Dairy Care System

Comprehensive Dairy Management

Project Report

Submitted by

Krishnendu Lal

Reg. No.: AJC23MCA-2042

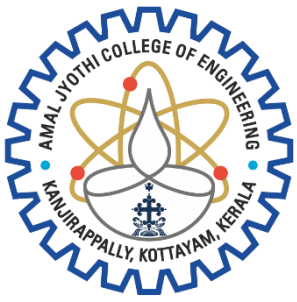
In Partial fulfillment for the Award of the Degree of

MASTER OF COMPUTER APPLICATIONS

(MCA TWO YEAR)

(Accredited by NBA)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY



AMAL JYOTHI COLLEGE OF ENGINEERING AUTONOMOUS

KANJIRAPPALLY

[Approved by AICTE, Accredited by NAAC, Accredited by NBA.
Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2023-2025

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 bona fide work of **KRISHNENDU LAL (Regno: AJC23MCA-2042)** carried out in partial fulfillment of the requirements for the award of the **Degree of Master of Computer Applications** at **Amal Jyothi College of Engineering Autonomous, Kanjirappally,** Affiliated to **APJ Abdul Kalam Technological University.** The project was undertaken during the period from **December 10, 2024,** to **March 27, 2025**.

Mr. Amal K Jose
Internal Guide

Ms. Meera Rose Mathew
Coordinator

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

External Examiner

AMAL JYOTHI
COLLEGE OF ENGINEERING
AUTONOMOUS
KANJIRAPPALLY

Department of Computer Applications

CERTIFICATE ON PLAGIARISM CHECK

1	Name of the Scholar	KRISHNENDU LAL
2	Title of the Publication	DAIRY CARE SYSTEM
3	Name of the Guide	Mr. AMAL K JOSE
4	Similar Content (%) identified	25%
5	Acceptable Maximum Limit	**25%**
6	Software Used	**TURNITIN**
8	No.of pages verified	78

Name & Signature of the Scholar: KRISHNENDU LAL

Name & Signature of the Guide: Mr. AMAL K JOSE

Name & Signature of the Head of the Department: Rev. Fr. Dr. RUBIN THOTTUPURATHU JOSE

Dept. Seal

DECLARATION

I hereby declare that the project report **"DAIRY CARE SYSTEM"** is a bona fide work done at **Amal Jyothi College of Engineering Autonomous, Kanjirappally**, Affiliated to **APJ Abdul Kalam Technological University**, towards the partial fulfilment of the requirements for the award of the **Master of Computer Applications (MCA)** during the period from **December 10, 2024** to **March 27, 2025**.

Date: 26-03-2025 **KRISHNENDU LAL**
KANJIRAPPALLY **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 **Ms. Meera Rose Mathew** for her valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **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 an advanced web-based platform designed to optimize and modernize dairy farm management. Built using Django, it integrates AI and Machine Learning to enhance efficiency, productivity, and profitability. The system provides role-based access for admins, farm owners, delivery agents, and customers, ensuring secure authentication and seamless management of livestock, milk production, product inventory, and deliveries.

Health monitoring features include ML-based disease prediction and milk yield analytics, while real-time tracking streamlines distribution. AI-powered dynamic pricing and a chatbot enhance customer engagement, and interactive dashboards offer insightful analytics into sales trends and consumer behavior. Payment processing supports Razor Pay for online transactions and Cash on Delivery options, while feedback collection ensures continuous improvement. The platform employs MySQL with cloud storage for scalability and AI-driven route optimization for delivery management.

With its cutting-edge technologies and data-driven decision-making capabilities, the Dairy Care System revolutionizes dairy farming, paving the way for a sustainable, efficient, and customer-centric industry transformation.

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
- ML - Machine Learning
- AI - Artificial Intelligence

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The evolution of dairy farming in new technology demands an efficient automated system to operate daily operations in modern farms. The Dairy Care System has made headlines for its user-friendliness, being an Internet-based application that carries with itself the principles and practices of sound management of dairy farms but is also integrated with state-of-the-art applications like AI, ML, and automation

Django is a powerful python web framework for developing this system. It gives a very strong data-based view to farm owners and employees and consumers on major activities like livestock management, milk production tracking, health monitoring, product distribution, and customer engagement. The system also encapsulates predictive analytics for disease detection and predicting the milk yield to take proactive decisions and better utilization of resources.

Features like security payment processing, real-time delivery tracking, role-based access control, and AI-powered virtual assistant feature make the Dairy Care System a very simple yet intelligent experience for the users. The dashboards are interactive and provide data-rich views that can help farm owners make better informed business decisions, increase their productivity, and improve their customer service.

The platform is set to redefine the future of dairy farming as it optimizes farm operations using state-of-the-art technologies while ensuring sustainability, profits, and empowering the agricultural sector digitally.

1.2 PROJECT SPECIFICATION

The Dairy Care System provides for a better management scheme of dairy farms, hence embracing automation and efficient handling of core activities. The system provides role-based access to users with differing responsibilities in such a way that the administrator, farm owners, delivery agents, and customers can perform their jobs effectively.

The project specification is detailed as below:

1. **User Roles and Authentication:**
    - User Roles:
        - Admin: The administrator of the whole system and overall farm operations.
        - Farm Owner: Relieve chores such as health monitoring, and milking operations.

- Delivery agent: Efficient handling of delivery of orders to customers.
- Customer: Acquires dairy products online, tracks their orders, and gives feedback.

- Authentication:
  - Secure login and registration with role-based access control.
  - User management for the admin, assign roles, and permissions.

2. Product and Inventory Management:

- Product Search and Purchase:
  - Browsing, searching, and buying dairy products can be done by customers.
  - Pay for it using Razor-Pay or Cash on Delivery.
  - Tracking delivery is integrated so products can be monitored where they travel.
- Inventory Management:
  - Owners can add, edit, remove products from the inventory.
  - It alerts the stock owner if low, and makes notifications not to run out of product.
  - Real-time inventory status is displayed on the dashboard.

3. Animal Health Management:

- Health Management:
  - It introduces Machine Learning for disease prediction and diagnosis for better management.

4. Payment Processing:

- Customer Payment:
  - Provides easy online payment with gateways like Razor-Pay.
  - Cash on Delivery with tracking of order delivery is also available.

5. Feedback and Analysis:

- Feedback System:
  - Customers can also leave feedback on the products.
- Feedback Analysis:
  - Admin or owner can analyse the feedback for product improvement.

6. Technology and Tools:

- Frontend:
  - HTML, CSS, JavaScript for intuitive user interface.
- Backend:
  - Django framework for server-side logic and application structure.

- MySQL database management for storing and managing data regarding users, products, livestock, and transactions.

- Payment Gateway:
  - Razor-Pay integration for online payments.
- Operating System:
  - This project will be developed and tested on Windows.
- IDE:
  - Built using Visual Studio Code for development and debugging.

7. System Workflow:

- User Interaction:
  - The users will interact with the system depending on their roles like the admin and farm owner who will manage it, delivery agent who will record data, and customers who will shop.
- Admin Dashboard:
  - The dashboard is centralized in the management of users, products, inventory, sales, and deliveries.
- Notification:
  - Automated notifications serve purpose of stock management and feedback analysis.

8. Database Design:

- Tables:
  - Users (Admin, Farm Owner, Delivery Agent, Customer), Products, Orders, Payments, Feedback, Livestock Health Records.
- Relationships:
  - Relational data management between users, products, orders, and health records to ensure that the data consistency is in place.

CHAPTER 2

SYSTEM STUDY

2.1INTRODUCTION

In the critical evaluation of the System Study for the Dairy Care System, a comparison of the proposed application and existing dairy management solutions is considered. Whereas the proposed study brings into consideration strengths and weaknesses concerning current systems largely characterized as being functional with minimal consideration for an integrated farm management approach.

The presented study analyzes existing systems to show how the Dairy Care System attempts to successfully address their limitations through special features like user credentials, inventory control, health monitoring, and integrated e-commerce capabilities. This ultimately points to the need for a more integrated and efficient solution for dairy farmers, which has been the key design consideration for the Dairy Care System.

2.2EXISTING SYSTEM

NATURAL SYSTEM STUDIED

The natural system in dairy farming is an orientation toward daily operations and processes in running a dairy farm. This includes looking after the livestock, observing animal health, controlling milk production, and managing the marketing and movement of dairy products. The natural system of a dairy farm, while working for its limited purpose, suffers from data inaccuracy, slow reaction to disease in animals, and efficient control on the scale of production.

Hence, with the further study of this natural system, the Dairy Care System seeks to digitalize and automate these processes, providing a modern solution integrating advanced tools for farm management, health monitoring, and product sales while optimizing the entire workflow of a dairy farm.

DESIGNED SYSTEM STUDIED

The Dairy Care System, as a whole, represents a very comprehensive digital platform designed to capture, streamline and modernize all the processes of running a dairy farm. Unlike the manual and traditional modes of management, the different processes of this system, such as health monitoring of livestock, inventory, and sales of products, are automated by the aid of technology.

The whole system will rely on the Django framework and will have user authentication, role-based

access, and an integrated e-commerce site. It will create a friendly user interface for farm owners, employees, and customers for easy navigation. This whole system will introduce efficiency, minimize the error of humans, and encourage data-based decision-making rather than just going by the traditional natural system of dairy management.

2.2.1 DRAWBACKS OF EXISTING SYSTEM

- Low Technology Adoption: A number of systems do not harness advanced technologies such as machine learning for forecasting milk production or detecting animal diseases.
- Concentration on Specific Functions: The majority of systems focus on the sale/distribution of products while overlooking the entire aspect of farm management such as livestock health and feeding schedules.
- Costly Setup and Maintenance: Systems such as DelPro and Lactanet are costly to set up and maintain, which limits their use among smaller farms.
- Absence of Modern User Authentication: Generally, existing systems have out-of-date authentication methods with no advanced role-based access types for different users.
- Complicated and Difficult to Navigate: Some platforms, such as Uniform-Agri and Lactanet, are extremely complex in appearance and thus tend to overload rather than enhance effectiveness in usage.
- Non-Integrated Payment Solutions: Only a handful of systems promise secure online payment gateway or cash-on-delivery options with tracking and location-based delivery.
- Inadequate support for farm monitoring: Milma and Binsar Farms do not possess the complete farm management tool needed to monitor the health and feeding of livestock as well as milking activities.
- Reliance on Technical Assistance: Platforms like Lactanet are not friendly in terms of customization and setup of farm-owners, as they rely much on external technical support.

2.3 PROPOSED SYSTEM

The Dairy Care System presents a complete solution to counter the limitations that other dairy management platforms have. Most of the current systems operate in specified niches like product sales or basic farm operational activities, while this proposed system brings all aspects of dairy farm management into one complete package. The system also enhances user authentication and role-based access for owners, employees, and customers, thereby making its use secure and easy for all

stakeholders. In addition, it provides an integrated e-commerce platform with secure payments through Razor-Pay and cash-on-delivery services.

Furthermore, the Dairy Care System is cost-effective and easy to set up, suitable for any size farm, unlike the far more costly and complex systems currently in use. There is also an advanced feedback system in place that allows the farm owner to gather feedback from customers and analyze it to spur improvement. The system will finally send alerts regarding low stock levels in real time to the admins for better management of inventory and prevention of shortages. Thus, the Dairy Care System indeed offers a state-of-the-art, scalable, and user-friendly solution for any dairy farm management needs.

2.3.1 ADVANTAGES OF PROPOSED SYSTEM

- All-inclusive Farm Management: It takes care of everything that happens on the farm, such as taking care of livestock health, milking, and product management. It really is the most comprehensive solution.
- User-friendly interface: This is simple and intuitive so all users with no matter what kind of technical background could use this very easily.
- Role-Based Access Control: Those unauthorized users will find themselves locked out of the data, so provide secure authentication and customized access for owners, employees, and customers without any issue of permission rights.
- Integrated E-commerce Platform: Customers can now browse and buy products online, offer payment options which include secure payments and cash on delivery, all at the back end offering convenience.
- Low Inventory alerts: These trigger alerts for admins once stock levels fall below a certain threshold and prevent stock-outs from occurring, hence improving stock management activities.
- Affordableed and Scalable: With a proper dairy management system, it would be very costly and complex for installation, yet here lies everything that you might look at as a scalable and less costly economy to install, from the small farm to the big farm.
- Feedback Collection and Analysis: This is about having a number of intelligent tools for feedback collection and analysis from customers in order to improve farm operations and the service offered continuously.
- Health and equipment monitoring: It tracks the health of animals and milking equipment maintenance. This ensures smooth operations and welfare of the resources within the farm.

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

The main focus of going through a feasibility study is that it should assess all parameters concerning whether the proposed project meets the objectives of the organization resources, labor, and time. This important study highlights the potential viability and prospects of the project for decision-makers and developers. To ascertain the feasibility and success potential of the proposed project, various elements associated with the system need to be scrutinized- such as the impacts on the organization and the fulfillment of user needs as well as optimal utilization of resources. The concern of possible feasibility of the proposed project involves many dimensions; each has its critical role in decision-making.

- Technical Feasibility
- Behavioral Feasibility
- Economic Feasibility

A feasibility study that has been well conducted often results in insights that can be quite useful for decision-makers in making sound judgments concerning the anticipated success of an initiative. It also identifies potential risks, challenges, and opportunities associated with the proposed venture before enabling stakeholders to create effective mitigation strategies.

3.1.1 Economical Feasibility

Evaluation of the economic feasibility for a Dairy Care System includes considering all the costs associated with development, implementation, and maintenance and the expected return on any investment. The costs accrue at a modest level because of the usage of open-source technologies, which take away the licensing fees from the factor. Besides, the use of open-source technologies ensures that there would be no licensing fees needed for the project itself. The system makes use of technology such as Django for the backend and HTML, CSS, and JavaScript for the frontend, so it does not cost so much in terms of development. They can also be cheap due to the lowest system requirements like Intel Core i5 types of processors and 8 GB of RAM, a 500 GB hard disk, which are rather ordinary and inexpensive hardware components. By default, it uses widely supported technologies, which adds to the minimal operational costs that an in-house team can operate on without incurring costs for maintenance and bug fixes. Those are just a few of the factors that make the Dairy Care System, an economically viable alternative because, its investments would relatively be lower at startup and along with running costs, the system remains sustainable and effective in terms of certain dairy farm management operations.

3.1.2 Technical Feasibility

Technical feasibility study primarily establishes the capabilities of the Dairy Care System, their operation within a proposed environment, and such integrations with other systems. The system utilizes the following frontend technologies to develop an adaptive and interactive user interface: HTML, CSS, and JavaScript. The backend was a combination of Django, a highly scalable and secure web framework in Python that accelerates the development of web apps.

MySQL, widely regarded as a powerful, open-source relational database system, manages the complete set of databases with exceptional data-handling capability. The system is set to work on PCs powered by Windows 11, giving it broad compatibility with the user environment. Consequently, the chosen technologies allow for smooth operation and scalability and allow for payment gateway integration for better functionality and user experience of the Dairy Care System.

3.1.3 Behavioral Feasibility

The degree of acceptance and adoption of the Dairy Care System by its users is assessed during the behavioral feasibility evaluation. The user interface of the system is simple enough to provide easy navigation for all concerned parties, such as farm owners, employees, and customers. Intuitive design ensures that little training beyond documentation and in-support resources to assist the user during transitory times is required.

Furthermore, the system is based on the requirements of its users, including but not limited to farm monitoring, health management, payment processing, and feedback analysis, all critical services to the day-to-day running of dairy farms. All these factors are felt strongly inclined towards obtaining user satisfaction, making the Dairy Care System quite suitable for acceptance at large with less possible resistance.

3.1.4 Feasibility Study Questionnaire

1. **Project Overview**:

   The Dairy Care System is a complete digital ecosystem that aids in the management of dairy farms by integrating inventory management, order processing, animal health, and customer interaction. It eliminates inefficiencies in stock tracking, disease management, and order fulfillment by suggesting role-based authentication, real-time stock alerts, and machine learning for disease detection. The core objectives of the system are to make farm

operations smooth, productive, and customer-satisfied through an organized or friendly user interface.

2. **System Scope**:

The system is developed as a full-scale implementation, several farm owners can now register, manage livestock, and sell products independently. This system does not focus solely on one farm; rather, it functions as a multi-vendor dairy management platform where different inhabitants of the farms can carry their business activities.

3. **Target Audience**:

The Dairy Care System is primarily aimed at farm owners and customers. Farm owners have at their convenience a centralized digital space where they can provide management for their livestock, inventory tracking, and direct sales of their dairy products to consumers. Real-time stock updates and automatic alerts for low stock help enhance their operations and improve product quality through customer feedback. The customers, on the other hand, have a handy online platform to browse and purchase fresh dairy products, track their orders, and give feedback about their purchases. The system closes the gap between the farm owners and their customers while ensuring a seamless farm-to-consumer experience with efficiency, transparency, and customer satisfaction in the dairy industry.

4. **Modules**:

- User Authentication & Role Management: A secure login application whereby different members are accessed as admins, farm owners, delivery agents, and consumers.
- Multi-Farm Product & Stock Management: Each farm owner can list products for that particular farm, control inventory, and be alerted to low stock.
- Order & Payment Processing: Customers place orders and make online payments (Razor-Pay) or pay Cash on Delivery while integrated tracking for delivery is provided.
- Animal Health Management: Machine learning for disease identification aids in the overall livestock health management system.
- Feedback & Analysis: Customers can submit product reviews while farm owners or admins analyze these reviews for improvements.
- Admin Dashboard/Notification: Administrators can manage users and sales while sending notifications via a centralized dashboard.

5. **User Roles**:

   - Admin - For overall system management, definition of roles, and proper coordination of activities.
   - Farm Owner - Self-lists and manages his/her dairy products, controls inventory, and sales.
   - Delivery Agent - Delivery of products and update of status of orders.
   - Customer - Browse available products and purchases, track order and feedback.

6. **System Ownership**:

   Not privately owned by a single farm, the system is rather a multi-farm platform where different farm owners can register and run their milk businesses on a self-sufficient basis under an admin-supervised scheme.

7. **Industry/Domain**:

   Under the Agriculture & Dairy Industry, the project concentrates on the digital transformation of dairy farm operations by giving the farm owners and consumers an online marketplace linking both.

8. **Data Collection Contacts**:

   V Biju,
   Jayanivas (H),Vazhoor
   Phone: 9447806271

9. **Questionnaire for Data Collection**:

   1) What are the main challenges in dairy farm management that the system seeks to solve?
      - Monitoring livestock health effectively and ensuring consistent milk quality.
      - Managing inventory and sales efficiently.
      - Reducing manual errors in record-keeping and scheduling.

   2) What advantages does the Dairy Care System have over existing management processes?
      - A centralized platform for every activity on the farm.
      - Low dependence on manual processing.
      - Evidence-based predictability using machine learning.
      - Improved customer-employee satisfaction with simplified processes.

3) What capabilities does this system bring on board in terms of technology?
   - Machine learning to predict disease detection and milk yield.
   - AI-powered virtual assistance to answer queries and provide support.

4) How does the system manage animal health records?
   - Health records and accurate dates for up-and-coming check-ups are given.

5) What features are planned for product inventory and sales management?
   - Creation of categories for dairy product management.
   - Easy to integrate with payment gateways for hassle-free transactions.
   - Availability updates on stock.

6) What way does the system handle feedback and customer engagement?
   - Structured product feedback submission form.
   - Identifying trends and areas for improvement through automated analysis.
   - Interact directly through virtual assistance for instant solutions.

7) What use does machine learning have in the system?
   - Detects potential health issues in animals before they get aggravated.

8) How is the system going to improve the delivery of dairy products?
   - Allows customers to track deliveries in real-time.
   - Improves delivery routing for time and cost efficiency.
   - Tracks the status of deliveries for record and accountability.

9) What users and roles does the system contain?
   - Admin: Total access to manage all operations.
   - Owner: Manage products, orders, health and feedbacks.
   - Employee: Manage product delivery.
   - Customer: A customer can search for products, place orders, and give feedback.

10) How does the system guarantee scalability for future adoption?
    - Constructed on scalable technologies such as Django and MySQL.
    - Modular architecture makes it easy to add new functionalities when required.
    - Integration of state-of-the-art technologies will allow upgrade and expansion.

3.1.5 Geotagged Photograph

3.1 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor - Intel Core i3

RAM - 4 G B

Hard disk - 5 0 0 G B S S D

3.2.2 Software Specification

Front End - HTML, CSS, JS, AJAX

Back End - Django

Database - MySQL

Client on PC - Windows 11

Technologies used - JS, HTML, CSS, AJAX, Prediction using ML and AI

3.3 SOFTWARE DESCRIPTION

3.3.1 HTML

HyperText Markup Language (HTML) is the most used and widely accepted mark-up language for web applications and documents. It provides the skeleton behind-a-curtain of the content that fills your web from title to paragraph tag, links, images, and every other piece of content. HTML maintains the backbone of each website, as it enables the browsers to interpret and display pertinent page information. As simple and easy, it allows the developers to create well-structured documents that both are readable by human eyes and can be deciphered by a machine. SEO gets its best shot from HTML when the document uses semantic tags to enhance a user's experience on the web.

3.3.2 MySQL

MySQL is designated as a reliable, high-performance, and quite user-friendly, powerful open-source RDBMS: MYSQL. Several applications for such database could be classified based on the usage community: either personal or business. MYSQL uses the structured query language (SQL)-for all database-related actions, doing many things easier for developers to manage and manipulate data. Strong Features: Transactions Security Data replication, etc. hence it is a web application favorite. It is one of those database engines used with the Django framework.

3.3.3 DJANGO

Django is one of the amazing and powerful open-source software web frameworks, based on python and giving the main context for rapid web application development and maintainability. The library is hugely similar to an MVT(MODEL-VIEW-TEMPLATE) architecture. It was designed for web applications with a structured and effective way of creating complex data and provides some of the derived core components and functionalities out of which one of the most important is above described. A key focus is a very robust OR-Mapping component that allows the application to interact with a database much more conveniently. For the purpose, we give a very new way of working with a Python object that consumes just a little raw SQL code, URL Dispatcher to match URL to view function, automatic admin interface for managing application data, and finally template engine for dynamic reusable UI. This is very secured by Django against the most generic web entry vulnerabilities built-in vessel-secured constructions. Authentication and authorization schemes are included with global request and response processing using middleware, and multiple databases or other scalable extensible frameworks. Definitely, superb web development within itself-from the simplest, least intimidating websites, to the most complex and high-load sites: this makes the frame user-palatable.

CHAPTER 4
SYSTEM DESIGN

4.1 INTRODUCTION

Developed from all aspects, conception is basically the first phase of design and the most inventive section of it when you are working toward generating any engineered product or system. It is one of the key factors that ensures a system will function efficiently. Design describes the efforts to apply a variety of techniques and principles which serve to define a process or system, in all its intricacies, for realizing it physically. Different methods are used to describe a machine or system and explain its working in sufficient detail to go for fabrication. In software development, design is always an important phase, considered as the very foundation of software development, which is true for any kind of software engineering paradigm. System design draws a blueprint for a machine or product to be created. Good design of the software is essential to the top performance and accuracy of the product. At this time, the focus of the design process shifts from the end-user to the programmers or database specialists. There are typically two main activities during the creation stage of design which are, Logical Design and Physical Design.

