Sri Lanka Institute of Advanced Technology Education (SLIATE)

Higher National Diploma in Information Technology

HNDIT 4052

Programming Individual Project

Project Title

DairyPro Management System

Supervisor

Mr.K.W.G.S.Konthasinghe

ACADEMIC YEAR-: 2022

YEAR- II, SEMESTER II

H.M.H.P.Disanayake

KUR/IT/2022/F/0074

Table of Contents

01.Executive Summary

1.1 Brief overview of the project

The DairyPro Management System is a web-based application designed to streamline and automate the operations of a dairy shop. It both dairy manager and customers, providing distinct features and interfaces for each user type. The system focuses on simplifying product management, order handling, and cash on delivery tracking and online payment while offering an intuitive user experience and robust data security through role-based access control.

1.2 Key objectives and outcomes

- Enable secure admin and user authentication.
- Allow admins to manage dairy product management and inventory, process orders, and analyze sales through real-time dashboards.
- Provide customers with seamless browsing, ordering, and delivery tracking experiences.
- Improve decision-making through data analytics and reporting.
- Increase operational efficiency, reduce manual work, time consuming and prepare the system for scalability.

1.3 Summary of major findings and results

The system introduces features such as product CRUD operations, order lifecycle management (from placement to delivery), cart functionality, and integrated sales analytics. It supports role-based access and offers a user-friendly interface backed by the MERN stack (MongoDB, Express.js, React.js, Node.js). Non-functional attributes like performance, security, availability, and usability are also addressed to ensure a reliable and efficient solution. The expected outcome is a scalable, secure, and time-saving management system that benefits both dairy business manager and their customers

02.Introduction

2.1 Background information

Dairy businesses traditionally rely on manual methods for managing daily operations like milk production tracking, product management, order processing, and sales analysis. These methods are prone to errors, inefficiencies, and time-consuming processes. With the advancement of technology, there's a growing need for digital solutions that streamline these activities, provide real-time insights, and support business scalability.

2.2 Purpose of the project

The purpose of the DairyPro Management System is to define and document the requirements of a software solution that enhances the operational efficiency of a dairy shop. This system is intended to serve as a guideline for developers, stakeholders, and users, enabling clear understanding of the system's objectives, functionality, and constraints.

2.3 Scope of the project

The project involves developing a web-based application that supports two main user roles: Admin and Customer. Admins can manage dairy products, monitor orders, analyze sales, and see to customer contact message, while Customers can browse products, place orders, and track deliveries. The system includes features such as authentication, role-based access, real-time analytics, and user-friendly interfaces. It leverages the MERN stack for development and MongoDB for data management.

2.4 Project objectives

- Automate dairy shop operations including product, order, sales, analytics management.
- Provide role-based dashboards for secure and efficient system usage.
- Enable real-time sales and inventory analytics for better decision-making.
- Enhance customer experience through easy product browsing and order tracking.
- Ensure system scalability and data security for future growth.

03.Requirements Analysis

3.1 Functional requirements

The DairyPro Management System supports the following key functional requirements:

- **User Authentication:**
  Users (customers) can register and log in securely with encrypted credentials.
- **Admin Authentication:**
  Admins have secure login access to manage the platform via the Admin Dashboard.
- **Product Browsing (Customer):**
  Customers can browse products by categories (e.g., Milk, Cheese, Yogurt), view descriptions, price, and unit details.
- **Product Management (Admin):**
  Admins can perform CRUD (Create, Read, Update, Delete) operations on dairy products.
- **Cart and Order Management (Customer):**
  Customers can add products to their cart, update quantity or units, and place orders via Cash on Delivery (COD).
- **Order Management (Admin):**
  Admins can view and update order statuses (e.g., Pending → Shipped → Delivered) and approve/reject COD orders.
- **View Messages (Admin):**
  Admins can view customer contact messages submitted through the system.
- **Sales Analytics (Admin):**
  Admins can access real-time data visualizations and reports on orders, revenue, and top-selling products.

