ready for release and complies with all specified functional and non-functional requirements.

Typically conducted by end-users, business stakeholders, or subject matter experts, UAT focuses on assessing the system's functionality, usability, and overall user experience. Test cases are designed to simulate real-world scenarios, verifying that the system behaves as expected across various use cases.

UAT plays a vital role in identifying any issues or defects that may have been overlooked in earlier testing stages. By performing user acceptance testing, organizations can reduce the risk of user dissatisfaction, enhance the overall user experience, and improve customer satisfaction with the software system before its release.

### **5.2.5 Automation Testing**

Automation testing is the process of using software tools to execute pre-scripted tests on a software application or system. The goal of automation testing is to improve the efficiency and accuracy of the testing process while reducing the time and cost involved in manual testing. Automation testing can be used for functional testing, regression testing, performance testing, and other types of software testing.

One of the main benefits of automation testing is that it can help to reduce the time and effort required for repetitive testing tasks. Automated tests can be executed more quickly and consistently than manual tests, and they can be run repeatedly without the need for human intervention. Automation testing can also help to identify defects and issues more quickly and accurately than manual testing, as it can perform a large number of tests in a short period of time.

### **5.2.6 Selenium Testing**

Selenium testing is an automated testing framework specifically designed for validating web applications, ensuring that they function correctly across various browsers and platforms. As an open-source tool, Selenium provides a variety of libraries and tools suitable for functional, regression, and load testing of web applications. One of its key features is cross-browser testing, which allows for testing across multiple browsers, including Chrome, Firefox, Internet Explorer, and Safari, ensuring consistent functionality. This framework emphasizes automated testing, enhancing testing efficiency by reducing the time and effort required while also improving accuracy and consistency in test results. Additionally, Selenium supports a record-and-playback feature, enabling testers to capture their interactions with a web application to generate reusable test scripts. Test scripts can be written in various programming languages, including Java, Python, C#, and Ruby, giving flexibility to testers based on their expertise. Furthermore, Selenium can seamlessly integrate with other testing

tools and frameworks like Jenkins, Maven, and TestNG, allowing for the establishment of comprehensive testing pipelines that include automated testing, continuous integration, and reporting. By leveraging these features, Selenium testing significantly enhances the testing process for web applications, ensuring quality and reliability.

## Test Case 1

### Code

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time

def test_login(driver, role, email, password, dashboard_title):
    try:
        driver.get("http://127.0.0.1:8000/login/")
        email_input = WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.NAME, "email"))
        )
        time.sleep(1)
        email_input.clear()
        email_input.send_keys(email)
        password_input = WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.NAME, "password"))
        )
        time.sleep(1)
        password_input.clear()
        password_input.send_keys(password)
        password_input.send_keys(Keys.RETURN)
        WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.ID, "logout"))
        )
        print(f"{role.capitalize()} login successful!")
        logout_button = driver.find_element(By.ID, "logout")
        logout_button.click()
        WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.NAME, "email"))
        )
        print(f"{role.capitalize()} logout successful!")
    except Exception as e:
        print(f"An error occurred during {role} login: {e}")

def test_invalid_login(driver):
    try:
```



```

    driver.get("http://127.0.0.1:8000/login/")
    email_input = WebDriverWait(driver, 10).until(
        EC.visibility_of_element_located((By.NAME, "email")))
    )
    password_input = driver.find_element(By.NAME, "password")
    time.sleep(1)
    email_input.clear()
    email_input.send_keys("invalid@example.com") # Invalid email
    time.sleep(1)
    password_input.clear()
    password_input.send_keys("InvalidPass123") # Invalid password
    password_input.send_keys(Keys.RETURN)
    WebDriverWait(driver, 10).until(
        EC.visibility_of_element_located((By.ID, "error"))) # Adjust the selector if needed
    )
    print("Invalid login attempt handled correctly.")
except Exception as e:
    print(f"An error occurred during invalid login attempt: {e}")
def run_tests():
    driver = webdriver.Edge() # Initialize Edge WebDriver (replace with your driver if needed)
    try:
        test_login(driver, "customer", "grocery18900@gmail.com", "Sree@123", "Customer Dashboard")
        test_login(driver, "farmer", "sreelakshmips2025@mca.ajce.in", "Farmer@123", "Farmer Dashboard")
        test_login(driver, "admin", "admin@gmail.com", "Admin@123", "Admin Dashboard")
        test_invalid_login(driver)
        print("All tests passed successfully!")
    except Exception as e:
        print(f"Test suite failed: {e}")
    finally:
        driver.quit()
if __name__ == "__main__":
    run_tests()

```

## Result

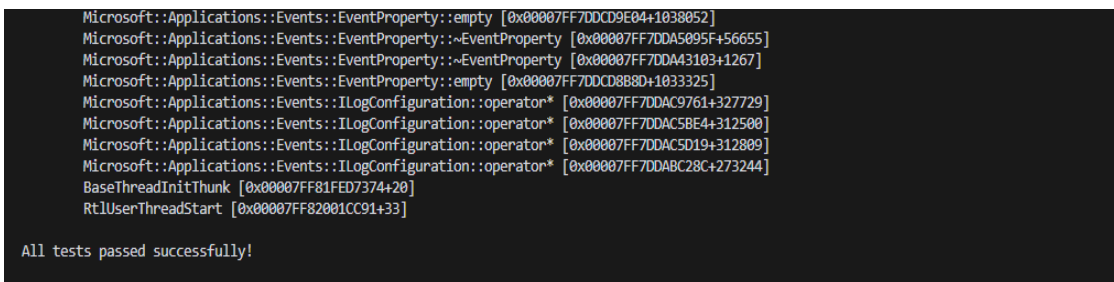


Figure 14: Result of login page

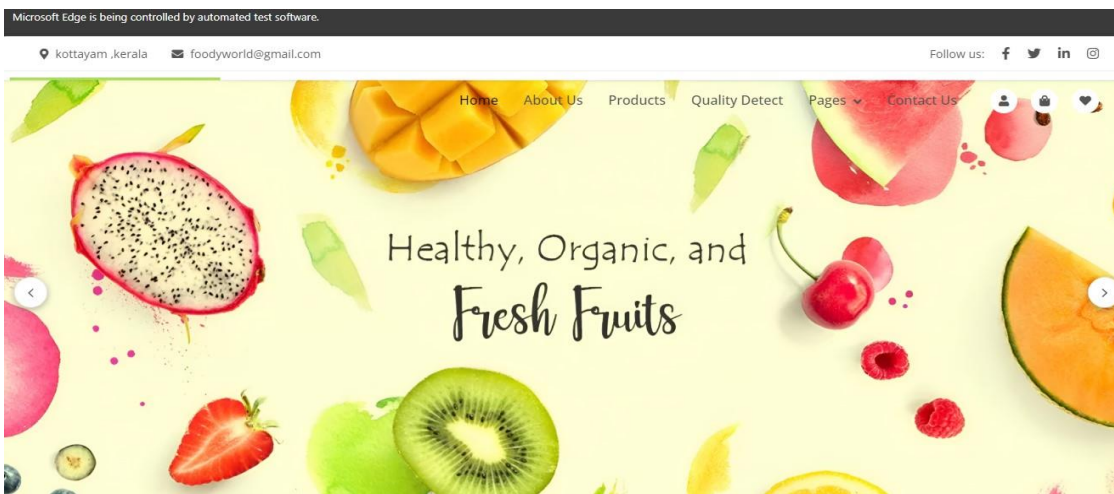


Figure 15: Dashboard

## Test Report

Test Case 1					
Project Name:Green Grocery					
Login Test Case					
Test Case ID: Test_1			Test Designed By: Sreelakshmi PS		
Test Priority(Low/Medium/High):			Test Designed Date: 17-10-2024		
Module Name: Login Page			Test Executed By: Sreelakshmi P S		
Test Title : Login			Test Execution Date: 17-10-2024		
Description: Verify Login with valid email and password					
Pre-Condition :User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/ Fai l)
1	Navigate to login page		Display Login page	Login page displayed	pass
2	Valid email entered	Email : Sreelakshmips2025@mca.ajce.in	User should be able to login	User logged in and navigate to the dashboard	pass
3	Valid password	Sree@123			
4	Click on to submit button				
Post-Condition:email and password is checked with database values for successful login					

## Test Case 2:

### Code