4.2 UML DIAGRAM

Unified Modeling Language, abbreviated as UML, is an all-encompassing means for specifying, visualizing, constructing, and documenting software systems. The UML was developed by the Object Management Group. UML, being a modeling language, stands in contrast to C++, Java, or COBOL, which can consequently be described as a programming language. UML can quite simply be treated as a graphical language with which to draw the schematics of software. UML is a widely accepted general visual modeling language that is adopted to visualize, specify, construct, and document software-intensive systems. Though it was primarily developed for modeling software systems, UML can also be used to model and understand processes in contexts distance apart from software-that is, in manufacturing unit processes. Thus, although UML is not itself a programming language, it may be employed to assist tools that produce code in different programming languages from UML diagrams.

UML consists of the ultimate eight core diagrams that would help for representing various faces of one 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

Use Cases serve as instruments for grasping and organizing the requirements of any system, especially with the building up or using of a product delivery website. Such instruments are graphically represented in so-called “Use case” diagrams, which are part of the Unified Modeling Language, a standardized way of modeling the objects and systems of the real world.

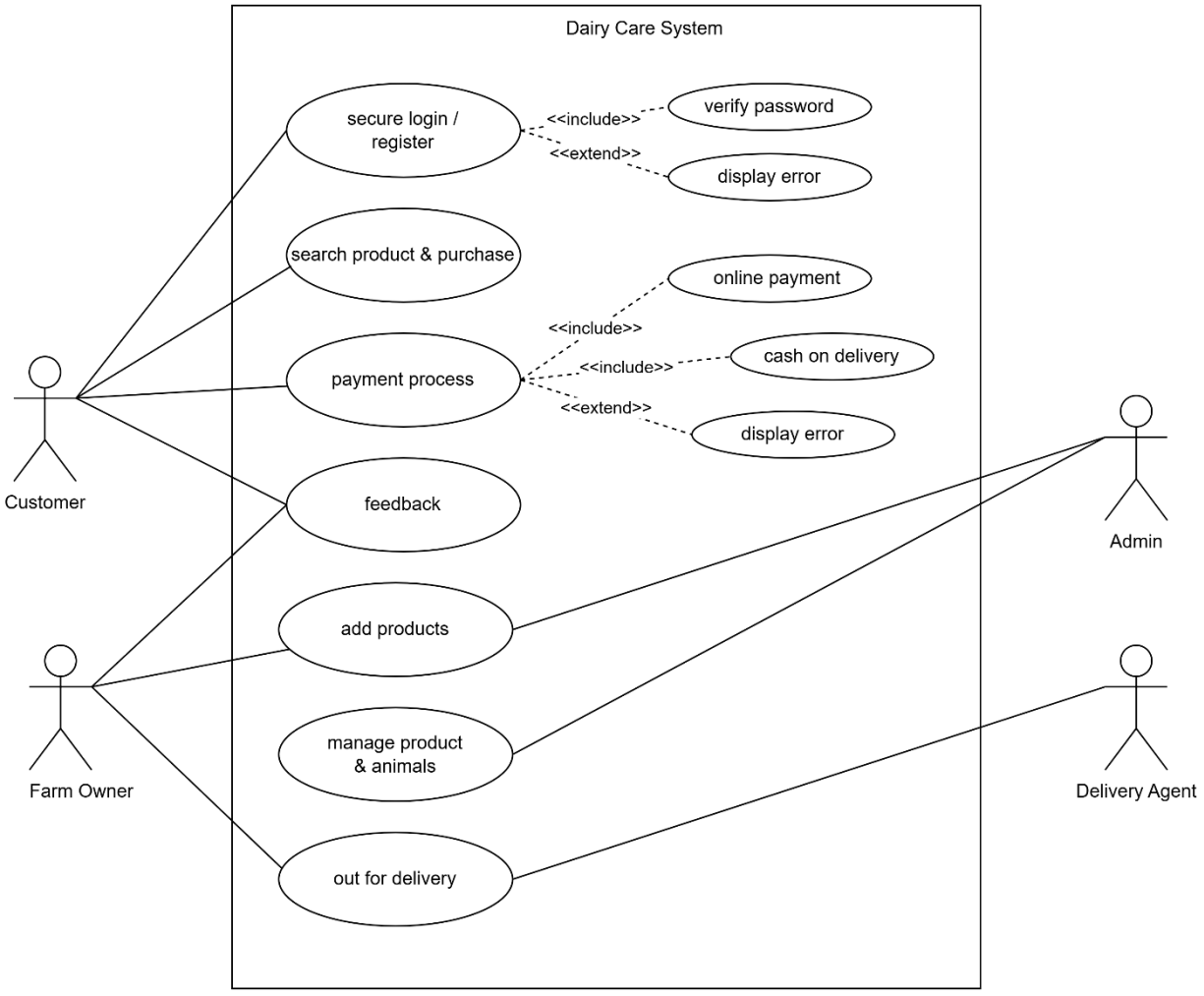


Figure 4.2.1: Use Case Diagram

4.2.2 SEQUENCE DIAGRAM

The sequence diagram is a type of interaction diagram that shows the intended sequence of events that occur when an object interacts with another object. It is also sometimes referred to as an event diagram or event scenario. The object of the sequence diagram, therefore, is to show how various constituents of the system cooperate and the exact order of these activities. Management, business professionals, and software designers often utilize these diagrams to outline and understand requirements for new and/or existing systems.

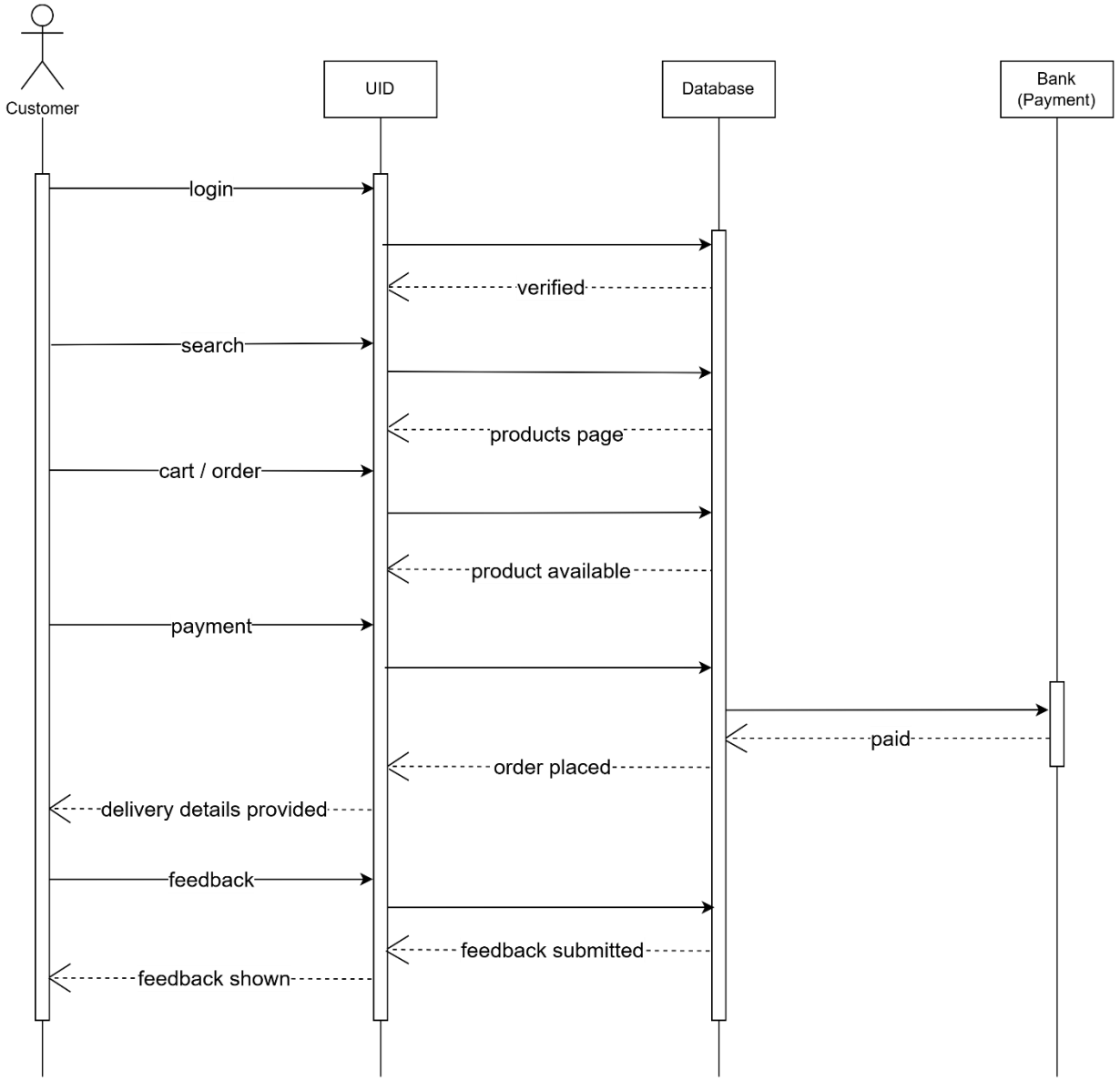


Figure 4.2.2: Sequence Diagram

4.2.3 State Chart Diagram

The name state chart diagram is most commonly called state chart, a diagram depicting the states an object undergoes in the system and how it ceases to exist in one state and enters another. It shows the behavior of the system over a series of entities that may be acting together in unison, a team, a bunch of students, a big body, or the organization itself. The most prominent infinitive form of state machine is showing how the different parts of the system interact with one another; how each system object changes in respond to an event; and the states or conditions each entity or component may find itself in.

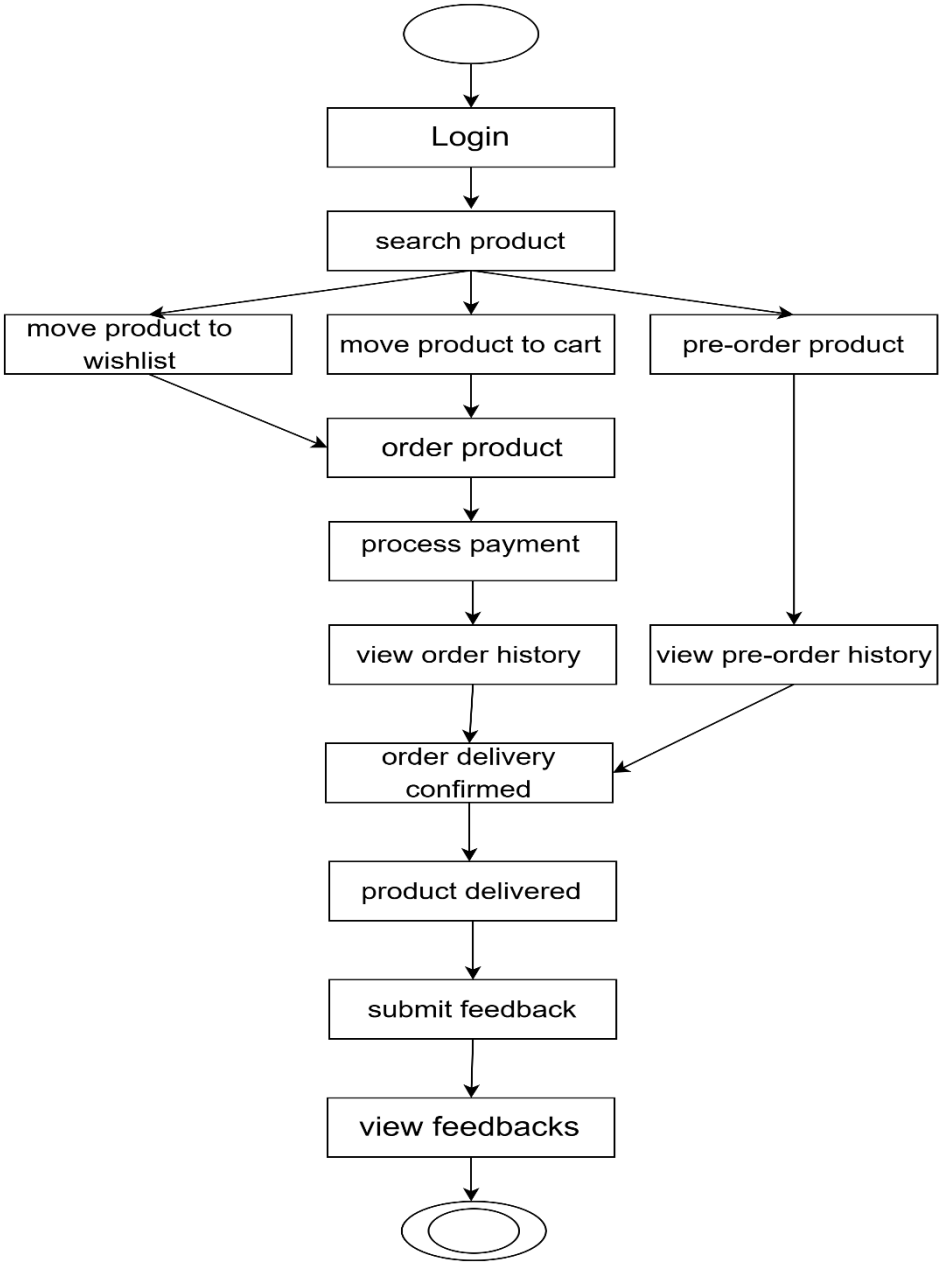


Figure 4.2.3: State Chart Diagram

4.2.4 Activity Diagram

This is a model of the many occurrences, happening either in parallel or subsequent directions, kind of like the film of a play. Understanding this flow of activities will help comprehend the sequence from one activity to the next. The activity diagrams examine the order of occurrence of activities: they can demonstrate sequential flows, parallel flows, alternative flows, or other behaviors. Other additional elements are imported into activity diagrams, such as forks and join nodes, to demonstrate each of these flows in practice, thus enabling an illustration of the operation of a system in a particular way.

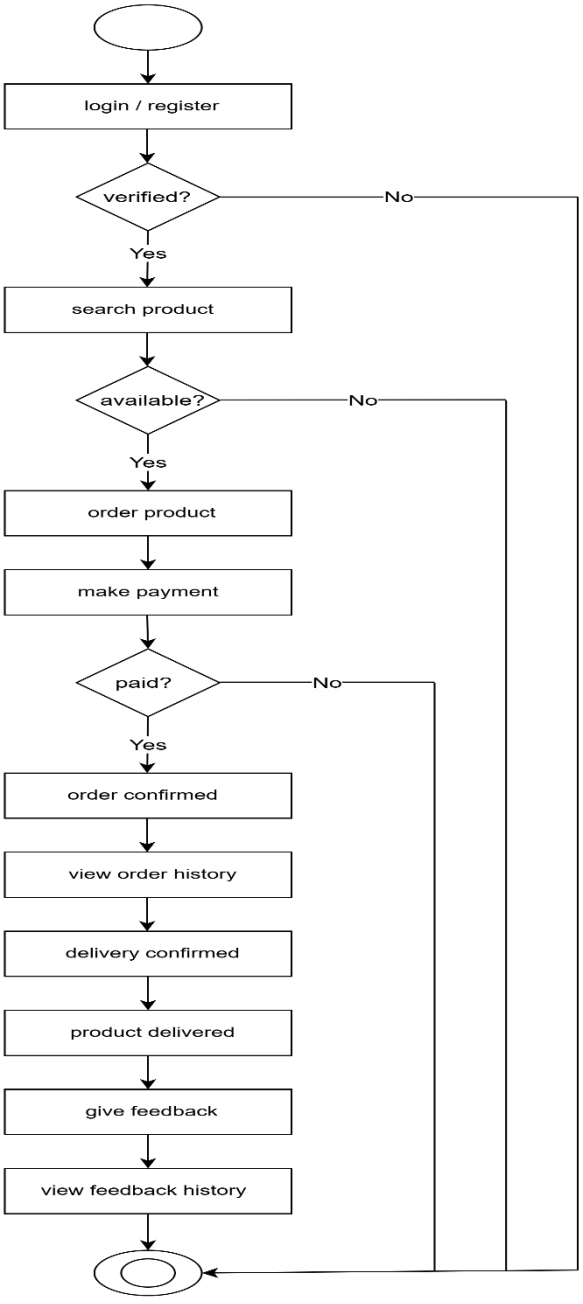


Figure 4.2.4: Activity Diagram

4.2.5 Class Diagram

Entire application may be viewed statically when the entire class diagram represents it; so all its parts and links with other parts can appear when the system is not active. The class diagram gives an internal structure of the system, that is, various elements that comprise the system and how these components interact with each other as well. In short, a class diagram can be considered a visual guide for software development by helping with creating functional applications.

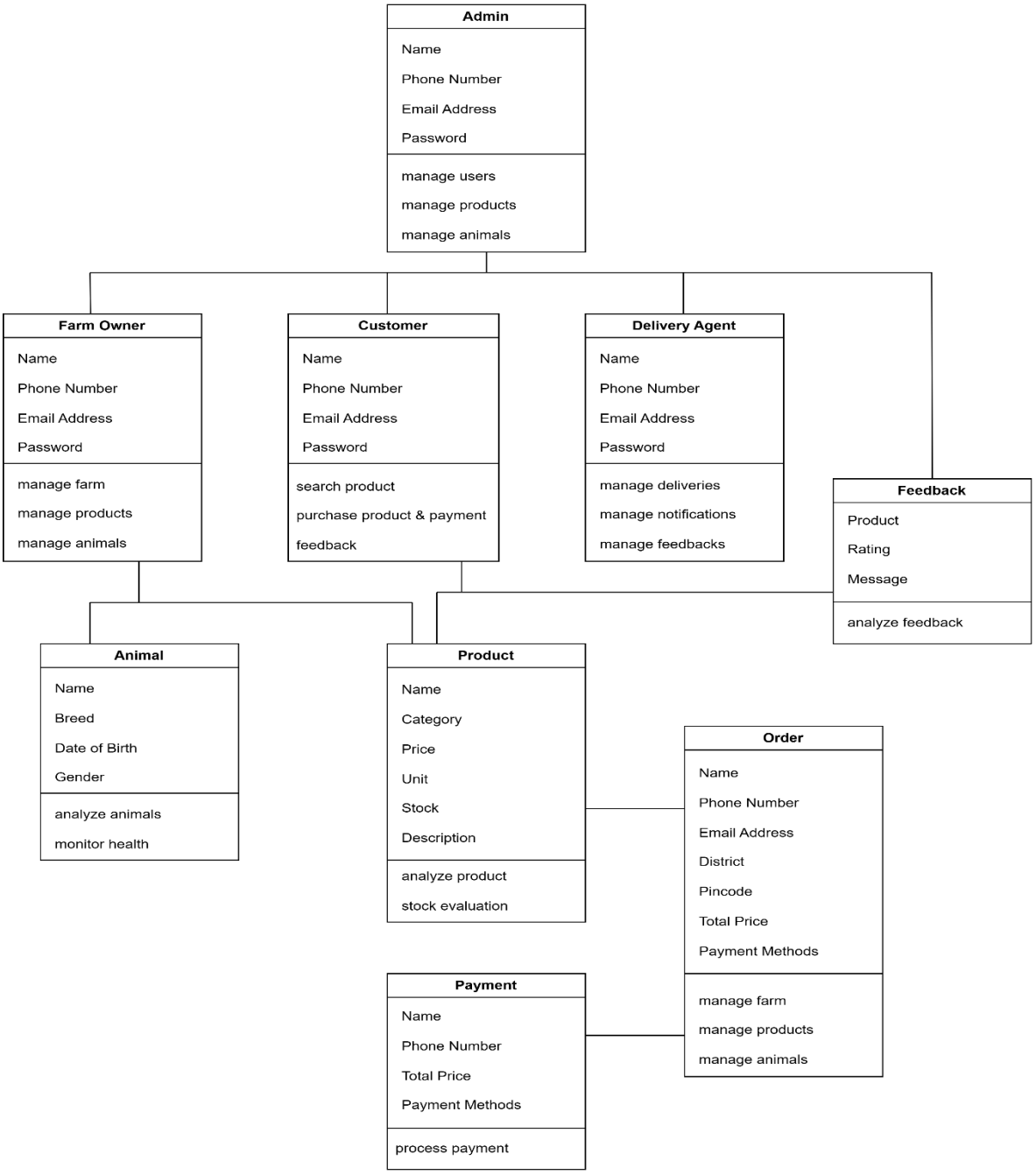


Figure 4.2.5: Class Diagram

4.2.6 Object Diagram

An object diagram may be regarded as a narrowly defined class diagram inheriting so much of its properties and drawing them with the intention of representation. Object diagrams show what multiple objects are tied to a single class. It shows a snapshot of specific object instances in an object-oriented system at a point in time. An object diagram can be similar but different from a class diagram. The contrast with class diagrams is that they are quite general and do not picture specific objects. This generality of class diagrams serves to better illuminate the functions and structure of the system spanned by the diagram.

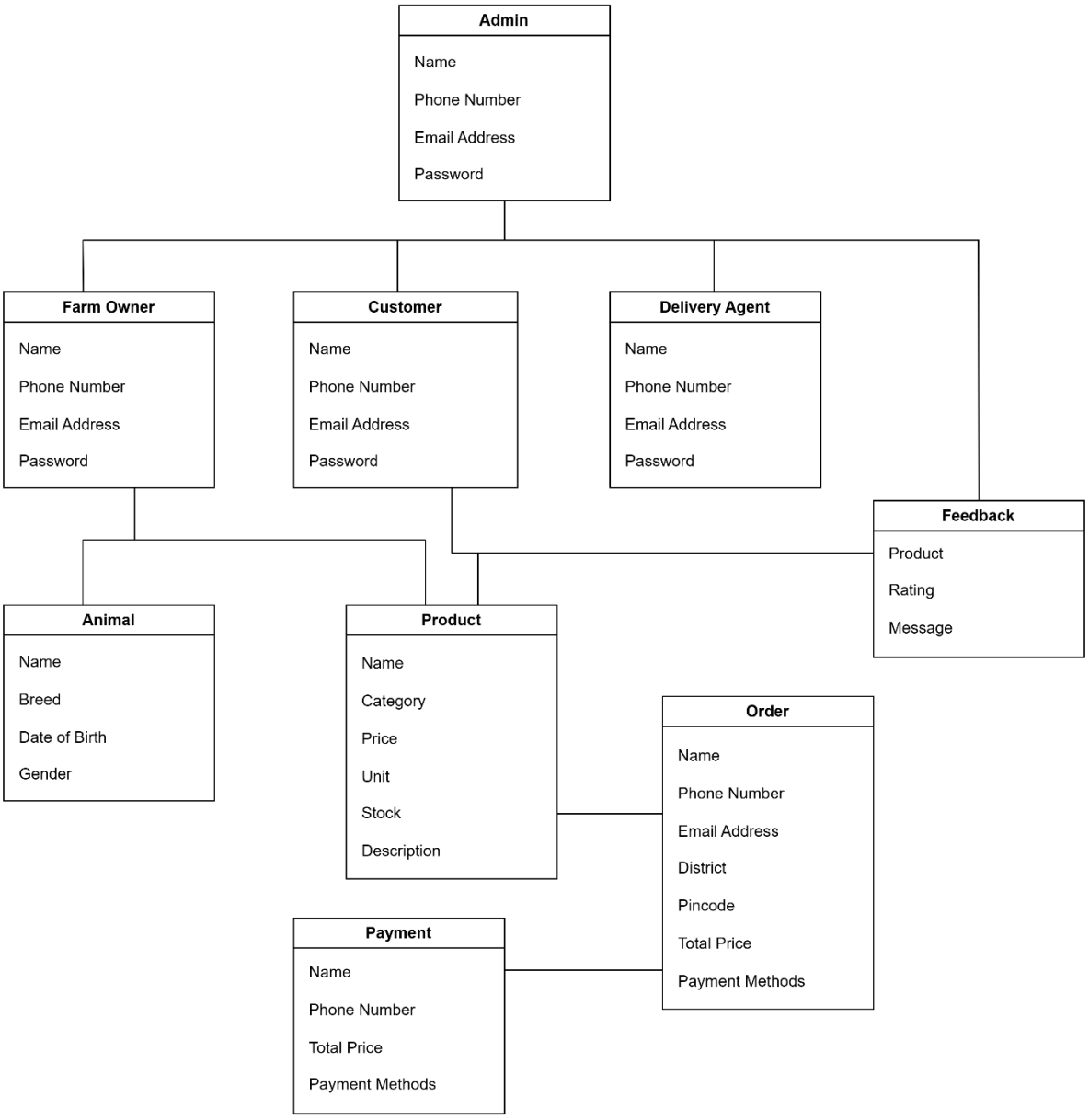


Figure 4.2.6: Object Diagram

4.2.7 Component Diagram

When complex object-oriented systems are fashioned, the component diagram provides a lot of help. It provides clients with a visual representation of the system as it shows the internal components such as programs, documents, and tools within nodes. A usage version is defined for using the element organization or connection. In component diagrams, components are used to represent system parts that can be modified and run independently. Internally, operations keep doing task running but require a specific type of way to run it, which can be imagined like a closed box that could only be opened in a proper way.

