Dairy Care System

Comprehensive Dairy Management

Mini Project Report

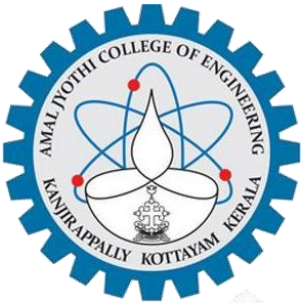
Submitted by

Krishnendu Lal

Reg. No.: AJC23MCA-2042

In Partial fulfillment for the Award of the Degree of

MASTER OF COMPUTER APPLICATIONS (MCA)



AMAL JYOTHI COLLEGE OF ENGINEERING AUTONOMOUS

KANJIRAPPALLY

[Approved by AICTE, Accredited by NAAC, Accredited by NBA.
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 Project report, "**DAIRY CARE SYSTEM"** is the bonafide work of **KRISHNENDU LAL (Regno: AJC23MCA-2042)** in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications under **Amal Jyothi College of Engineering Autonomous, Kanjirappally** during the year 2024-25.

Mr. Amal K Jose
Internal Guide

Mr. Binumon Joseph
Coordinator

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

DECLARATION

I hereby declare that the project report **"DAIRY CARE SYSTEM"** is a bona fide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Master of Computer Applications (MCA) from Amal Jyothi College of Engineering Autonomous during the academic year 2024-2025.

Date: 08-11-2024
KANJIRAPPALLY

KRISHNENDU LAL
Reg: AJC23MCA-2042

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** 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 **Mr. Amal K Jose** for his 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.

KRISHNENDU LAL

ABSTRACT

The Dairy Care System is a robust web application built using Django, designed to optimize the management of dairy farms. It simplifies operations by tracking livestock, monitoring milk production, and overseeing animal health records. Key features include user authentication and role management, allowing secure access for owners, employees, and customers. The system also supports product management, enabling owners to add, update, and monitor product listings. Customers can search for and purchase dairy products online using payment gateways like Razor-Pay or offline through cash on delivery.

In addition to product management, the Dairy Care System provides tools for farm monitoring, where employees can add, update, and monitor animal listing and monitor health status. The system also facilitates handling of customer product payments efficiently. Feedback collection and analysis are integrated into the platform, allowing owners to assess customer input to make improvements. By offering a wide range of features, the Dairy Care System enhances farm productivity and ensures smooth day-to-day operations.

CONTENT

List of Abbreviations

- UML - Unified Modelling Language
- RDBMS - Relational Database Management System
- 1NF - First Normal Form
- 2NF - Second Normal Form
- 3NF - Third Normal Form
- IDE - Integrated Development Environment
- HTML - HyperText Markup Language
- JS - JavaScript
- CSS - Cascading Style Sheets
- UI - User Interface
- HTTP - Hypertext Transfer Protocol
- PK - Primary Key
- FK - Foreign Key
- SQL - Structured Query Language
- CRUD - Create, Read, Update, Delete

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The Dairy Care System is a web-based platform designed to modernize and streamline dairy farm operations, offering an integrated solution for managing livestock, milk production, product inventory, sales, and overall farm activities. Developed using Django, this system enhances farm efficiency, productivity, and profitability by automating routine tasks and providing a seamless user experience for farm owners, employees, and customers. The platform offers secure role-based access, allowing users to manage different aspects of farm operations according to their roles.

Farm owners can manage product listings, track inventory, and handle both online and offline sales, while employees can monitor and track the health and condition of livestock. Customers, on the other hand, can search for and purchase products through a user-friendly interface, with options for online payments via Razor-Pay or cash on delivery. Additionally, the system facilitates feedback submission and analysis, helping farm owners improve services based on input from both customers and employees.

Built with HTML, CSS, and JavaScript for the frontend and Django with MySQL for the backend, the Dairy Care System provides a comprehensive solution for the modern dairy farming industry, transforming traditional practices into data-driven, efficient processes.

1.2 PROJECT SPECIFICATION

The Dairy Care System is designed to enhance the management of dairy farms through automation and efficient handling of core activities. The system offers role-based access to users with varying responsibilities, ensuring that farm owners, employees, and customers can perform their tasks efficiently.
Below are the detailed specifications of the project:

1. User Roles and Authentication:

- User Roles:
    - Owner (Admin): Manages the entire system, products, employees, and overall farm operations.
    - Employee: Handles tasks like feeding, health monitoring, and milking operations.
    - Customer: Purchases dairy products online or offline, tracks orders, and provides feedback.

- Authentication:
    - Secure login and registration system with role-based access controls.
    - Admin can manage users, assign roles, and set permissions.

2. Product and Inventory Management:

- Product Search and Purchase:
    - Customers can browse, search, and purchase dairy products.
    - Payment options include online methods like Razor-Pay or Cash on Delivery.
    - Integration with delivery tracking to monitor product shipments.
- Inventory Management:
    - Owners can add, update, and remove products from the inventory.
    - Low-stock alerts and notifications to avoid product shortages.
    - Real-time inventory status displayed on the dashboard.

3. Animal Health Management:

- Health Management:
    - Veterinarians can record animal health data and prescribe medication.

4. Payment Processing:

- Customer Payments:
    - Online payment gateways like Razor-Pay for seamless transactions.
    - Cash on Delivery option with location sharing for order delivery.

5. Feedback and Analysis:

- Feedback System:
    - Customers can submit feedback on products.
- Feedback Analysis:
    - Admin or owners can analyse feedback for product improvement.

6. Technologies and Tools:

- Frontend:
    - HTML, CSS, JavaScript for creating an intuitive user interface.
- Backend:
    - Django framework for server-side logic and application structure.
    - MySQL database for managing and storing data related to users, products, livestock, and transactions.
- Payment Gateway:
    - Integration with Razor-Pay for secure online payments.

- Operating System:
    - Developed and tested on Windows.
- IDE:
    - Built using Visual Studio Code for development and debugging.

7. System Workflow:

- User Interaction:
    - Users interact with the system based on their roles—admin or owner manage, employees record data, and customers shop.
- Admin Dashboard:
    - Centralized dashboard to manage users, products, inventory, sales, and deliveries.
- Notifications:
    - Automated notifications for stock management and feedback analysis.

8. Database Design:

- Tables:
    - Users (Admin or Owner, Employee, Customer)
    - Products
    - Orders
    - Payments
    - Feedback
    - Livestock Health Records
- Relationships:
    - Relational data management between users, products, orders, and health records to ensure data consistency.

CHAPTER 2

SYSTEM STUDY

2.1INTRODUCTION

The System Study for the Dairy Care System is a critical evaluation aimed at comparing the proposed application with existing dairy management solutions. This study highlights the strengths and weaknesses of current systems, which often focus on specific functionalities rather than providing a comprehensive approach to farm management.

By analyzing these existing systems, we can demonstrate how the Dairy Care System addresses their limitations through innovative features such as user authentication, inventory management, health monitoring and integrated e-commerce capabilities. Ultimately, this study emphasizes the need for a more integrated and efficient solution for dairy farmers, which the Dairy Care System is designed to provide.

2.2EXISTING SYSTEM

2.2.1 NATURAL SYSTEM STUDIED

In the context of dairy farming, the natural system revolves around the daily operations and processes involved in managing a dairy farm. This includes the care of livestock, monitoring animal health, managing milk production, and overseeing the sale and distribution of dairy products. The system functions through a series of interdependent activities where farmers and employees track and manage tasks such as animal health checks, and product management.

Traditionally, these tasks are often handled manually or with limited technological support, leading to inefficiencies in operations and decision-making. The natural system of a dairy farm, although effective, is prone to challenges such as data inaccuracy, delayed responses to animal health issues, and difficulty in scaling production efficiently.

By studying this natural system, the Dairy Care System aims to digitize and streamline these processes, providing a modern solution that integrates advanced tools for farm management, health monitoring and product sales, thereby optimizing the entire workflow of a dairy farm.

2.2.2 DESIGNED SYSTEM STUDIED

The designed system in the context of the Dairy Care System focuses on creating a comprehensive digital platform to streamline and modernize the operations of a dairy farm. Unlike the traditional,

manual management processes, this system leverages technology to automate critical tasks such as livestock health monitoring, inventory management, and product sales.

Built using the Django framework, the designed system introduces user authentication, role-based access and integrated e-commerce. It provides a user-friendly interface for farm owners, employees, and customers, ensuring easy navigation and accessibility. By enhancing efficiency, reducing human error, and enabling data-driven decision-making, this designed system significantly improves upon the traditional, natural system of dairy farm management.

2.3 DRAWBACKS OF EXISTING SYSTEM

- Limited Technology Integration: Many systems lack advanced tools like machine learning for forecasting milk production or detecting animal diseases.
- Focus on Specific Areas: Most systems focus on product sales or distribution, neglecting comprehensive farm management aspects like livestock health and feeding schedules.
- High Setup and Maintenance Costs: Systems like DelPro and Lactanet are expensive to install and maintain, making them less accessible for smaller farms.
- Lack of Modern User Authentication: Existing systems often use outdated authentication methods and do not support advanced role-based access for different users.
- Complexity and Difficult Navigation: Some platforms, like Uniform-Agri and Lactanet, have overly complex interfaces that can overwhelm users and hinder effective usage.
- Insufficient Customer Feedback Mechanisms: Many systems fail to provide robust feedback collection and analysis tools for customer and employee inputs.
- Lack of Integrated Payment Solutions: Few systems offer secure online payment gateways or cash-on-delivery options with tracking and location-based delivery services.
- Limited Support for Farm Monitoring: Systems like Milma and Binsar Farms do not include comprehensive farm management tools for tracking livestock health, feeding, and milking operations.
- Dependence on Technical Support: Platforms such as Lactanet rely heavily on external technical support, making customization and setup difficult for farm owners

2.4 PROPOSED SYSTEM

The Dairy Care System offers a comprehensive solution that addresses the limitations of existing dairy management platforms. Unlike current systems, which often focus on specific areas like

product sales or basic farm operations, this proposed system integrates all aspects of dairy farm management into one cohesive platform. The system also provides enhanced user authentication with role-based access for owners, employees, and customers, ensuring security and ease of use for all stakeholders. Additionally, it includes an integrated e-commerce platform with secure payment options Razor-Pay and cash-on-delivery services.

Moreover, the Dairy Care System is designed to be cost-effective and easy to install, making it accessible for farms of all sizes, unlike more expensive and complex existing systems. It also features an advanced feedback mechanism, allowing farm owners to collect and analyze feedback from customers, driving continuous improvement. Finally, the system sends real-time notifications to admins for low stock levels, ensuring better inventory management and preventing shortages. Overall, the Dairy Care System provides a modern, scalable, and user-friendly solution to dairy farm management.

2.5 ADVANTAGES OF PROPOSED SYSTEM

- Comprehensive Farm Management: Covers all essential farm operations, including livestock health, milking, and product management, offering a complete solution.
- User-Friendly Interface: Designed for ease of use with a simple and intuitive interface, accessible to users of all technical backgrounds.
- Role-Based Access Control: Secure authentication with tailored access for owners, employees, and customers, ensuring the right functionalities are available to the right users.
- Integrated E-commerce Platform: Allows customers to browse and purchase products online with secure payment options and cash on delivery, enhancing convenience.
- Real-time Inventory Notifications: Alerts admins when stock levels are low, preventing shortages and improving inventory management.
- Cost-Effective and Scalable: More affordable and easier to install compared to other complex and expensive dairy management systems, with scalability for different farm sizes.
- Feedback Collection and Analysis: Built-in tools to collect and analyze feedback from customers, driving continuous improvement in farm operations and customer service.
- Health and Equipment Monitoring: Tracks animal health and milking equipment maintenance, ensuring the smooth operation and welfare of the farm's resources.

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

The primary purpose of conducting a feasibility study is to comprehensively assess whether the proposed project can successfully meet the organization's objectives in terms of available resources, labor, and time. This crucial study allows the developers and decision-makers to gain valuable insights into the project's potential viability and prospects. By carefully examining various aspects of the proposed system, such as its impact on the organization, its ability to fulfill user requirements, and the optimal utilization of resources, a feasibility study helps in determining the project's feasibility and potential success. The assessment of the proposed project's feasibility involves multiple dimensions, each playing a critical role in the decision-making process.

- Technical Feasibility
- Behavioral Feasibility
- Economic Feasibility

A well-conducted feasibility study provides valuable insights to decision-makers, allowing them to make informed judgments about the project's potential success. It assists in identifying potential risks, challenges, and opportunities associated with the proposed endeavor, enabling stakeholders to devise effective mitigation strategies.

3.1.1 Economical Feasibility

The economic feasibility of the Dairy Care System involves assessing the costs associated with its development, implementation, and maintenance, while considering the potential return on investment (ROI). The project’s development costs are kept low by utilizing open-source technologies, which eliminates licensing fees. By leveraging frameworks like Django for the backend and HTML, CSS, and JavaScript for the frontend, the system ensures cost-effectiveness during the development phase. Additionally, hardware costs are minimal, as the system requires standard and affordable hardware components, such as an Intel Core i5 processor, 8 GB of RAM, and a 500 GB hard disk. The use of widely supported technologies also contributes to low operational costs, allowing in-house teams to handle maintenance and bug fixes without incurring significant expenses. Overall, these factors make the Dairy Care System economically viable, with low initial investments and manageable ongoing costs, ensuring a sustainable and efficient solution for dairy farm management.

3.1.2 Technical Feasibility

Technical feasibility evaluates the Dairy Care System's capacity to operate within the current technical environment and its ability to integrate with other systems. The system employs a combination of frontend technologies, including HTML, CSS, JavaScript, to create a responsive and interactive user interface. On the backend, it leverages Django, a robust Python-based web framework that offers scalability, security, and rapid development capabilities.

The database is managed using MySQL, a reliable open-source relational database system capable of handling data efficiently. Additionally, the system is designed to run on PCs with Windows 11, ensuring broad compatibility with most user environments. The selected technologies not only support seamless operation and scalability but also facilitate integration with payment gateways, enhancing the overall functionality and user experience of the Dairy Care System.

3.1.3 Behavioral Feasibility

Behavioral feasibility assesses the likelihood of user acceptance and adoption of the Dairy Care System. The system is designed with a user-friendly interface, making it easy to navigate for all user roles, including farm owners, employees, and customers. Its intuitive design ensures that minimal training is required, with documentation and support resources readily available to assist users during the transition phase.

Additionally, the system is tailored to meet the specific needs of its users, offering features such as farm monitoring, health management, payment processing, and feedback analysis, all of which are essential to the daily operations of dairy farms. These factors contribute to a high likelihood of user satisfaction, making the Dairy Care System well-suited for widespread adoption with minimal resistance.

3.1.4 Feasibility Study Questionnaire

1) What are the primary challenges you face in managing your dairy farm?

- Tracking and managing the health records of animals.
- Monitoring milk production and managing product inventory and sales.
- Ensuring timely payment processing for customers.
- Collecting and analyzing feedback to improve services.

2) How do you currently track milk production and animal health records?

Currently, these records might be tracked manually or through basic digital tools, but the Dairy Care System will provide a centralized platform for more efficient and accurate tracking.

3) What methods do you use for product inventory management?

Product inventory may be managed using basic inventory management software. The Dairy Care System will enhance this by providing a dedicated module for product and inventory management, allowing for real-time updates and tracking.

4) How do you process wages for employees and payments by customers?

Payments and wages are processed manually or using third-party payment systems. The Dairy Care System will integrate payment processing for both online (Razor-Pay) and offline (Cash on Delivery) transactions, streamlining the process for customers.

5) What type of feedback system do you have in place for customer reviews?

There might be a basic feedback collection method in place. The Dairy Care System will implement a structured feedback submission and analysis module to gather and analyze customer feedback effectively.

6) What are the common health issues you encounter with your livestock?

Common health issues include nutritional deficiencies, infections, and diseases. The Dairy Care System's health management module will help in monitoring and addressing these issues promptly.

7) How do you manage the delivery of milk and other dairy products?

Deliveries may be managed through manual scheduling and coordination. The Dairy Care System will improve this with a delivery management module, integrating location sharing and delivery tracking for efficient management.

8) What tools do you use for monitoring feeding schedules and nutritional data?

Feeding schedules and nutritional data might be tracked manually or using basic digital tools. The Dairy Care System will offer a dedicated module for collecting and managing feeding details, ensuring proper nutrition and health of the livestock.

9) How would you benefit from integrating machine learning for forecasting milk production?

Machine learning integration will enable accurate forecasting of milk yield based on historical data, allowing for better planning and optimization of resources, leading to increased productivity and profitability.

10) What improvements do you seek in the existing system for dairy farm management?

- A centralized platform for managing all aspects of the dairy farm.
- Enhanced tracking and monitoring of livestock health and milk production.
- Streamlined product and inventory management.
- Integrated payment processing and delivery management.
- Effective collection and analysis of feedback for continuous improvement.

3.1 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor - Intel Core i5

RAM - 8 G B

Hard disk - 5 0 0 G B S S D

3.2.2 Software Specification

Front End - HTML, CSS, JavaScript (JS)

Back End - Django

Database - MySQL

Client on PC - Windows 11

Technologies used - JS, HTML, CSS

3.3 SOFTWARE DESCRIPTION

3.3.1 HTML

HTML (Hypertext Markup Language) is the standard markup language used for creating web pages and applications. It provides the basic structure for web content by utilizing elements and tags to define headings, paragraphs, links, images, and other types of content. HTML serves as the backbone of any website, enabling browsers to interpret and display information. Its simplicity and accessibility allow developers to create well-structured documents that are easily readable by both humans and machines. By using semantic elements, HTML enhances search engine optimization (SEO) and improves the overall user experience on the web.

3.3.2 MySQL

MySQL is a powerful, open-source relational database management system (RDBMS) known for its reliability, performance, and ease of use. It supports a wide range of applications, from small to large-scale enterprise solutions. MySQL uses structured query language (SQL) for database operations, allowing developers to efficiently manage and manipulate data. Its robust features, such as transactions, security, and data replication, make it a preferred choice for web applications, including those built on the Django framework.

3.3.3 DJANGO

Django is a popular and powerful open-source web framework written in Python, designed to facilitate rapid development and maintainable web applications. It follows the Model-View Template (MVT) architectural pattern, which is similar to the Model-View-Controller (MVC) pattern. Django provides a structured and efficient way to build web applications, offering several key components and features. At its core, Django includes a robust Object-Relational Mapping (ORM) system that simplifies database interactions, allowing developers to work with Python objects instead of raw SQL queries. It also includes a URL dispatcher for mapping URLs to view functions, an automatic admin interface for managing application data, and a templating engine for creating dynamic and reusable user interfaces. Django places a strong emphasis on security, with built-in features to protect against common web vulnerabilities. It offers authentication and authorization systems, middleware support for global request and response processing, and compatibility with various databases. The framework's scalability, extensibility, and a vibrant community of developers make it a popular choice for building web applications, from simple websites to complex, high-traffic platforms.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

The initial stage of developing any engineered product or system is the design phase, which involves a creative approach. A well-crafted design plays a critical role in ensuring the successful functioning of a system. Design is defined as the process of employing various techniques and principles to define a process or system in enough detail to enable its physical realization. This involves using different methods to describe a machine or system, explaining how it operates, in sufficient detail for its creation. In software development, design is a crucial step that is always present, regardless of the development approach. System design involves creating a blueprint for building a machine or product. Careful software design is essential to ensure optimal performance and accuracy. During the design phase, the focus shifts from the user to the programmers or those working with the database. The process of creating a system typically involves two key steps: Logical Design and Physical Design.

4.2 UML DIAGRAM

UML, which stands for Unified Modeling Language, is a standardized language used for specifying, visualizing, constructing, and documenting the elements of software systems. The Object Management Group (OMG) is responsible for the creation of UML, with the initial draft of the UML 1.0 specification presented to OMG in January 1997. Unlike common programming languages such as C++, Java, or COBOL, UML is not a programming language itself. Instead, it is a graphical language that serves as a tool for creating software blueprints. UML is a versatile and general-purpose visual modeling language that facilitates the visualization, specification, construction, and documentation of software systems. While its primary use is in modeling software systems, UML's applications are not limited to this domain. It can also be employed to represent and understand processes in various contexts, including non-software scenarios like manufacturing unit processes. It's important to note that UML is not a programming language, but it can be utilized with tools that generate code in different programming languages based on UML diagrams.

UML encompasses nine core diagrams that aid in representing various aspects of a system.

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Activity diagram

- State chart diagram
- Deployment diagram
- Component diagram

4.2.1 USE CASE DIAGRAM

A use case is a tool for understanding a system's requirements and organizing them, especially in the context of creating or using something like a product delivery website. These tools are represented using "use case" diagrams within the Unified Modeling Language, a standardized way of creating models for real-world things and systems.