```

from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import Select # For dropdowns
from selenium.webdriver.common.keys import Keys
import time

def add_subcategory(driver, email, password, subcategory_name, category_name, product_category_name):
    driver.get("http://127.0.0.1:8000/login/")
    try:
        email_input = WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.NAME, "email"))
        )
        email_input.send_keys(email)
        time.sleep(1)
        password_input = WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.NAME, "password"))
        )
        password_input.send_keys(password)
        time.sleep(1)
        password_input.send_keys(Keys.RETURN)
        time.sleep(2)
        print("Login successful!")
        driver.get("http://127.0.0.1:8000/add_subcategory/")
        WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.ID, "subcategory_name"))
        )
        subcategory_input = driver.find_element(By.ID, "subcategory_name")
        subcategory_input.send_keys(subcategory_name)
        time.sleep(1)
        print(f"Subcategory name '{subcategory_name}' entered.")
        category_dropdown = Select(driver.find_element(By.ID, "category"))
        category_dropdown.select_by_visible_text(category_name)
        time.sleep(1)
        print(f"Category '{category_name}' selected.")

        product_category_dropdown = Select(driver.find_element(By.ID, "product_category"))
        product_category_dropdown.select_by_visible_text(product_category_name)
        time.sleep(1)
        print(f"Product Category '{product_category_name}' selected.")
        submit_button = driver.find_element(By.XPATH, "//button[@type='submit']")
        submit_button.click()
        time.sleep(2)
        WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.CLASS_NAME, "alert-success"))
        )
        print(f"Subcategory '{subcategory_name}' added successfully.")
    except Exception as e:
        print(f"An error occurred: {e}")
    finally:
        driver.quit()
driver = webdriver.Edge()
add_subcategory(driver, "admin@gmail.com", "Admin@123", "Kashmir", "Fruits", "apple")

```

### Result

```


DevTools listening on ws://127.0.0.1:63235/devtools/browser/d5766cd2-d617-4fec-a2fc-23d5fbccb6c9
[13188:15572:1018/152759.896:ERROR:edge_auth_errors.cc(523)] EDGE_IDENTITY: Get Default OS Account
citSignInFailure, Secondary Error: kAccountProviderFetchError, Platform error: 0, Error string:

Login successful!
Subcategory name 'Kashmir' entered.
Category 'Fruits' selected.
Product Category 'apple' selected.

```

Figure 16: Result of Subcategory add page

Microsoft Edge is being controlled by automated test software.

 Admin

[Dashboard](#)  
[Manage Users](#)  
[Category](#) ▾  
[Product](#) ▾  
[Sub Category](#) ▾  
[Price Charts](#) ▾  
[Compare Prices](#)  
[Logout](#)

### Add Subcategory

Subcategory Name:

Category:

Product Category:

[Add Subcategory](#)

Figure 17:Subcategory Page

## Test report

Test Case 2					
Project Name: Green Grocery					
User Registration Test Case					
Test Case ID: Test_2			Test Designed By: Sreelakshmi PS		
Test Priority (Low/Medium/High): High			Test Designed Date: 17-10-2024		
Module Name:Subcategory page			Test Executed By: Sreelakshmi P S		
Test Title: Add Subcategory			Test Execution Date: 17-10-2024		
Description: Add subcategory based on the category and product name					
Pre-Condition: Admin enter the subcategory					
Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fai l)
1	Navigate to the subcategory page		Display subcategory page	Subcategory displayed	pass
2	Valid subcategory	Subcategory: Kashmiri	Sub category added successfully	User should be view the category	pass
3	Choose category	Category name: fruits			
4	Choose product name	Products name: apple			
10	Click on Submit button				
Post-Condition: Enter the subcategory choose the product and category if the subcategory is already existing show an error					

### Test case 3

#### Code