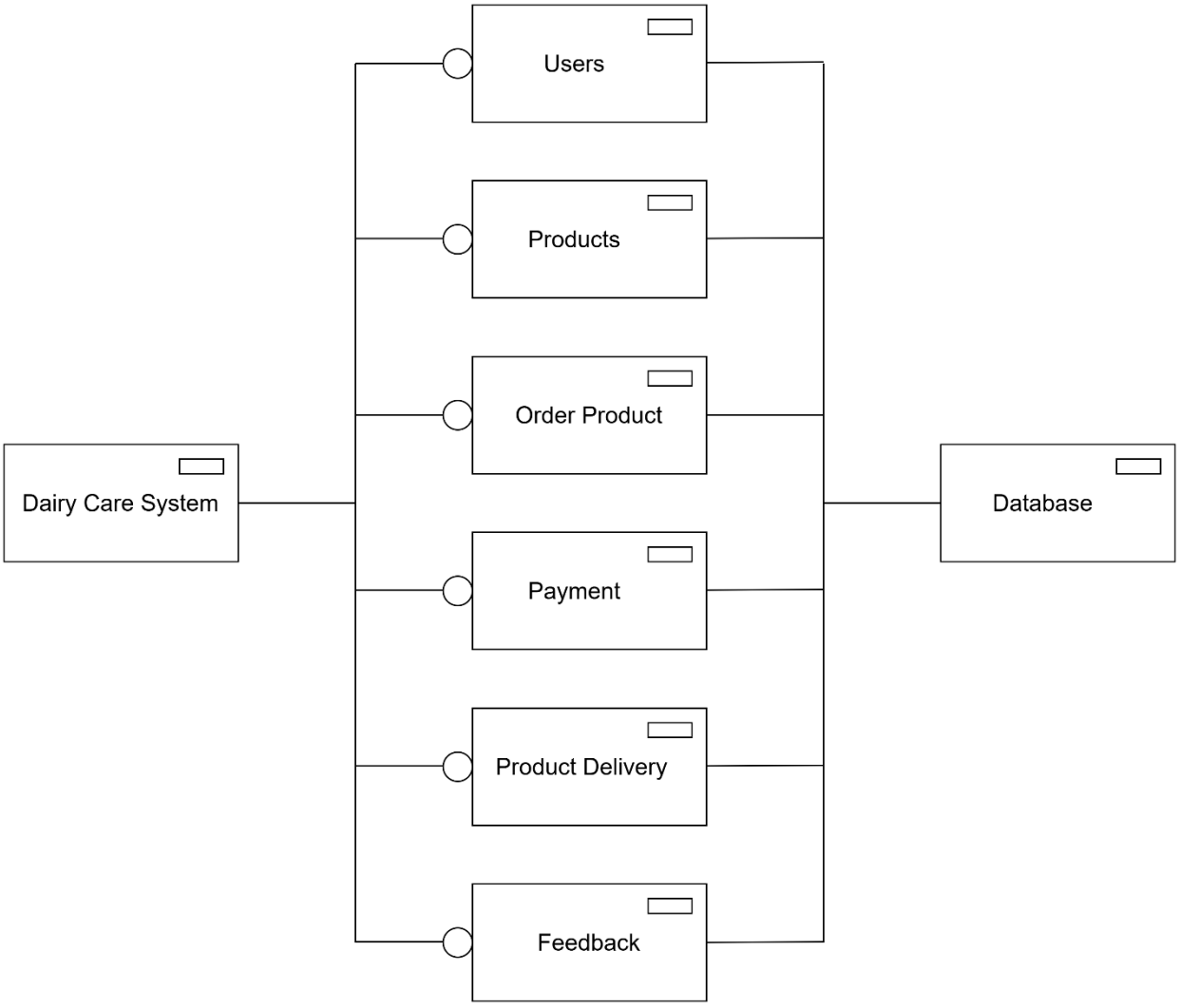


Figure 4.2.7: Component Diagram

4.2.8 Deployment Diagram

The deployment diagram displays the way software implements physical computers or server connections. It provides a static view of the system into which this arrangement of nodes and directly relates them to each other. Thus, this diagram concerns the location of programs on machines and describes how software is arranged concerning the actual physical real-world system of the computers.