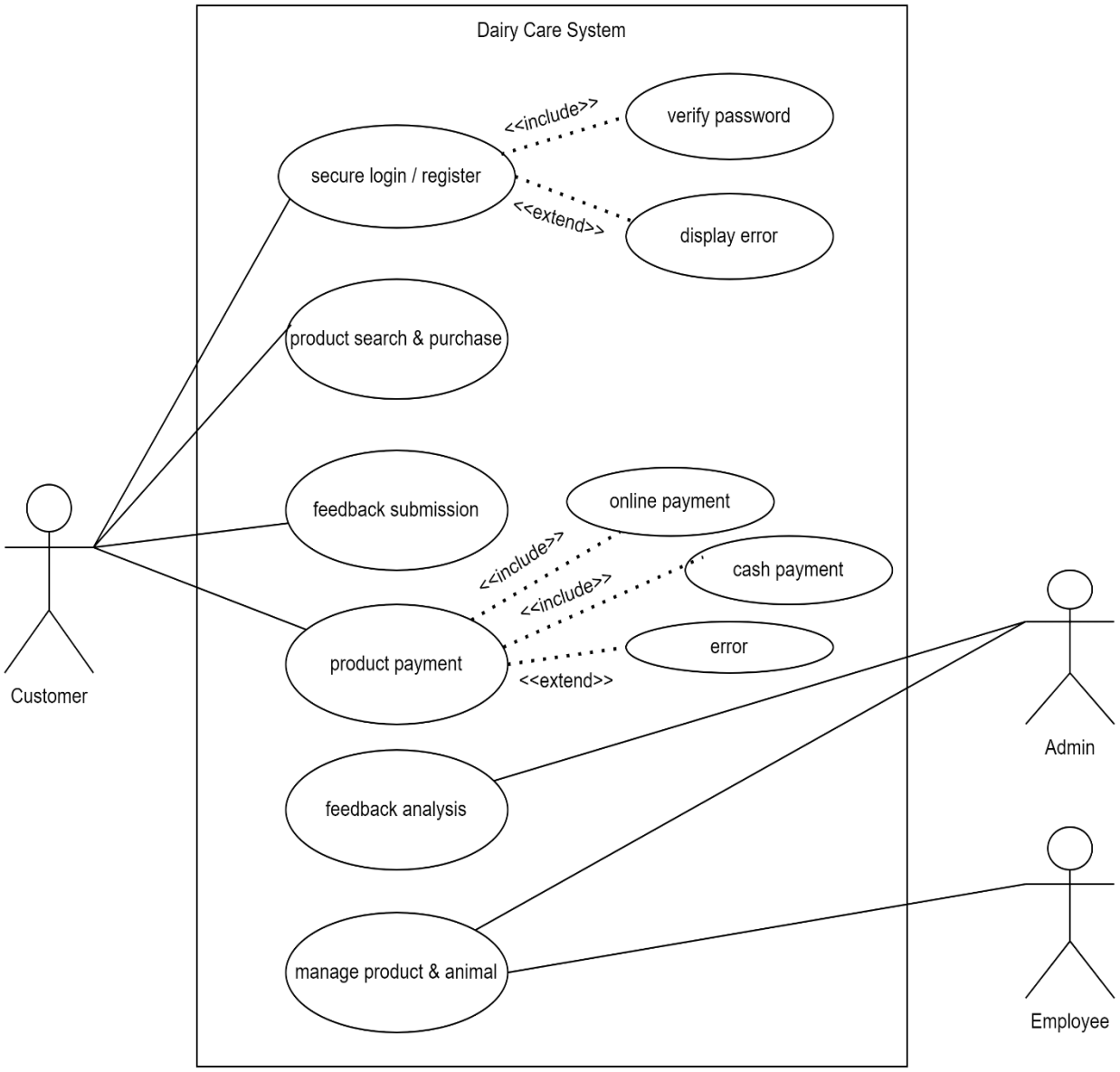


Figure 4.2.1: Use Case Diagram

4.2.2 SEQUENCE DIAGRAM

A sequence diagram illustrates the specific order in which objects interact with each other, showcasing the sequential flow of events. This type of diagram is also known as event diagrams or event scenarios. Sequence diagrams serve the purpose of elucidating how various components of a system collaborate and the precise sequence in which these actions occur. These diagrams find frequent application among business professionals and software developers, aiding in the understanding and depiction of requirements for both new and existing systems.

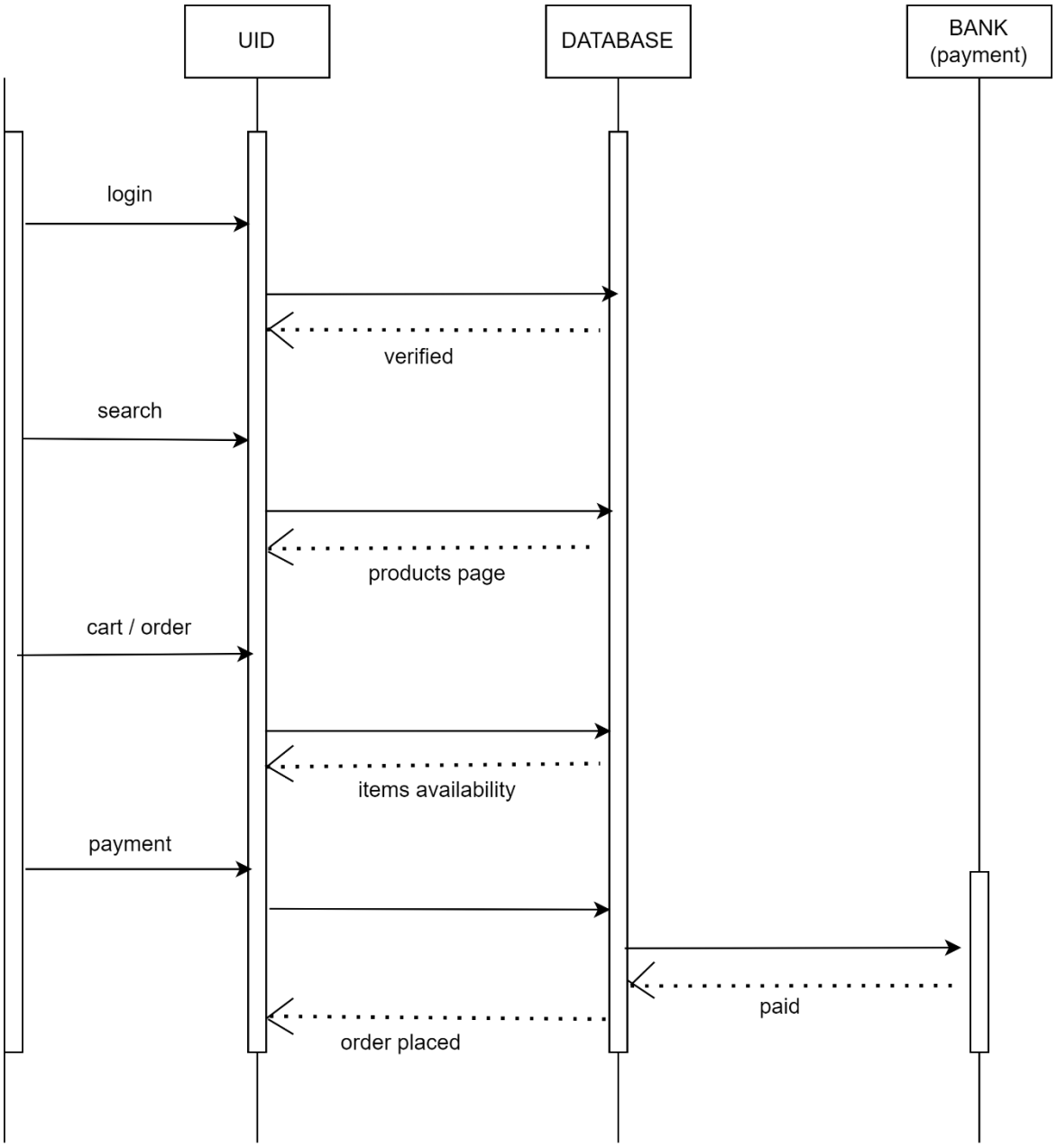


Figure 4.2.2: Sequence Diagram

4.2.3 State Chart Diagram

A state machine diagram, also known as a state chart, visually represents the various states an object undergoes within a system and the sequence in which these states are traversed. It serves as a record of the system's behavior, illustrating the collaborative functioning of a group of entities, whether it's a team, a collection of students, a large assembly, or an entire organization. State machine diagrams are a valuable method for depicting the interactions of diverse components within a system, outlining how objects evolve in response to events and elucidating the diverse conditions that each entity or component can inhabit.

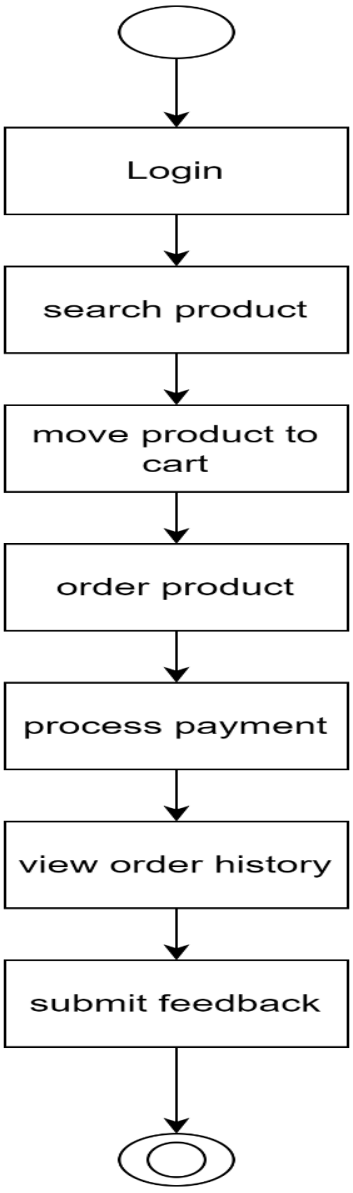


Figure 4.2.3: State Chart Diagram

4.2.4 Activity Diagram

An activity diagram is a visual representation of how events unfold simultaneously or sequentially. It aids in understanding the flow of activities, emphasizing the progression from one task to the next. Activity diagrams focus on the order in which tasks occur and can depict various types of flows, including sequential, parallel, and alternative paths. To facilitate these flows, activity diagrams incorporate elements such as forks and join nodes, aligning with the concept of illustrating the functioning of a system in a specific manner.

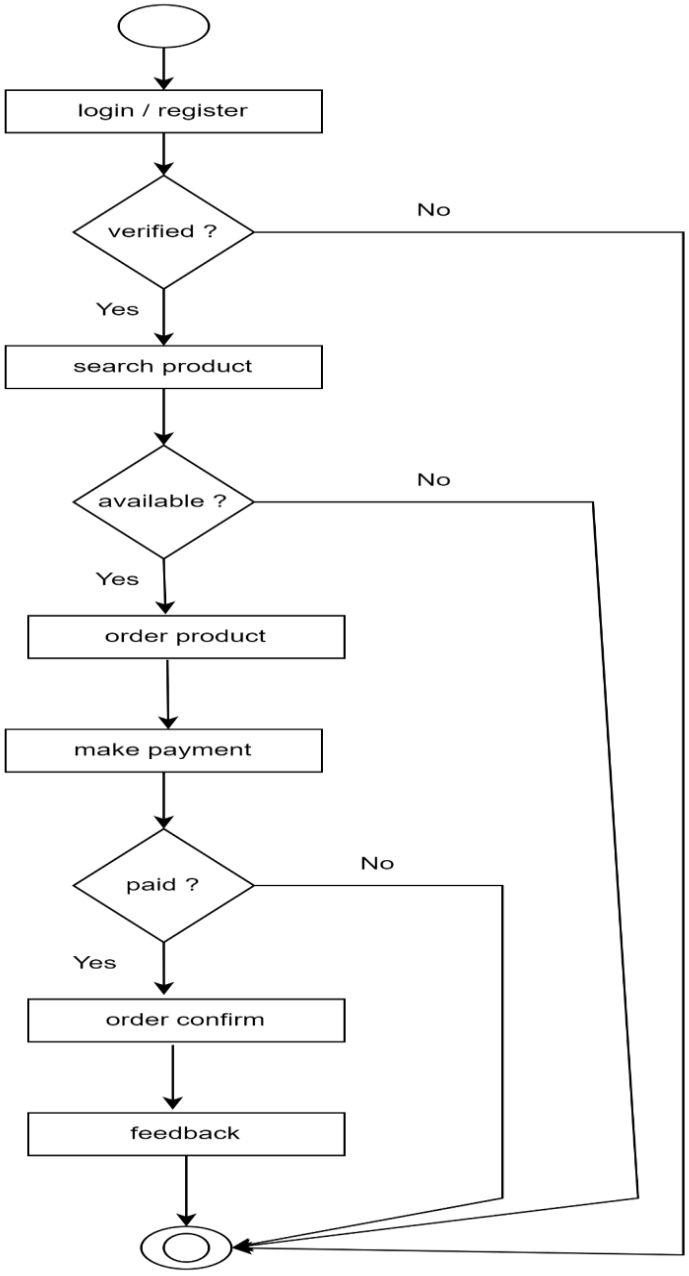


Figure 4.2.4: Activity Diagram

4.2.5 Class Diagram

A class diagram serves as a static blueprint for an application, illustrating its components and their relationships when the system is in a dormant state. It provides insights into the system's structure, showcasing the various elements it comprises and how they interact. In essence, a class diagram acts as a visual guide for software development, aiding in the creation of functional applications.

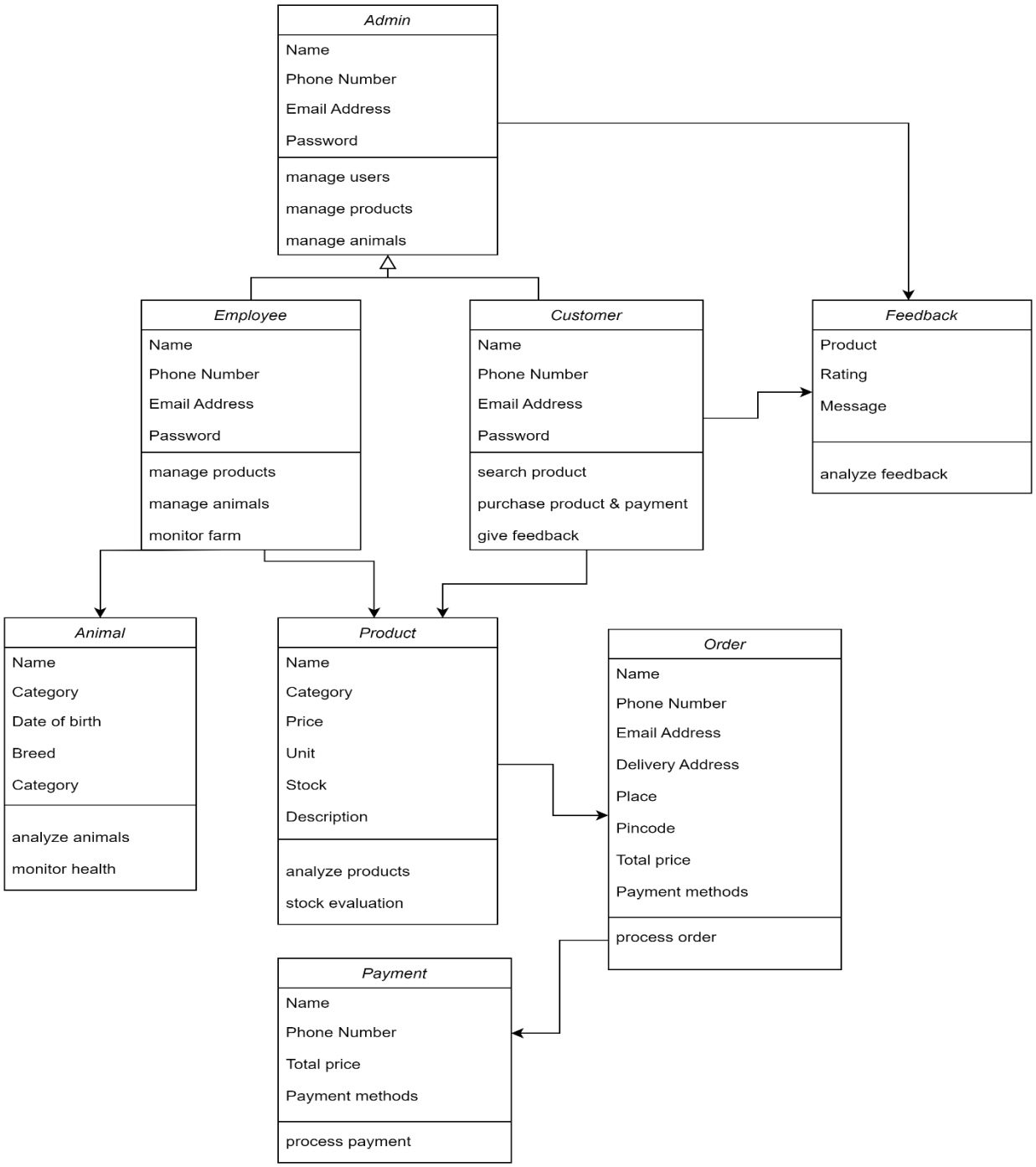


Figure 4.2.5: Class Diagram

4.2.6 Object Diagram

Object diagrams are derived from class diagrams and rely on them to provide a visual representation. They offer an illustration of a collection of objects related to a particular class. It provides a snapshot of objects in an object-oriented system at a specific point in time. Object diagrams and class diagrams share similarities, but they also have distinctions. Class diagrams are more generalized and do not portray specific objects. This abstraction in class diagrams simplifies the comprehension of a system's functionality and structure.

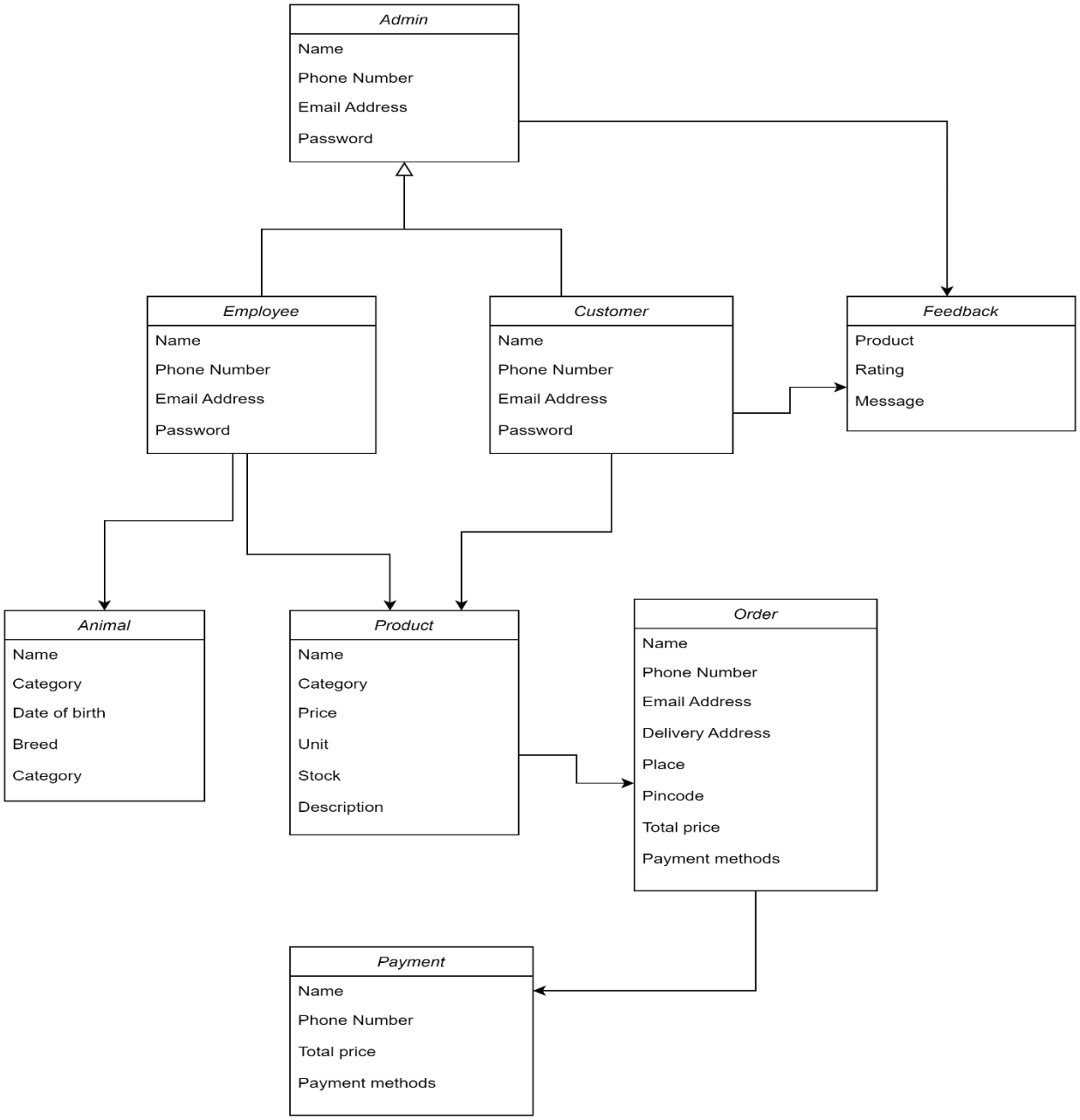


Figure 4.2.6: Object Diagram

4.2.7 Component Diagram

A component diagram serves the purpose of breaking down a complex system that utilizes objects into more manageable segments. It offers a visual representation of the system, showcasing its internal components such as programs, documents, and tools within the nodes.