```

from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import Select # Import Select for dropdowns
from selenium.webdriver.common.keys import Keys
import time
import logging
from selenium.webdriver.remote.remote_connection import LOGGER
LOGGER.setLevel(logging.ERROR)
def slow_typing(element, text, delay=0.2):
    for char in text:
        element.send_keys(char)
        time.sleep(delay)
def admin_add_product(driver, email, password):
    driver.get("http://127.0.0.1:8000/login/")
    try:
        print("Attempting to log in as admin...")
        email_input = WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.NAME, "email"))
        )
        slow_typing(email_input, email) # Slow typing for email
        print(f"Entered email: {email}")
        password_input = WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.NAME, "password"))
        )
        slow_typing(password_input, password)
        print("Entered password.")
        password_input.send_keys(Keys.RETURN)
        print("Admin login successful!")
        print("Navigating to Add Product Category page...")
        driver.get("http://127.0.0.1:8000/add_product_category/")
        WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.ID, "category"))
        )
        category_dropdown = Select(driver.find_element(By.ID, "category"))
        category_dropdown.select_by_visible_text("Fruits")
        print("Fruits category selected.")
        product_input = driver.find_element(By.ID, "product_name")
        slow_typing(product_input, "Apple") # Slow typing for product name
        print("Product name 'Apple' entered.") # Debugging line
        submit_button = driver.find_element(By.XPATH, "//button[@type='submit']")
        submit_button.click()
        WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.CLASS_NAME, "alert-success"))
        )
        print("Category 'Fruits' and product 'Apple' added successfully.")
    except Exception as e:
        print(f"An error occurred: {e}") # Custom error message without showing the full stack trace
    finally:
        # Close the browser when done
        driver.quit()
        print("Test completed successfully.")
driver = webdriver.Edge()
admin_add_product(driver, "admin@gmail.com", "Admin@123") # Replace with actual admin credentials

```


## Result

```
DevTools listening on ws://127.0.0.1:63420/devtools/browser/cdc303b7-e6d2-47f8-a75b-1c1c5824b3c2
[9480:9424:1018/153122.922:ERROR:edge_auth_errors.cc(523)] EDGE_IDENTITY: Get Default OS Account
tSignInFailure, Secondary Error: kAccountProviderFetchError, Platform error: 0, Error string:

Attempting to log in as admin...
Entered email: admin@gmail.com
Entered password.
Admin login successful!
Navigating to Add Product Category page...
Category dropdown found.
Fruits category selected.
Product name 'Apple' entered.
```

**Figure 18:Category Result**

Microsoft Edge is being controlled by automated test software.

**Admin**

[Dashboard](#)  
[Manage Users](#)  
[Category ▾](#)  
[Product ▾](#)  
[Sub Category ▾](#)  
[Price Charts ▾](#)  
[Compare Prices](#)  
[Logout](#)

### Add Product Category

Category:

Product Name:

[Add Product Category](#)

**Figure 19: Category Page**

**Test Report**

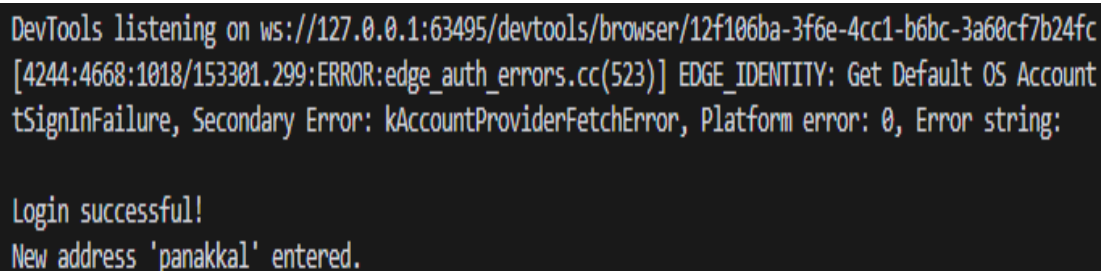
Test Case 3					
Project Name: Fabie’s Kids Wear					
category Test Case					
Test Case ID: Test_3			Test Designed By: Sreelakshmi PS		
Test Priority(Low/Medium/High):			Test Designed Date: 17-10-2024		
Module Name: product page			Test Executed By :Sreelakshmi P S		
Test Title : Add product			Test Execution Date: 17-10-2024		
Description: Add products based on category					
Pre-Condition :User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/ Fai l)
1	Navigate to product page		Display product	product displayed	pass
2	Choose the category	Category name: fruits	Display the category		pass
2	Valid ProductName entered	Product name: apple	User should be able to view	User can view the product	pass
3	Click on to Add button				
Post-Condition: product is added by admin based on the category that the user can view					

## Test Case 4

### Code

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.keys import Keys
import time
def slow_typing(element, text, delay=0.2):
    for char in text:
        element.send_keys(char)
        time.sleep(delay)
def edit_deliveryboy_address(driver, email, password, new_address):
    driver.get("http://127.0.0.1:8000/login/")
    try:
        email_input = WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.NAME, "email")))
        slow_typing(email_input, email)
        password_input = WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.NAME, "password")))
        slow_typing(password_input, password)
        password_input.send_keys(Keys.RETURN)
        print("Login successful!")
        driver.get("http://127.0.0.1:8000/deliveryboy/profile/edit/")
        address_input = WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.ID, "address")))
        address_input.clear()
        slow_typing(address_input, new_address)
        print(f"New address '{new_address}' entered.")
        submit_button = driver.find_element(By.XPATH, "//button[@type='submit']")
        submit_button.click()
        WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.CLASS_NAME, "alert-success")))
        print("Address updated successfully.")
```

### Result



DevTools listening on ws://127.0.0.1:63495/devtools/browser/12f106ba-3f6e-4cc1-b6bc-3a60cf7b24fc  
 [4244:4668:1018/153301.299:ERROR:edge\_auth\_errors.cc(523)] EDGE\_IDENTITY: Get Default OS Account  
 tSignInFailure, Secondary Error: kAccountProviderFetchError, Platform error: 0, Error string:  
  
 Login successful!  
 New address 'panakkal' entered.

**Figure 20:Result of profile edit page**



Microsoft Edge is being controlled by automated test software.