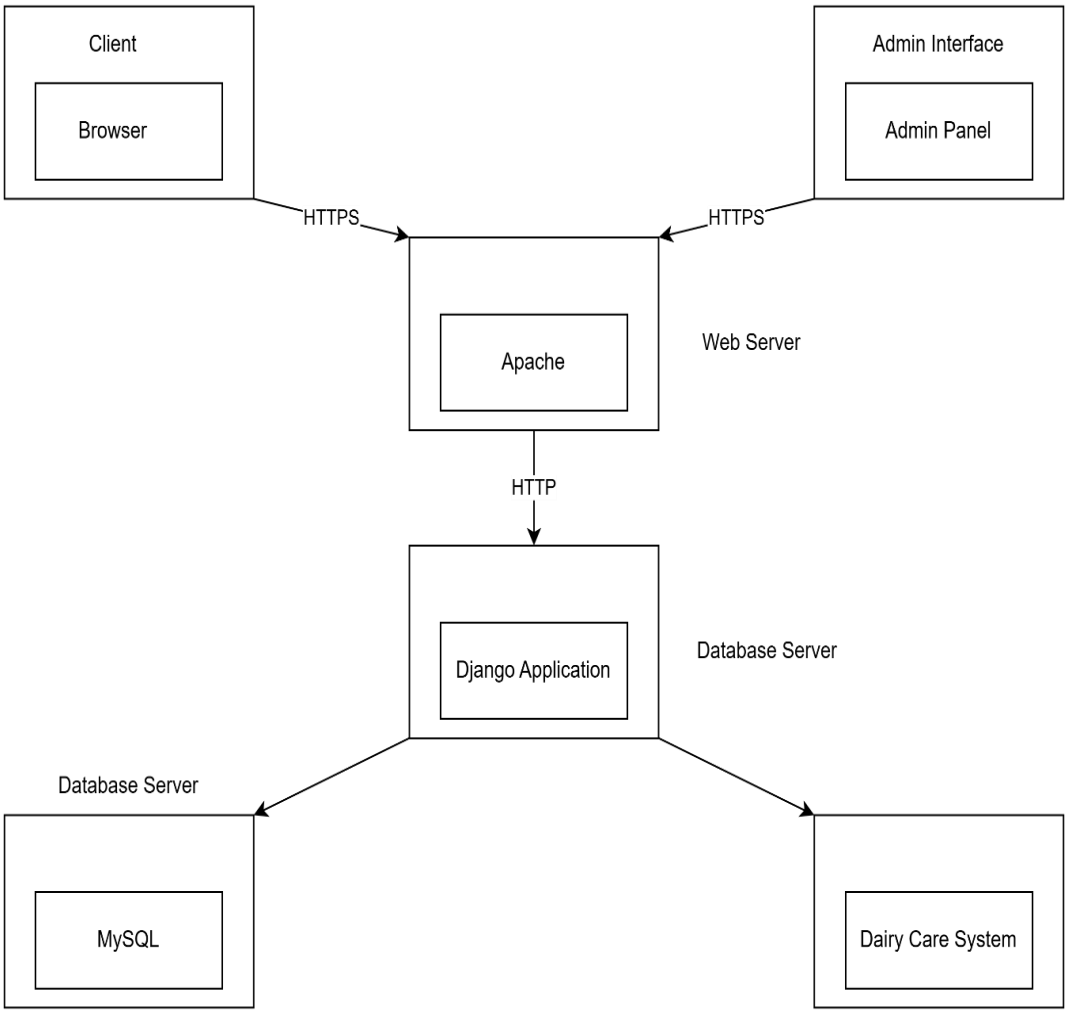


Figure 4.2.8: Deployment Diagram

4.3 USER INTERFACE DESIGN USING FIGMA

4.3.1 Home Page

Figure 4.3.1: Home Page

4.3.2 Registration Page

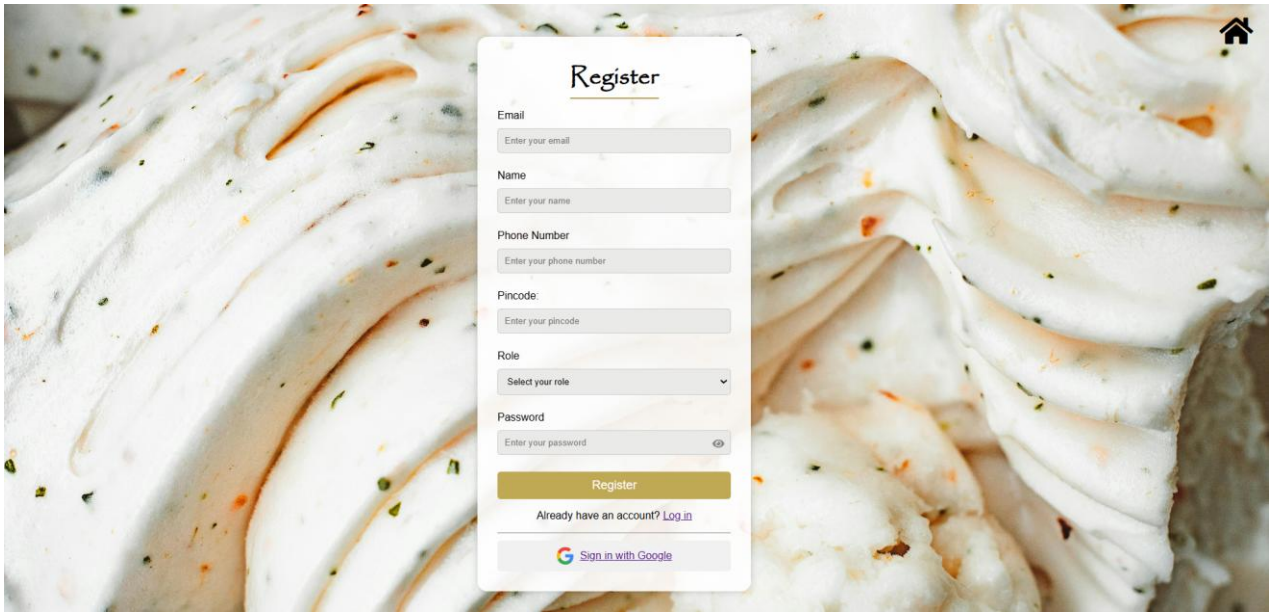


Figure 4.3.2: Registration Page

4.3.3 Login Page

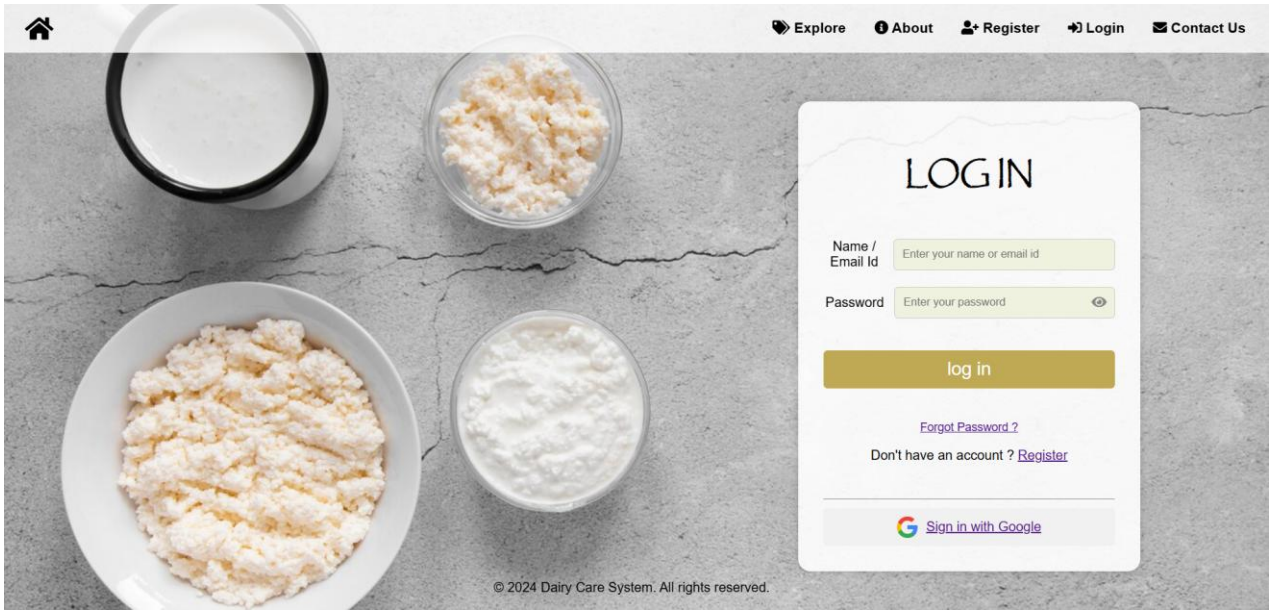


Figure 4.3.3: Login Page

4.3.4 Delivery Agent Page

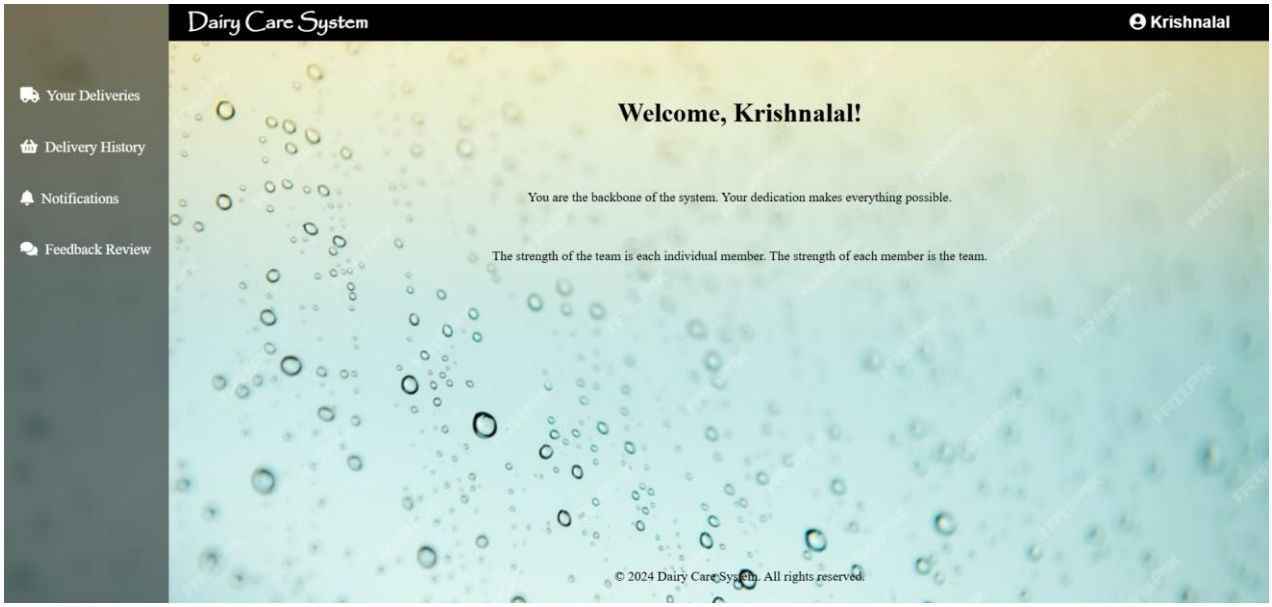


Figure 4.3.4: Delivery Agent Page

4.3.5 Deliveries Page

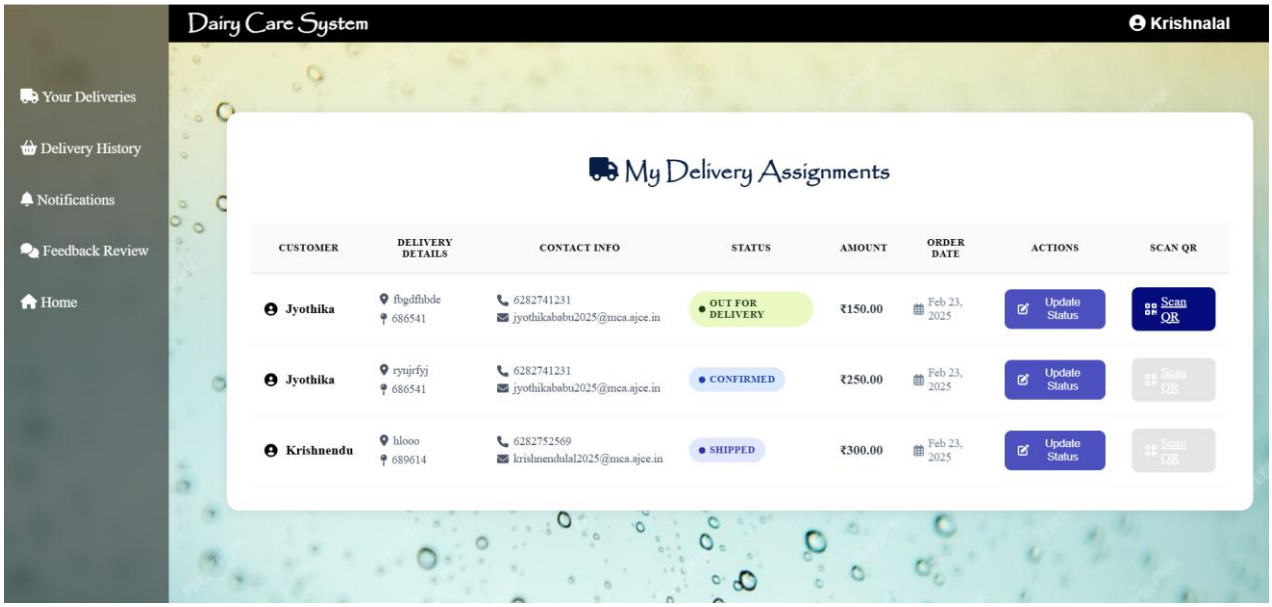


Figure 4.3.5: Deliveries Page

4.3.6 Status Updation Page

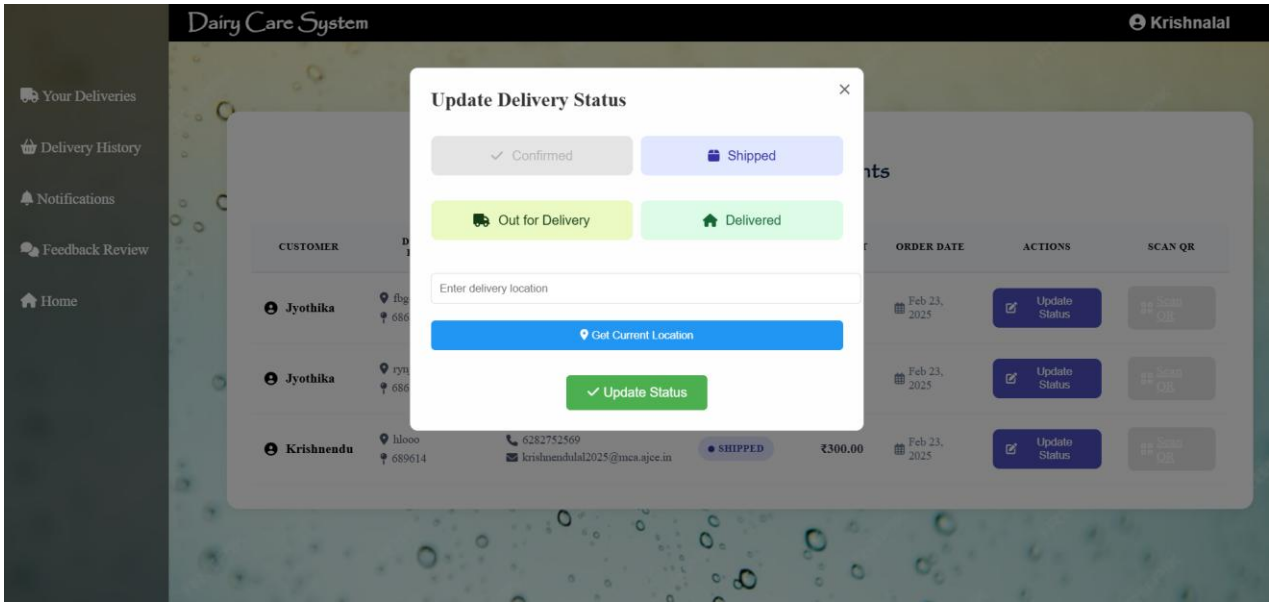


Figure 4.3.6: Status Updation Page

4.3.7 Delivery History Page

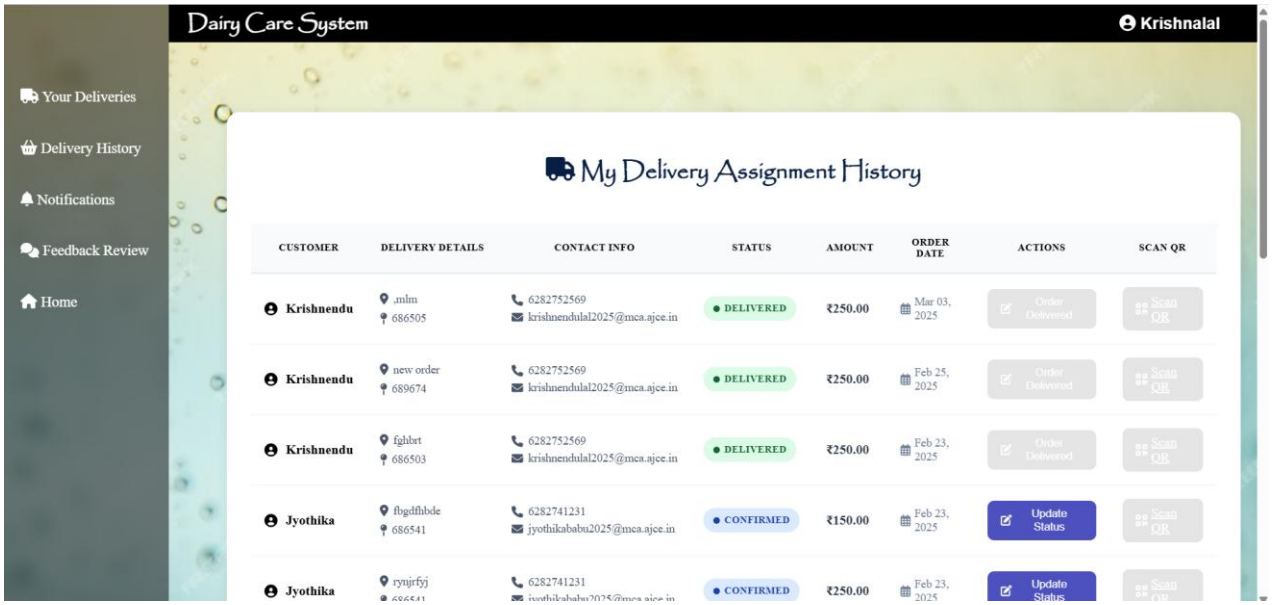


Figure 4.3.7: Delivery History Page

4.3.8 Delivery Confirmation Page

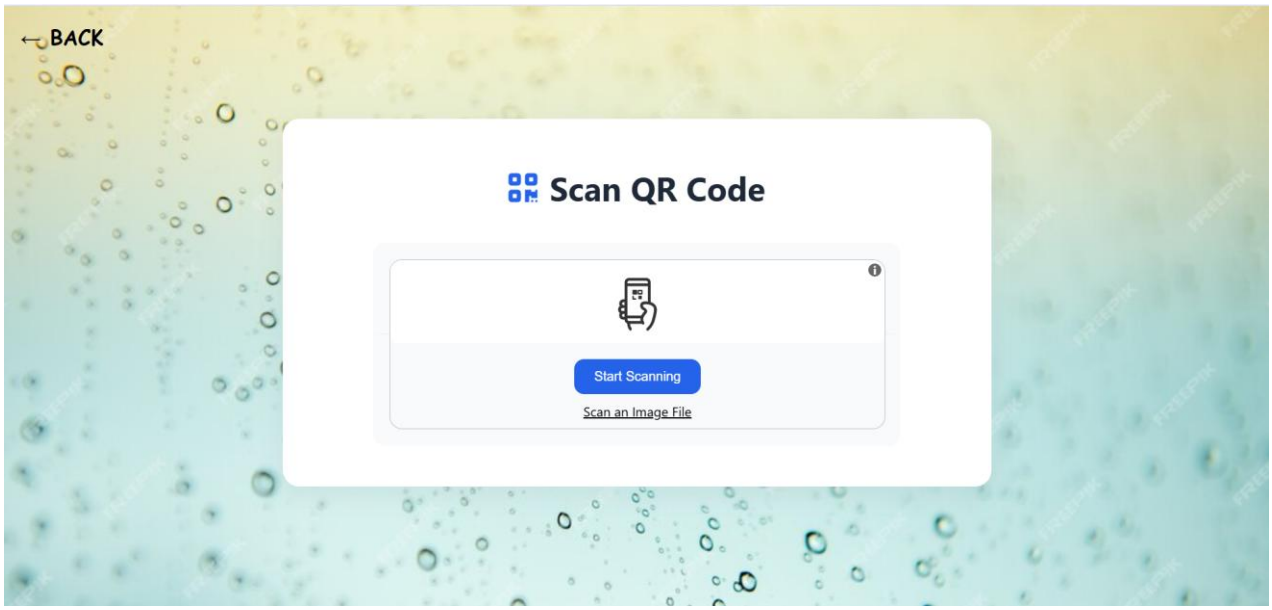


Figure 4.3.8: Delivery Confirmation Page

4.4 DATABASE DESIGN

4.4.1 Relational Database Management System (RDBMS)

Considering the definition above, a database is carefully designed apart from a data warehouse. The database system is an organized system for the storage, retrieval, and manipulation of information, the integrity and security of data stored in this database is of paramount importance to consider. There are two major steps to establishing a database.

- Firstly, one must work to understand the user's requirements and to create a database structure that matches the user's requirements.
- Essentially, the first step is to create a plan on how the data are organized that is not dependent on any specific computer program.
- Then the plan is then utilized to design for the specific computer program that is going to build the database system. This step deals with how the data gets stored using the chosen computer program.

The following are issues of concern during database development:

- Data Integrity: An assurance of correctness and consistency of data in the database.
- Data Independence: The ability of changing the physical structure of storing data without influencing an application/program's capability of accessing that data.

4.4.2 Normalization

The relational system is a method of depicting a database as an assemblage of relations. Each relationship operates like a computer information map or bulk of data. From technical usages in the definition of a table, a row refers to a tuple, and the headers of the columns are called attributes while the entire tabular set is termed a relation. In essence, a relational database is defined as a collection of tables that are formed in a special way. A row in a story represents a collection of associated entities.

Relations, Domains & Attributes

The tables regarded as containers consist of organized and related information. The rows in a table are referred to as tuples. In this sense, a tuple is an ordered list of n items. The columns within a table are considered to be attributes, each representing a property of the data. In a relational database, tables are linked to one another to maintain coherence and meaningful

association among different sets of data. This relational structure is the backbone of a well-structured database.

A domain is defined as a set of atomic values originating from the same common source or data type. Naming domains therefore becomes tremendously helpful when dealing with describing the meaning of the values that fall within the domain. Moreover, an important observation is that values that lie within a domain are truly indivisible. The key between the tables is the key in primary keys and foreign keys. An important principle in constructing the relationship is Entity Integrity and Referential Integrity. Entity Integrity refers to the primary key should be crosscutting, not null for it to be individualized and held as intact among the relationships existing between slave databases.

Normalization means organizing data for some efficient storage and to prevent redundancy yet ensuring correctness and reliability. This means laying off unwanted columns and breaking up large tables into smaller manageable tables. Normalization eliminates the risk of making errors when modifying, deleting, or entering data. Two concepts central to its normal form in data modeling are keys and relationships.

A key is a feature of the data, which, when applied, serves to distinguish one row from three other rows in the table. There are two categories of keys:

- The primary key distinguishes all similar records within the same table.
- The foreign key serves as a reference to identify records from another table.

1. First Normal Form (1NF): The first normal form describes the conditions under which it should be that no attribute may have multi-valued characteristics; every value is indivisible. Thus, having respect to each tuple, the assigned value must be of that data type which a given attribute is defined to accept. A table under 1NF implies that there cannot be another table within it, nor can a data type of table: thus ensuring all values are atomic and refer to a single instance. To get into 1NF, data is most often organized into different tables, each having its own key as per loading requirements. Thus, new relationships are set up for different kinds of data or related groups of data in the name of avoiding redundancy. One relationship is labelled first normal form if all it observes is primary key constraints.

   Eg: Below given is my user registration data storage table that has information about users id, name, email, phone number, pincode and place

Before applying 1NF:

User_id	Name	Phone no	Pincode	Place
1	Ram	869475236/ 987459826	686505	Kottayam
2	Gopika	9685741236	686503	Kottayam
3	Kalpana	7569482631	686504	Kottayam
4	Mahesh	6521459832	686503	Kottayam

After applying 1NF:

User_id	Name	Phone no	Pincode	Place
1	Ram	8694752369	686505	Kottayam
1	Ram	987459826	686505	Kottayam
2	Gopika	9685741236	686503	Kottayam
3	Kalpana	7569482631	686504	Kottayam
4	Mahesh	6521459832	686503	Kottayam

2. Second Normal Form (2NF): The second normal form follows the principle that information should not depend only on the part of the primary information that is used for organizing the data. It requires that the information be grouped into separate categories, based on each determinant and its related details. The original primary key should be maintained in reference to any dependent attribute. This step helps in removing attributes that depend only on part of the key. In simple terms, A table will be in second normal form if some data exist that can uniquely identify each instance, and all other attributes, descriptions, and meanings are functions of that primary means.

   Eg: Below given is my user registration data storage table that has information about users id, name, email, phone number, pincode and place

   Before applying 2NF:

User_id	Name	Phone no	Pincode	Place
1	Ram	8694752369	686505	Kottayam
2	Gopika	9685741236	686503	Kottayam
3	Kalpana	7569482631	686504	Kottayam
4	Mahesh	6521459832	686503	Kottayam

After applying 2NF:

User_id	Name	Phone no	Pincode
1	Ram	8694752369	686505
2	Gopika	9685741236	686503
3	Kalpana	7569482631	686504
4	Mahesh	6521459832	686503

Pincode	Place
686505	Kottayam
686504	Kottayam
686503	Kottayam

3. Third Normal Form (3NF): Database normalization is necessary in giving independence to tables and right over attributes called independence of a table. A table in 3NF does not contain columns that depend on any other non-key columns thus the attributes are functionally dependent only on the primary key. It destroys the relationship with non-key attributes to take out dependencies on attributes which are not part of the primary key. For a table to be 3NF, it should have already been in Second Normal Form (2NF) and should not allow for any non-key attributes to depend on any other non-key attribute in the same table, consequently avoiding the possibility for transitive dependency to exist. The resulting advantages of this process are data integrity, redundancy-free systems, and a better organized structure of the database.

   Eg: Below given is my user registration data storage table that has information about users id, name, email, phone number, pincode and place

   Before applying 3NF:

User_id	Name	Phone no	Pincode	Place
1	Ram	8694752369	686505	Kottayam
2	Gopika	9685741236	686503	Kottayam
3	Kalpana	7569482631	686504	Kottayam
4	Mahesh	6521459832	686503	Kottayam

After applying 3NF:

User_id	Name	Phone no	Pincode
1	Ram	8694752369	686505
2	Gopika	9685741236	686503
3	Kalpana	7569482631	686504
4	Mahesh	6521459832	686503

Pincode	Place_id
686505	1
686504	1
686503	1

Place_id	Place
1	Kottayam

4.4.3 Sanitization

While going through plentiful built-in mechanisms for the data sanitization, Django, to start with, generally supplies an incredibly wide list of field types, attached with automatic validation and concealment.

Then, it is much revered because form validation exposes yet other avenues for achieving sanitization of user input with regard to automated checking on data. Within the forms of Django are such pre-defined methods and validators by which form fields may by all means make use of: some validators check whether numerical values fall within specific ranges, and some check whether the uploaded file is in a specific format. Through this large number of built-in features split into several sub-categories in Django, integrity and safety are ensured, hence a good environment for web applications.

4.4.4 Indexing

An index is meant to maintain a single piece of information or set of information that is arranged in order according to its values: an arrangement of index entries that allows fast and easy searching for anything that matches or falls into a particular range. An index lets you find information from the database without having to check every single record during the operation of the database. An index may well work like a guide map to information in a database. An index greatly facilitates fast look-up of data and helps find records in a specified order. A single index can be created based on one or more columns of 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	pincode	Int	Not Null	Pincode of the user
9	preated_at	Timestamp	Not Null	Timestamp for when the user was created
10	updated_at	Timestamp	Not Null	Timestamp for the last update to the user's information
11	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 image_table, 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 image_table
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	view_count	Int	Default: 0	Stores the number of views for the item
12	added_at	Timestamp	Not Null	Product added time
13	updated_at	Timestamp	Not Null	Product last updated time

4.5.3 image_table

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

4.5.4 order_table

Primary Key: order_id

Foreign Key: user_id reference product_table, deliveryboy_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	Null False	Order date
4	status	Varchar (50)	Null False	Order status
5	total amount	Decimal (10, 2)	Null False	Total amount to be paid
6	payment_status	Varchar (50)	Null False	Payment status
7	name	Varchar (50)	Null False	Name of customer
8	email	Varchar (50)	Null False	Email of customer
9	phone	Int	Null False	Phone number of user
10	address	Varchar (50)	Null False	Address of customer
11	pincode	Int(6)	Null False	Pincode of Customer
12	deliveryboy_id	Int	Foreign Key	Foreign Key to users_table

4.5.5 orderItems_table

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	total_price	Decimal (10, 2)	Not Null	Total amount

4.5.6 wishlist_table

Primary Key: item_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	item_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_table

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	breed	Varchar (50)	Not Null	Livestock breed
5	date_of_birth	Timestamp	Not Null	Date of birth of livestock
6	gender	Varchar (50)	Not Null	Gender of the livestock
7	milk_capacity	Decimal (10, 2)	Not Null	Milking of livestock
8	status	Boolean	Not Null	Livestock status (1 or 0)
9	created_at	Timestamp	Not Null	Livestock added time
10	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_table

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 diseaseImage_table

Primary Key: image_id

Foreign Key: user_id reference users_table

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	image_id	Int	Primary Key	Key for this table
2	added_by_id	Int	Foreign Key	Foreign key to users_table
3	image	Blob	Not Null	Animal image file path
4	diagnosis	TextField	Not Null	Disease Diagnosed
5	uploaded_at	Timestamp	Not Null	animal image time

4.5.13 notifications_table

Primary Key: notification_id

Foreign Key: product_id reference product_table, farmer_id reference users_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
4	created_at	Timestamp	Not Null	notification send time
5	product_id	Int	Not Null	Foreign key to product_table
6	farmer_id	Int	Not Null	Foreign key to users_table

4.5.14 preorder_table

Primary Key: preorder_id

Foreign Key: user_id reference users_table

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	preorder_id	Int	Primary Key	Key for this table
2	user_id	Int	Foreign Key	Foreign key to users_table
3	order_delivery _date	Timestamp	Not Null	Preorder delivery date
4	status	Varchar (50)	Not Null	Order status
5	quantity	Decimal (10, 2)	Not Null	Quantity to be preordered
6	additional_notes	TextField	Not Null	Additional improvements in product manufacturing
7	farmer_id	Int	Foreign Key	Foreign Key to users_table
8	product_id	Int	Foreign Key	Foreign Key to users_table
9	created_At	Timestamp	Not Null	Preordered date

4.5.15 deliverystatus_table

Primary Key: delivery_id

Foreign Key: order_id reference order_table

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	delivery_id	Int	Primary Key	Key for this table
2	order_id	Int	Foreign Key	Foreign key to order_table
3	updated_at	Timestamp	Not Null	Status updation date
4	status	Varchar (50)	Not Null	Delivery status
5	location	Varchar (50)	Not Null	Location of status updation

4.5.16 paymentstatus_table

Primary Key: payment_id

Foreign Key: farmer_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	farmer_id	Int	Foreign Key	Foreign key to users_table
3	is_paid	Boolean	Default (0)	Indicates whether the payment is made
4	last_paid_amount	Decimal (10,2)	Default (0.00)	Stores the last paid amount
5	last_paid_date	Datetime	Null	Stores the date of the last payment

CHAPTER 5
SYSTEM TESTING

5.1 INTRODUCTION

Software testing is the procedure of verifying if a computer application operates according to the prescribed methods. It has been particularly turned over to know whether the software performs correctly according to the specified requirements and standards. Validation is inspecting and evaluating software for meeting the specifications criteria. Software testing is an evaluation of a program performance, typically accompanied by means such as code inspection and program walkthrough. Validation concern makes sure that an end product corresponds to the user's needs and requirements.

The main principles and objectives on which software testing is based include;

- Test is executing the program to find errors within it.
- Good test-case must be one that is very likely finding one or more of its undiscovered errors.
- An Effective with regard to tests is that of discovering previously unknown errors.

If a test case works properly and achieves the desired performance, it may also reveal defects in the programming software. This indicates that the computer program would work as intended and perform reasonably well. Evaluation of a computer program is divided into the following three major areas:

- Assessment for correctness
- Effectiveness of implementation
- Examination of computational complexity.

5.2 TEST PLAN

Test plan is essentially an all-encompassing written instructions for carrying out various types of testing in terms of the aspects it would cover. It is almost like a map which shows the various steps to engage in when evaluating a computer program. Software builders write directions for utilizing the program and organizing the information to be to provide for proper functioning of that program. Even every element of that program is being ensured to work according to specifications. For software to be well tested, it should be proven to the others-that is, the ITG (Information Technology Group), who are convened to ensure that the software works as required, but not the program developers testing their handiwork.

The levels of testing consist of:

- Unit testing
- Integration testing
- Data Validation Testing
- Output Testing

5.2.1 Unit Testing

Unit testing emphasizes the design's smallest unit such as software components or modules inside a software system. Testing crucial control paths inside a module with design document verification to hunt defects. This checks how demanding the tests are for each little part of the program and what portions remain untested within the program. An example of contemporary testing is one where the unit test may be testing some internal functions of the code while testing with respect to other parts of the program.

To start with, the data flow has to be validated for all parts of the computer program. If the proper data is not going into it and coming out of it, all other tests are meaningless. Then, designing means considering everything that could go wrong and designing accordingly. This might well include rerouting the process or stopping it altogether. Some errors in their design were detected and corrected.

After writing specifications for each part, these were subsequently tested and checked out in strict isolation. A feature that enabled code laying off was ensuring everything works up to expectations.

5.2.2 Integration Testing

Integration testing could be said to belong to the most important phases of software development that deals with program construction and error detection in the interfacing of two or more program components.

The main purpose is to take the previously tested smaller parts and assemble the program based on its specification. The scope of this testing technique is to conduct a full assessment of the program, ensuring that it behaves accurately and reasonably. Well-identified defects are likely

to lead to the identification of new ones because correcting problems generates extra testing and fixing activity within integration testing. After the careful evaluation of individual components of the system, they are integrated together to see how they actually work with one another as supposed. In addition, a genuine effort is made towards harmonizing all programs in terms of standardization, so that uniformity exists, as opposed to different version.

5.2.3 Validation Testing or System Testing

The last phase of testing is a systemic inspection to ensure that all parts of the system have demonstrated the expected characteristic interaction, involving several types of instructions and building blocks. This mode of testing is referred to as Black Box testing or System testing.

Black Box Testing decides whether software performs well against requirements or not. It's an important help for software engineers by identifying all possible faults with mixed types of user input. Black Box Testing basically involves all functions, interfaces, data access, performance, initialization and termination process error states. This is an important technique, ensuring that the software meets its intended requirements and, thus, works in what it is supposed to do.

5.2.4 Output Testing or User Acceptance Testing

The user is satisfied and hence fits the standards of the company so the system test is performed. While developing or modifying the computer program it is necessary to have closeness to the end users.

That is possible due to the following:

- Coping Screen Design.
- Output Screen Design.

Different forms of data have been used during system testing. The test data preparation is the main work in this phase. Once the test data is prepared, it gets to the step of using it to test the system under investigation. Where the problems are found with respect to this test, they get corrected following acceptable ways. The record of these will be maintained over time for further reference and improvement. This makes the system function without any defect in the performance and satisfies the requirements of the users and the organization as well.

5.2.5 Automation Testing

Automated testing does the process of testing just before the usage of software to verify its scrutiny and standards by writing its instructions fed into the automated tools. UIAutomation testing in particular is the idiomatic term for tools involved in automation testing. Instead of manual testing, where the testers clicked through the application to confirm its working, use of scripts for automation has been developed for different scenarios. The automated tests are primarily useful if one has to run them at the same time on several computers. That leads to the application of a much more identical test process.

5.2.5 Selenium Testing

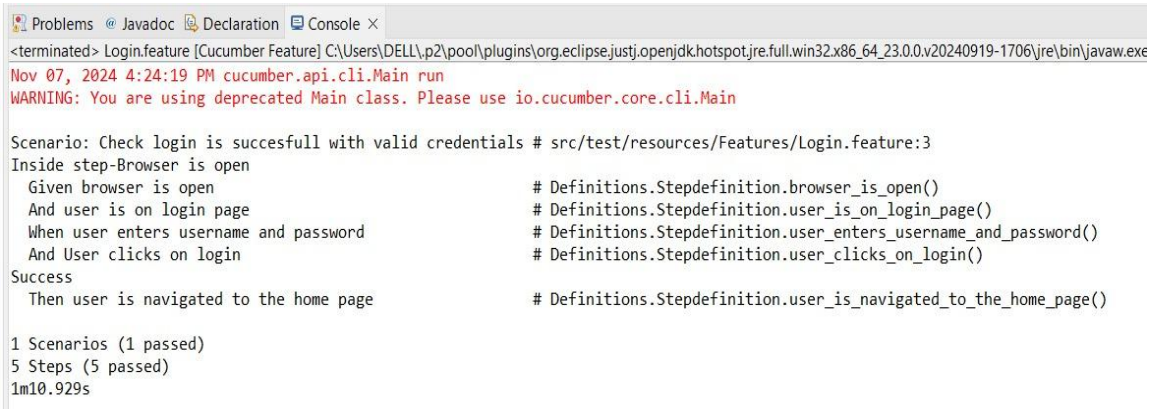
Selenium is designed as a free but very handy tool for automating web testing. For a web developer, it aids in making life easier by testing. Therefore, we can say Selenium automation testing means testing using Selenium. However, Selenium is not a single tool; rather, it is a collection of tools, each of which serves its purpose in the realm of automation testing. Manual testing, though, is an integral part of application development, and at times it just becomes too tedious. This is when Jason Huggins decided to put forth a brilliant way of automating the whole process of testing so as to replace manual tasks. He initially developed a tool called JavaScriptTestRunner, which enabled the automated testing of websites but subsequently renamed it Selenium in the year 2004.