- **Role-Based Access Control:**

  Differentiated access for Admins and Customers to ensure secure and appropriate data handling.

3.2 Non-functional requirements

- **Performance:**

  The system ensures fast data access, quick report generation, and smooth navigation.

- **Security:**

  Implements authentication and role-based authorization to protect sensitive data.

- **Availability:**

  Cloud deployment ensures high uptime and continuous access.

- **Usability:**

  A user-friendly interface with minimal learning curve designed for efficiency.

3.3Use case diagrams

- Below page there Dairypro management system usecase diagram

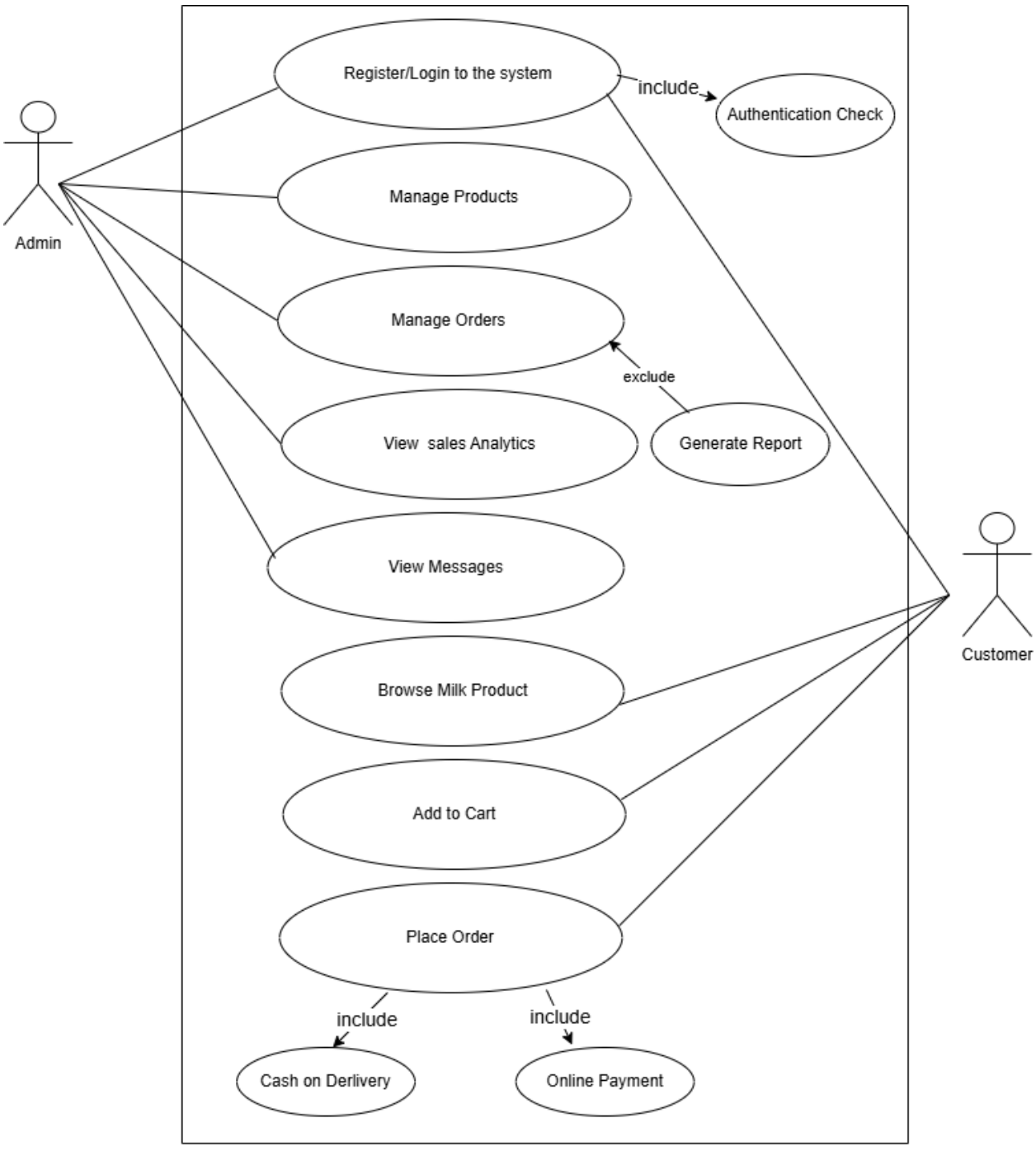

Figure 01 -: Usecase Diagram

3.4 Requirements specification

The requirements are clearly defined in the SRS under both functional and non-functional categories, as well as described through system behavior such as:

- System shall allow users to register and log in securely.
- System shall allow admins to manage inventory through CRUD operations.