This diagram elucidates the connections and organization of elements within a system, resulting in the creation of a usable system. In the context of a component diagram, a component refers to a system part that can be modified and operates independently. It retains the secrecy of its internal operations and requires a specific method to execute a task, resembling a concealed box that functions only when operated correctly.

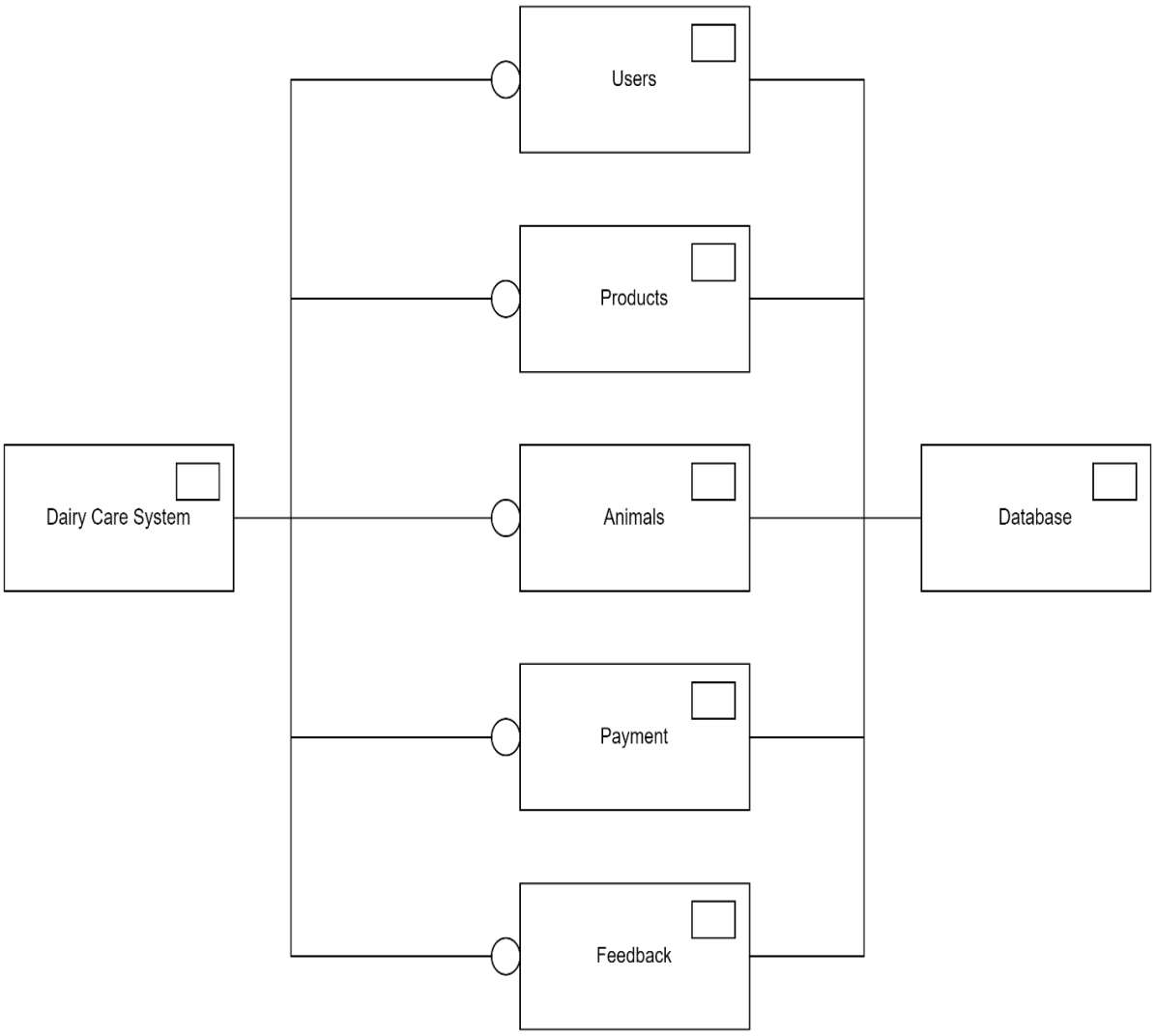


Figure 4.2.7: Component Diagram

4.2.8 Deployment Diagram

A deployment diagram provides a visual representation of how software is positioned on physical computers or servers. It depicts the static view of the system, emphasizing the arrangement of nodes and their connections.

This type of diagram delves into the process of placing programs on computers, elucidating how software is constructed to align with the physical computer system.