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("krishnendulal113@gmail.com");
                   driver.findElement(By.id("password")).sendKeys("Krishna@113");
                    }

                     @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 Page Test					
Test Case ID: Test_1			**Test Designed By: Krishnendu Lal**		
Test Priority (Low/Medium/High): High			**Test Designed Date: 25/01/2025**		
Module Name: Login Module			**Test Executed By: Mr. Amal K Jose**		
Test Title: Login Page Test			**Test Execution Date: 25/01/2025**		
Description: Verify that a user can successfully log in using a valid email and 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		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				
Post-Condition: User is validated with database and successfully logs into Userpage					

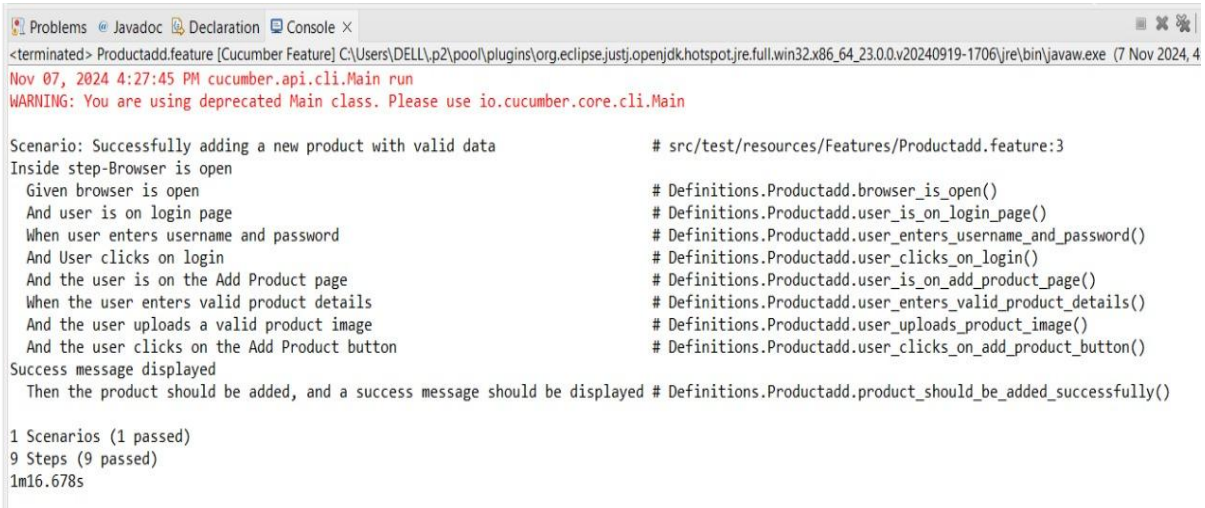
Test Case 2: Add Products Test

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");
        driver.quit();
    }
}
```

Screenshot



Test report

Test Case 2					
Project Name: Dairy Care System					
Add Products Test					
Test Case ID: Test_2			**Test Designed By: Krishnendu Lal**		
Test Priority(Low/Medium/High): High			**Test Designed Date: 10/02/2025**		
Module Name: Product Adding Module			**Test Executed By: Mr. Amal K Jose**		
Test Title: Add Products Test			**Test Execution Date: 10/02/2025**		
Description: Verify that the farm owner can successfully add a new product to the system					
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		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 successfully filled in and logged in	Username and password fields are successfully filled in and logged in	Pass
3	Owner is on add product page		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		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					

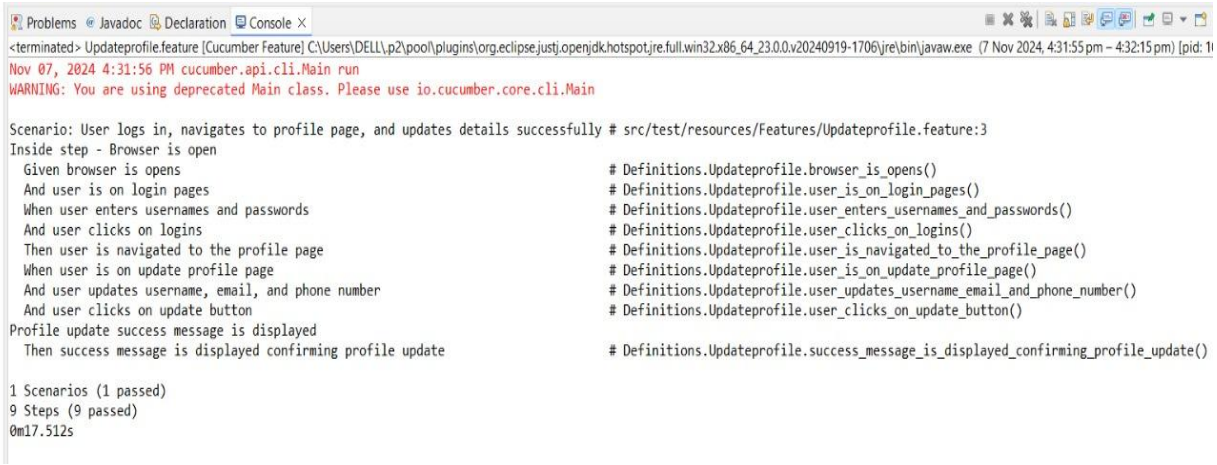
Test Case 3: Update Profile Test

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\\DELL\\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");
      Thread.sleep(2000);
```

```
    }
    @When("user is on update profile page")
    public void user_is_on_update_profile_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 {
        driver.findElement(By.id("username")).clear();
        driver.findElement(By.id("username")).sendKeys("Krishnendu");
        driver.findElement(By.id("email")).clear();
       driver.findElement(By.id("email")).sendKeys("krishnendulal2025@mca.ajce.in");
        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					
Test Case ID: Test_3			**Test Designed By: Krishnendu Lal**		
Test Priority (Low/Medium/High): High			**Test Designed Date: 15/02/2024**		
Module Name: Update Profile Module			**Test Executed By: Mr. Amal K Jose**		
Test Title: Update Profile Test			**Test Execution Date: 15/02/2024**		
Description: Verify that a logged-in user can successfully update their profile details					
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		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		User page should be displayed	User page should be displayed	Pass
6	Users view their profile		User should view their profile	User should view their profile	Pass
7	User updates their profile		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.					

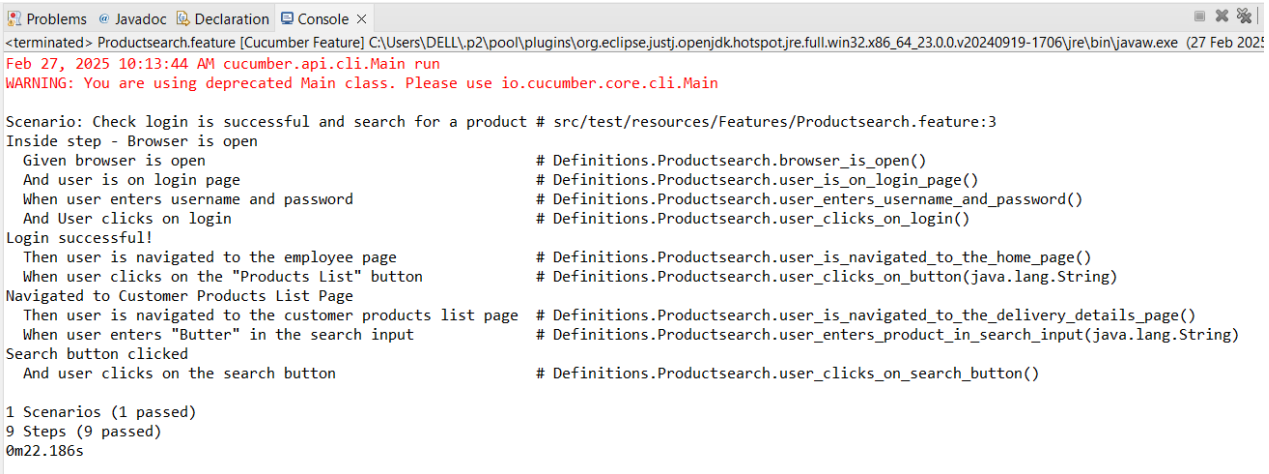
Test Case 4: Product Search Test

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.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import java.time.Duration;
import io.cucumber.java.After;
import io.cucumber.java.en.*;

public class Productsearch {

   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();
      wait = new WebDriverWait(driver, Duration.ofSeconds(5));
   }
   @And("user is on login page")
   public void user_is_on_login_page() {
      driver.navigate().to("http://127.0.0.1:8000/login");
   }
   @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();
   }
   @Then("user is navigated to the employee page")
   public void user_is_navigated_to_the_home_page() {
      wait.until(ExpectedConditions.urlToBe("http://127.0.0.1:8000/customerpage"));
      System.out.println("Login successful!");
   }
   @When("user clicks on the {string} button")
   public void user_clicks_on_button(String buttonId) {
```

```
        WebElement productsButton =
wait.until(ExpectedConditions.elementToBeClickable(By.id("products")));
        productsButton.click();
    }
    @Then("user is navigated to the customer products list page")
    public void user_is_navigated_to_the_delivery_details_page() {
        wait.until(ExpectedConditions.urlToBe("http://127.0.0.1:8000/custproductslist"));
        System.out.println("Navigated to Customer Products List Page");
    }

    @When("user enters {string} in the search input")
    public void user_enters_product_in_search_input(String productName) {
        WebElement searchInput =
wait.until(ExpectedConditions.elementToBeClickable(By.id("search-bar")));
        searchInput.sendKeys(productName);
    }

    @And("user clicks on the search button")
    public void user_clicks_on_search_button() {
        WebElement searchButton =
wait.until(ExpectedConditions.elementToBeClickable(By.id("search")));
        searchButton.click();
        System.out.println("Search button clicked");
    }
    @After
    public void tearDown() {
        if (driver != null) {
            driver.quit();
        }
    }
}
```

Screenshot



Test report

Test Case 4					
Dairy Care System					
Product Search Test					
Test Case ID: Test_4			**Test Designed By: Krishnendu Lal**		
Test Priority(Low/Medium/High): High			**Test Designed Date: 24/02/2025**		
Module Name: Product Module			**Test Executed By: Mr. Amal K Jose**		
Title: Product Search Test			**Test Execution Date: 24/02/2025**		
Description: Verify that users can search for a product by entering its name in the search bar					
Pre-Condition: User has valid username and password					
Step	**Test Step**	**Test Data**	**ExpectedResult**	**Actual Result**	**Status (Pass/ Fail)**
1	Navigate to login page		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 User page		User page should be displayed	User page displayed	Pass
6	Navigate to products list page		User should be displayed products list page	User was displayed the products list page	Pass
7	User search a product based on product name		User should be able to search a product based on products name	User was able to search a product based on products name	Pass
Post-Condition: User is validated with database and successfully logs into Users page and navigate to Products list page and search products based on the categories given					

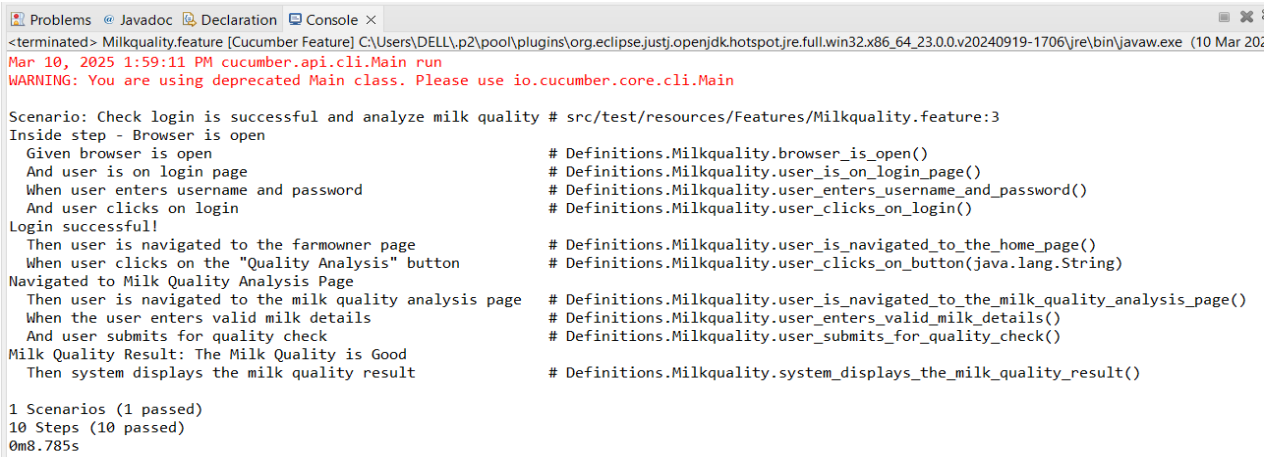
Test Case 5: Milk Quality Analysis Test

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.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import java.time.Duration;
import io.cucumber.java.After;
import io.cucumber.java.en.*;

public class Milkquality {
   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();
      wait = new WebDriverWait(driver, Duration.ofSeconds(5));
   }
   @And("user is on login page")
   public void user_is_on_login_page() {
      driver.navigate().to("http://127.0.0.1:8000/login");
   }
   @When("user enters username and password")
   public void user_enters_username_and_password() {
      driver.findElement(By.id("username")).sendKeys("shylavlal123@gmail.com");
      driver.findElement(By.id("password")).sendKeys("Shyla@123");
   }
   @And("user clicks on login")
   public void user_clicks_on_login() {
      driver.findElement(By.id("login")).click();
   }
   @Then("user is navigated to the farmowner page")
   public void user_is_navigated_to_the_home_page() {
     wait.until(ExpectedConditions.urlToBe("http://127.0.0.1:8000/farmownerpage"));
      System.out.println("Login successful!");
   }
   @When("user clicks on the {string} button")
   public void user_clicks_on_button(String buttonId) {
      WebElement milkQualityButton =
wait.until(ExpectedConditions.elementToBeClickable(By.id("milkquality")));
```

```
        milkQualityButton.click();
    }
    @Then("user is navigated to the milk quality analysis page")
    public void user_is_navigated_to_the_milk_quality_analysis_page() {
wait.until(ExpectedConditions.urlToBe("http://127.0.0.1:8000/milkqualityanalysis"));
        System.out.println("Navigated to Milk Quality Analysis Page");
    }

    @When("the user enters valid milk details")
        public void user_enters_valid_milk_details() {
            driver.findElement(By.id("phLevel")).sendKeys("6.53");
            driver.findElement(By.id("fatContent")).sendKeys("5");
            driver.findElement(By.id("density")).sendKeys("1039");
            driver.findElement(By.id("temperature")).sendKeys("6.2");
        }
    @And("user submits for quality check")
    public void user_submits_for_quality_check() {
        WebElement submitButton =
wait.until(ExpectedConditions.elementToBeClickable(By.id("analyze_button")));
        submitButton.click();
    }
    @Then("system displays the milk quality result")
    public void system_displays_the_milk_quality_result() {
        WebElement resultText =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("result")));
        System.out.println("Milk Quality Result: " + resultText.getText());
    }
    @After
    public void tearDown() {
        if (driver != null) {
            driver.quit();
        }
    }
}
```

Screenshot



Test report

Test Case 5					
Dairy Care System					
Milk Quality Analysis Test					
Test Case ID: Test_5			**Test Designed By: Krishnendu Lal**		
Test Priority(Low/Medium/High): High			**Test Designed Date: 10/03/2025**		
Module Name: Milk Module			**Test Executed By: Mr. Amal K Jose**		
Test Title: Milk Quality Analysis Test			**Test Execution Date: 10/03/2025**		
Description: Verify that the system correctly analyzes milk quality based on software-based parameters					
Pre-Condition: User has valid username and password					
Step	**Test Step**	**Test Data**	**ExpectedResult**	**Actual Result**	**Status (Pass/ Fail)**
1	Navigate to login page		Login form should be displayed	Login form is displayed Pass	Pass
2	Provide valid Email	Email : shylavlal123@gmail.com	User should be able to login	User logs in	Pass
3	Provide valid password	Password: Shyla@123			
4	Click on login button				
5	Navigated to User page		User page should be displayed	User page displayed	Pass
6	Navigate to milk quality analysis page		User should be displayed milk quality analysis page	User was displayed the milk quality analysis page	Pass
7	User enters valid details of milk for analysis		User should be able to analyze the milk quality based on entered data	User was able to analyze the milk quality	Pass
Post-Condition: User is validated with database and successfully logs into Farm Owner page and navigate to Milk Quality Analysis page and analyze quality of milk based on the entered data.					

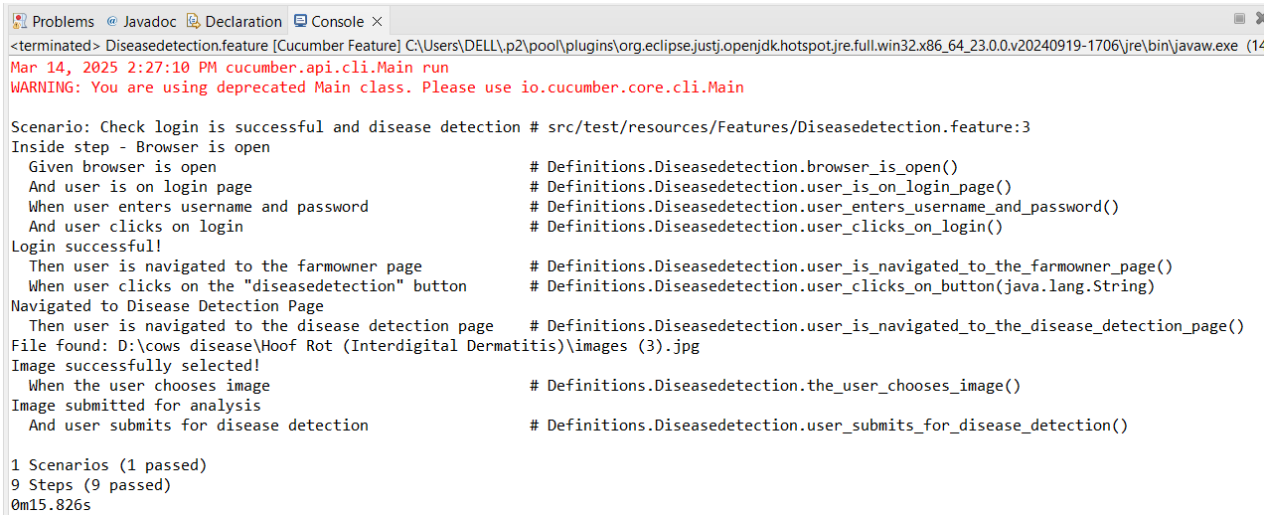
Test Case 6: Disease Detection Test

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.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import java.time.Duration;
import java.io.File;
import io.cucumber.java.en.*;

public class Diseasedetection {
   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();
      wait = new WebDriverWait(driver, Duration.ofSeconds(5));
   }
   @And("user is on login page")
   public void user_is_on_login_page() {
      driver.navigate().to("http://127.0.0.1:8000/login");
   }
   @When("user enters username and password")
   public void user_enters_username_and_password() {
      driver.findElement(By.id("username")).sendKeys("shylavlal123@gmail.com");
      driver.findElement(By.id("password")).sendKeys("Shyla@123");
   }
   @And("user clicks on login")
   public void user_clicks_on_login() {
      driver.findElement(By.id("login")).click();
   }
   @Then("user is navigated to the farmowner page")
   public void user_is_navigated_to_the_farmowner_page() {
     wait.until(ExpectedConditions.urlToBe("http://127.0.0.1:8000/farmownerpage"));
      System.out.println("Login successful!");
   }
   @When("user clicks on the {string} button")
   public void user_clicks_on_button(String buttonId) {
      WebElement diseaseDetectionButton =
```

```
wait.until(ExpectedConditions.elementToBeClickable(By.id(buttonId)));
        diseaseDetectionButton.click();
    }
    @Then("user is navigated to the disease detection page")
    public void user_is_navigated_to_the_disease_detection_page() {
wait.until(ExpectedConditions.urlToBe("http://127.0.0.1:8000/diseasedetection"));
        System.out.println("Navigated to Disease Detection Page");
    }
    @When("the user chooses image")
    public void the_user_chooses_image() {
        WebElement uploadInput =
wait.until(ExpectedConditions.presenceOfElementLocated(By.id("disease_image")));
        // Make sure the file path is correct
        String imagePath = "D:\\cows disease\\Hoof Rot (Interdigital Dermatitis)\\images (3).jpg";
        File file = new File(imagePath);
        if (file.exists()) {
            System.out.println("File found: " + imagePath);
            uploadInput.sendKeys(file.getAbsolutePath());
            System.out.println("Image successfully selected!");
        } else {
            System.out.println("Error: File not found at " + imagePath);
        }
    }
    @And("user submits for disease detection")
    public void user_submits_for_disease_detection() {
        WebElement submitButton =
wait.until(ExpectedConditions.elementToBeClickable(By.id("analyze_result")));
        submitButton.click();
        System.out.println("Image submitted for analysis");
    }
}
```

Screenshot



Test report

Test Case 6					
Dairy Care System					
Disease Detection Test					
Test Case ID: Test_6			**Test Designed By: Krishnendu Lal**		
Test Priority(Low/Medium/High): High			**Test Designed Date: 14/03/2025**		
Module Name: Milk Module			**Test Executed By: Mr. Amal K Jose**		
Test Title: Disease Detection Test			**Test Execution Date: 14/03/2025**		
Description: Verify that the system accurately detects diseases in dairy animals using image processing					
Pre-Condition: User has valid username and password					
Step	**Test Step**	**Test Data**	**ExpectedResult**	**Actual Result**	**Status (Pass/ Fail)**
1	Navigate to login page		Login form should be displayed	Login form is displayed Pass	Pass
2	Provide valid Email	Email : shylavlal123@gmail.com	User should be able to login	User logs in	Pass
3	Provide valid password	Password: Shyla@123			
4	Click on login button				
5	Navigated to User page		User page should be displayed	User page displayed	Pass
6	Navigate to disease detection page		User should be displayed disease detection page	User was displayed the disease detection page	Pass
7	User chooses image for detection		User should be able to detect disease based on chosen image	User was able to detect disease from the image	Pass
Post-Condition: User is validated with database and successfully logs into Farm Owner page and navigate to disease detection page and detect disease based on the chosen image.					

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

The stage of implementation is where the intended system moves to reality and becomes operational. Gaining confidence and trust from the user is critical in ensuring the success of the system. The very critical stage focuses much on user training and educational materials. The actual change is effected very often during or just after user training. It signifies the shipment of the new system into action after design work is done-a working model delivered from a conceptual design. Implementation is often said to refer to the process of shifting the manner of doing things-whether by replacing an old system with a new or modifying the old one. In order to build a system that fulfills the needs of the organization, it is crucial that this process is done in the right manner.

The following activities characterize system implementation:

- An exhaustive planning and evaluation of the existing system and its shortcoming.
- Strategies for transition are designed.

6.2 IMPLEMENTATION PROCEDURES

The acceptable implementation process involves the final acceptance of software after it was installed on hardware that will be used. A non-user or a set of users may still sign off on the development of the project. This individual could be positioned at senior management, which would create a certain amount of skepticism towards the new software from the very beginning. It would be vital to appease these early concerns to prevent strong resistance later on.

For a smooth transition, some critical success factors would involve:

- Communicating Benefits: Users would need to feel that new software truly is better than the old one in order to have confidence in it.
- User Training: This is about providing training for users to feel confident and, unlike the old program, comfortable using the application.

The end-results evaluation will need to check the server program operationality on the server; otherwise, any outcome could be far away from what was planned.

6.2.1 User Training

Teaching users is an important nexus in making people know the system efficiencies and how to adapt to a system. It acts as an important vehicle for building user comfort and confidence in a system. This need becomes important as the system becomes much more complex than it is already. User training is meant to cover inputting information, handling errors, querying databases, and using tools to perform reporting and key work. Ultimately, the program is meant to empower a person with knowledge and skills to use the system more effectively.

6.2.2 Training on the Application Software

The second stage of training focuses-on actually familiarizing users to a new application after the basics of computer skills have been covered. The training should instruct-how to understand and master entering new system application screens, getting help, error messages and resolution procedures. It is designed to train end users or groups of users to operate the system and its components effectively. The training program is likely to differ depending on user segments and even individuals in various roles within the organization.

6.2.3 System Maintenance

Software maintenance is very complex. It is during the maintenance phase of the software life cycle that the application performs important critical tasks while still smoothly operating. Therefore, the application is put into operation and will continuously require keeping in touch with it to ensure that optimal performance continues on the system. This is a significant part of development maintenance because it is what provides the possibility of altering the system to suit the environmental condition within which it exists. Maintenance is what allows the system to change according to the environment within which it exists. Maintenance is more than just finding bugs in the code and fixing them. It includes those actions designed to add toward a greater stability and efficiency of the system.