**DELIVERY BOY DASHBOARD**

- Dashboard
- My Deliveries
- Orders
- Profile
- Settings
- Logout

**DELIVERY BOY PROFILE**

**Name:** sreekanth

**Email:** psreelakshmi1891@gmail.com

**Phone:** 8978909908

**Address:** panakkal

**Pincode:** 686509

[Edit Profile](#)

Figure 20 : Delivery Boy profile page

## Test Report

Test Case 4					
Project Name: Fabie’s Kids Wear					
Subcategory Test Case					
Test Case ID: Test_4			Test Designed By: Sreelakshmi PS		
Test Priority(Low/Medium/High):			Test Designed Date: 17-10-2024		
Module Name: Profile Edit			Test Executed By :Sreelakshmi P S		
Test Title : Edit the profile page			Test Execution Date: 17-10-2024		
Description: Edit the Delivery Boy profile page					
Pre-Condition :Delivery boy has valid email and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fai l)
1	Navigate to Delivery Boy dashboard		Display dashboard	Enter into the dashboard	pass
2	Valid profile details	Address: parakkadavil	Delivery boy should be able to edit their profile	Delivery boy can view the profile	pass
4	Click on to Add button				
Post-Condition: Edit the profile page of the Delivery boy					

## **CHAPTER 6**

### **IMPLEMENTATION**

## **6.1 INTRODUCTION**

Implementation is the process of turning a plan or idea into action. In the context of a project, implementation refers to the stage where the plan is put into action, and the project is executed. This stage involves the use of resources and the coordination of activities to achieve project goals.

During the implementation stage, project managers must ensure that the project is executed efficiently and effectively. They need to monitor progress and make adjustments to the plan as needed. It is essential to have clear communication with team members and stakeholders to ensure that everyone is working towards the same goal.

The success of a project depends heavily on the implementation stage. A well-planned project may not succeed if the implementation is poorly executed. Therefore, it is crucial to have a detailed plan, adequate resources, and a skilled team to ensure a successful implementation. Additionally, effective monitoring and control mechanisms should be put in place to detect any issues and address them promptly.

## **6.2 IMPLEMENTATION PROCEDURES**

Implementation procedures are the set of steps and processes that a project team follows to execute a project plan. These procedures outline the specific actions that need to be taken to ensure that the project is completed successfully. The implementation procedures typically include defining the scope of the project, developing a project plan, assigning roles and responsibilities, mobilizing resources, executing the project plan, monitoring progress, and closing the project. Following these procedures helps to ensure that the project is completed on time, within budget, and to the required quality standards.

The implementation procedures typically start with defining the scope of the project. This involves identifying the project goals, objectives, deliverables, timelines, and resources required. Once the scope is defined, the project team develops a project plan that outlines the tasks to be completed, the resources required, timelines, and milestones. The plan also identifies potential risks and mitigation strategies.

### **6.2.1 User Training**

User training is an important part of implementing a new system or technology. It means teaching people how to use the new system or technology effectively. This helps them to

understand how to operate the system and to use it to its full potential.

User training is important because it ensures that everyone who needs to use the system can use it effectively. It also helps to prevent mistakes and problems that might occur if people don't know how to use the system properly. The training can include written guides, videos, and in-person training sessions. It's important to provide training before the new system is put into use so that everyone has time to learn how to use it.

### **6.2.2 Training on the Application Software**

Training on the application software is an important aspect of implementation procedures for any new software or technology. This involves educating users on how to effectively use the application software for their day-to-day tasks. It is important to provide training on the specific features and functions of the software that users will be using regularly.

Training on the application software can include various formats such as videos, user manuals, or online tutorials. In-person training sessions can also be arranged to provide hands-on experience to the users. It is crucial to ensure that the training is delivered in a way that is easy to understand and follow for all users.

### **6.2.3 System Maintenance**

System maintenance is an important aspect of implementation procedures for any new system or technology. It refers to the ongoing process of maintaining and updating the system to ensure that it continues to operate effectively and efficiently. This involves monitoring the system regularly to identify any issues and resolving them promptly.

System maintenance can include various activities such as updating the software, fixing bugs and errors, replacing hardware, and performing backups. It is crucial to establish a schedule for regular maintenance tasks to ensure that the system operates smoothly and to prevent any major issues.

Regular system maintenance can have several benefits such as improved system performance, increased uptime, reduced downtime, and enhanced security. It is important to allocate adequate resources and assign responsibility for the maintenance tasks to ensure that the system is maintained effectively and efficiently.

### **6.2.4 Hosting**

- Log into Koyeb using your GitHub account at [koyeb.com](https://koyeb.com).
- Create a New App by clicking " Create" on the Dashboard and choosing "App" for

deployment.

- Select GitHub as your source, connect your GitHub account, and choose your repository and branch to deploy.
- Configure Build Settings with the appropriate build and run commands for your app (e.g., npm install and npm start).
- Set Up Environment Variables if your app requires them, by adding key-value pairs in the environment settings.
- Deploy the App by clicking "Deploy Now" after reviewing all settings.
- Monitor Deployment using the deployment logs until it completes successfully.
- Access Your Site using the provided Preview URL to check if the app is live and functional.
- Configure an External Database if needed, and update environment variables with the database connection details.

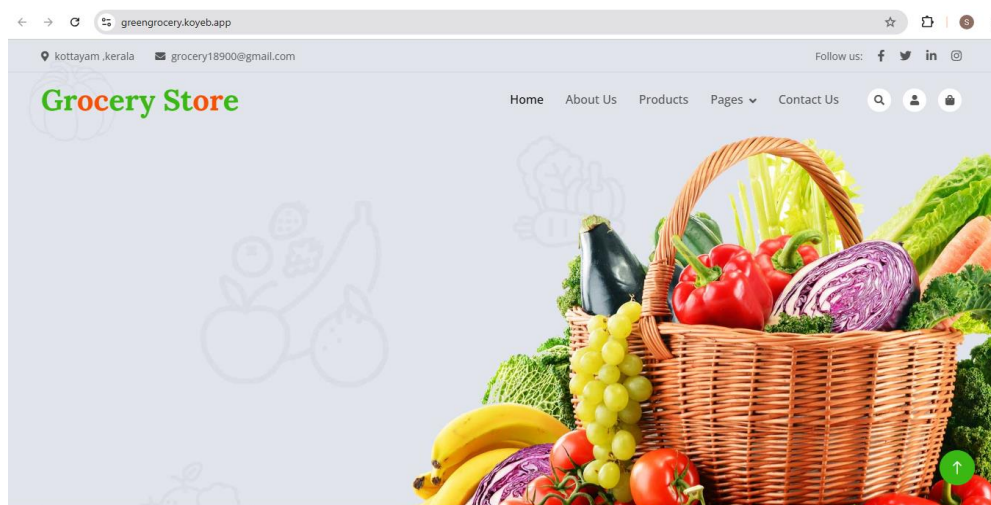
**Hosted Website: Koyeb.app, <https://app.koyeb.com/>**

**Hosted Link: <https://greengrocery.koyeb.app/>**

**Hosted Link QR Code**



**Screenshot**



**Figure 21: Customer dashboard**

## **CHAPTER 7**

### **CONCLUSION AND FUTURE SCOPE**

## 7.1 CONCLUSION

The proposed online grocery store system leverages Python and Django for robust web development and utilizes SQLite for efficient data management. The system addresses the limitations of traditional grocery shopping by providing a convenient platform for purchasing fresh fruits and vegetables directly from local farmers. It features dedicated modules for customers, suppliers, delivery personnel, and administrators, enhancing inventory management, streamlining order processing, and allowing real-time tracking of deliveries. Additionally, the implementation of quality detection using machine learning ensures that only the best produce is offered to customers, enhancing overall satisfaction. This user-friendly system supports secure payment options and personalized customer service, making it both technically and behaviorally feasible. Ultimately, it aims to thrive in the competitive online grocery market while promoting local agriculture and ensuring customer satisfaction.