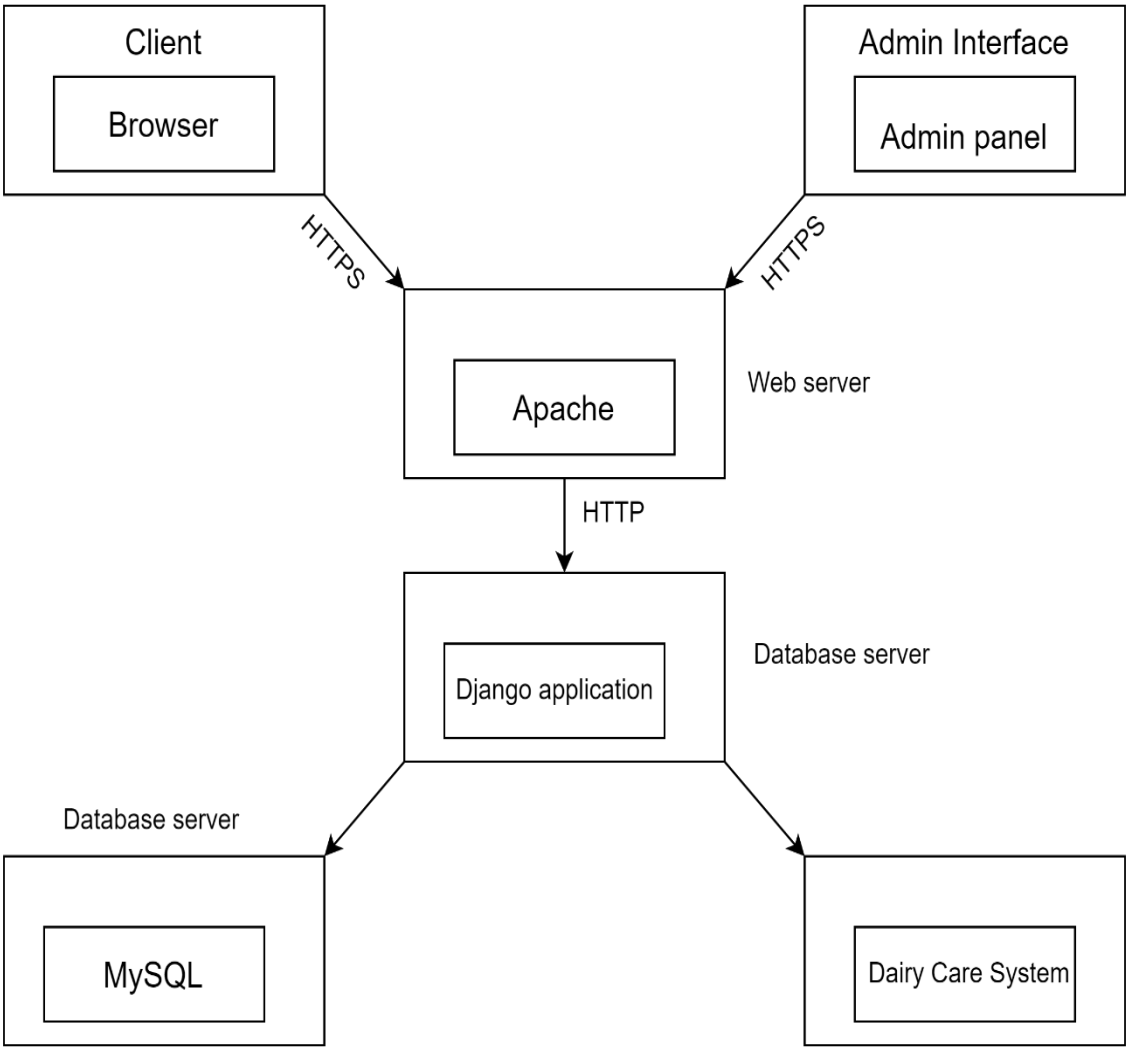


Figure 4.2.8: Deployment Diagram

4.3 USER INTERFACE DESIGN USING FIGMA

4.3.1 Form Name: Home Page

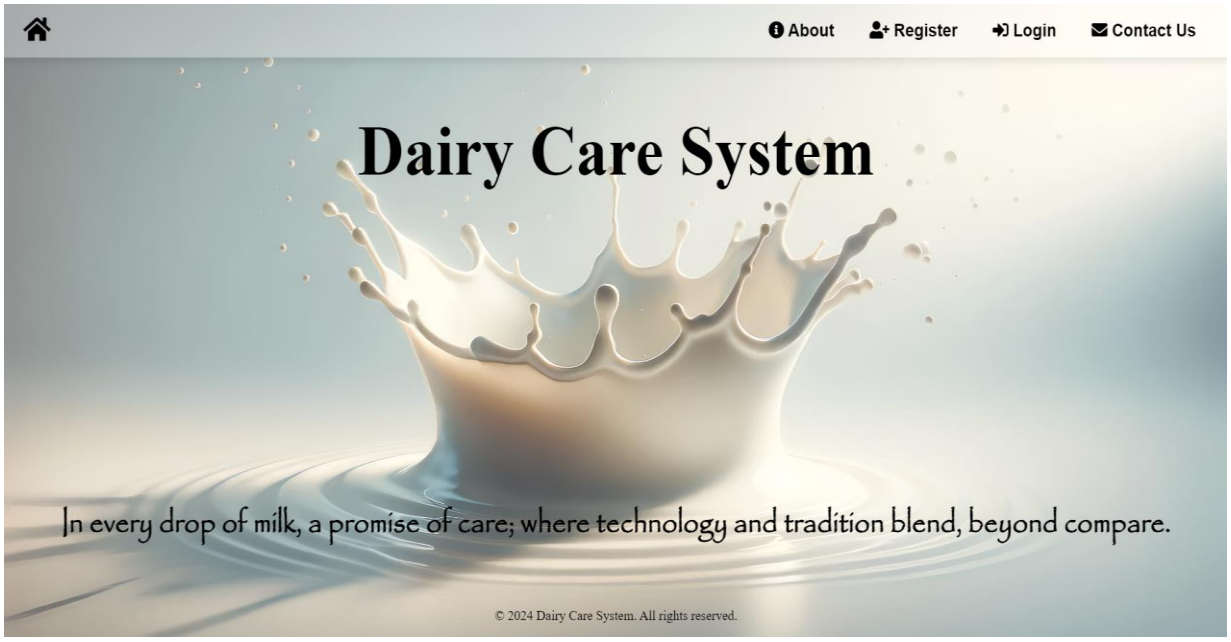


Figure 4.3.1: Home Page

4.3.2 Form Name: Registration Page

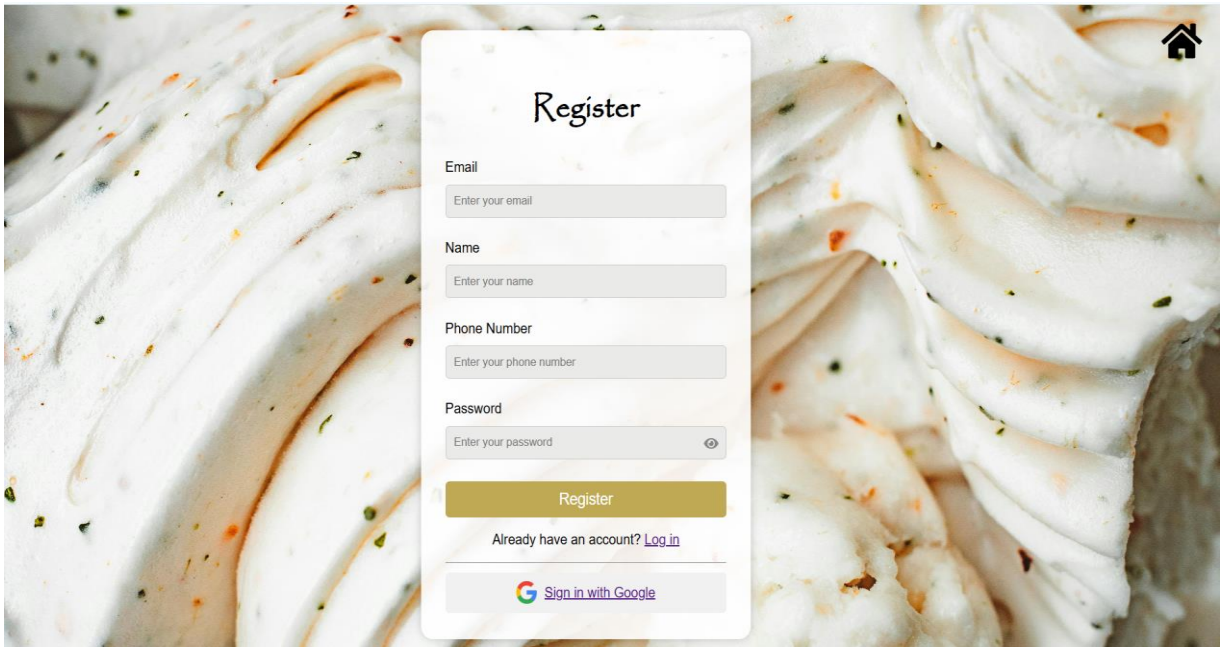


Figure 4.3.2: Registration Page

4.3.3 Form Name: Login Page

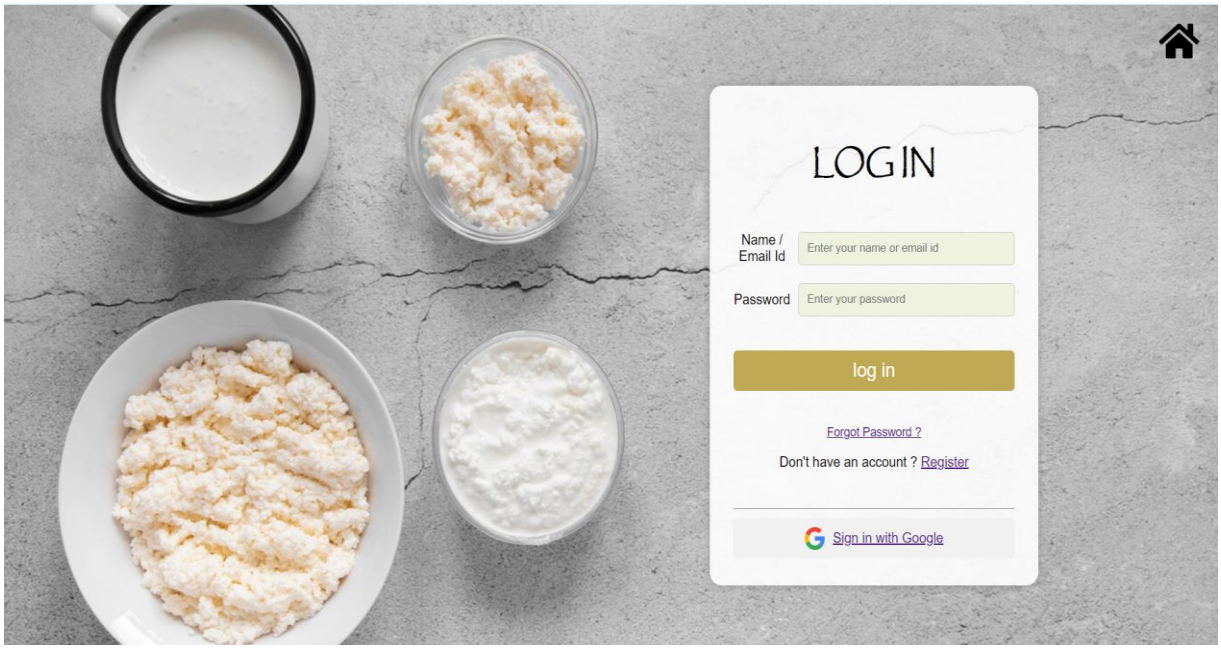


Figure 4.3.3: Login Page

4.3.4 Form Name: Forgot Password Page

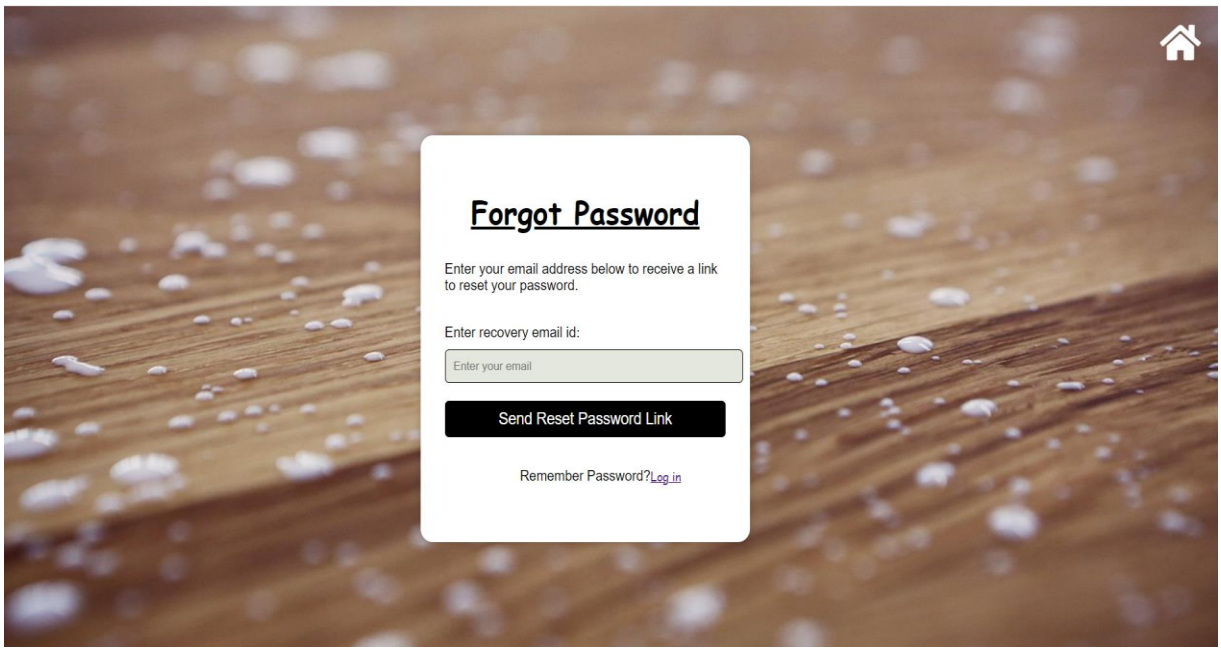


Figure 4.3.4: Forgot Password Page

4.3.5 Form Name: Customer Page

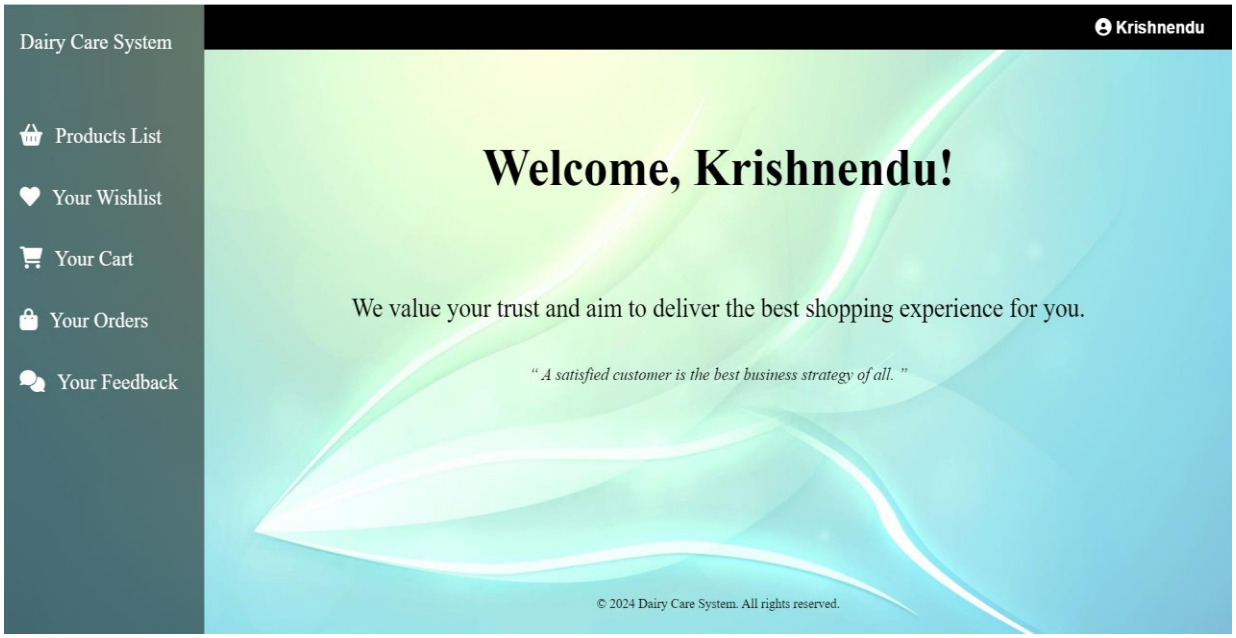


Figure 4.3.5: Customer Page

4.3.6 Form Name: Cart Page

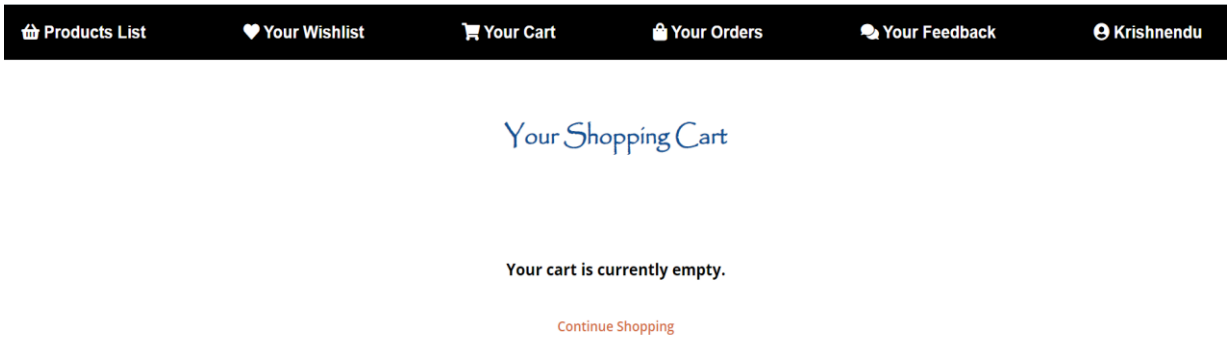


Figure 4.3.6: Cart Page

4.3.7 Form Name: Order Page

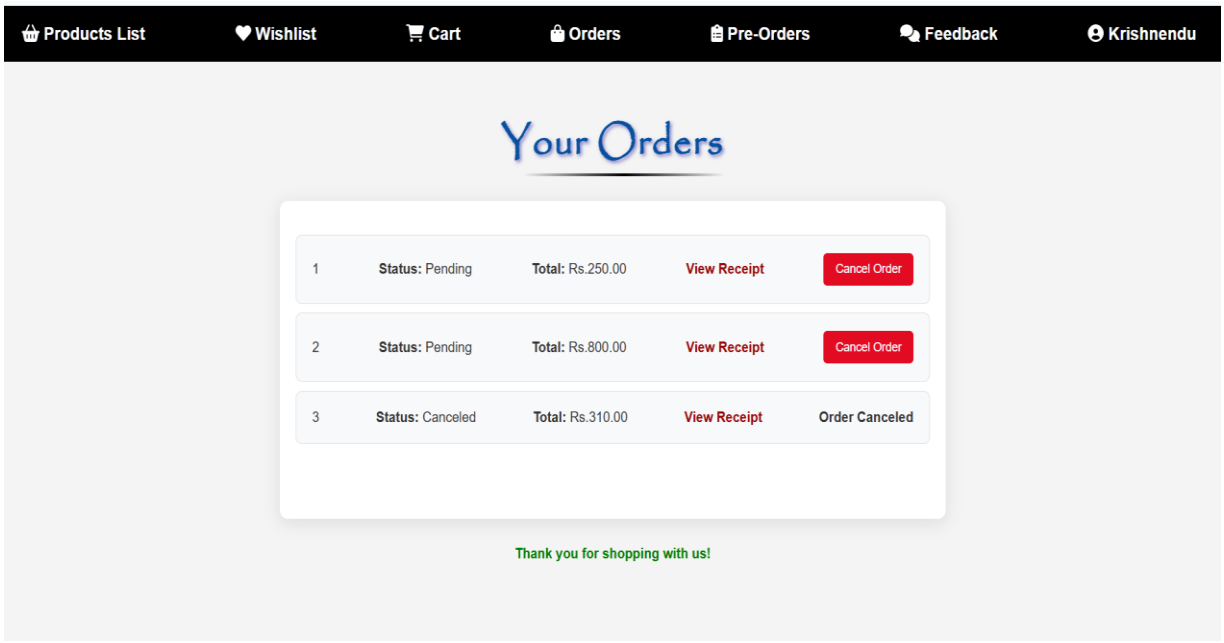


Figure 4.3.7: Order Page

4.3.8 Form Name: Feedback Page

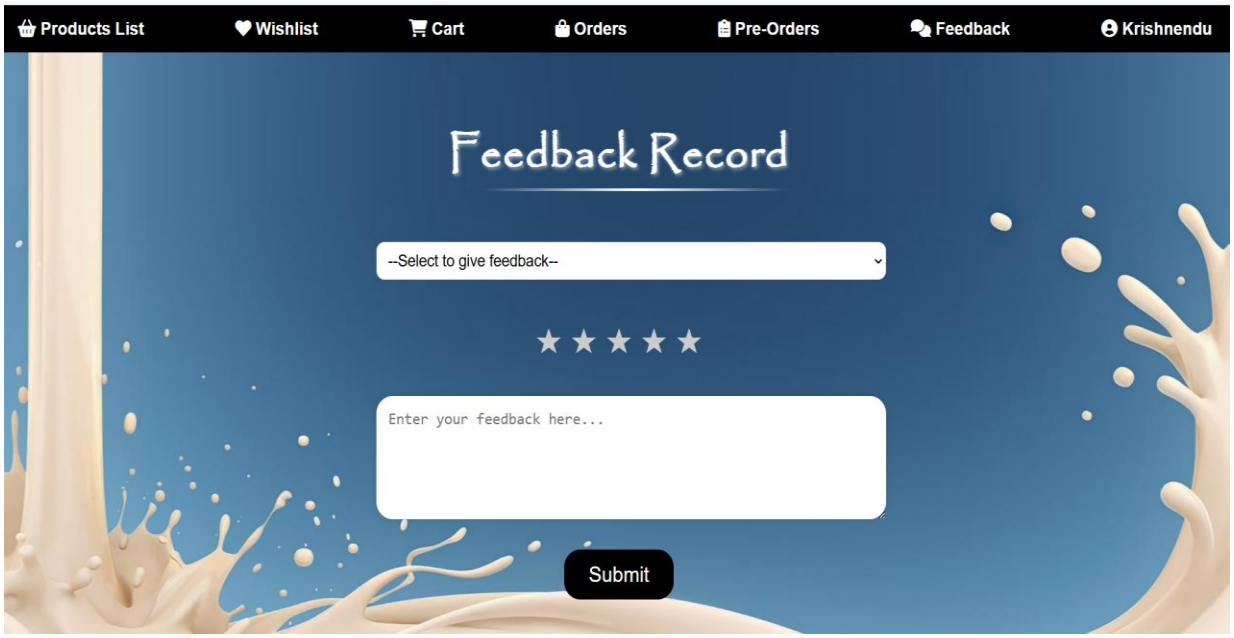


Figure 4.3.8: Feedback Page

4.4 DATABASE DESIGN

4.4.1 Relational Database Management System (RDBMS)

A database is a system designed to store and facilitate the retrieval and utilization of information. The integrity and security of the data within a database are of paramount importance. Creating a database involves two essential steps. Initially, it is crucial to understand the user's requirements and establish a database structure that aligns with their needs. The first step entails devising an organizational plan for the information, which is not reliant on any particular computer program.

Subsequently, this plan is employed to craft a design tailored to the specific computer program that will be employed to construct the database system. This phase is centered on determining how data will be stored within the chosen computer program.

Key aspects of database development include:

- Data Integrity: Ensuring the accuracy and consistency of data stored within the database.
- Data Independence: The ability to modify the data storage structure without affecting the application's access to the data.

4.4.2 Normalization

A way of showing a database as a group of connections is called a relational model. Every relationship is like a chart of information or a group of data. In the formal way of talking about tables, a row is called a tuple, a column header is called an attribute, and the table is called a relation. A bunch of tables with special names make up a relational database. A row in a story shows a group of things that are connected.

Relations, Domains & Attributes

Tables serve as containers for organized and related information. Within a table, the rows are referred to as tuples, and a tuple is essentially an ordered list containing n items. Columns in a table are synonymous with characteristics, each representing a specific aspect of the data.

In a relational database, tables are interconnected, ensuring coherence and meaningful associations between different sets of data. This relational structure forms the foundation of a well-structured database.

A domain can be thought of as a collection of fundamental values with a shared source or data type. Naming domains is a helpful practice as it aids in comprehending the significance of the values they encompass. It's essential to recognize that values within a domain are indivisible.

- Table relationships are established using keys, with the primary key and foreign key being the primary types. Entity Integrity and Referential Integrity are fundamental principles that guide these relationships.
- Entity Integrity mandates that no Primary Key should contain null values, reinforcing the unique and integral nature of primary keys in database relationships.

Normalization is the process of organizing data to ensure its efficient storage and to minimize redundancy while maintaining accuracy and reliability. This approach involves eliminating unnecessary columns and breaking down large tables into smaller, more manageable parts. Normalization helps prevent errors when adding, removing, or modifying data. In data modeling, two key concepts are integral to its normal form: keys and relationships.

Keys serve as unique identifiers to distinguish one row from all others in a table. There are two main types of keys: the primary key, which groups similar records within a table, and the foreign key, which acts as a special code to identify specific records in another table.

1. First Normal Form (1NF): This rule dictates that an attribute can only contain a single value that cannot be further divided. Each value in a tuple must match the attribute's type. In 1NF, tables cannot contain other tables or have tables as part of their data. This form ensures that values are indivisible and represent a single entity. Achieving 1NF often involves organizing data into separate tables, with each table having a designated key based on project requirements. New relationships are established for various data categories or related groups, eliminating redundant information. A relationship adhering to primary key constraints alone is termed the first normal form.
2. Second Normal Form (2NF): In simpler terms, 2NF stipulates that no additional information should be associated with only part of the primary information used for data organization. This involves breaking down the information and creating separate groupings for each part along with its related details. It's essential to maintain a connection between the original primary key and any data dependent on it. This step helps remove data that relies on only a portion of the key. When a set of information has a primary means of identifying each item, and all other details depend solely on that primary means, it is considered to be in the second normal form.

3. Third Normal Form (3NF): It is a critical concept in database normalization, aiming to achieve table independence and control over attributes. In 3NF, a table should not contain columns that depend on other non-key columns, ensuring that each column is functionally dependent only on the primary key. It breaks down relationships involving non-key attributes to eliminate dependencies on attributes not part of the primary key. To meet the criteria for 3NF, data must already be in the Second Normal Form (2NF), and non-key attributes should not rely on other non-key attributes within the same table, avoiding transitive dependencies. This process enhances data integrity, reduces redundancy, and optimizes database structure and organization.

4.4.3 Sanitization

Django incorporates various in-built mechanisms for data sanitization. To begin with, it provides a diverse range of field types that come with automatic validation and sanitization capabilities. For instance, the CharField performs automatic validation and cleaning of text input, while the EmailField ensures that email addresses adhere to the correct format. These field types serve as protective measures to prevent the storage of malicious or invalid data in the database.

Moreover, Django strongly advocates for the utilization of form validation to sanitize user input. Django's forms are equipped with predefined validation methods and validators that can be applied to form fields. These validators execute a range of checks and cleansing operations, such as confirming that numerical values fall within specified ranges or verifying that uploaded files adhere to specific formats. These combined features in Django promote data integrity and security, making it a reliable choice for web application development.

4.4.4 Indexing

The index keeps track of a certain piece of information or group of information, arranged in order of its value. Arranging the index entries helps find things quickly and easily that match exactly or are within a certain range. Indexes make it easy to find information in a database without having to search through every record every time the database is used. An index is like a roadmap for finding information in a database. It helps you look up data quickly and also makes it easy to find records that are in a certain order. It can be based on one or more columns in the table.

4.5 TABLE DESIGN

4.5.1 Users_table

Primary key: user_id

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	User_id	Int	Primary Key	Unique identifier for each user
2	Name	Varchar (50)	Unique, Not Null	Name for user login
3	Email	Varchar (50)	Unique, Not Null	User's email address
4	Phone	Varchar (10)	Unique	User's contact phone number
5	Password	Varchar (50)	Not Null	Encrypted user password
6	Role	Boolean	Not Null	Role of the user
7	Status	Boolean	Default 1	Status indicating if the user is active (1) or inactive (0)
8	Created_at	Timestamp	Not Null	Timestamp for when the user was created
9	Updated_at	Timestamp	Not Null	Timestamp for the last update to the user's information
10	Reset_token	Varchar (255)	Null	Token used for password reset functionality

4.5.2 Products_table

Primary Key: product_id

Foreign Key: image_id reference imagetable ， employee_id reference users_table

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Product_id	Int	Primary Key	Key for this table
2	Image_id	Int	Foreign Key	Foreign key to imagetbl
3	Product_name	Varchar (50)	Not Null	Product name
4	Product_description	Text	Not Null	Product description
5	Product_quantity	Decimal (10, 2)	Not Null	Available quantity of product

6	Product_unit	Varchar (50)	Not Null	Unit of measurement
7	Product_price	Decimal (10, 2)	Not Null	Product price
8	Product_category	Varchar (50)	Not Null	Product category
9	Employee_id	Int	Foreign Key	Foreign key to users_table
10	Status	Boolean	Default 1	Product status
11	Added_at	Timestamp	Not Null	Product added time
12	Updated_at	Timestamp	Not Null	Product last updated time

4.5.3 Imagetable

Primary Key: image_id

Foreign Key: product_id reference product_table

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Image_id	Int	Primary Key	Key for this table
2	Product_id	Int	Foreign Key	Foreign key to product_table
3	Image	Blob	Not Null	Product image path
4	Added_at	Timestamp	Not Null	Image added time
5	Updated_at	Timestamp	Not Null	Image last updated time

4.5.4 order_table

Primary Key: order_id

Foreign Key: user_id reference users_table

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Order_id	Int	Primary Key	Key for this table
2	User_id	Int	Foreign Key	Foreign key to product_table
3	Order_date	Timestamp	Not Null	Order date
4	Status	Varchar (50)	Not Null	Order status

4	Total amount	Decimal (10, 2)	Not Null	Total amount to be paid
5	Payment_status	Varchar (50)	Not Null	Payment status
6	Payment_method	Varchar (50)	Not Null	Payment method
7	Name	Varchar (50)	Not Null	Name of customer
8	Email	Varchar (50)	Not Null	Email of customer
9	Phone	Int	Not Null	Phone number of user
10	Address	Varchar (50)	Not Null	Address of customer
11	Created_at	Timestamp	Not Null	Order added date
12	Updated_at	Timestamp	Not Null	Order last updated date

4.5.5 orderItems

Primary Key: orderitem_id

Foreign Key: order_id reference order_table, product_id reference products_table

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Orderitem_id	Int	Primary Key	Key for this table
2	Order_id	Int	Foreign Key	Foreign key to orders_table
3	Product_id	Int	Foreign Key	Foreign key to products_table
4	Quantity	Decimal (10, 2)	Not Null	Quantity ordered
5	Price_per_unit	Decimal (10, 2)	Not Null	Price per item
6	Total_price	Decimal (10, 2)	Not Null	Total amount
7	Created_at	Timestamp	Not Null	Order added time

4.5.6 wishlist

Primary Key: id

Foreign Key: user_id reference users_table, product_id reference products_table

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Orderitem_id	Int	Primary Key	Key for this table
2	User_id	Int	Foreign Key	Foreign key to orders_table
3	Product_id	Int	Foreign Key	Foreign key to products_table
4	Added_at	Timestamp	Not Null	Item added to wishlist time

4.5.7 feedback table

Primary Key: feedback_id

Foreign Key: user_id reference users_table, product_id reference products_table

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Feedback_id	Int	Primary Key	Key for this table
2	User_id	Int	Foreign Key	Foreign key to orders_table
3	Product_id	Int	Foreign Key	Foreign key to products_table
4	Rating	Int	Not Null	Product rating
5	Feedback_text	Text	Not Null	Feedback message
6	Added_at	Timestamp	Not Null	Item added to wishlist time

4.5.8 cart

Primary Key: cart_id

Foreign Key: user_id reference users_table, product_id reference products_table

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Cart_id	Int	Primary Key	Key for this table
2	User_id	Int	Foreign Key	Foreign key to orders_table
3	Product_id	Int	Foreign Key	Foreign key to products_table
4	Quantity	Decimal (10, 2)	Not Null	Order quantity
5	Added_at	Timestamp	Not Null	Item added to cart time

4.5.9 animals_table

Primary Key: animal_id

Foreign Key: user_id reference users_table

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Animal_id	Int	Primary Key	Key for this table
2	Animal_name	Varchar (50)	Not Null	Name of the livestock
3	Added_by	Int	Foreign Key	Foreign key to users_table
4	Category	Varchar (50)	Not Null	Livestock category
5	Breed	Varchar (50)	Not Null	Livestock breed
6	Date_of_birth	Timestamp	Not Null	Date of birth of livestock
7	Gender	Varchar (50)	Not Null	Gender of the livestock
8	Milk_capacity	Decimal (10, 2)	Not Null	Milking of livestock
9	Status	Boolean	Not Null	Livestock status (1 or 0)
10	Added_at	Timestamp	Not Null	Livestock added time
11	Updated_at	Timestamp	Not Null	Livestock last updated time

4.5.10 animalhealth_table

Primary Key: animal_id

Foreign Key: user_id reference users_table

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Health_id	Int	Primary Key	Key for this table
2	Animal_id	Int	Foreign Key	Foreign key to animals_table
3	Checkup_date	Timestamp	Not Null	Date of last checkup
4	Health_status	Varchar (50)	Not Null	Livestock health status
5	Vaccinations	Varchar (50)	Not Null	Livestock vaccinations taken

6	Treatment_details	Timestamp	Not Null	Treatments for livestock
7	Veterinarian_name	Varchar (50)	Not Null	Veterinarian attended
8	Next_checkup_date	Timestamp	Not Null	Next checkup date
9	Added_by	Int	Foreign Key	Foreign key to users_table
10	Created_at	Timestamp	Not Null	Health status added time
11	Updated_at	Timestamp	Not Null	Health status last updated

4.5.11 animalImages

Primary Key: image_id

Foreign Key: animal_id reference animal_table

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Image_id	Int	Primary Key	Key for this table
2	Animal_id	Int	Foreign Key	Foreign key to animals_table
3	Image	Blob	Not Null	Animal image file path
4	Added_at	Timestamp	Not Null	animal added time

4.5.12 payment_table

Primary Key: payment_id

Foreign Key: user_id reference users_table

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Payment_id	Int	Primary Key	Key for this table
2	User_id	Int	Foreign Key	Foreign key to users_table
3	Amount	Decimal (10, 2)	Not Null	Total amount to be paid
4	Payment_date	Timestamp	Not Null	Payment date
5	Payment_method	Int	Not Null	Method of payment
6	Added_at	Timestamp	Not Null	Payment conducted time

4.5.13 notification_table

Primary Key: notification_id

Foreign Key: product_id reference product_table

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Notification_id	Int	Primary Key	Key for this table
2	Message	Varchar(50)	Not Null	Notification message
3	Is_read	Boolean	Not Null	Flag to note the status of notification
6	Created_at	Timestamp	Not Null	notification send time

CHAPTER 5
SYSTEM TESTING

5.1 INTRODUCTION

Software testing is an essential procedure employed to verify if a computer program functions as intended. It is conducted to ensure that the software performs its designated tasks accurately and complies with the prescribed requirements and standards. Validation is the process of inspecting and assessing software to confirm its adherence to the specified criteria. Software testing is a method for assessing the performance of a program, often in conjunction with techniques like code inspection and program walkthroughs. Validation ensures that the software aligns with the user's expectations and requirements.

Several principles and objectives guide the process of software testing, including:

- Testing is the practice of executing a program with the primary aim of identifying errors.
- An effective test case is one that has a high likelihood of uncovering previously undiscovered errors.
- A successful test is one that exposes previously undiscovered errors.

When a test case operates effectively and accomplishes its objectives, it can detect flaws within the software. This demonstrates that the computer program is functioning as intended and is performing well. The process of evaluating a computer program encompasses three primary aspects:

- Correctness assessment
- Evaluation of implementation efficiency
- Examination of computational complexity

5.2 TEST PLAN

A test plan serves as a comprehensive set of instructions for conducting various types of tests. It can be likened to a map that outlines the steps to follow when evaluating a computer program. Software developers create instructions for both using and organizing the necessary information for the program's proper functionality. They ensure that each component of the program performs its intended functions. To ensure the thorough testing of the software, a group known as the ITG (Information Technology Group) is responsible for verifying its functionality, instead of relying solely on the software's creators for testing.