6.2.4 Hosting

Hosting is the means through which all files pertaining to one's website are kept and even referred to remote servers out of which they can be accessed by visitors who want to view the website over the internet. For example, a website owner wanting to have his site viewable

across the internet by many people should host it on a server such that others can view it. There are so many types of hosting like shared hosting for example dedicated hosting, VPS hosting, cloud hosting, and so many others. The option of hosting usually depends on the magnitude of the site itself, the amount of traffic, and personal requirements for control and flexibility from the owner's end.

AWS

AWS actually stands for Amazon Web Services, a cloud-hosting service that offers a hassle-free platform for hosting web applications, APIs, static sites, and databases for its clients. It shields the entire configuration, scaling, and load balancing of applications so that development efforts can focus fully on application construction while infrastructure maintenance is left for all those not developing. With AWS support for countless languages and frameworks, including Django, it even provides an easy way to integrate into Git for continuous deployment.

Procedure for hosting a website on AWS

Step 1: Log in to your AWS account.

Step 2: Create new EC2 instance for your hosting needs.

Step 3: Create a requirements.txt including all of your dependencies that are going to be installed.

Step 4: Upload the instance for deployment of your project.

Step 5: Configure related settings on the AWS service (like security groups, environment variables, domain settings) and deploy.

Hosted Website: AWS

Hosted Link: https://krishnendulal.github.io/dairycare/

Hosted QR:

DAIRY CARE SYSTEM

Screenshot

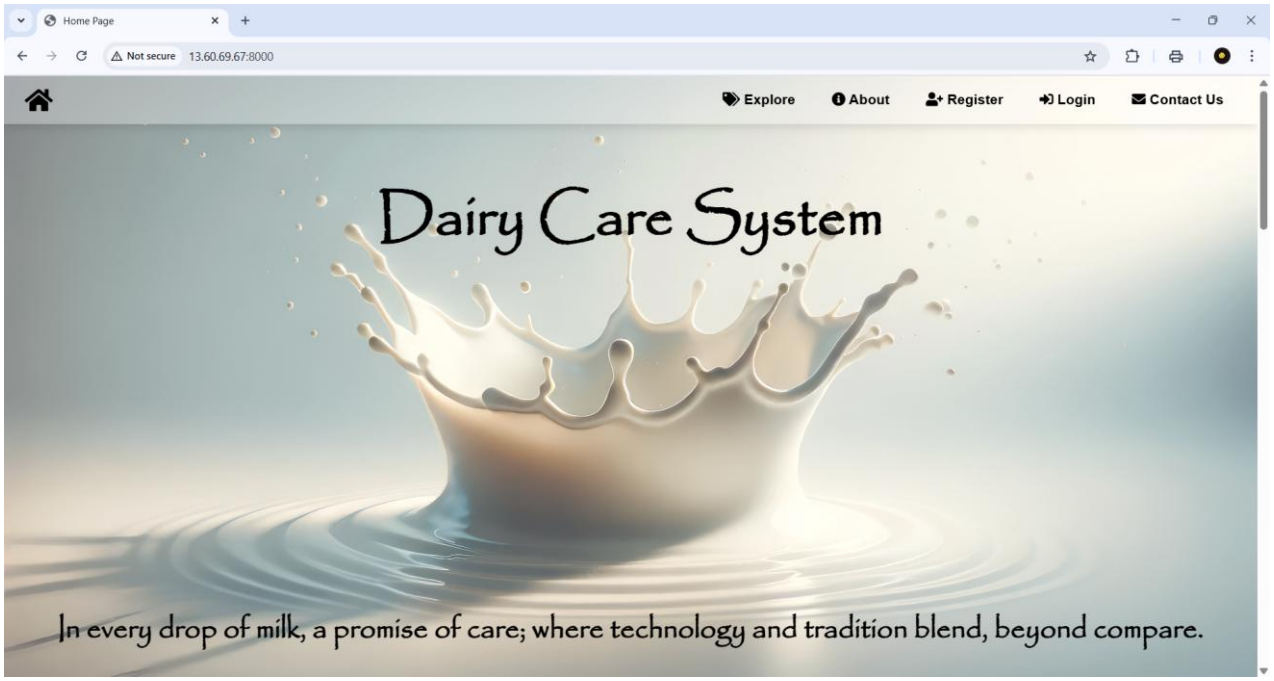


Figure: Hosted Landing page

CHAPTER 7
CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The Dairy Care System integrates all pertinent features, such as product and inventory management. On analysis, Dairy Care System should be well grounded and trusted to carry out the management of dairy farm operations, specifically tailored to meet the needs of a private dairy farm, addressing all complexities in the management of livestock, farm activity monitoring, and product sales of simple and secure mode of access. The system, with its whole spectrum of functions, has undeniably made life easier for dairy farm owners, employees, and customers alike, thus providing them with a powerful tool in dairy farm management.

The Dairy Care System integrates key functionalities like product and inventory management, animal health surveillance, and payments using an intuitive interface. Admins are given centralized access to user upkeep, inventory management, farm monitoring, and order processing. By virtue of a strong backend with Django and a fresh design in the front end of the system, reliability, security, and efficiency in deli daily operations are guaranteed. This project provides a real solution to dairy management, enhancing the operational efficiency of the farm and thereby contributing to achieving productive farm activities.

7.2 FUTURE SCOPE

The future aspect of this Dairy Care System project looks bright as it can take a throw in extensions and improvements for effective management of dairy farm operations. Some of the improvements would be the installation of IoT devices that may perform real-time monitoring of livestock health, environmental conditions, and automatic feeding systems. These would provide the system's continuous data update and allow for health monitoring to be more accurate with preventive treatment.

Other Important additions include widening the analytics and reports features to provide predictive insights on the expected amounts of production, sales trends, and resources needed. AI-based recommendations for animal nutrition and prediction for maintenance on the farm will probably help strengthen farm operations, cost reduction, and optimization in productivity. The future considerations would even further enhance the Dairy Care System's strength, flexibility, and feasibility as these types of dairy farms managerial environments evolve in the future.

CHAPTER 8
BIBLIOGRAPHY

REFERENCES:

- Django for Beginners: Build Websites with Python and Django by William S. Vincent
- Python Crash Course by Eric Matthes
- Disease detection using VGG16 CNN architecture by S. Shinde
- Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow by Aurélien Géron
- A Comprehensive Review on Image-Based Plant Disease Detection Using Deep Learning by Patel et al.
- Python Crash Course by Eric Matthes
- Django for Professionals: Production-ready Django by William S. Vincent
- Machine Learning Approaches for Livestock Disease Prediction, Journal of Veterinary Science & AI

WEBSITES:

- docs.djangoproject.com
- docs.python.org/3/
- razorpay.com/docs/
- getbootstrap.com
- keras.io/
- pytorch.org/docs/
- scikit-learn.org/stable/documentation.html

CHAPTER 9

APPENDIX

9.1 Sample Code

Delivery Agent Page

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <script src="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-
beta3/js/all.min.js"></script>
   <title>My Deliveries</title>
   <link rel="stylesheet" href="{% static 'css/headsidenav.css' %}">
   <link rel="stylesheet" href="{% static 'css/deliveryagent.css' %}">
   <style>
      body {
         background: url('{% static "images/employeepage.jpg" %}') no-repeat center center fixed;
         background-size: cover;
         margin: 0;
         margin-left: 250px;
      }
   </style>
   <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/sweetalert2@11/dist/sweetalert2.min.css">
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">
</head>
<body>
   <div id="mySidenav" class="sidenav"><br><br><br><br>
      <a href="{% url 'deliveryboyassigns' %}"><i class="fas fa-truck"></i> Your
Deliveries</a><br>
      <a href="{% url 'deliverydetailsall' %}"><i class="fas fa-shopping-basket"></i> Delivery
History</a><br>
      <a href="{% url 'employeepage' %}"><i class="fas fa-home home-icon"></i>
Home</a><br>
   </div>

   <ul>
      <div class="brand-name"></i><span>Dairy Care System</span></div>
      <div class="header">
         <div class="nav-item">
            <li>
               <b><a href="#" class="username" onclick="toggleDropdown()"><i class="fas fa-
user-circle"></i> {{ username }}</a></b>
               <div class="dropdown-content" id="dropdown-content">
                  <a href="{% url 'userprofile' %}"><i class="fas fa-user"></i> Profile</a>
                  <!-- Change Logout link to trigger a custom SweetAlert -->
                  <a href="#" onclick="confirmLogout()"><i class="fas fa-sign-out-alt"></i>
```

```
  Logout</a>
                </div>
              </li>
            </div>
          </div>
        </ul><br><br><br><br><br><br>

        <div class="delivery-container">
          <h1 class="page-title"><i class="fas fa-truck"></i> My Delivery Assignments</h1>
          {% csrf_token %}
          {% if assigned_orders %}
            <table class="orders-table">
              <thead>
                <tr>
                  <th>Customer</th>
                  <th>Delivery Details</th>
                  <th>Contact Info</th>
                  <th>Status</th>
                  <th>Amount</th>
                  <th>Order Date</th>
                  <th>Actions</th>
                  <th>Scan QR</th>
                </tr>
              </thead>
              <tbody>
                {% for order in assigned_orders %}
                  <tr data-order-id="{{ order.id }}">
                    <td>
                      <div class="customer-info">
                        <i class="fas fa-user-circle"></i>
                        <strong>{{ order.name }}</strong>
                      </div>
                    </td>
                    <td class="address-cell">
                      <div class="contact-info">
                        <span><i class="fas fa-map-marker-alt"></i>{{ order.delivery_address
  }}</span>
                        <span><i class="fas fa-map-pin"></i>{{ order.pincode }}</span>
                      </div>
                    </td>
                    <td>
                      <div class="contact-info">
                        <span><i class="fas fa-phone"></i>{{ order.phone }}</span>
                        <span><i class="fas fa-envelope"></i>{{ order.email }}</span>
                      </div>
                    </td>
                    <td>
                      <span class="status-badge status-{{ order.status|lower }}">
                        <i class="fas fa-circle" style="font-size: 8px;"></i>
                        {{ order.get_status_display }}
                      </span>
```

```
                    </td>
                    <td class="price-column">₹{{ order.total_price }}</td>
                    <td>
                      <div class="order-date">
                        <i class="far fa-calendar-alt"></i>
                        {{ order.order_date|date:"M d, Y" }}
                      </div>
                    </td>
                    <td>
                      <button class="update-status-btn"
                              onclick="openStatusModal('{{ order.id }}')"
                              {% if order.status == 'Cancelled' or order.status == 'Delivered' %}
disabled style="opacity: 0.5; cursor: not-allowed; background-color: #cccccc;" {% endif %}>
                        <i class="fas fa-edit"></i>
                        {% if order.status == 'Cancelled' %} Order Cancelled
                        {% elif order.status == 'Delivered' %} Order Delivered
                        {% else %} Update Status
                        {% endif %}
                      </button>
                    </td>
                    <td>
                      <a href="{% url 'scan_qr' %}?{{ order.id }}"
                        class="qr-scanner-btn"
                        {% if order.status != 'Out' %}
                        style="opacity: 0.5; cursor: not-allowed; background-color: #cccccc;"
                        onclick="return false;"
                        {% endif %}>
                         <i class="fas fa-qrcode"></i> Scan QR
                      </a>
                    </td>
                  </tr>
                {% endfor %}
              </tbody>
            </table>
          {% else %}
            <div class="empty-state">
              <i class="fas fa-box-open"></i>
              <p>No deliveries have been assigned to you yet.</p>
            </div>
          {% endif %}
        </div>

        <div id="statusModal" class="modal">
          <div class="modal-content">
            <span class="close">&times;</span>
            <h3 class="modal-title">Update Delivery Status</h3>

            <div class="status-options">
              <button class="status-btn confirmed" onclick="selectStatus('Confirmed')">
                <i class="fas fa-check"></i> Confirmed
              </button>
```

```
            <button class="status-btn shipped" onclick="selectStatus('Shipped')">
              <i class="fas fa-box"></i> Shipped
            </button>
            <button class="status-btn out" onclick="selectStatus('Out')">
              <i class="fas fa-truck"></i> Out for Delivery
            </button>
            <button class="status-btn delivered" onclick="selectStatus('Delivered')">
              <i class="fas fa-home"></i> Delivered
            </button>
          </div>

          <div class="location-section">
            <input type="text" id="location" placeholder="Enter delivery location" class="location-
input">
            <button class="fetch-location-btn" onclick="getCurrentLocation()">
              <i class="fas fa-map-marker-alt"></i> Get Current Location
            </button>
          </div>

          <div class="modal-footer">
            <button class="update-confirm-btn" onclick="submitStatusUpdate()">
              <i class="fas fa-check"></i> Update Status
            </button>
          </div>
        </div>
      </div>

      <div id="qrScannerModal" class="modal">
        <div class="modal-content">
          <div class="modal-header">
            <h3><i class="fas fa-qrcode"></i> Scan QR Code</h3>
            <span class="close" onclick="closeQRScannerModal()">&times;</span>
          </div>
          <div class="modal-body">
            <div id="reader"></div>
            <div id="result"></div>
          </div>
        </div>
      </div>

      <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
      <script src="https://unpkg.com/html5-qrcode"></script>
      <script>

        let currentOrderId = null;
        let selectedStatus = null;
        let html5QrcodeScanner = null;
        let currentOrderIdForQR = null;

        function openStatusModal(orderId) {
          currentOrderId = orderId;
```

```
    const row = document.querySelector(`tr[data-order-id="${orderId}"]`);
    const statusBadge = row.querySelector('.status-badge');
    const currentStatus = statusBadge.textContent.trim();
    document.getElementById('statusModal').style.display = 'block';

    selectedStatus = null;
    document.querySelectorAll('.status-btn').forEach(btn => {
        btn.classList.remove('selected');
    });

    updateStatusButtons(currentStatus);
}

function closeStatusModal() {
    document.getElementById('statusModal').style.display = 'none';
}
document.querySelector('.close').onclick = function() {
    closeStatusModal();
}

window.onclick = function(event) {
    let modal = document.getElementById('statusModal');
    if (event.target == modal) {
        closeStatusModal();
    }
}

function submitStatusUpdate() {
    const locationInput = document.getElementById('location').value;

    if (!selectedStatus) {
        Swal.fire({
            icon: 'error',
            title: 'Status Required',
            text: 'Please select a delivery status'
        });
        return;
    }
    if (!locationInput) {
        Swal.fire({
            icon: 'error',
            title: 'Location Required',
            text: 'Please enter or fetch current location'
        });
        return;
    }

    updateStatus(selectedStatus, locationInput);
}

function selectStatus(status) {
```

```
            const button = event.target.closest('.status-btn');
            if (!button.disabled) {
                selectedStatus = status;
                document.querySelectorAll('.status-btn').forEach(btn => {
                    btn.classList.remove('selected');
                });
                button.classList.add('selected');

                if (status === 'Delivered') {
                    window.location.href = "{% url 'scan_qr' %}" + "?" + currentOrderId;
                    return;
                }
            }
        }

        function getCurrentLocation() {
            if (navigator.geolocation) {
                navigator.geolocation.getCurrentPosition(
                    function(position) {
                        fetch(`https://nominatim.openstreetmap.org/reverse?format=json&lat=${position.c
oords.latitude}&lon=${position.coords.longitude}`)
                            .then(response => response.json())
                            .then(data => {
                                document.getElementById('location').value = data.display_name;
                            })
                            .catch(error => {
                                console.error('Error fetching address:', error);
                                document.getElementById('location').value =
                                    `${position.coords.latitude}, ${position.coords.longitude}`;
                            });
                    },
                    function(error) {
                        Swal.fire({
                            icon: 'error',
                            title: 'Location Error',
                            text: 'Unable to fetch location. Please enter manually.'
                        });
                    }
                );
            } else {
                Swal.fire({
                    icon: 'error',
                    title: 'Geolocation Not Supported',
                    text: 'Your browser does not support geolocation. Please enter location manually.'
                });
            }
        }

        function updateStatus(status, location) {
            if (!currentOrderId) return;
```

```
    const csrfToken = document.querySelector('[name=csrfmiddlewaretoken]').value;
    fetch('/update_order_status/', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'X-CSRFToken': csrfToken
        },
        body: JSON.stringify({
            order_id: currentOrderId,
            status: status,
            location: location
        })
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            const row = document.querySelector(`tr[data-order-id="${currentOrderId}"]`);
            const statusCell = row.querySelector('.status-badge');
            statusCell.className = `status-badge status-${status.toLowerCase()}`;
            statusCell.innerHTML = `
                <i class="fas fa-circle" style="font-size: 8px;"></i>
                ${status}
            `;

            closeStatusModal();

            Swal.fire({
                icon: 'success',
                title: 'Status Updated!',
                text: `Order status has been updated to ${status}`,
                timer: 2000,
                showConfirmButton: false
            });
        } else {
            Swal.fire({
                icon: 'error',
                title: 'Update Failed',
                text: 'Failed to update the status. Please try again.'
            });
        }
    })
    .catch(error => {
        console.error('Error:', error);
        Swal.fire({
            icon: 'error',
            title: 'Error',
            text: 'An error occurred while updating the status.'
        });
    });
}
```

```
function openQRScanner(orderId) {
    currentOrderIdForQR = orderId;
    document.getElementById('qrScannerModal').style.display = 'block';
    if (!html5QrcodeScanner) {
        html5QrcodeScanner = new Html5QrcodeScanner(
            "reader", { fps: 10, qrbox: 250 });

        html5QrcodeScanner.render(onScanSuccess, onScanError);
    }
}

function closeQRScannerModal() {
    document.getElementById('qrScannerModal').style.display = 'none';
    if (html5QrcodeScanner) {
        html5QrcodeScanner.clear();
        html5QrcodeScanner = null;
    }
}
function onScanSuccess(decodedText, decodedResult) {
    html5QrcodeScanner.clear();
    const urlPattern = `confirm-delivery/${currentOrderIdForQR}/`;
    if (decodedText.includes(urlPattern)) {
        Swal.fire({
            title: 'Confirm Delivery',
            text: 'Are you sure you want to mark this order as delivered?',
            icon: 'question',
            showCancelButton: true,
            confirmButtonColor: '#3085d6',
            cancelButtonColor: '#d33',
            confirmButtonText: 'Yes, mark as delivered',
            cancelButtonText: 'No, cancel',
            allowOutsideClick: false // Prevent clicking outside to dismiss
        }).then((result) => {
            if (result.isConfirmed) {
                const csrfToken = document.querySelector('[name=csrfmiddlewaretoken]').value;

                fetch(`/confirm-delivery/${currentOrderIdForQR}/`, {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json',
                        'X-CSRFToken': csrfToken
                    }
                })
                .then(response => response.text())
                .then(data => {
                    Swal.fire({
                        title: 'Success!',
                        text: 'Delivery confirmed successfully!',
                        icon: 'success'
                    }).then(() => {
                        // Refresh the page to update the status
```

```
                        location.reload();
                    });
                })
            } else {
                Swal.fire({
                    title: 'Invalid QR Code',
                    text: 'This QR code does not match the order',
                    icon: 'error'
                });
            }
            closeQRScannerModal();
        }
        function onScanError(error) {
            console.warn(`QR scan error: ${error}`);
        }
            const currentIndex = statusOrder[currentStatus] || 0;
            const statusButtons = document.querySelectorAll('.status-btn');
            statusButtons.forEach(button => {
                const buttonText = button.textContent.trim();
                let buttonStatus;