## 7.2 FUTURE SCOPE

The future scope of the online grocery store system includes enhancing the existing quality detection system through advanced machine learning for predictive analytics and personalized recommendations. Developing a dedicated mobile app will improve accessibility and facilitate real-time notifications for customers. Implementing augmented reality (AR) features can offer an interactive shopping experience, while sustainability initiatives, such as carbon footprint tracking and eco-friendly packaging, will promote responsible shopping. Expanding payment options to include cryptocurrencies and mobile wallets will cater to a broader audience. Additionally, robust multi-channel marketing strategies can enhance customer engagement, and creating a community forum will encourage users to share recipes and tips. Finally, exploring geographical expansion will diversify product offerings and reach more customers, ensuring the online grocery store remains competitive in the evolving e-commerce landscape.

## **CHAPTER 8**

### **BIBLIOGRAPHY**



**REFERENCES:**

- 1) Smith, A., & Doe, J. (2018). Automatic Fruit Quality Detection System.
- 2) Johnson, B., & Lee, K. (2020). Ripeness Classification of Bananas Using Machine Learning.
- 3) Wang, C., & Zhang, L. (2019). Sensor-Based Quality Assessment of Vegetables.
- 4) Patel, D., & Singh, M. (2017). Image Analysis Techniques for Fruit Ripeness Detection.
- 5) Martinez, E., & Garcia, F. (2021). Deep Learning for Ripeness Classification of Tomatoes.
- 6) Kim, F., & Park, H. (2022). Multi-Spectral Imaging for Quality Control of Fruits.
- 7) Lee, G., & Chen, I. (2016). Non-Destructive Quality Assessment of Fruits Using NIR Spectroscopy.
- 8) Ali, H., & Kumar, J. (2019). IoT-Based Smart Grocery System for Quality Monitoring.
- 9) Kumar, I., & Reddy, P. (2020). Automated Ripeness Detection of Mangoes Using Image Processing.
- 10) Brown, J., & White, K. (2021). Evaluation of Fruit Quality Using Hyperspectral Imaging.
- 11) Johnson, L., & Patel, N. (2022). Fruit Ripeness Prediction Using Machine Learning.
- 12) Anderson, O., & Kumar, P. (2019). Real-Time Quality Monitoring of Fresh Produce Using Smart Sensors.

**WEBSITES:**

- <https://www.djangoproject.com/>
- <https://www.w3schools.com/>
- <https://developer.mozilla.org/>
- <https://stackoverflow.com/>
- <https://www.geeksforgeeks.org/>

## **CHAPTER 9**

### **APPENDIX**

## 9.1 Sample Code

### Login.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Login</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">
  { % load static % }
  <link rel="stylesheet" type="text/css" href="{ % static 'styles.css' % }">
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
  <link href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-
awesome.min.css" rel="stylesheet">
</head>
<body>
  <div class="container-fluid">
    <div class="row">
      <div class="col-lg-6 col-md-6 form-container">
        <div class="col-lg-8 col-md-12 col-sm-9 col-xs-12 form-box text-center">
          <div class="logo mt-5 mb-3">
            </div>
          <div class="heading mb-3">
            <h4>Login into your account</h4>
          </div>
          { % if error % }
          <div class="alert alert-danger">{{ error }}</div>
          { % endif % }
          <form method="post" action="{ % url 'login' % }" onsubmit="return
validateForm()">
            { % csrf_token % }
```

```

    <div class="form-input">
        <span><i class="fa fa-envelope"></i></span>
        <input type="email" name="email" placeholder="Email Address"
required>

        <div class="error-message" id="emailError"></div>
    </div>
    <div class="form-input">
        <span><i class="fa fa-lock"></i></span>
        <input type="password" name="password" placeholder="Password"
required>

        <div class="error-message" id="passwordError"></div>
    </div>
    <div class="row mb-3">
        <div class="col-6 d-flex">
            <div class="custom-control custom-checkbox">
                <input type="checkbox" class="custom-control-input" id="cb1">
            </div>
        </div>
        <div class="col-6 text-right">
            <a href="{% url 'forgot_password' %}" class="forget-link">Forgot
password</a>
        </div>
    </div>
    <div class="text-left mb-3">
        <button type="submit" class="btn btn-primary">Login</button>
    </div>
    <!--<div class="mb-3">or login with</div>
    <div class="row mb-3">
        <div class="col-4">
            <a href="#" class="btn btn-block btn-social btn-facebook">
                <i class="fa fa-facebook"></i>
            </a>
        </div>
    </div>

```

```

        <div class="col-4">
            <a href="#" class="btn btn-block btn-social btn-google">
                <i class="fa fa-google"></i>
            </a>
        </div>
        <div class="col-4">
            <a href="#" class="btn btn-block btn-social btn-twitter">
                <i class="fa fa-twitter"></i>
            </a>
        </div>
    </div>-->
    <div>Don't have an account?
        <a href="{ % url 'register' % }" class="register-link">Register here</a>
    </div>
</form>
</div>
</div>
<div class="col-lg-6 col-md-6 d-none d-md-block image-container"></div>
</div>
</div>
<script>
$(document).ready(function() {
    // Real-time validation
    $('input[name="email"]').on('input', function() {
        validateEmailField($(this));
    });
    $('input[name="password"]').on('input', function() {
        validatePasswordField($(this));
    });
});
function validateForm() {
    let isValid = true;
    if (!validateEmailField($('input[name="email"]'))) isValid = false;

```