The objectives of testing should be clearly defined and measurable. A well-structured test plan should encompass details about the frequency of failures, the associated repair costs, the occurrence rate of issues, and the time required for complete testing.

The levels of testing typically include:

- Unit testing
- Integration testing
- Data validation testing
- Output testing

5.2.1 Unit Testing

Unit testing checks the smallest part of a software design - the software component or module. Testing important control paths within a module using the design guide to find errors. This means how difficult the tests are for each small part of a program and what parts of the program haven't been tested yet. Unit testing is a type of testing that looks at how the code works inside and can be done at the same time for different parts of the program.

Before starting any other test, we need to check if the data flows correctly between different parts of the computer program. If the information doesn't move in and out correctly, all other checks are pointless. When designing something, it's important to think about what could go wrong and make a plan for how to deal with those problems. This can mean redirecting the process or stopping it completely. The Sell-Soft System was tested by looking at each part by itself and trying different tests on it. Some mistakes in the design of the modules were discovered and then fixed. After writing the instructions for different parts, each part is checked and tried out separately. We got rid of extra code and made sure everything works the way it should.

5.2.2 Integration Testing

Integration testing is a critical process in software development that involves constructing program while simultaneously identifying errors in the interaction between different program components.

The primary objective is to utilize tested individual parts and assemble them into a program according to the initial plan. This comprehensive testing approach assesses the entire program to ensure its proper and correct functionality.

As issues are identified and resolved during integration testing, it's not uncommon for new problems to surface, leading to an ongoing cycle of testing and refinement. After each individual component of the system is thoroughly examined, these components are integrated to ensure they function harmoniously. Additionally, efforts are made to standardize all programs to ensure uniformity rather than having disparate versions.

5.2.3 Validation Testing or System Testing

The final phase of testing involves a comprehensive examination of the entire system to ensure the correct interaction of various components, including different types of instructions and building blocks. This testing approach is referred to as Black Box testing or System testing.

Black Box testing is a method employed to determine if the software functions as intended. It assists software engineers in identifying all program issues by employing diverse input types. Black Box testing encompasses the assessment of errors in functions, interfaces, data access, performance, as well as initialization and termination processes. It is a vital technique to verify that the software meets its intended requirements and performs its functions correctly.

5.2.4 Output Testing or User Acceptance Testing

System testing is conducted to assess user satisfaction and alignment with the company's requirements. During the development or update of a computer program, it's essential to maintain a connection with the end-users.

This connection is established through the following elements:

- Input Screen Designs.
- Output Screen Designs.

To perform system testing, various types of data are utilized. The preparation of test data plays a crucial role in this phase. Once the test data is gathered, it is used to evaluate the system under investigation. When issues are identified during this testing, they are addressed by following established procedures. Records of these corrections are maintained for future reference and improvement. This process ensures that the system functions effectively and meets the needs of both users and the organization.

5.2.5 Automation Testing

Automated testing is a method employed to verify that software functions correctly and complies with established standards before it is put into official use. This type of testing relies on written instructions that are executed by testing tools. Specifically, UI automation testing involves the use of specialized tools to automate the testing process. Instead of relying on manual interactions where individuals click through the application to ensure its proper functioning, scripts are created to automate these tests for various scenarios. Automating testing is particularly valuable when it is necessary to conduct the same test across multiple computers simultaneously, streamlining the testing process and ensuring consistency.

5.2.6 Selenium Testing

Selenium is a valuable and free tool designed for automating website testing. It plays a crucial role for web developers as it simplifies the testing process. Selenium automation testing refers to the practice of using Selenium for this purpose. Selenium isn't just a single tool; it's a collection of tools, each serving distinct functions in the realm of automation testing. Manual testing is a necessary aspect of application development, but it can be monotonous and repetitive. To alleviate these challenges, Jason Huggins, an employee at ThoughtWorks, devised a method for automating testing procedures, replacing manual tasks. He initially created a tool named the JavaScriptTestRunner to facilitate automated website testing, and in 2004, it was rebranded as Selenium.

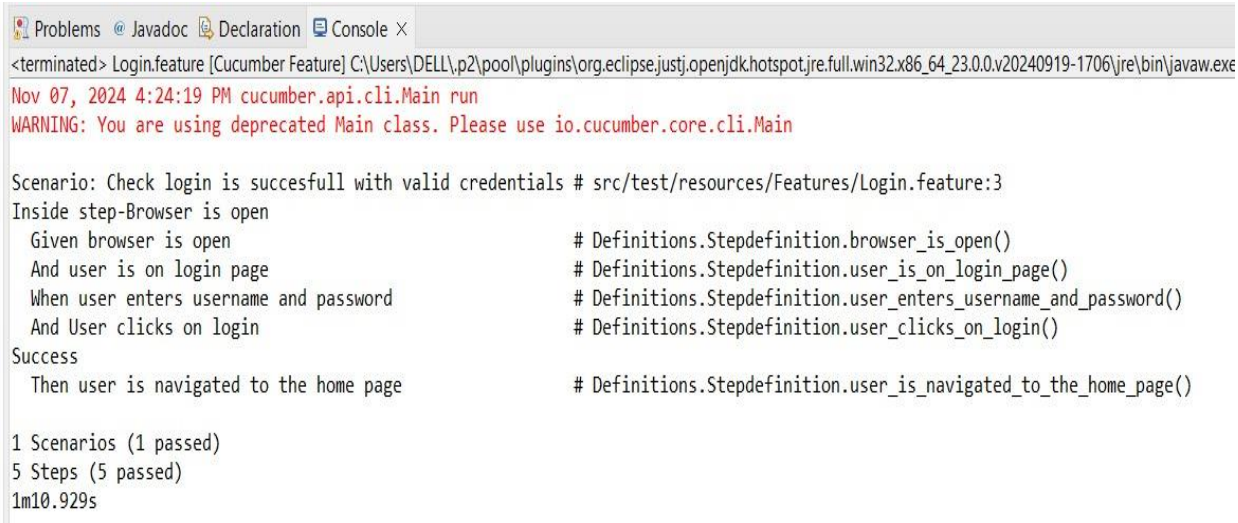
Test Case 1: Login Page Test

Code

```
package Definitions;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class Stepdefinition {

		WebDriver driver=null;

		@Given("browser is open")
		public void browser_is_open() {
		 System.out.println("Inside step-Browser is open");
	System.setProperty("webdriver.gecko.marionette","C:\\Users\\DELL\\eclipse-
workspace\\2025\\src\\test\\resources\\Driver\\geckodriver.exe");
		 driver=new FirefoxDriver();
		 driver.manage().window().maximize();
		}


		@And("user is on login page")
		public void user_is_on_login_page() throws Exception {
		 driver.navigate().to("http://127.0.0.1:8000/login");
		 Thread.sleep(2000);
		}
```

```
@When("user enters username and password")
public void user_enters_username_and_password() throws Throwable{
driver.findElement(By.id("username")).sendKeys("admin11@gmail.com");
 driver.findElement(By.id("password")).sendKeys("Admin@11");
 }


 @When("User clicks on login")
  public void user_clicks_on_login() {
    driver.findElement(By.id("login")).click(); }


 @Then("user is navigated to the home page")
 public void user_is_navigated_to_the_home_page() throws Exception {
         System.out.println("Success");

 }
}
```

Screenshot



Test Report

Test Case 1					
Project Name: Dairy Care System					
Login Test Case					
Test Case ID: Test_1			**Test Designed By: Krishnendu Lal**		
Test Priority (Low/Medium/High): High			**Test Designed Date: 30/10/2024**		
Module Name: Login Module			**Test Executed By: Mr. Amal K Jose**		
Test Title: Login Test Case			**Test Execution Date: 30/10/2024**		
Description:			User has a valid email/password		
Pre-Condition: User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass/ Fail)**
1	Navigate to login page	N/A	Login form should be displayed	Login form is displayed	Pass
2	Provide valid Email	krishnendulal2025@mca.ajce.in	User should be able to login	User logs in	Pass
3	Provide valid password	Krishna@2025			
4	Click on login button	N/A			
Post-Condition: User is validated with database and successfully logs into Userpage					

Test Case 2: Add Products Test

Code

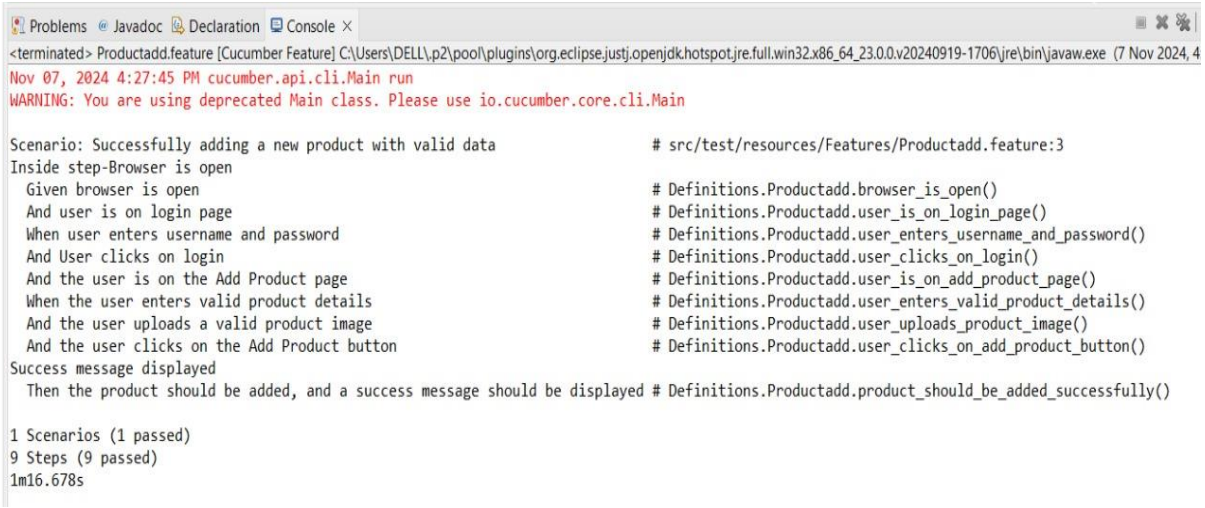
```
package Definitions;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
```

```
import org.openqa.selenium.firefox.FirefoxDriver;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import org.junit.Assert;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import java.time.Duration;

public class Productadd {
   WebDriver driver=null;
   WebDriverWait wait;

   @Given("browser is open")
         public void browser_is_open() {
          System.out.println("Inside step-Browser is open");
         System.setProperty("webdriver.gecko.marionette","C:\\Users\\DELL\\eclipse-
workspace\\2025\\src\\test\\resources\\Driver\\geckodriver.exe");
          driver=new FirefoxDriver();
          driver.manage().window().maximize();
         }

         @And("user is on login page")
         public void user_is_on_login_page() throws Exception {
          driver.navigate().to("http://127.0.0.1:8000/login");
          Thread.sleep(2000);
         }

         @When("user enters username and password")
        public void user_enters_username_and_password() throws Throwable{
          driver.findElement(By.id("username")).sendKeys("admin11@gmail.com");
          driver.findElement(By.id("password")).sendKeys("Admin@11");
         }
```

```
        @When("User clicks on login")
         public void user_clicks_on_login() {
          driver.findElement(By.id("login")).click(); }

    @And("the user is on the Add Product page")
    public void user_is_on_add_product_page() {
       driver.findElement(By.id("product")).click();
       driver.findElement(By.id("addproduct")).click();
    }

    @When("the user enters valid product details")
    public void user_enters_valid_product_details() {
       driver.findElement(By.id("productName")).sendKeys("Peda");
       driver.findElement(By.id("productDescription")).sendKeys("A rich, creamy, traditional Indian sweet made from condensed milk, flavored with cardamom, and garnished with nuts");
       driver.findElement(By.id("productCategory")).sendKeys("Dessert");
       driver.findElement(By.id("productQuantity")).sendKeys("6");
       driver.findElement(By.id("quantityUnit")).sendKeys("kg");
       driver.findElement(By.id("productPrice")).sendKeys("400");
    }

    @And("the user uploads a valid product image")
    public void user_uploads_product_image() {
       WebElement uploadElement = driver.findElement(By.name("product_images"));
       uploadElement.sendKeys("D:\\pictures\\peda.jpg"); // replace with path to an image file
    }

    @And("the user clicks on the Add Product button")
    public void user_clicks_on_add_product_button() {
       driver.findElement(By.id("add_prod")).click();
    }
```

```
    @Then("the product should be added, and a success message should be displayed")
    public void product_should_be_added_successfully() {
        System.out.println("Success message displayed");
        // Close the browser after verification
        driver.quit();
    }
}
```

Screenshot



Test report

Test Case 2	
Project Name: Dairy Care System	
Add Product Test Case	
Test Case ID: Test_2	**Test Designed By: Krishnendu Lal**
Test Priority(Low/Medium/High): High	**Test Designed Date: 03/11/2024**
Module Name: Product Adding Module	**Test Executed By: Mr. Amal K Jose**
Test Title: Add Product	**Test Execution Date: 03/11/2024**
Description:	Test to add a product by owner
Pre-Condition: User has valid username and password	

Step	Test Step	Test Data	Expecte dResult	Actual Result	Status (Pass/ Fail)
1	Navigate to login page	N/A	Login form should be displayed	Login form is displayed	Pass
2	Owner is logged in	"Admin" for username and "admin@11" for password	Username and password fields are successfu lly filled in and logged in	Username and password fields are successfully filled in and logged in	Pass
3	Owner is on add product page	N/A	Owner landed on the add product form page	Owner landed on the add product form page	Pass
4	Owner enters valid product details	Owner enters the data of product details	Owner enters valid product details	Owner enters valid product details	Pass
5	Owner submit the form	N/A	Owner submits the valid form	Owner submits the valid form	Pass
Post-Condition: The Owner is validated with database and successfully logs into Owner page and the product is added successfully					

Test Case 3: Update Profile Test

Code

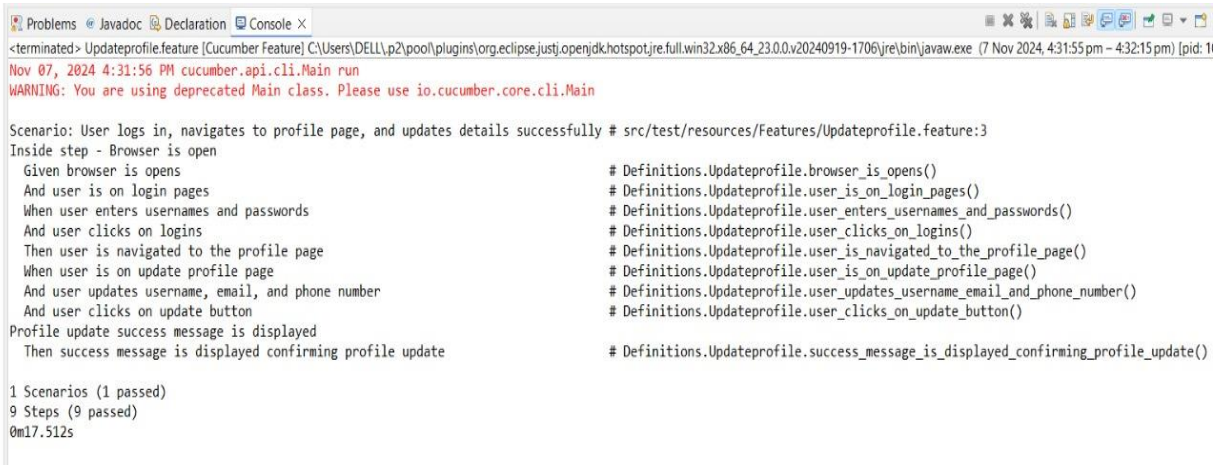
```
package Definitions;


import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
```

```
public class Updateprofile {

    WebDriver driver = null;

    @Given("browser is opens")
    public void browser_is_opens() {
        System.out.println("Inside step - Browser is open");
        System.setProperty("webdriver.gecko.marionette", "C:\\Users\\Sona Maria
Sebastian\\eclipse-
workspace\\CucumberJava\\src\\test\\resources\\drivers\\geckodriver.exe");
        driver = new FirefoxDriver();
        driver.manage().window().maximize();
    }

    @And("user is on login pages")
    public void user_is_on_login_pages() throws InterruptedException {
        driver.navigate().to("http://127.0.0.1:8000/login");
        Thread.sleep(2000);
    }

    @When("user enters usernames and passwords")
    public void user_enters_usernames_and_passwords() {
driver.findElement(By.id("username")).sendKeys("krishnendulal2025@mca.ajce.in");
        driver.findElement(By.id("password")).sendKeys("Krishna@2025");
    }

    @And("user clicks on logins")
    public void user_clicks_on_logins() {
        driver.findElement(By.id("login")).click();
    }

    @Then("user is navigated to the profile page")
    public void user_is_navigated_to_the_profile_page() throws InterruptedException {
        driver.navigate().to("http://127.0.0.1:8000/updateuserprofile");  // URL for the
```

```
update profile page
        Thread.sleep(2000);
    }

    @When("user is on update profile page")
    public void user_is_on_update_profile_page() {
        // Confirm that user is on the profile update page
        WebElement updateProfileHeading = driver.findElement(By.tagName("h1"));
        assert(updateProfileHeading.getText().equals("Profile"));
    }

    @And("user updates username, email, and phone number")
    public void user_updates_username_email_and_phone_number() throws
InterruptedException {

        // Update the username
        driver.findElement(By.id("username")).clear();
        driver.findElement(By.id("username")).sendKeys("Krishnendu");

        // Update the email
        driver.findElement(By.id("email")).clear();
       driver.findElement(By.id("email")).sendKeys("krishnendulal2025@mca.ajce.in");

        // Update the phone number
        driver.findElement(By.id("phone")).clear();
        driver.findElement(By.id("phone")).sendKeys("6282752569");
        Thread.sleep(2000);
    }

    @And("user clicks on update button")
    public void user_clicks_on_update_button() {
        driver.findElement(By.cssSelector(".btn")).click();
    }
```

```
    @Then("success message is displayed confirming profile update")
    public void success_message_is_displayed_confirming_profile_update() throws
InterruptedException {
        System.out.println("Profile update success message is displayed");
    }
}
```

Screenshot



Test report

Test Case 3	
Project Name: Dairy Care System	
Update Profile Test Case	
Test Case ID: Test_3	**Test Designed By: Krishnendu Lal**
Test Priority (Low/Medium/High): High	**Test Designed Date: 03/11/2024**
Module Name: Update Profile Module	**Test Executed By: Mr. Amal K Jose**
Test Title: Update Profile	**Test Execution Date: 03/11/2024**
Description:	Test to update user profile
Pre-Condition: User has valid username and password	

Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/ Fail)
1	Navigate to login page	N/A	Login form should be displayed	Login form is displayed	Pass
2	Provide valid Email	Email : krishnendulal2025@mca.ajce.in	User should be able to login	User logs in	Pass
3	Provide valid password	Password: Krishna@2025			
4	Click on login button				
5	Navigated to Userpage	N/A	User page should be displayed	User page should be displayed	Pass
6	Users view their profile	N/A	User should view their profile	User should view their profile	Pass
7	User updates their profile	N/A	User should update their details	User should update their details	Pass
Post-Condition: User is validated with database and successfully logs into Userpage and navigated to profile page and updates his/her profile.					