                if (buttonText.includes('Confirmed')) buttonStatus = 'Confirmed';
                else if (buttonText.includes('Shipped')) buttonStatus = 'Shipped';
                else if (buttonText.includes('Out')) buttonStatus = 'Out';
                else if (buttonText.includes('Delivered')) buttonStatus = 'Delivered';
                if (statusOrder[buttonStatus] <= currentIndex) {
                    button.disabled = true;
                    button.style.opacity = '0.5';
                    button.style.cursor = 'not-allowed';
                    button.style.backgroundColor = '#cccccc';
                    button.style.color = '#666666';
                    button.classList.remove('selected');
                } else {
                    button.disabled = false;
                    button.style.removeProperty('opacity');
                    button.style.removeProperty('cursor');
                    button.style.removeProperty('background-color');
                    button.style.removeProperty('color');
                    if (buttonStatus === 'Confirmed') button.className = 'status-btn confirmed';
                    else if (buttonStatus === 'Shipped') button.className = 'status-btn shipped';
                    else if (buttonStatus === 'Out') button.className = 'status-btn out';
                    else if (buttonStatus === 'Delivered') button.className = 'status-btn delivered';
                }
            });
        }
    </script>
</body>
</html>
```

9.2 Screen Shots

9.2.1 Home Page

Figure 9.2.1: Home Page

9.2.2 Registration Page

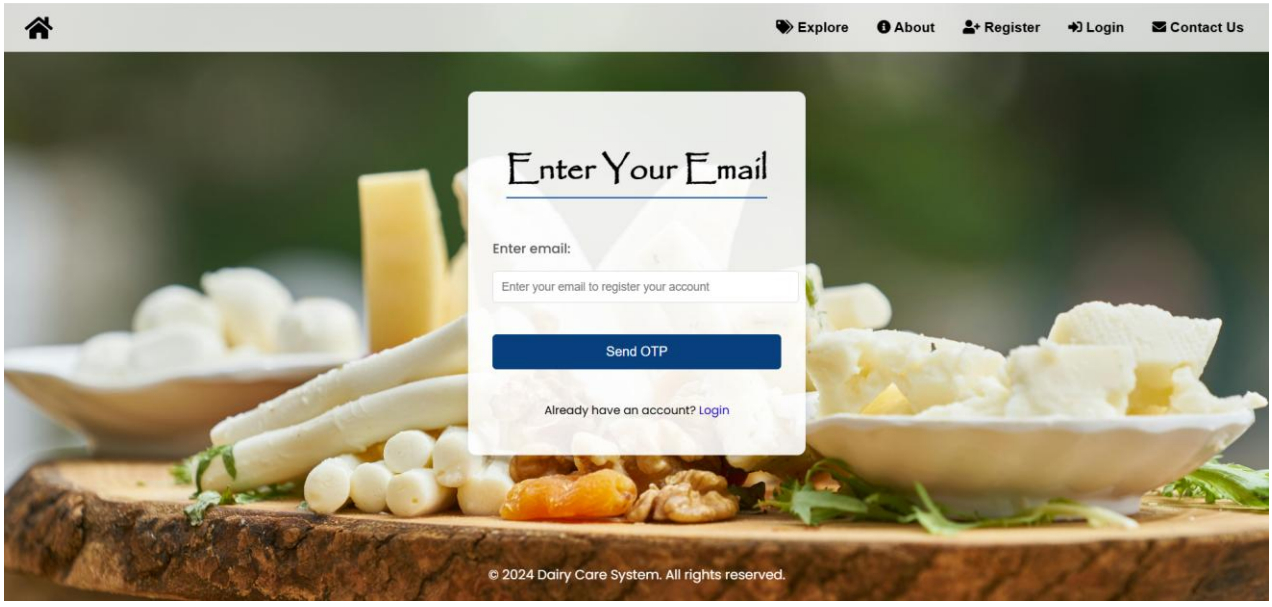


Figure 9.2.2: Registration Page

9.2.3 Login Page

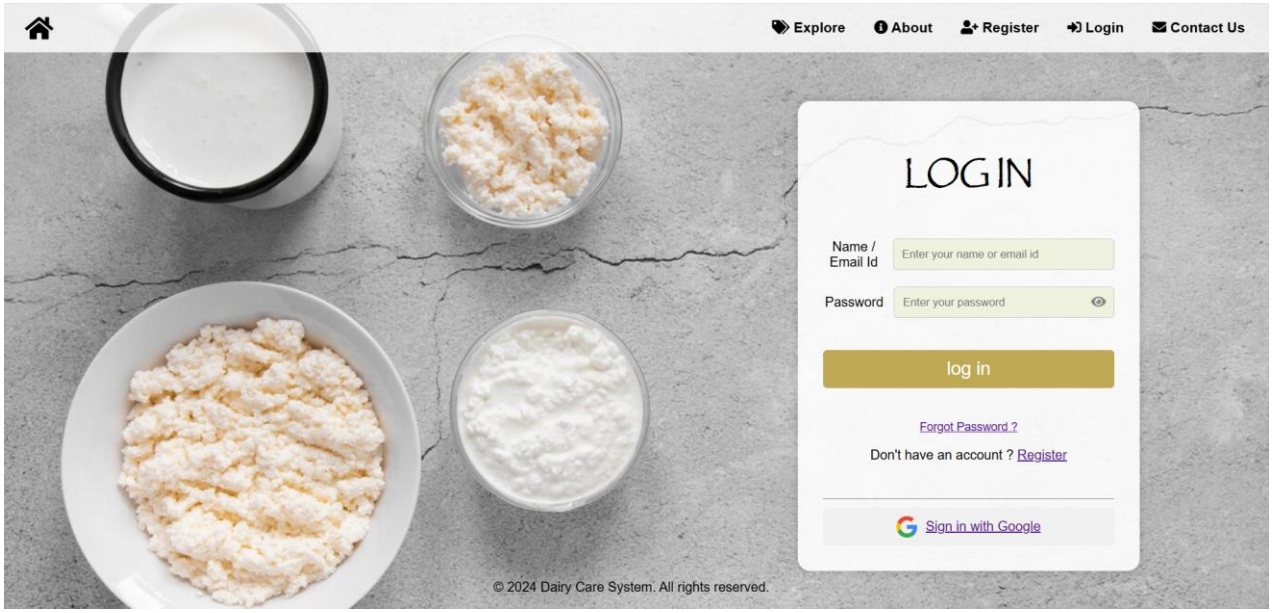


Figure 9.2.3: Login Page

9.2.4 Customer Page

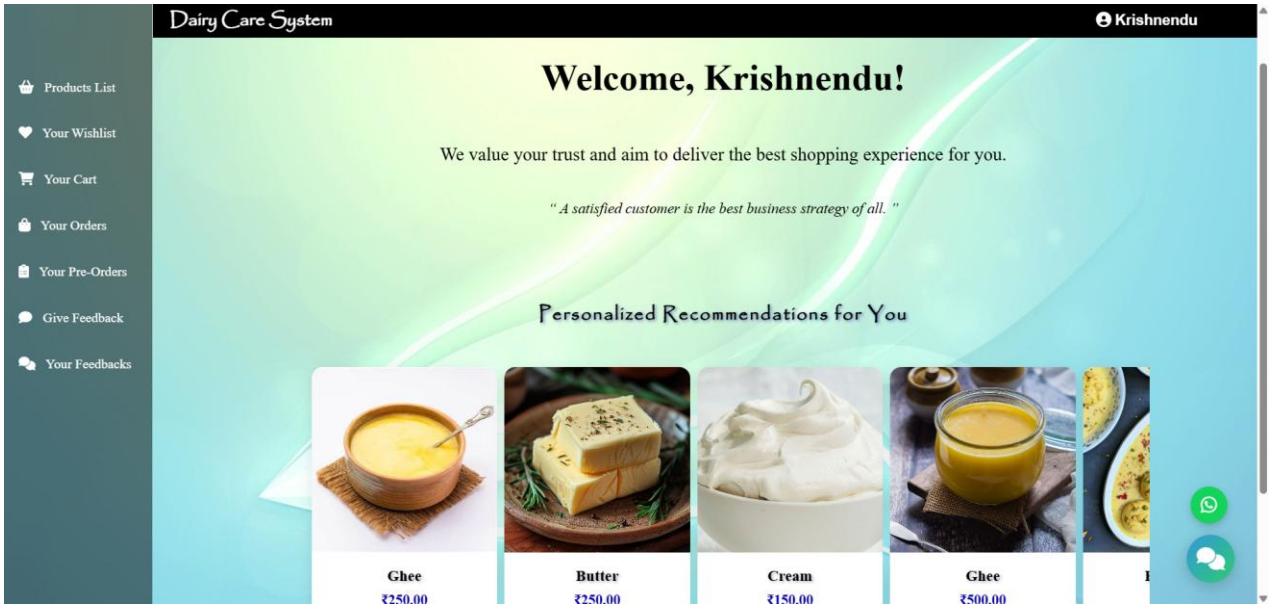


Figure 9.2.4: Customer Page

9.2.5 Products Page

Figure 9.2.5: Products Page

9.2.6 Products Details Page

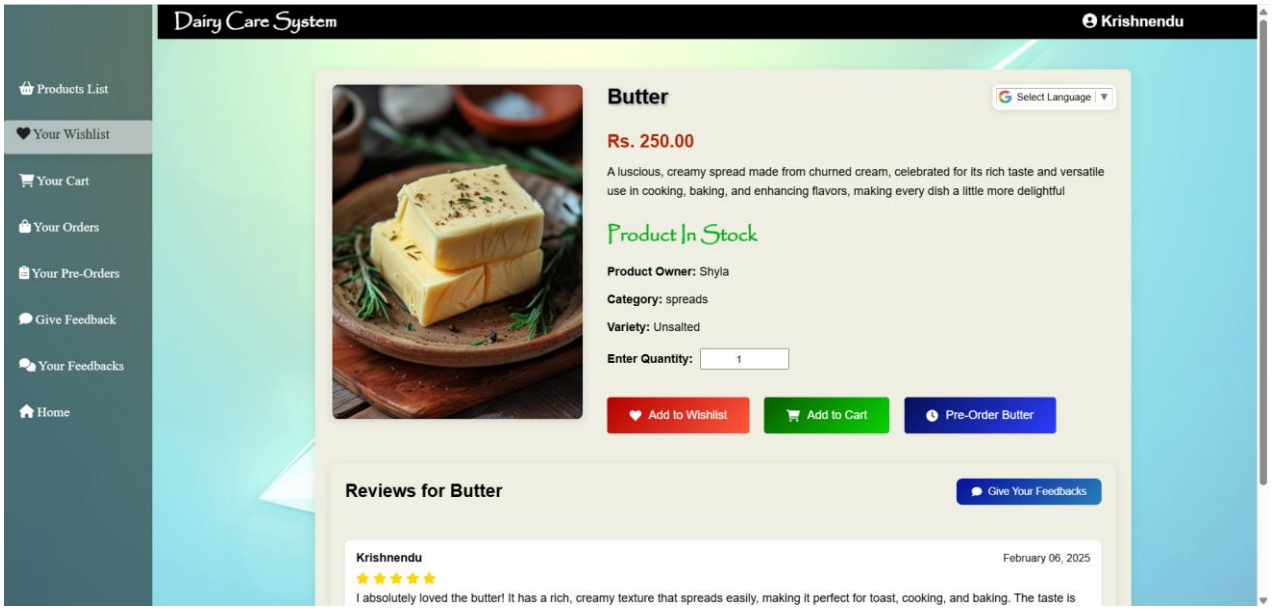


Figure 9.2.6: Products Details Page

9.2.7 Cart Page

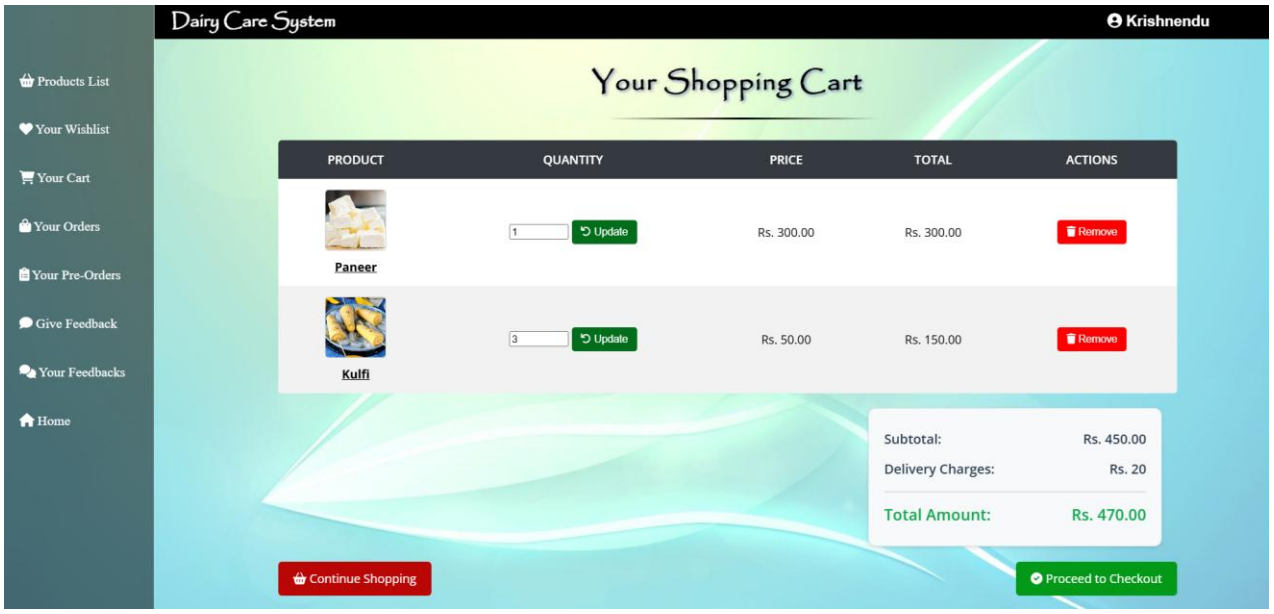


Figure 9.2.7: Cart Page

9.2.8 Form Name: Payment Page

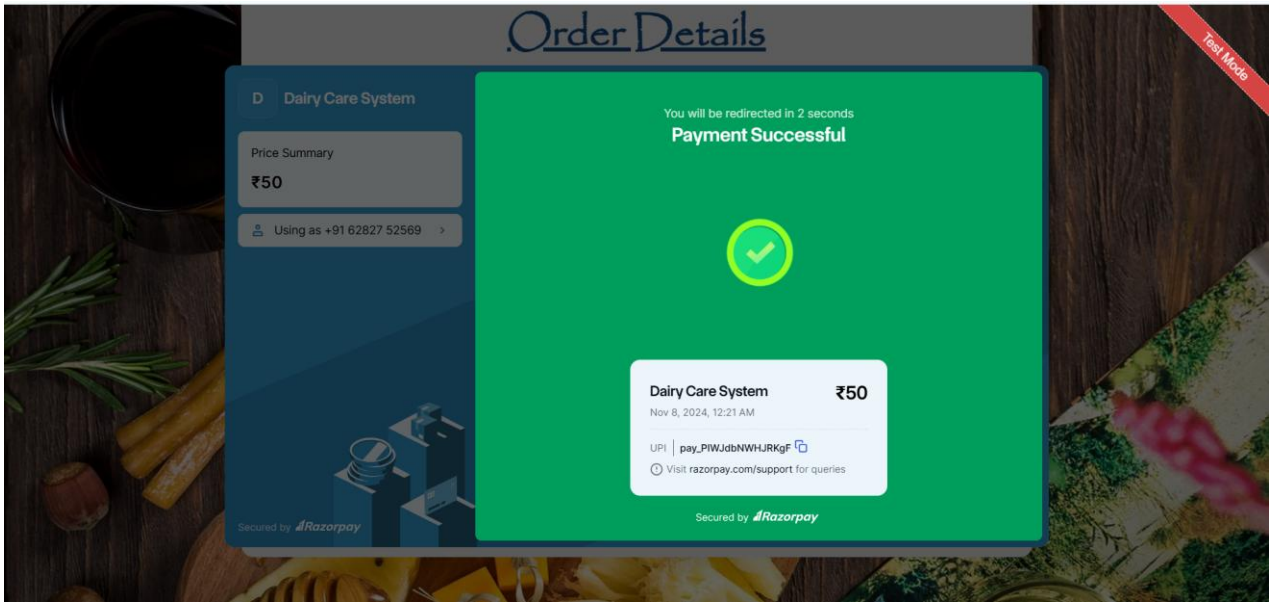


Figure 9.2.8: Payment Page

9.2.9 Form Name: Order Receipt Page

Figure 9.2.9: Order Receipt Page

9.2.10 Form Name: Order Tracking Page

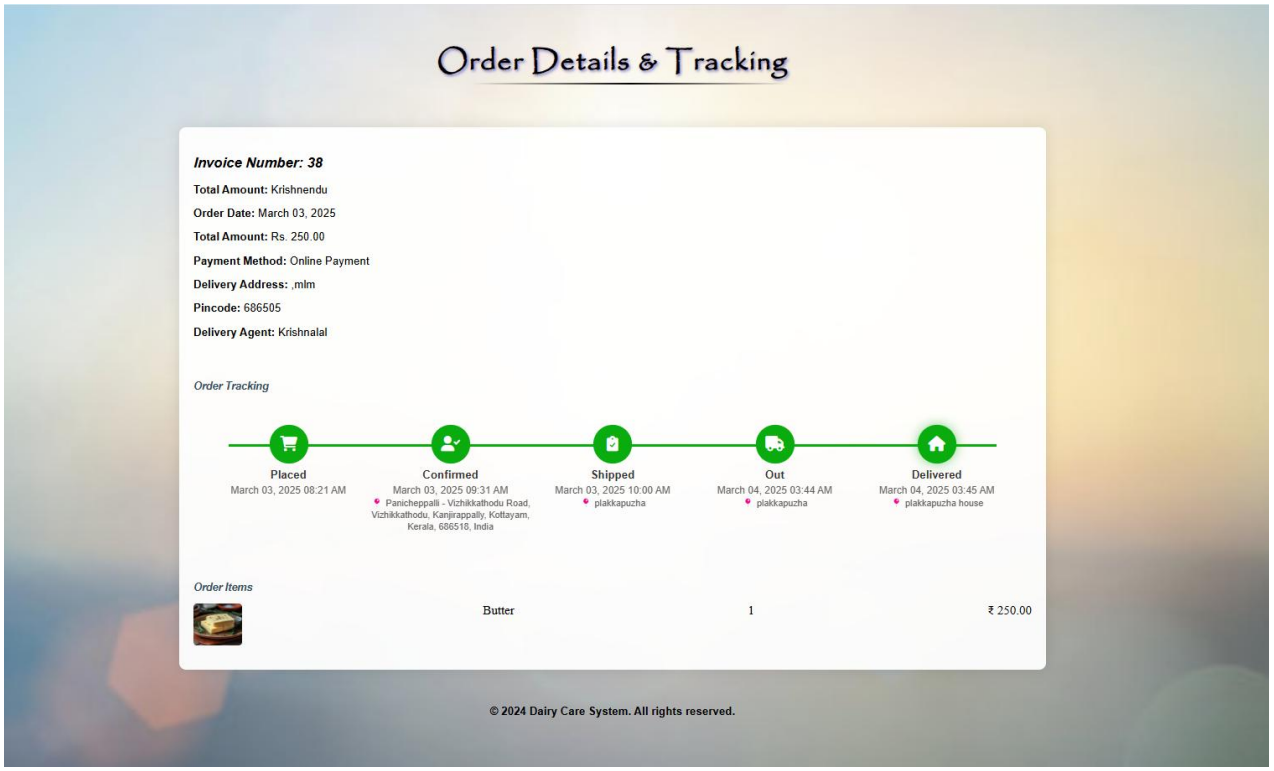


Figure 9.2.10: Order Tracking Page

9.2.11 Form Name: Delivery Agent Page

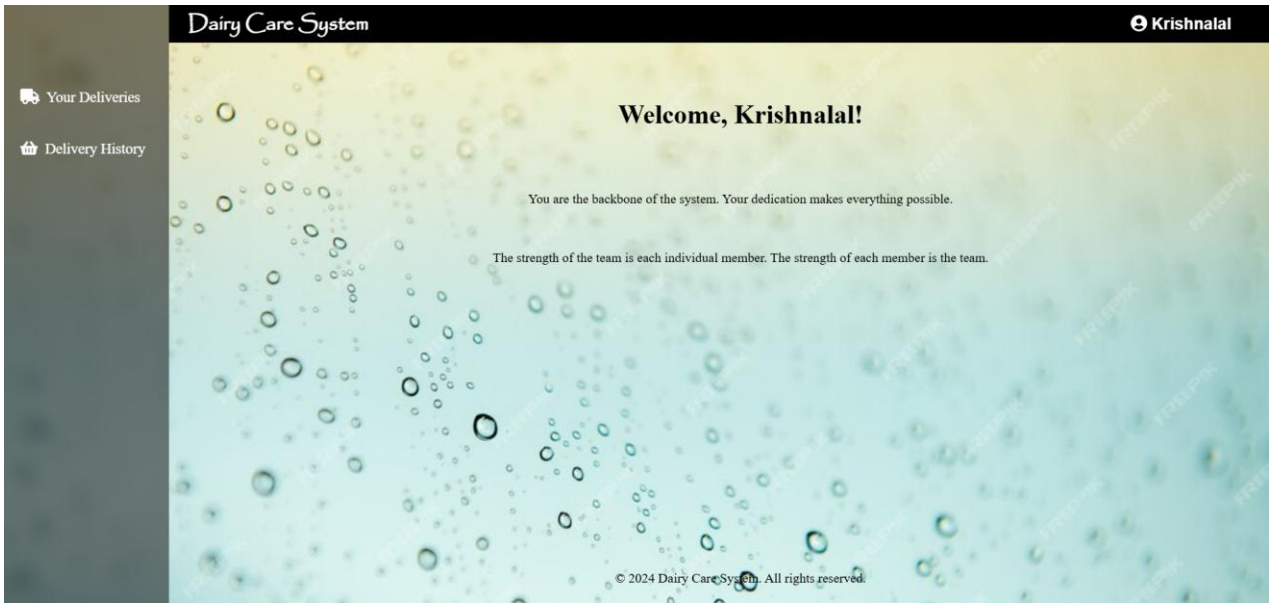


Figure 9.2.11: Delivery Agent Page

9.2.12 Form Name: Active Deliveries Page

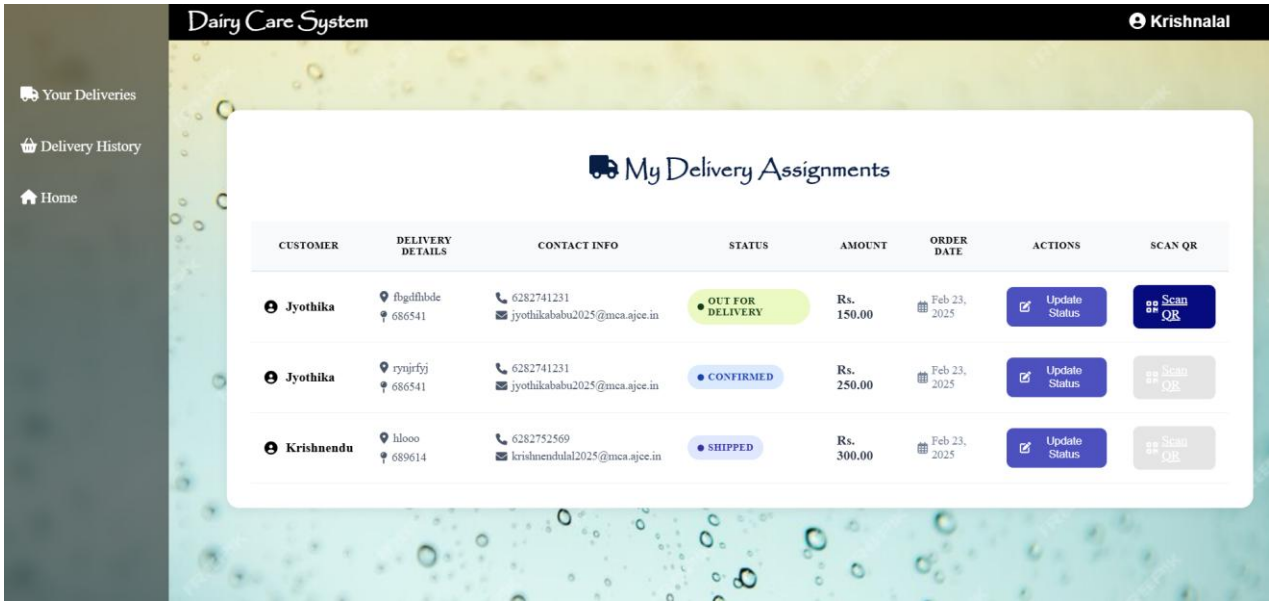


Figure 9.2.12: Active Deliveries Page

9.2.13 Form Name: Farm Owner Page

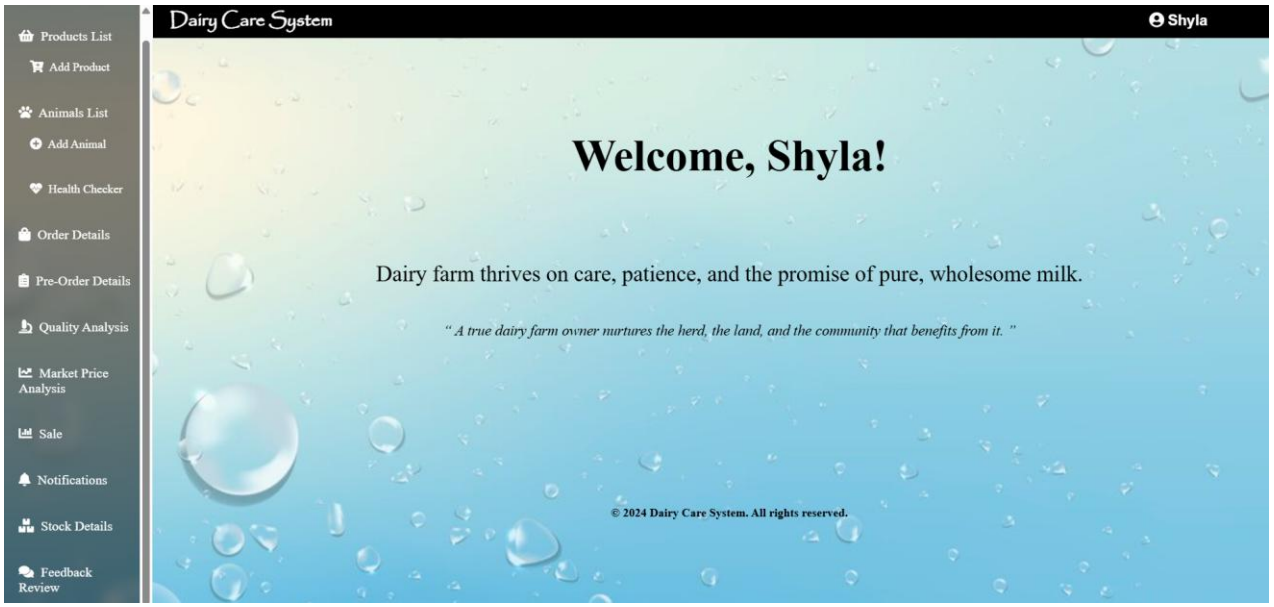


Figure 9.2.13: Farm Owner Page

9.2.14 Form Name: Disease Detection Page

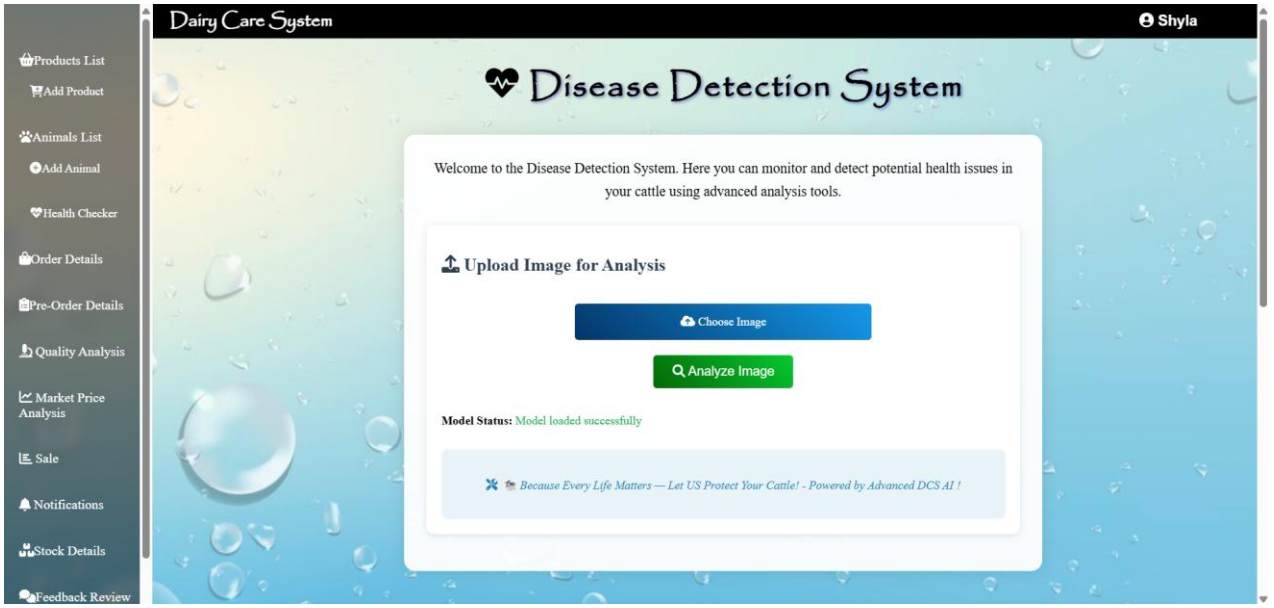
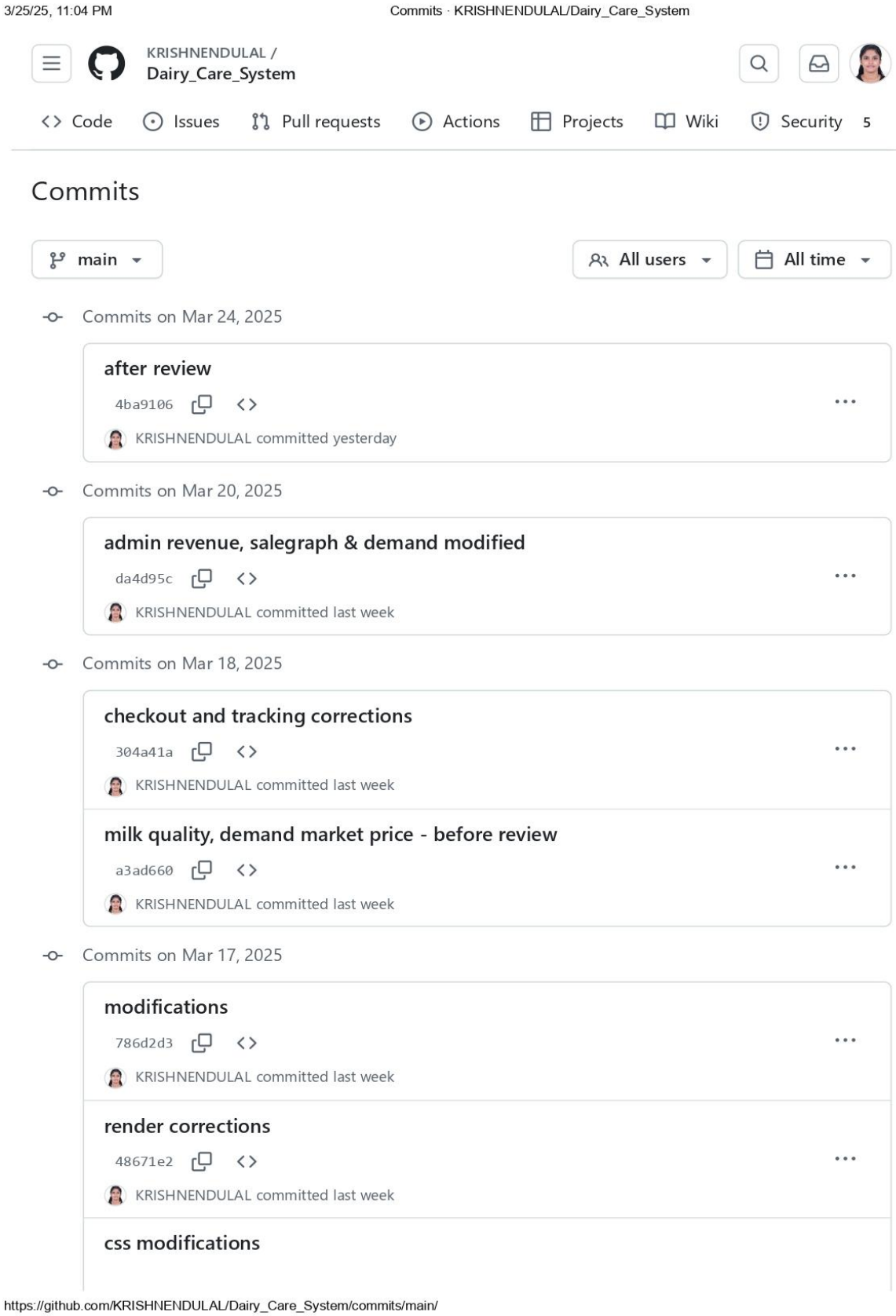


Figure 9.2.14: Disease Detection Page

9.3 Git Log


3/25/25, 11:04 PM
Commits · KRISHNENDULAL/Dairy_Care_System
KRISHNENDULAL /
Dairy_Care_System
Code
Issues
Pull requests
Actions
Projects
Wiki
Security 5
Commits
main
All users
All time
Commits on Mar 24, 2025
after review
4ba9106
KRISHNENDULAL committed yesterday
Commits on Mar 20, 2025
admin revenue, salegraph & demand modified
da4d95c
KRISHNENDULAL committed last week
Commits on Mar 18, 2025
checkout and tracking corrections
304a41a
KRISHNENDULAL committed last week
milk quality, demand market price - before review
a3ad660
KRISHNENDULAL committed last week
Commits on Mar 17, 2025
modifications
786d2d3
KRISHNENDULAL committed last week
render corrections
48671e2
KRISHNENDULAL committed last week
css modifications
https://github.com/KRISHNENDULAL/Dairy_Care_System/commits/main/
1/5

11bd32a

KRISHNENDULAL committed last week

cart modification , pre order modification

6d59ebc

KRISHNENDULAL committed last week

Commits on Mar 14, 2025

disease detection ai analysis , analytics graph

603a3ca

KRISHNENDULAL committed 2 weeks ago

Commits on Mar 11, 2025

delivery module complete (status disabling, qr scanning), order place mail confirmation

a6a68c5

KRISHNENDULAL committed 2 weeks ago

Commits on Mar 10, 2025

milkqualityanalysis, ml image preview, multilingual

a240d43

KRISHNENDULAL committed 2 weeks ago

Commits on Mar 5, 2025

id encryption & disease detection

99a2a0d

KRISHNENDULAL committed 3 weeks ago

Commits on Feb 24, 2025

deliveryagent auto assign using pincode & qr scanning confirmation

eae85cb

KRISHNENDULAL committed last month

Commits on Feb 21, 2025

deliveryagent qr scanning

c4278e5

KRISHNENDULAL committed on Feb 21