```
        if (!validatePasswordField($('input[name="password"]'))) isValid = false;
        return isValid;
    }
    function validateEmailField(input) {
        const re = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;
        const isValid = re.test(String(input.val()).toLowerCase());
        showErrorMessage('email', isValid ? '' : 'Invalid email format');
        return isValid;
    }
    function validatePasswordField(input) {
        const isValid = input.val().trim() !== '';
        showErrorMessage('password', isValid ? '' : 'Please enter your password');
        return isValid;
    }
    function showErrorMessage(field, message) {
        const errorElement = $('#${field}Error');
        if (message) {
            errorElement.text(message).show();
            $('input[name="${field}"]').addClass('error');
        } else {
            errorElement.text('').hide();
            $('input[name="${field}"]').removeClass('error');
        }
    }
}
</script>
</body>
</html>
```

### Views.py

```
def login_view(request):
    if request.method == 'POST':
        email = request.POST['email']
        password = request.POST['password']
```

```

if email == 'admin@gmail.com' and password == 'Admin@123':
    # Use Django's login function
    return redirect('admin_dashboard')

try:
    user = User.objects.get(email=email, password=password)
    login(request, user) # Use Django's login function

    if user.role == 'farmer':
        return redirect('farmer_dashboard')
    elif user.role == 'customer':
        return redirect('customer_dashboard')
    elif user.role == 'deliveryboy':
        return redirect('deliveryboy_dashboard')
except User.DoesNotExist:
    return render(request, 'login.html', {'error': 'Invalid credentials'})
return render(request, 'login.html')

```

### Cart.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>View Cart</title>
    <meta content="width=device-width, initial-scale=1.0" name="viewport">
    { % load static % }
    <link href="{ % static 'img/favicon.ico' % }" rel="icon">
    <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap
" rel="stylesheet">
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-
beta3/css/all.min.css" rel="stylesheet">
    <link href="{ % static 'css/bootstrap.min.css' % }" rel="stylesheet">
    <link href="{ % static 'css/style.css' % }" rel="stylesheet">

```

```

</head>
<body>
  <!-- Navbar Start -->
  <div class="container-fluid fixed-top px-0 wow fadeIn" data-wow-delay="0.1s">
    <div class="top-bar row gx-0 align-items-center d-none d-lg-flex">
      <div class="col-lg-6 px-5 text-start">
        <small class="ms-4"><i class="fa fa-envelope me-
2"></i>grocery18900@gmail.com</small>
      </div>
      <div class="col-lg-6 px-5 text-end">
        <small>Follow us:</small>
        <a class="text-body ms-3" href="#"><i class="fab fa-facebook-f"></i></a>
        <a class="text-body ms-3" href="#"><i class="fab fa-twitter"></i></a>
        <a class="text-body ms-3" href="#"><i class="fab fa-linkedin-in"></i></a>
        <a class="text-body ms-3" href="#"><i class="fab fa-instagram"></i></a>
      </div>
    </div>
    <nav class="navbar navbar-expand-lg navbar-light py-lg-0 px-lg-5 wow fadeIn"
data-wow-delay="0.1s">
      <a href="index.html" class="navbar-brand ms-4 ms-lg-0">
        <!-- Navbar Brand Logo Here -->
      </a>
      <button type="button" class="navbar-toggler me-4" data-bs-toggle="collapse"
data-bs-target="#navbarCollapse">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarCollapse">
        <div class="navbar-nav ms-auto p-4 p-lg-0">
          <a href="{ % url 'customer_dashboard' % }" class="nav-item nav-link
active">Home</a>
          <a href="#" class="nav-item nav-link">About Us</a>
          <a href="{ % url 'list_category_products' % }" class="nav-item nav-
link">Products</a>

```



```

    <div class="nav-item dropdown">
      <a href="#" class="nav-link dropdown-toggle" data-bs-
toggle="dropdown">Pages</a>
      <div class="dropdown-menu m-0">
        <a href="#" class="dropdown-item">Blog Grid</a>
        <a href="#" class="dropdown-item">Our Features</a>
        <a href="{% url 'price_chart_customer' %}" class="dropdown-
item">View Price Charts</a>
      </div>
    </div>
    <div>
      <a href="{% url 'contact' %}" class="nav-item nav-link">Contact Us</a>
    </div>
    <a class="btn-sm-square bg-white rounded-circle ms-3" href="">
      <small class="fa fa-shopping-bag text-body"></small>
    </a>
  </div>
</nav>
</div>
<div class="page-header wow fadeIn" data-wow-delay="0.1s">
  <div class="container text-left"> <!-- Added text-left class here -->
    <h3 class="display-3 mb-3 animated slideInDown">Cart Items</h3>
    <nav aria-label="breadcrumb animated slideInDown">
      <ol class="breadcrumb mb-0">
        <li class="breadcrumb-item"><a class="text-body"
href="#">Home</a></li>
        <li class="breadcrumb-item"><a class="text-body"
href="#">Pages</a></li>
        <li class="breadcrumb-item text-dark active" aria-current="page">Cart</li>
      </ol>
    </nav>
  </div>
</div>
<div class="container">

```

```

<div class="cart-container">
  {% if cart_items %}
  <table>
    <thead>
      <tr>
        <th>Image</th>
        <th>Product</th>
        <th>Category Name</th>
        <th>Quantity(Kg)</th>
        <th>Price (Rs)</th>
        <th>Actions</th>
      </tr>
    </thead>
    <tbody>
      {% for item in cart_items %}
      <tr>
        <td></td>
        <td>{{ item.product.product_name.product_name }}</td>
        <td>{{ item.product.category.name }}</td>
        <td>
          <form method="post" action="{{ % url 'update_cart' item.id % }}">
            {% csrf_token %}
            <input type="number" step="0.5" min="0.5" max="{{ {
item.product.stock }}" name="quantity" value="{{ item.quantity }}" required>
            <button type="submit" class="btn btn-primary btn-
sm">Update</button>
          </form>
        </td>

        <td>Rs{{ item.total_price }}</td>
        <td>
          <form method="post" action="{{ % url 'remove_from_cart' item.id