Test Case 4: Feedback Test

Code

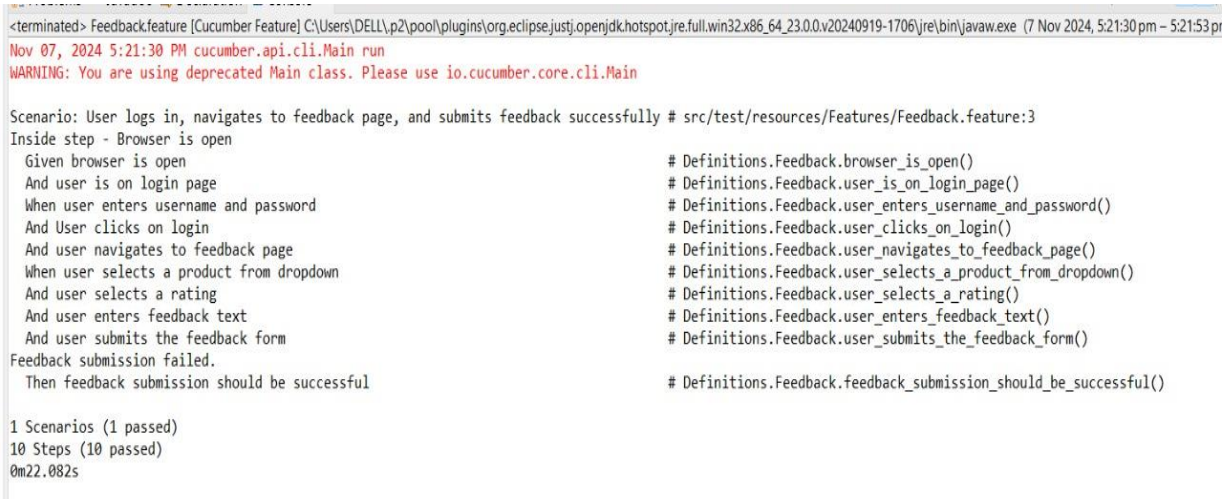
```
package Definitions;


import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.Select;
import org.openqa.selenium.support.ui.WebDriverWait;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.When;
```

```
import io.cucumber.java.en.Then;

public class Feedback {
    WebDriver driver = null;

    WebDriverWait wait;

    @Given("browser is open")
    public void browser_is_open() {
        System.out.println("Inside step - Browser is open");
        System.setProperty("webdriver.gecko.marionette",      "C:\\Users\\DELL\\eclipse-
workspace\\2025\\src\\test\\resources\\Driver\\geckodriver.exe");
        driver = new FirefoxDriver();
        driver.manage().window().maximize();
    }

    @And("user is on login page")
    public void user_is_on_login_page() throws InterruptedException {
        driver.navigate().to("http://127.0.0.1:8000/login");
        Thread.sleep(2000);
    }

    @When("user enters username and password")
    public void user_enters_username_and_password() {
driver.findElement(By.id("username")).sendKeys("krishnendulal2025@mca.ajce.in");
        driver.findElement(By.id("password")).sendKeys("Krishna@2025");
    }

    @And("User clicks on login")
    public void user_clicks_on_login() {
        driver.findElement(By.id("login")).click();
    }

    @And("user navigates to feedback page")
```

```
public void user_navigates_to_feedback_page() throws InterruptedException {
    // Navigating to feedback page after login
    driver.findElement(By.id("feed")).click();
    Thread.sleep(2000);
}

@When("user selects a product from dropdown")
public void user_selects_a_product_from_dropdown() {
    // Select a product from dropdown
    WebElement productDropdown = driver.findElement(By.id("product"));
    Select selectProduct = new Select(productDropdown);
    selectProduct.selectByIndex(1); // Select the first product (update index as needed)
}

@And("user selects a rating")
public void user_selects_a_rating() {
    // Select 5-star rating
    WebElement starRating = driver.findElement(By.id("star1"));
    starRating.click();
}

@And("user enters feedback text")
public void user_enters_feedback_text() {
    // Enter feedback
    WebElement feedbackText = driver.findElement(By.id("feedback"));
    feedbackText.sendKeys("This is a test feedback submission.");
}

@And("user submits the feedback form")
public void user_submits_the_feedback_form() {
    // Submit the feedback form
    driver.findElement(By.id("submit")).click();
}
```

```
    @Then("feedback submission should be successful")
    public void feedback_submission_should_be_successful() {
        // Check for success message
        try {
            WebElement successMessage = driver.findElement(By.className("success"));
            System.out.println("Feedback submitted successfully: " +
successMessage.getText());
        } catch (Exception e) {
            System.out.println("Feedback submission failed.");
        } finally {
            driver.quit();
        }
    }
}
```

Screenshot



Test report

Test Case 4					
Dairy Care System					
Feedback Test Case					
Test Case ID: Test_4			**Test Designed By: Krishnendu Lal**		
Test Priority(Low/Medium/High): High			**Test Designed Date: 04/11/2024**		
Module Name: Feedback Module			**Test Executed By: Mr. Amal K Jose**		
Test Title: Feedback Recording			**Test Execution Date: 04/11/2024**		
Description:			Test to record feedback from users		
Pre-Condition: User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigate to login page	N/A	Login form should be displayed	Login form is displayed Pass	Pass
2	Provide valid Email	Email : krishnendulal2025@mca.ajce.in	User should be able to login	User logs in	Pass
3	Provide valid password	Password: Krishna@2025			
4	Click on login button				
5	Navigated to Userpage	N/A	User page should be displayed	User page should be displayed	Pass
6	Navigate to feedback page	N/A	User should be displayed feedback page	User should be displayed feedback page	Pass
7	User record their feedback	N/A	User should be able to record feedback	User should be able to record feedback	Pass
Post-Condition: User is validated with database and successfully logs into Userpage and navigate to feedback page and enter his/her feedback					

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

Implementation is the stage where a planned system transforms into a tangible and operational entity. Building trust and instilling confidence in users is of paramount importance for the success of the system. This is a critical phase that places a strong emphasis on user training and the creationof informative materials. The actual transition often occurs during or after user training. It signifies the act of putting a new system into action after its design phase, turning a conceptual design into a functional one.

Implementation involves the process of transitioning from the old method of operation to the new one, whether it replaces the old system entirely or makes incremental changes. Ensuring that the process is executed accurately is vital to establish a system that aligns well with the organization's requirements. System implementation encompasses the following tasks:

- Meticulous planning.
- Assessment of the existing system and its constraints.
- Designing methods to facilitate the transition.

6.2 IMPLEMENTATION PROCEDURES

Software implementation involves the installation of software in its intended location and ensuringthat it functions as intended. In some organizations, an individual who is not directly involved in using the software may oversee and approve the project's development. Initially, there may be someskepticism regarding the software, and it's essential to address these concerns to prevent strong resistance.
To ensure a successful transition, several key steps are important:

- Communicating Benefits: Users need to understand why the new software is an improvement over the old one, instilling trust in its capabilities.
- User Training: Providing training to users is crucial to make them feel confident and comfortable using the application.

To evaluate the outcome, it is essential to verify that the server program is running on the server. If the server is not operational, the expected outcomes will not be achieved.

6.2.1 User Training

User training is a critical component of ensuring that individuals know how to use and adapt to a new system. It plays a vital role in fostering user comfort and confidence with the system. Even when a system becomes more complex, the need for training becomes even more apparent.

User training covers various aspects, including inputting information, error handling, querying databases,and utilizing tools for generating reports and performing essential tasks. It aims to empower individuals with the knowledge and skills needed to effectively interact with the system.

6.2.2 Training on the Application Software

Once the fundamental computer skills have been covered, the next step is to instruct individuals onusing a new software program. This training should provide a comprehensive understanding of the new system, including navigation through screens, accessing help resources, error management, andresolution procedures. The training aims to equip users or groups with the knowledge and skills necessary to effectively utilize the system or its components. It's important to note that training maybe tailored differently for various groups of users and individuals in different roles within the organization to cater to their specific needs and requirements.

6.2.3 System Maintenance

Maintaining a system's functionality is a complex challenge in software development. In the maintenance phase of the software life cycle, the software performs critical functions and operatessmoothly. Once a system is operational, it requires ongoing care and maintenance to ensure its continued optimal performance. Software maintenance is a crucial aspect of the development process as it ensures that the system can adapt to changes in its environment. Maintenance involvesmore than just identifying and correcting errors in the code. It encompasses various tasks that contribute to the system's stability and effectiveness.

6.2.4 Hosting

Hosting refers to the process of storing and serving website files on a remote server, which can be accessed by visitors over the internet. When you create a website, you need to have it hosted on a server so that it can be available for others to view. There are various types of hosting options available such as shared hosting, dedicated hosting, VPS hosting, cloud hosting, etc. The choice of hosting depends on the size of the website, its traffic volume, and the level of control and flexibility required by the owner.

Render

Render is a cloud hosting service that provides a streamlined platform for deploying web applications, APIs, static sites, and databases. It simplifies the process of hosting applications by automating tasks such as server setup, scaling, and load balancing, which allows developers to focus on building their applications rather than managing infrastructure. Render supports various languages and frameworks, including Django, and offers seamless integration with Git for continuous deployment.

Procedure for hosting a website on render

Step 1: Login to your render account

Step 2: Create a new project

Step 3: Create a requirements.txt file which will include all your dependencies to be installed.

Step 4: Upload the content which is to be hosted into GitHub

Step 5: Change the setting of your project to include the host also and then deploy.

Hosted Website: Render

Hosted Link: https://dairy-care-system.onrender.com

Hosted QR:

Screenshot

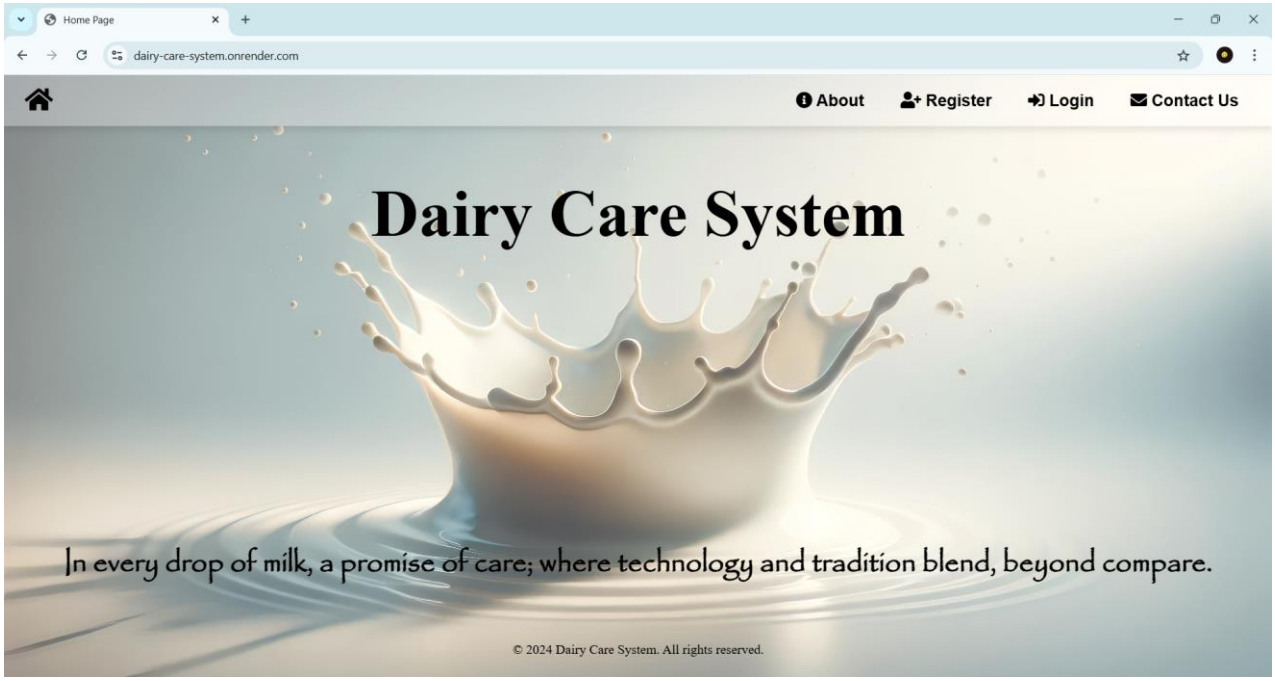


Figure: Hosted Landing page

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The analysis of the Dairy Care System demonstrates it as a well-structured and efficient platform for managing dairy farm operations, tailored to meet the unique needs of a private dairy farm. The system effectively addresses the complexity of managing livestock, monitoring farm activities, and overseeing product sales, while maintaining a user-friendly interface and secure environment. Its comprehensive feature set supports farm owners, employees, and customers, making the system a valuable tool for streamlining dairy farm management.

The Dairy Care System integrates essential features, including product and inventory management, animal health monitoring, and payment processing, all accessible through an intuitive interface. For the admin, the system provides a centralized dashboard for user and inventory management, farm monitoring, and order processing. By leveraging Django for backend functionality and a modern frontend design, the system ensures reliability, security, and efficiency in daily operations. This project offers a practical solution for dairy management, enhancing operational effectiveness and contributing to improved productivity across farm activities.

7.2 FUTURE SCOPE

The future scope of the Dairy Care System project is promising, with potential expansions and improvements to further support the efficient management of dairy farm operations. Enhancements could include the integration of IoT devices for real-time monitoring of livestock health, environmental conditions, and automated feeding systems. Such advancements would provide continuous data updates to the system, enabling more precise health monitoring and proactive care.

Additionally, expanding the analytics and reporting features to provide predictive insights on milk production, sales trends, and resource needs could further aid in decision-making. Incorporating AI-based recommendations for animal nutrition and predictive maintenance of farm could help optimize farm operations, reduce costs, and improve productivity. These future enhancements would make the Dairy Care System even more robust, adaptable, and valuable in the evolving landscape of dairy farm management.

CHAPTER 8
BIBLIOGRAPHY

REFERENCES:

- Django for Beginners: Build Websites with Python and Django by William S. Vincent
- Python Crash Course by Eric Matthes

WEBSITES:

- https://docs.djangoproject.com
- https://docs.python.org/3/
- https://razorpay.com/docs/
- https://getbootstrap.com

CHAPTER 9

APPENDIX

9.1 Sample Code

Login Page

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login Page</title>
    <link rel="stylesheet" href="{% static 'css/login.css' %}">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css"> <!-- Font Awesome CDN -->
    <style>
        body {
            background: url('{% static "images/login.jpg" %}') no-repeat center center fixed;
            background-size: cover;
            font-family: Arial, sans-serif;
        }

        .form-group input {
            width: 100%; /* Full width */
            padding: 10px; /* Same padding for both fields */
            box-sizing: border-box; /* Makes sure padding doesn't affect overall width */
        }

        .password-input-wrapper {
            position: relative;
            width: 100%;
        }

        .password-input-wrapper input {
            width: 100%; /* Ensure password input takes full width */
```

```
            padding: 10px; /* Same padding as username */
            box-sizing: border-box; /* Ensure padding and icon don't affect the width */
        }

        .toggle-password {
            position: absolute;
            right: 10px;
            top: 50%;
            transform: translateY(-50%);
            cursor: pointer;
            color: #888;
        }
        .toggle-password:hover {
            color: #333;
        }

        footer {
            text-align: center;
            margin-top: 10px;
            color: #000000;
        }
    </style>
</head>
<body>

    <a href="{% url 'home' %}">
        <i class="fas fa-home home-icon"></i>
    </a>

    <div class="container">
        <div class="login-box">
            <h2><b>LOG IN</b></h2><br>
            <form id="loginForm" method="POST">
                {% csrf_token %}
```

```
            <div class="form-group">
            <label for="username">Name / Email Id</label>
           <input type="text" id="username" name="username" placeholder="Enter your name
or email id">
              <div class="error-message" id="usernameError"></div>
           </div>
           <div class="form-group">
              <label for="password">Password</label>
              <div class="password-input-wrapper">
                 <input type="password" id="password" name="password" placeholder="Enter
your password">
                 <i class="fas fa-eye toggle-password" id="togglePassword"></i>
              </div>
              <div class="error-message" id="passwordError"></div>
           </div><br>
           <button type="submit" id="login">log in</button><br><br>

           {% if messages %}
           <div class="error-message" id="formError">
              {% for message in messages %}
                 {{ message }}<br>
              {% endfor %}
           </div>
           {% endif %}

          <p class="forgot-password"><a href="forgotpassword">Forgot Password ?</a></p>
          <p class="no-account">Don't have an account ? <a
href="regmailverify">Register</a></p><br>
           <hr>
           <div class="google-signin">
              <img src='{% static "images/google.png" %}' alt="Google Logo">
              <a href="{% url 'social:begin' 'google-oauth2' %}">  Sign in with Google</a>
           </div>
        </form>
```

```
    </div>
</div>

<script>
    document.addEventListener('DOMContentLoaded', function () {
        function validateLoginForm() {
            let username = document.getElementById('username');
            let password = document.getElementById('password');
            let isValid = true;

            // Reset error messages
            document.getElementById('usernameError').textContent = '';
            document.getElementById('passwordError').textContent = '';

            // Validate Username/Email (No changes here)
            let usernameError = '';
            if (!username.value) {
                usernameError = 'Name or Email is required.';
                isValid = false;
            } else if (!validateEmail(username.value) && !validateUsername(username.value))
            {
                usernameError = 'Enter a valid email or name.';
                isValid = false;
            }
            document.getElementById('usernameError').textContent = usernameError;

            // Validate Password (Only check for minimum 8 characters)
            let passwordError = '';
            if (!password.value) {
                passwordError = 'Password is required.';
                isValid = false;
            } else if (password.value.length < 8) {
                passwordError = 'Password must be at least 8 characters long.';
                isValid = false;
```

```
        }
        document.getElementById('passwordError').textContent = passwordError;

        return isValid;
    }

    function validateEmail(email) {
        const emailPattern = /^[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$/;
        return emailPattern.test(email);
    }

    function validateUsername(username) {
        const usernamePattern = /^[a-zA-Z0-9._-]{3,150}$/;
        return usernamePattern.test(username);
    }

    function checkFieldOrder(field) {
        let username = document.getElementById('username').value;
        let password = document.getElementById('password').value;

        if (field === 'password' && !username) {
            document.getElementById('usernameError').textContent = 'Please enter name/
Email before entering Password.';
            document.getElementById('password').value = '';  // Reset the password field
        }

        if (field === 'username' && password) {
            document.getElementById('passwordError').textContent = '';
        }
    }

    document.getElementById('username').addEventListener('input', function() {
        checkFieldOrder('username');
    });
```

```
        document.getElementById('password').addEventListener('input', function() {
            checkFieldOrder('password');
        });

        document.getElementById('loginForm').addEventListener('submit', function(event) {
            if (!validateLoginForm()) {
                event.preventDefault();
            }
        });

        const togglePassword = document.getElementById('togglePassword');
        const password = document.getElementById('password');