Commits on Feb 19, 2025

training checkpoint
8c04393
KRISHNENDULAL committed on Feb 19

dataset trained notebook
988d9ae
KRISHNENDULAL committed on Feb 19

trained modal
b4ab966
KRISHNENDULAL committed on Feb 19

recommand & image search
14868e0
KRISHNENDULAL committed on Feb 19

Commits on Feb 14, 2025

sample recommendation
48aa51a
KRISHNENDULAL committed on Feb 14

Dataset Training
af93ff7
KRISHNENDULAL committed on Feb 14

Commits on Feb 13, 2025

deliveryboy automatic assign
a73b98c
KRISHNENDULAL committed on Feb 13

Commits on Feb 12, 2025

jupyter installation
1adcde4
KRISHNENDULAL committed on Feb 12

Commits on Feb 11, 2025

explore and image search half

57a93e0

KRISHNENDULAL committed on Feb 11

Commits on Feb 10, 2025

env activated

b59c54e

KRISHNENDULAL committed on Feb 10

subcategory & QR

123f09e

KRISHNENDULAL committed on Feb 10

Commits on Feb 6, 2025

template & senti pips

b66da31

KRISHNENDULAL committed on Feb 6

Commits on Feb 4, 2025

Documents Upload

8d09c4d

KRISHNENDULAL committed on Feb 4

Commits on Feb 2, 2025

Disease Detection Starting

2cca560

KRISHNENDULAL committed on Feb 2

delivery agent complete

00d0f88

KRISHNENDULAL committed on Feb 2

Commits on Jan 27, 2025

delivery agent starting

0e1e44e

KRISHNENDULAL committed on Jan 27

Commits on Jan 16, 2025

chatbot and voice sample

2074492

KRISHNENDULAL committed on Jan 16

Commits on Dec 31, 2024

template modifications

86d1de7

KRISHNENDULAL committed on Dec 31, 2024

Commits on Dec 19, 2024

Main Project beginning

88bd4a6

KRISHNENDULAL committed on Dec 19, 2024

Commits on Nov 12, 2024

Final Submission of Mini Project

86f3579

KRISHNENDULAL committed on Nov 12, 2024

Commits on Nov 8, 2024

commit 41 (contactus)

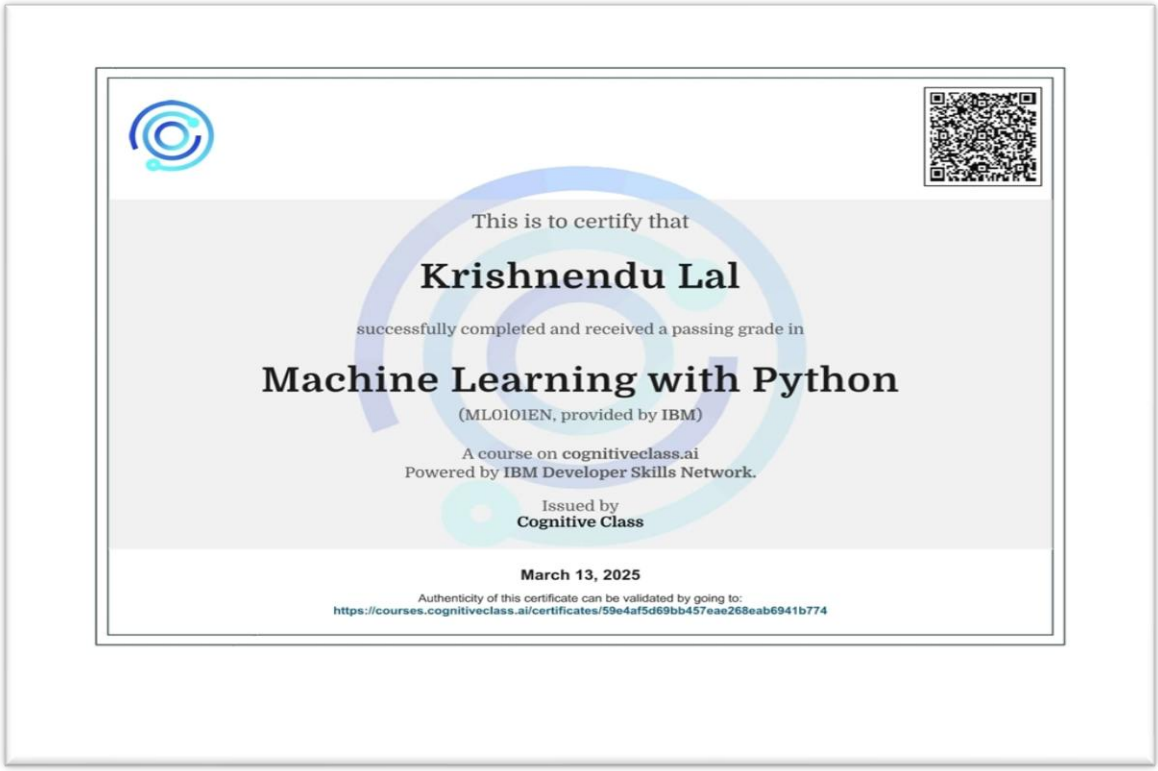
3bb14bf

KRISHNENDULAL committed on Nov 8, 2024

Previous Next

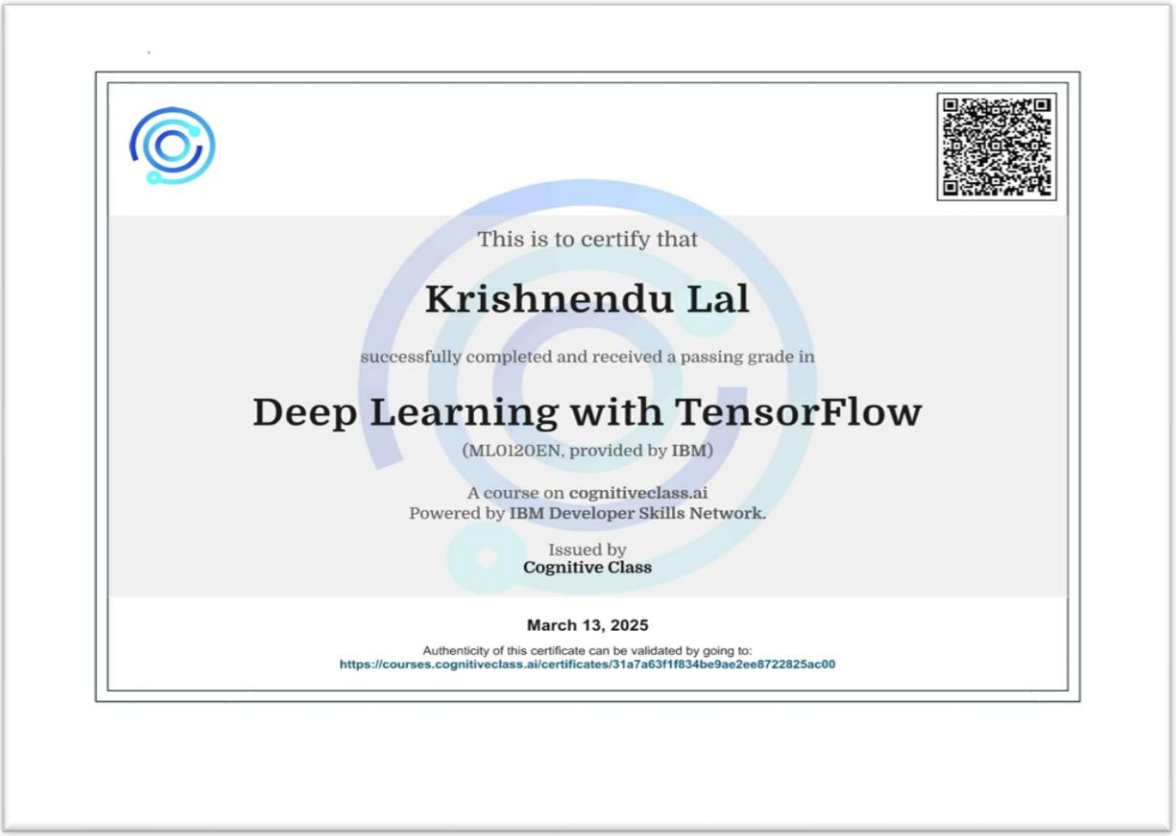
9.4 Certificates

9.4.1 Machine Learning with Python



9.4.2 Introduction to TensorFlow using Keras

9.4.3 Deep Learning with TensorFlow



9.4.4 Fundamentals of Machine Learning

9.4.5 Fundamentals of Generative AI

9.5 Plagiarism Report

Krishna_Project6

ORIGINALITY REPORT

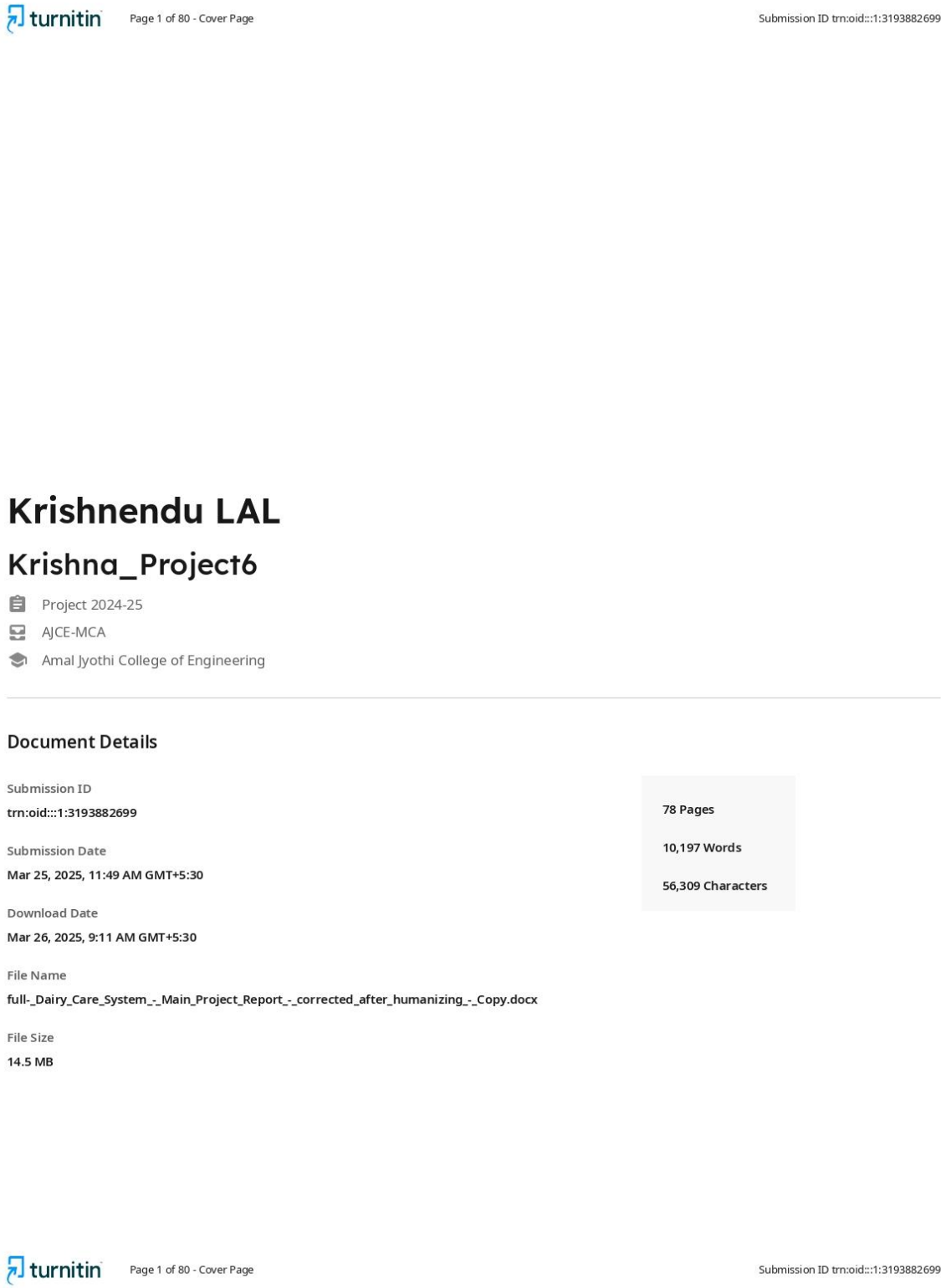
25%	12%	6%	24%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Amal Jyothi College of Engineering Student Paper	23%
2	www.coursehero.com Internet Source	<1%
3	idoc.tips Internet Source	<1%
4	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1%
5	Submitted to St. Ignatius High School Student Paper	<1%
6	Submitted to Segi University College Student Paper	<1%
7	Shailesh Kumar Shivakumar. "Chapter 5 Models, Tools, and Templates Used in Digital Project Management", Springer Science and Business Media LLC, 2018 Publication	<1%
8	Submitted to Te Pūkenga trading as the Open Polytechnic Student Paper	<1%
9	ajce.irins.org Internet Source	<1%
10	dspace.chitkara.edu.in Internet Source	<1%


11
fdocuments.us
Internet Source
<1%
12
www.ppc.biba.uni-bremen.de
Internet Source
<1%
13
Lee Chao. "Database Development and Management", Auerbach Publications, 2019
Publication
<1%
14
Paul C. Jorgensen. "Modeling Software Behavior - A Craftsman's Approach", Auerbach Publications, 2019
Publication
<1%
Exclude quotes Off
Exclude matches Off
Exclude bibliography Off

9.6 Plagiarism AI Report



turnitin Page 1 of 80 - Cover Page Submission ID trn:oid:::1:3193882699

Krishnendu LAL

Krishna_Project6

Project 2024-25

AJCE-MCA

Amal Jyothi College of Engineering

Document Details

Submission ID
trn:oid:::1:3193882699

Submission Date
Mar 25, 2025, 11:49 AM GMT+5:30

Download Date
Mar 26, 2025, 9:11 AM GMT+5:30

File Name
full-_Dairy_Care_System_-_Main_Project_Report_-_corrected_after_humanizing_-_Copy.docx

File Size
14.5 MB

78 Pages

10,197 Words

56,309 Characters

turnitin Page 1 of 80 - Cover Page Submission ID trn:oid:::1:3193882699

turnitin Page 2 of 80 - AI Writing Overview Submission ID trn:oid:::1:3193882699

0% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups

1 AI-generated only 0%
Likely AI-generated text from a large-language model.

2 AI-generated text that was AI-paraphrased 0%
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Disclaimer
Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (it may misidentify writing that is likely AI generated as AI generated and AI paraphrased or likely AI generated and AI paraphrased writing as only AI generated) so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

turnitin Page 2 of 80 - AI Writing Overview Submission ID trn:oid:::1:3193882699