```

```

% } ">

        { % csrf_token % }
        <button type="submit" class="btn btn-danger btn-
sm">Remove</button>
    </form>
</td>
</tr>
{ % endfor % }
</tbody>
</table>
<div class="cart-summary">
    <p><strong>Total Price: Rs{ { grand_total } }</strong></p>
    <form action="{ % url 'checkout' % }" method="POST">
        { % csrf_token % }
        <button type="submit">Proceed to Checkout</button>
    </form>
</div>
{ % else % }
<p>Your cart is empty.</p>
{ % endif % }
</div>
</div><br><br><br><br>
<footer>
    <p>&copy; Green Grocery</p>
</footer>
<script src="{ % static 'js/bootstrap.bundle.min.js' % }"></script>
</body>
</html>

```

## 9.2 Screen Shots

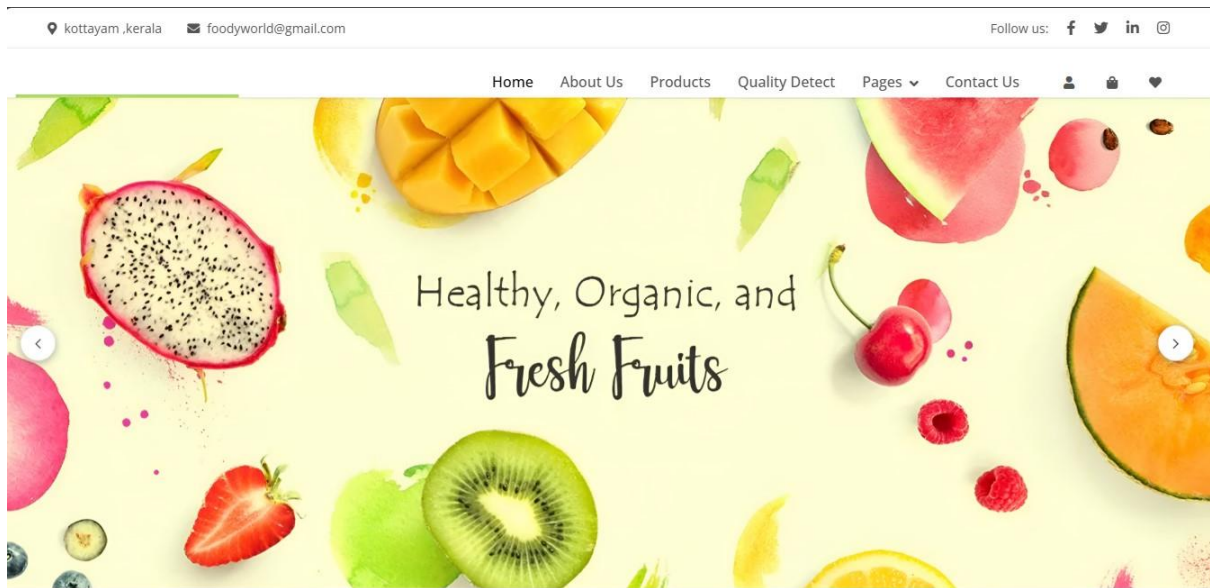


Figure 22 : Customer Dashboard

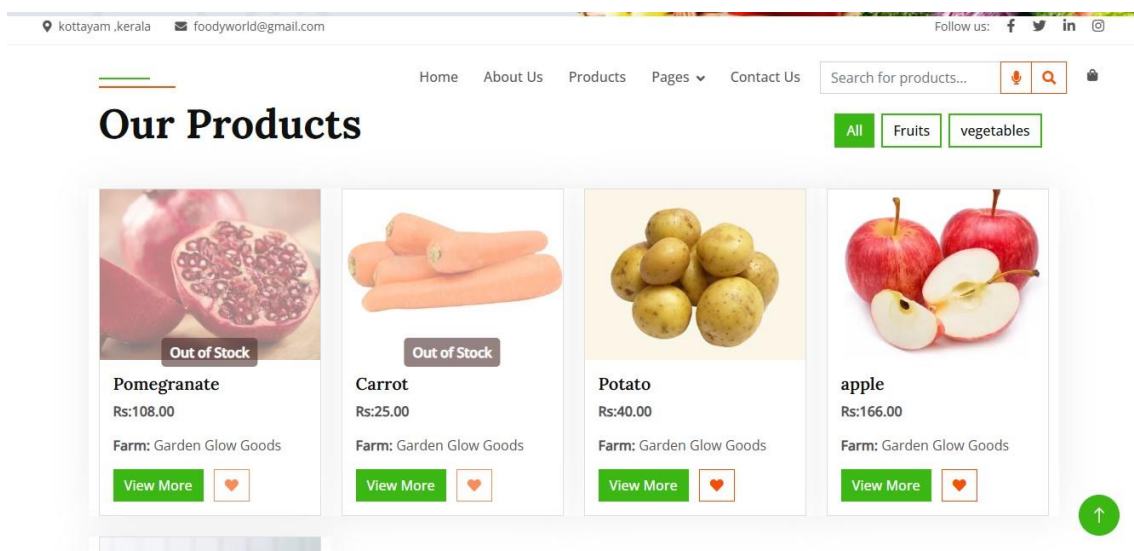


Figure 23 : product list

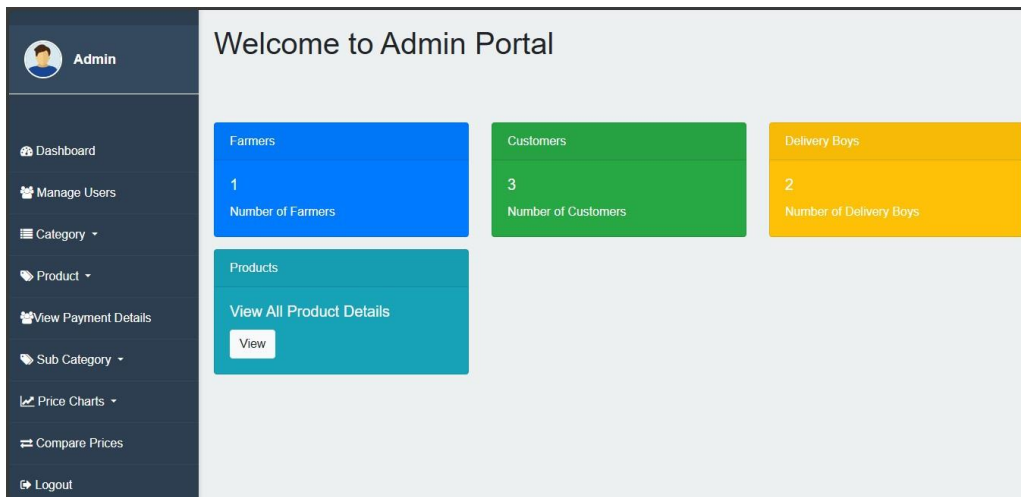


Figure 24: Admin Dashboard

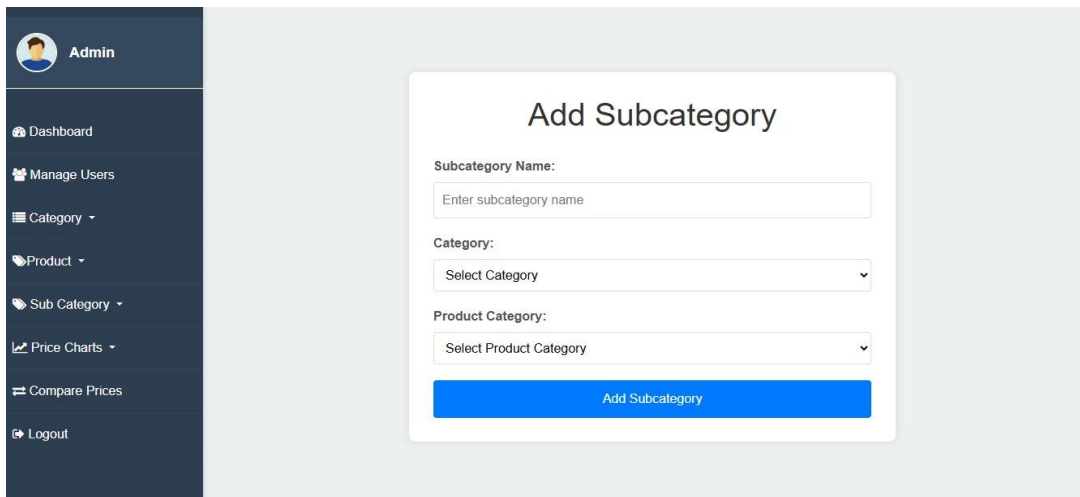


Figure 25: Subcategory page



Figure 26: Farmer Dashboard

Farmer Dashboard

Dashboard

Add Product

View Products

Orders

Profile

Logout

Product List

	Quantity (kg)	Stock Quantity	Shelf Life	Form Factor	Organic	Common Name	Product Image	Actions
	1	0	8 Days	Whole	True	Alu,aloo,aalu,aaloo,allu,aloo		<div><div></div><div></div></div>