        togglePassword.addEventListener('click', function () {
            const type = password.getAttribute('type') === 'password' ? 'text' : 'password';
            password.setAttribute('type', type);
            this.classList.toggle('fa-eye');
            this.classList.toggle('fa-eye-slash');
        });
    });
  </script>
</body>
</html>
```

9.2 Screen Shots

9.2.1 Home Page

Figure 9.2.1: Home Page

9.2.2 Registration Page

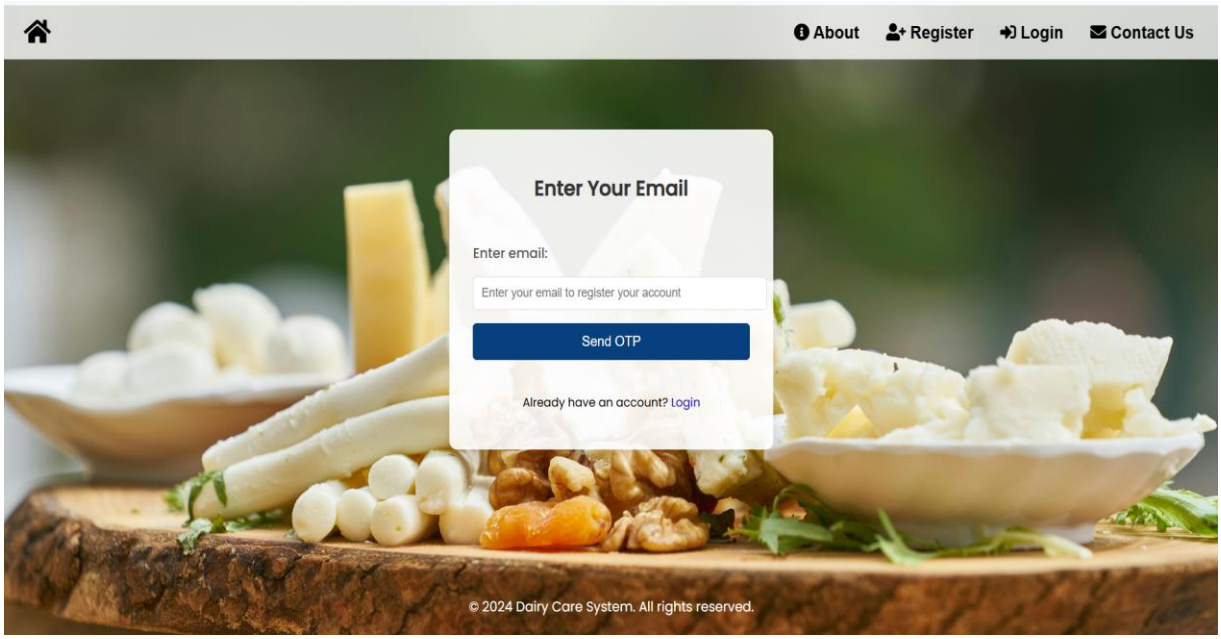


Figure 9.2.2: Registration Page

9.2.3 Login Page

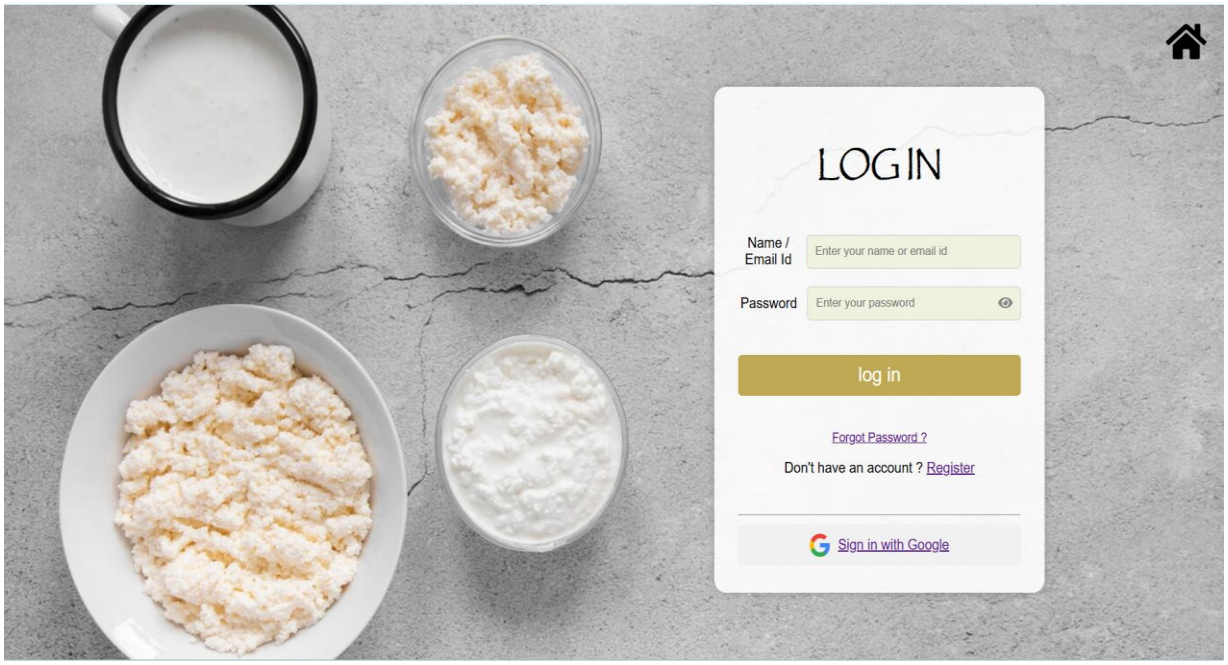


Figure 9.2.3: Login Page

9.2.4 Customer Page

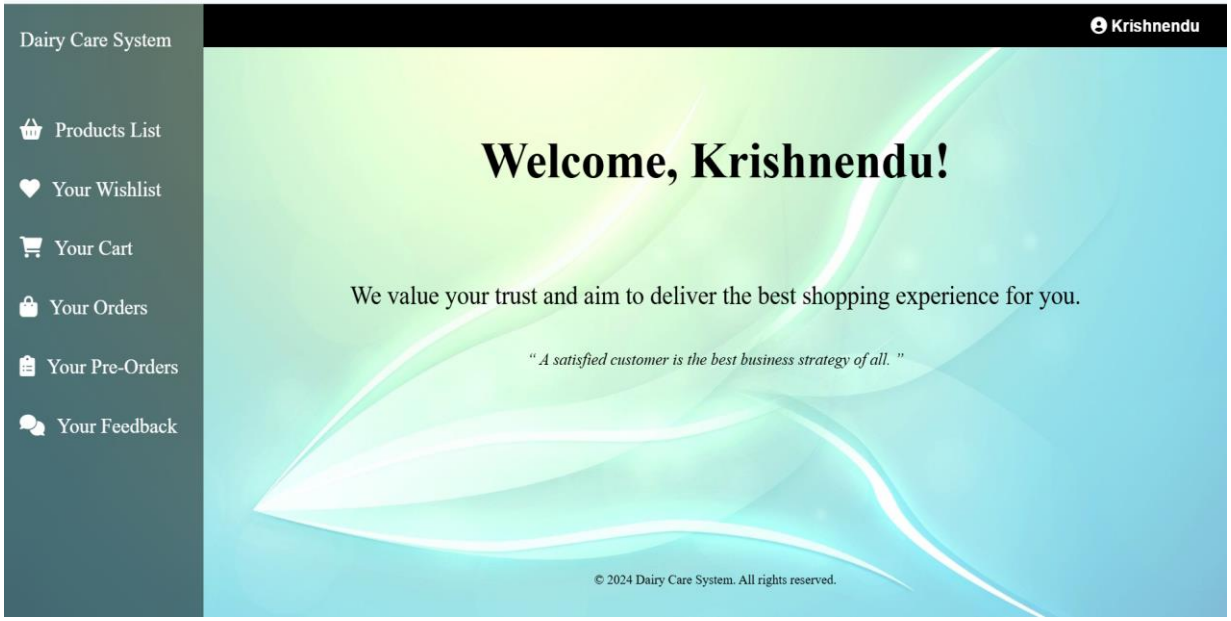


Figure 9.2.4: Customer Page

9.2.5 Products Page

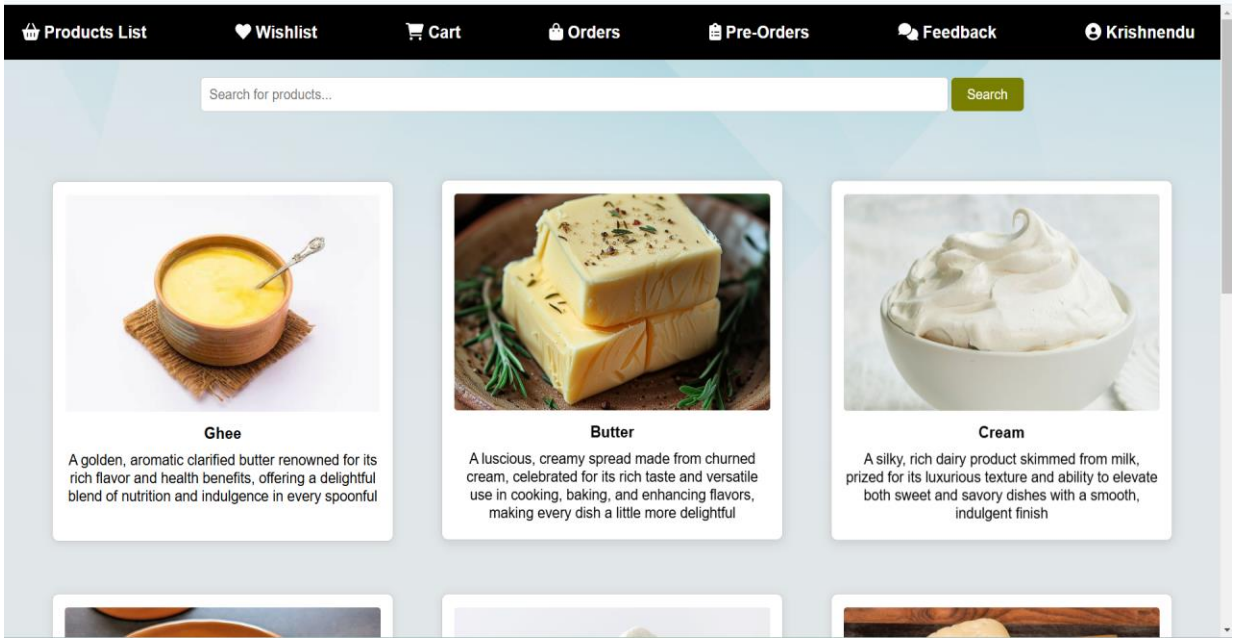


Figure 9.2.5: Products Page

9.2.6 Products Details Page

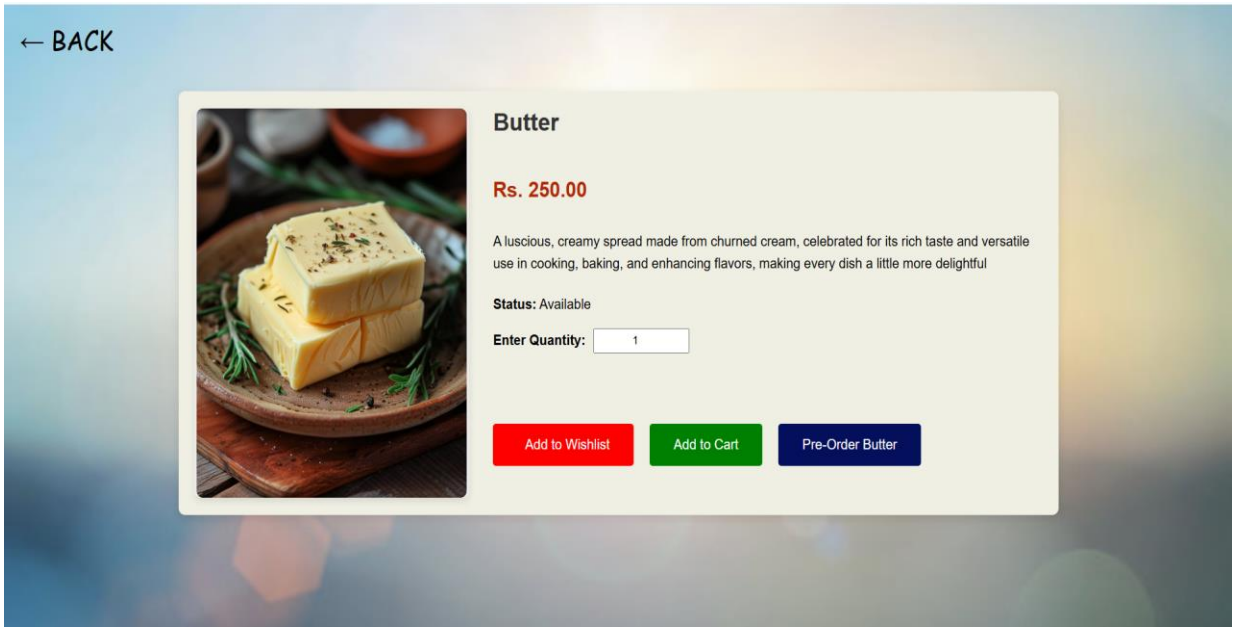


Figure 9.2.6: Products Details Page

9.2.7 Cart Page

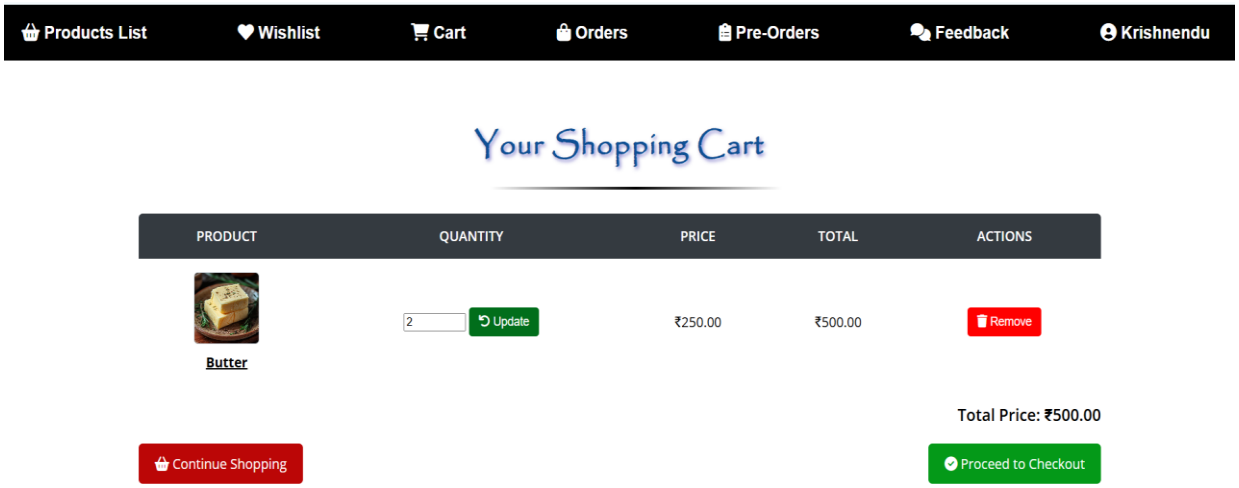


Figure 9.2.7: Cart Page

9.2.8 Form Name: Payment Page

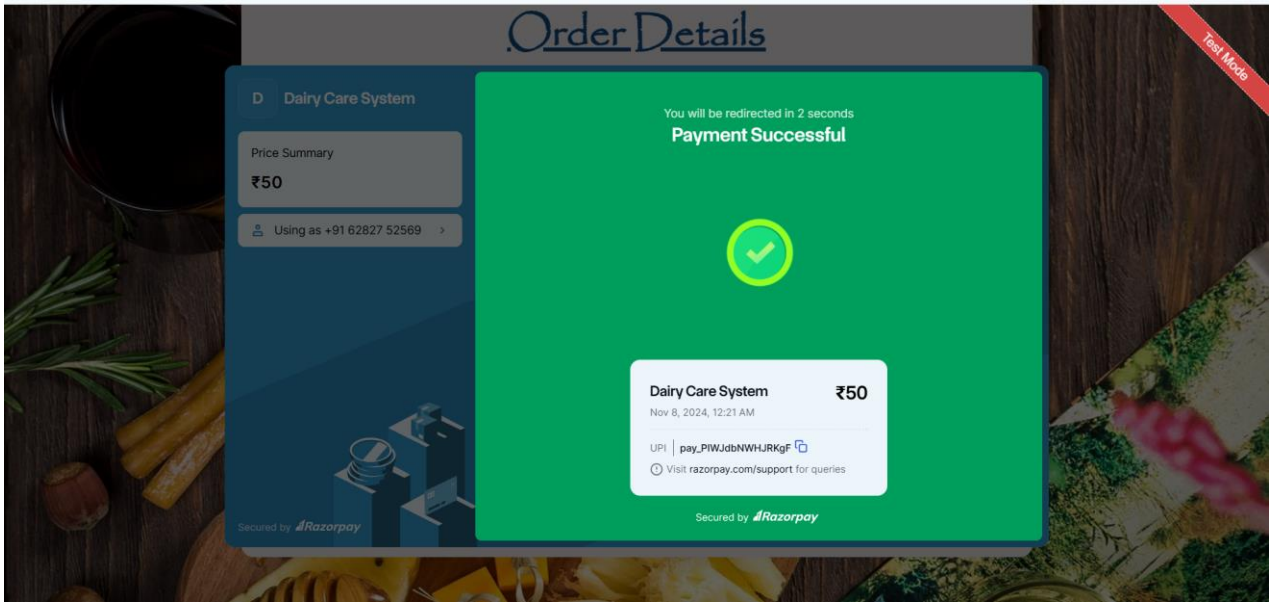


Figure 9.2.8: Payment Page

9.2.9 Form Name: Order Receipt Page

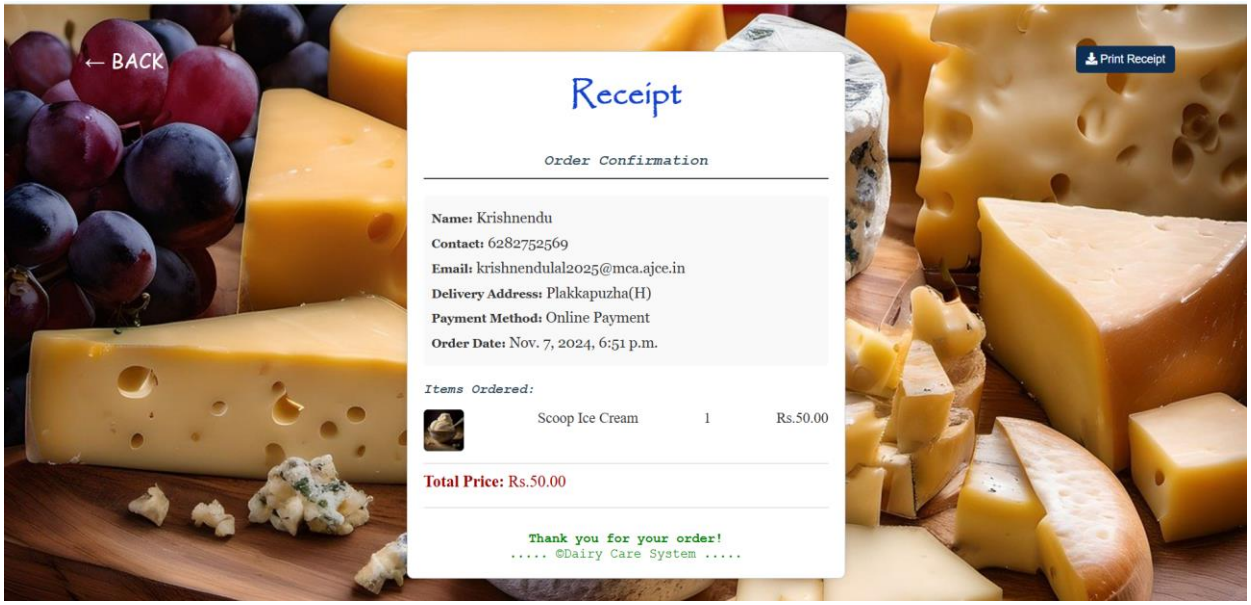


Figure 9.2.9: Order Receipt Page

9.2.10 Form Name: Order Page

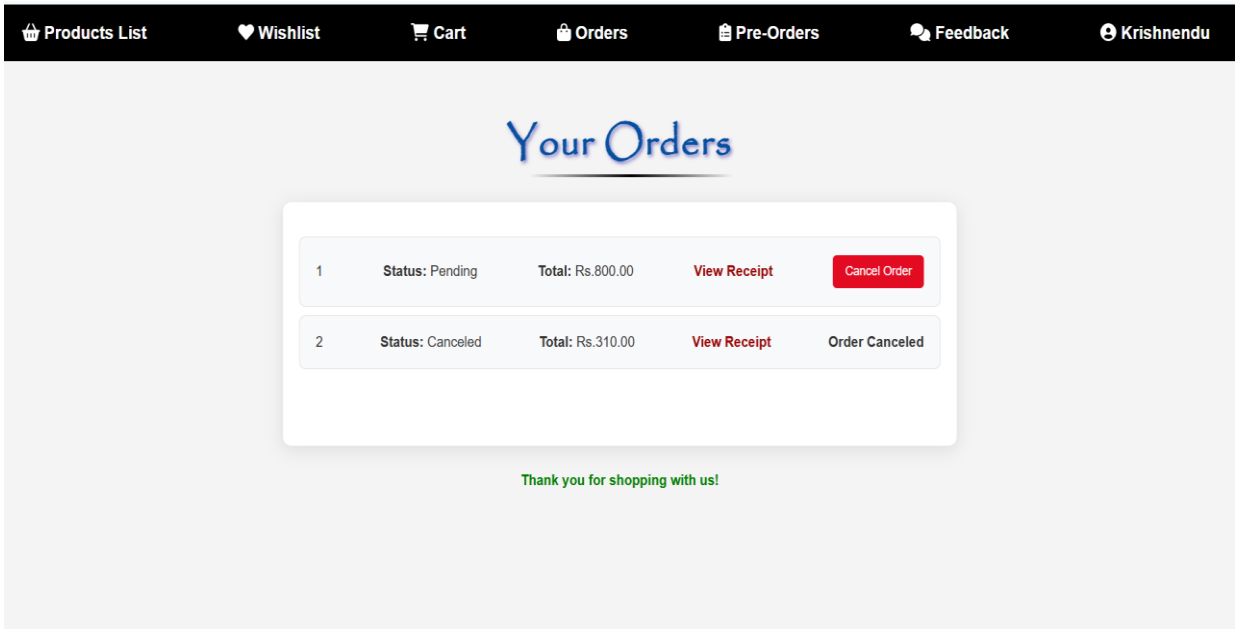


Figure 9.2.10: Order Page

9.2.11 Form Name: Pre-Ordering Page

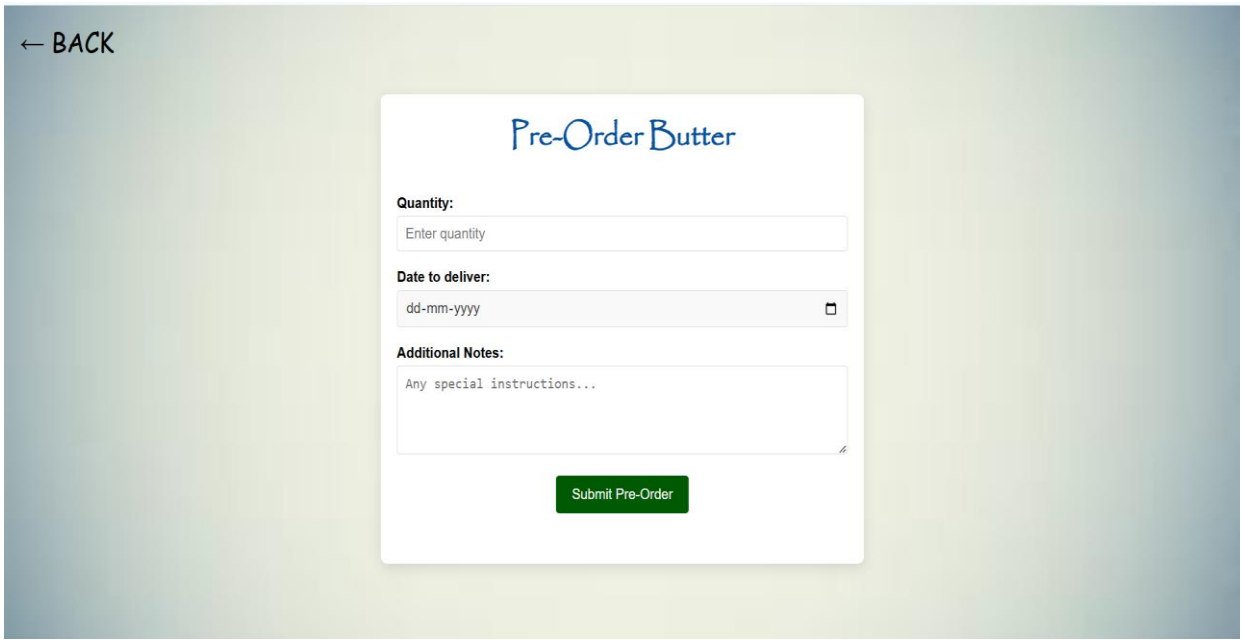


Figure 9.2.11: Pre-Ordering Page

9.2.12 Form Name: Pre-Orders Page

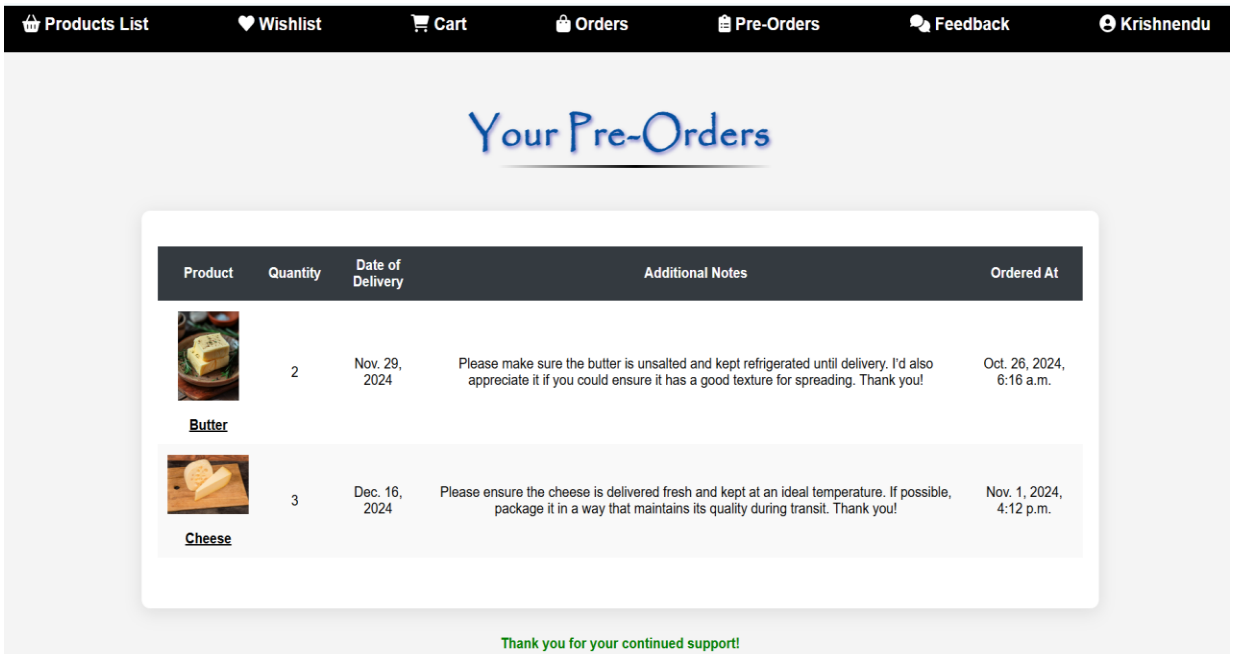


Figure 9.2.12: Pre-Orders Page

9.2.13 Form Name: Wishlist Page

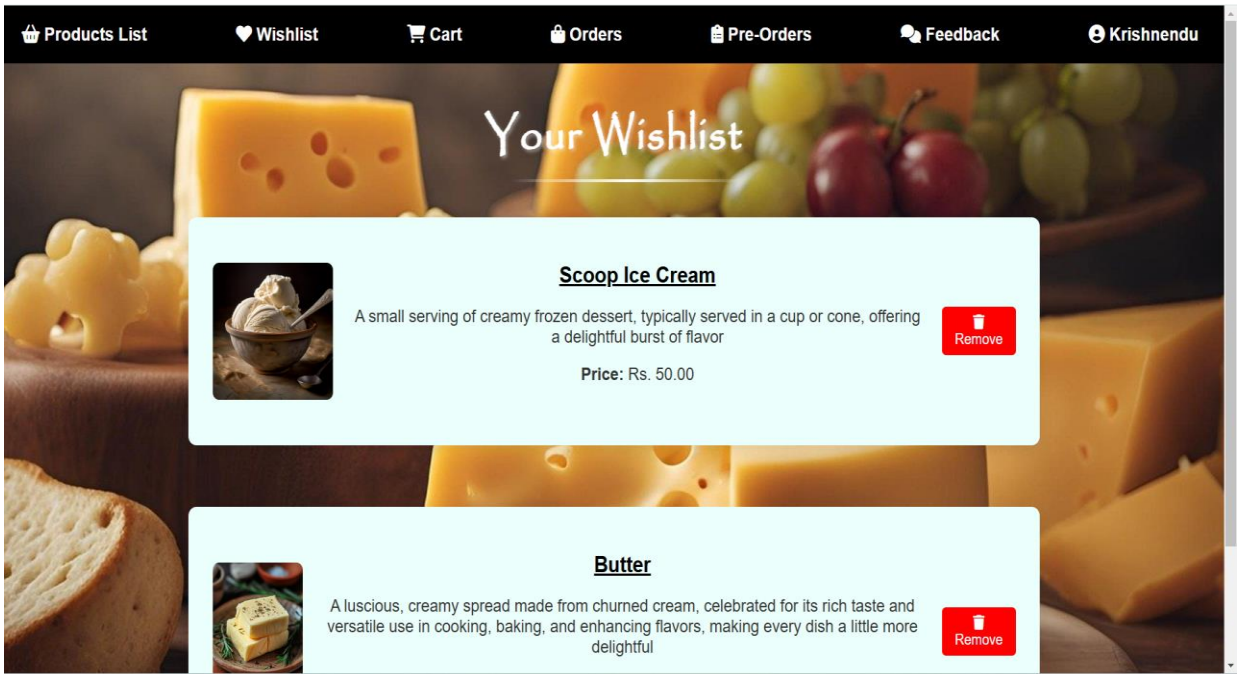


Figure 9.2.13: Wishlist Page

9.2.14 Form Name: Feedback Page

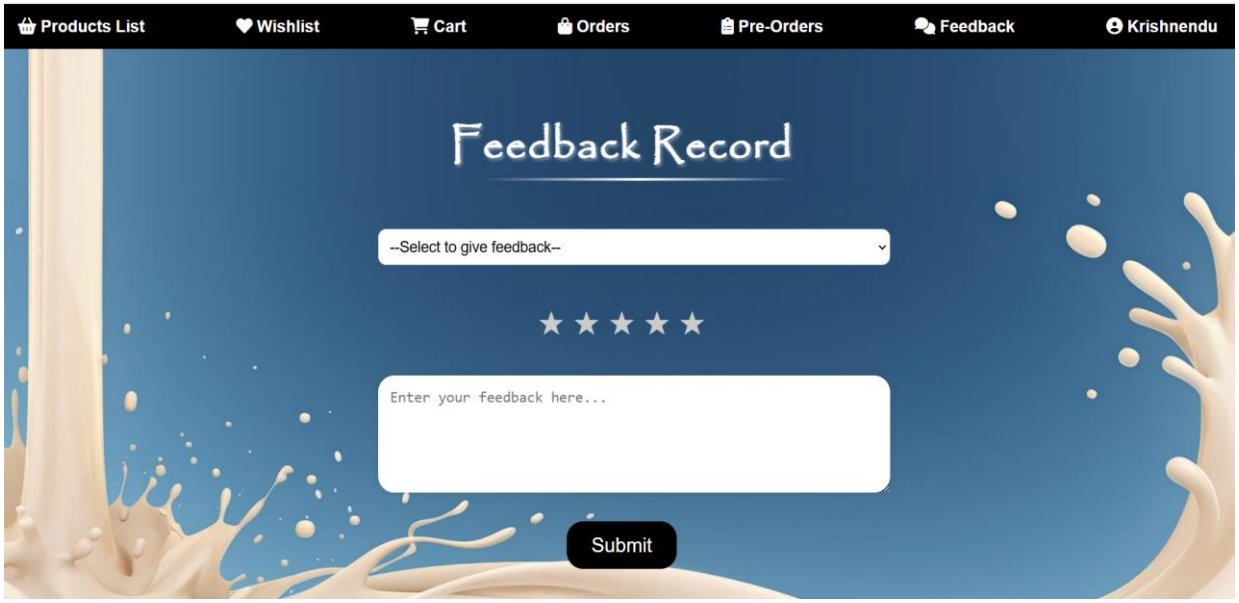


Figure 9.2.14: Feedback Page

9.2.15 Form Name: Admin Page

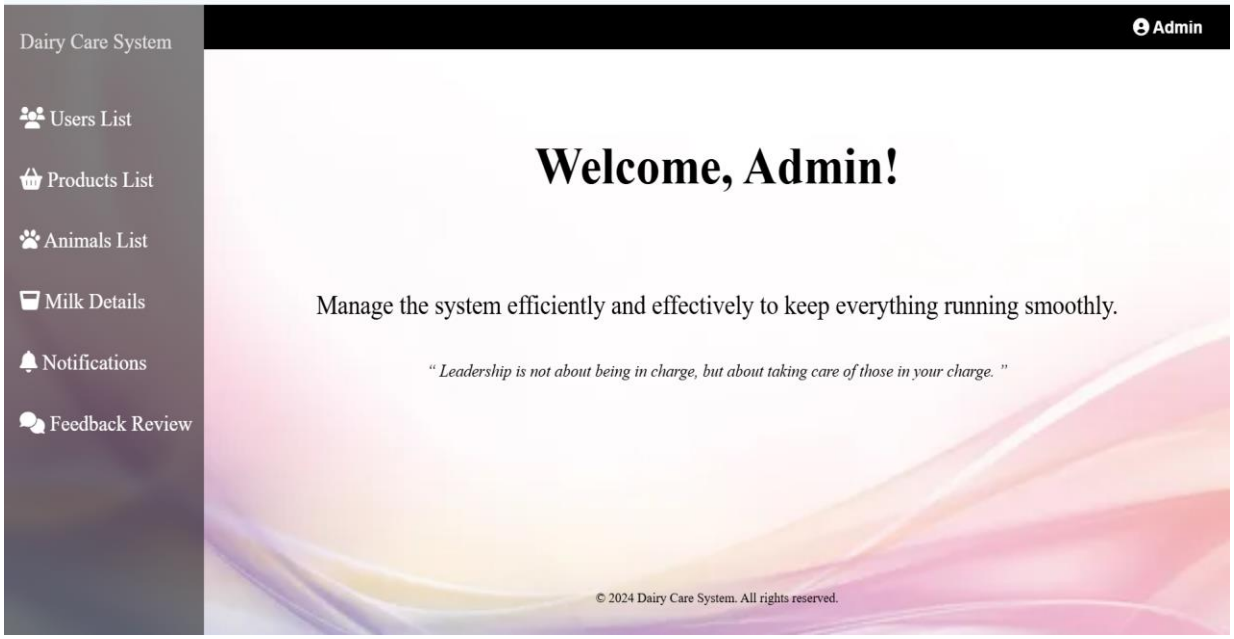


Figure 9.2.15: Admin Page

9.2.16 Form Name: Employee Adding Page

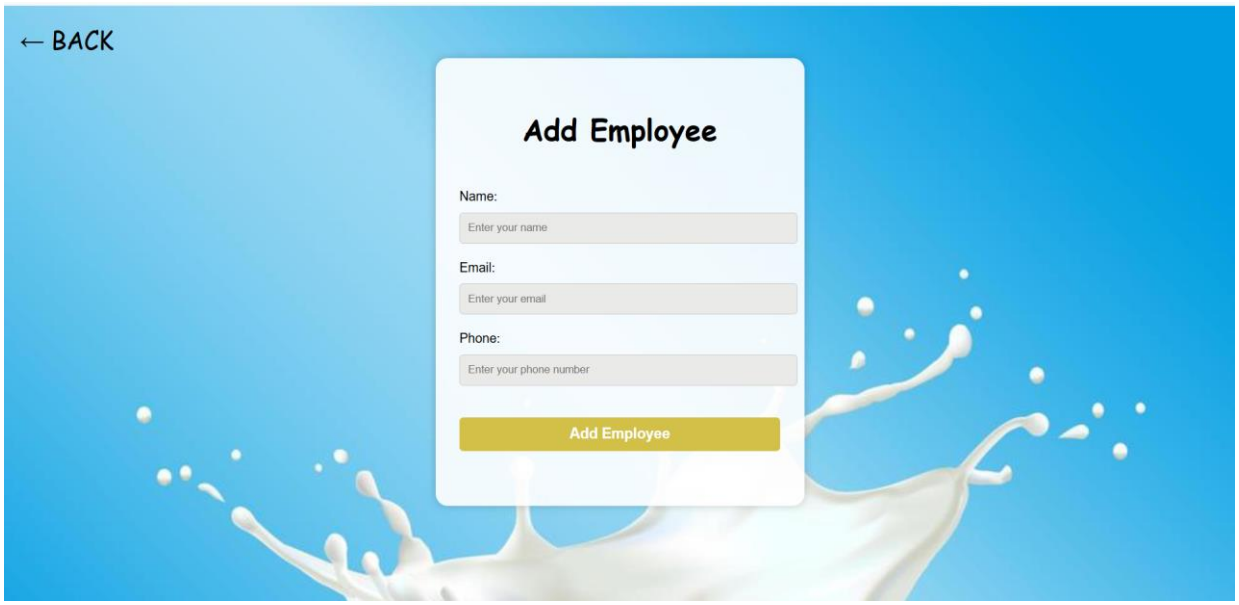


Figure 9.2.16: Employee Adding Page

9.2.17 Form Name: Feedback Review Page

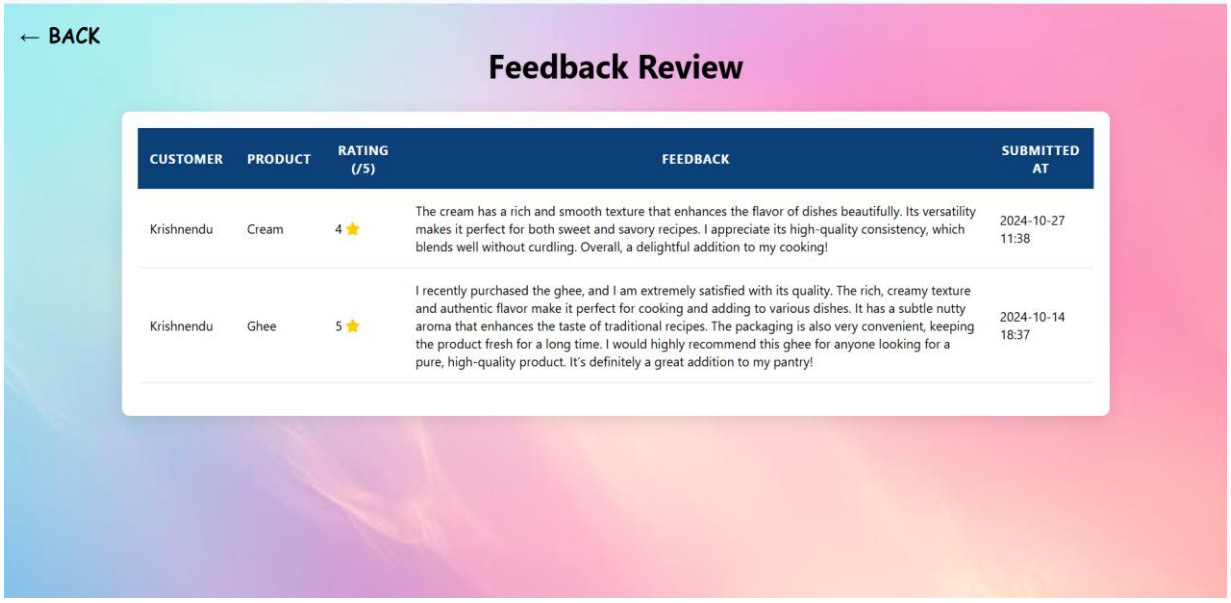


Figure 9.2.17: Feedback Review Page

CHAPTER 10

GIT LOG

KRISHNENDULAL / Dairy_Care_System

Code Issues Pull requests Actions Projects Wiki Security In

Commits

main All users All time

Commits on Nov 6, 2024

commit (hosting)

249d8fe

KRISHNENDULAL committed 2 days ago

commit 40(almost complete)

9849f82

KRISHNENDULAL committed 2 days ago

Commits on Nov 5, 2024

commit 39 (Payment complete)

e481148

ommitted 2 days ago

Browse repository at this point

Commits on Nov 4, 2024

corrected

017c2e9

KRISHNENDULAL committed 3 days ago

commit 38(almost ,2 address)

f9e52cf

KRISHNENDULAL committed 3 days ago

Commits on Oct 27, 2024

commit 37 (pre-order & billing two)

52d5d32

KRISHNENDULAL committed 2 weeks ago

Commits on Oct 25, 2024

commit 36(post request)

9e4844f

KRISHNENDULAL committed 2 weeks ago

Commits on Oct 24, 2024

commit 34 (pass encrypt)

01eca86

KRISHNENDULAL committed 2 weeks ago

Commits on Oct 21, 2024

commit 34 (Animal Details)

9634723

KRISHNENDULAL committed 2 weeks ago

commit 33 (order placement)

c23ad0d

KRISHNENDULAL committed 3 weeks ago

Commits on Oct 18, 2024

commit 32 (background designing)

5b42b9a

KRISHNENDULAL committed 3 weeks ago

Commits on Oct 15, 2024

commit 31 (feedback)

f176df0

KRISHNENDULAL committed 3 weeks ago

Commits on Oct 14, 2024

commit 30 (otp registration)

54d280a

KRISHNENDULAL committed 3 weeks ago

Commits on Oct 10, 2024

commit 29

555ed99

KRISHNENDULAL committed last month

Commits on Oct 9, 2024

commit 28 (wishlist)

f6e152d

KRISHNENDULAL committed last month

Commits on Oct 8, 2024

commit 27

791e9dd

KRISHNENDULAL committed on Oct 8

commit 26

d2f28ec

KRISHNENDULAL committed on Oct 8

Commits on Oct 6, 2024

commit 25

140b346

KRISHNENDULAL committed on Oct 6

Commits on Oct 1, 2024

commit 24

376f545

KRISHNENDULAL committed on Oct 1

Commits on Sep 30, 2024

commit 23

67a4fb9

KRISHNENDULAL committed on Sep 30

commit 22

1d5b4a1

KRISHNENDULAL committed on Sep 30

Commits on Sep 27, 2024

commit 21

f6c2c70

KRISHNENDULAL committed on Sep 27

Commits on Sep 26, 2024

commit 20 (Google log in)

d692c79

KRISHNENDULAL committed on Sep 26

Commits on Sep 25, 2024

commit 19

1fa0f02

KRISHNENDULAL committed on Sep 25

commit 18

29e53f5

KRISHNENDULAL committed on Sep 25

commit 16

2dc3804

KRISHNENDULAL committed on Sep 25

commit 15

305899a

KRISHNENDULAL committed on Sep 25

commit 14

3378ad9

KRISHNENDULAL committed on Sep 25

Commits on Sep 24, 2024

commit 13

`11db4bc`

KRISHNENDULAL committed on Sep 24

commit twelve

`7ab1600`

KRISHNENDULAL committed on Sep 24

commit eleven

`d31ffa9`

KRISHNENDULAL committed on Sep 24

commit ten

`efb85eb`

KRISHNENDULAL committed on Sep 24

Commits on Sep 23, 2024

commit nine

`e251503`

KRISHNENDULAL committed on Sep 23

commit eight

`b53384d`

KRISHNENDULAL committed on Sep 23

seventh commit

`f66310b`

KRISHNENDULAL committed on Sep 23

Previous Next ›

KRISHNENDULAL /
Dairy_Care_System

Code Issues Pull requests Actions Projects Wiki Security In

Commits

main All users All time

Commits on Sep 23, 2024

seventh commit
0cf440f
KRISHNENDULAL committed on Sep 23

Commits on Sep 22, 2024

sixth commit
6fc053b
KRISHNENDULAL committed on Sep 22

Commits on Sep 21, 2024

fifth commit
46c711f
KRISHNENDULAL committed on Sep 21

Commits on Sep 20, 2024

commit
d757b54
KRISHNENDULAL committed on Sep 20

third commit
be5737e
KRISHNENDULAL committed on Sep 20

second commit
9d1dd2b

KRISHNENDULAL committed on Sep 20

first commit

bdf7968

KRISHNENDULAL committed on Sep 20

Previous Next