	1	0	5 Days	Whole	True	Gajar,gaajar		<div><div></div><div></div></div>
	1	30	9 Days	Whole	True	Alu,aloo,aalu,aaloo,allu,aloo		<div><div></div><div></div></div>
enols.	1	3	12 Days	Whole	True	Sev,seb		<div><div></div><div></div></div>

Figure 27:Product list page

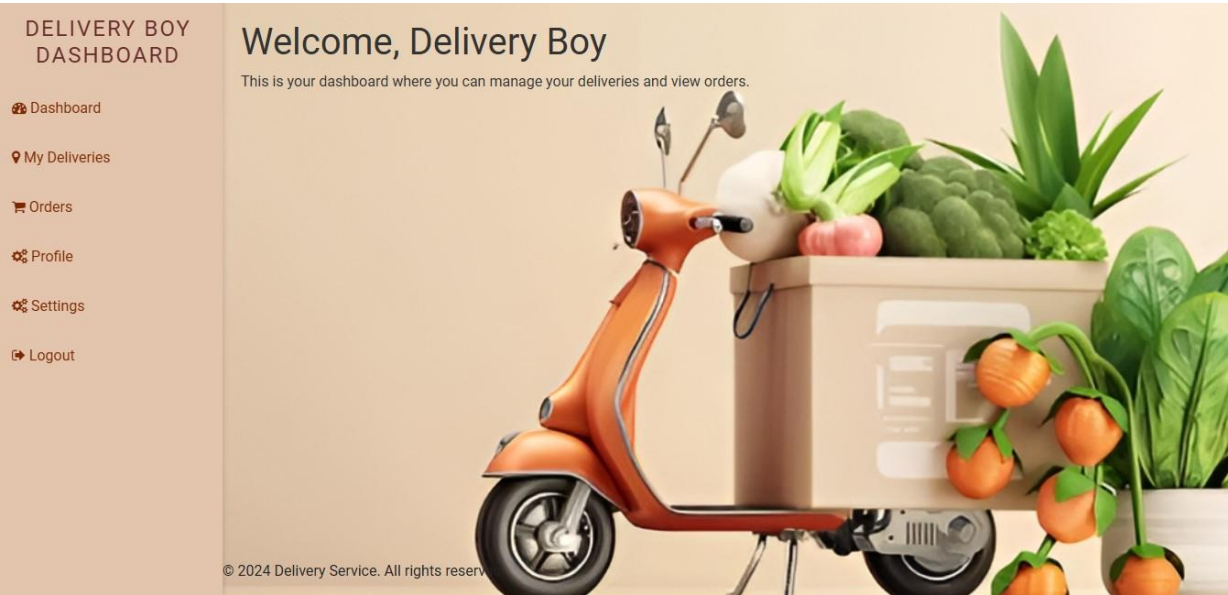


Figure 28:DeliveryBoy Dashboard

## 9.3 Git Log

11/7/24, 9:51 AM Commits · pssreelakshmi/Miniproject\_2-

Code Issues Pull requests Actions Projects Wiki Security

### Commits

main All users All time

Commits on Nov 4, 2024

**fifth commit**

c851818 ...

pssreelakshmi committed 3 days ago

Commits on Oct 29, 2024

**fourth commit**

da36c31 ...

pssreelakshmi committed last week

Commits on Oct 28, 2024

**third commit**

6936822 ...

pssreelakshmi committed last week

**second commit**

55746a3 ...

pssreelakshmi committed last week

Commits on Oct 26, 2024

**first commit**

5c751bf ...

pssreelakshmi committed 2 weeks ago

Commits on Aug 12, 2024

**Add files via upload**

Verified df621bd ...

pssreelakshmi authored on Aug 12

Commits on Jul 16, 2024

**Add files via upload**

Verified 1eea74f ...

pssreelakshmi authored on Jul 16

**Add files via upload**

Verified 031d659 ...

pssreelakshmi authored on Jul 16

< Previous Next