- System shall provide role-based dashboards for Admins and Customers.
- System shall generate analytics on sales, orders, and customer behavior.
- System shall support order tracking and confirmation for customers.

04. System Design

4.1 Architectural Design

The DairyPro Management System is built using the MERN stack (MongoDB, Express.js, React.js, Node.js), following a client-server architecture:

- Frontend (Client):

Developed using React.js, providing a web-based dashboard with role-based interfaces (Admin and Customer).

- Backend (Server):

Built with Node.js and Express.js, handling business logic, routing, authentication, and API endpoints.

- Database:

MongoDB is used to store data related to users, products, orders, and messages.

- Communication:

The frontend and backend communicate via Postman APIs, enabling CRUD operations and data synchronization.

4.2 Detailed design

4.2.1Database Design

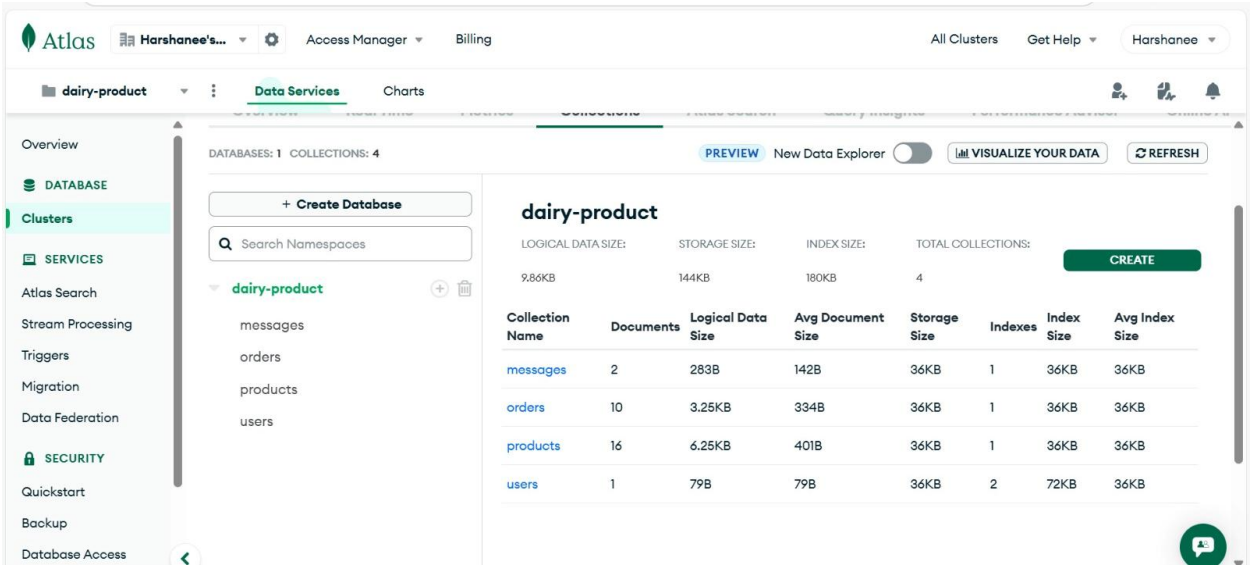

Figure 02 -: Overall Database

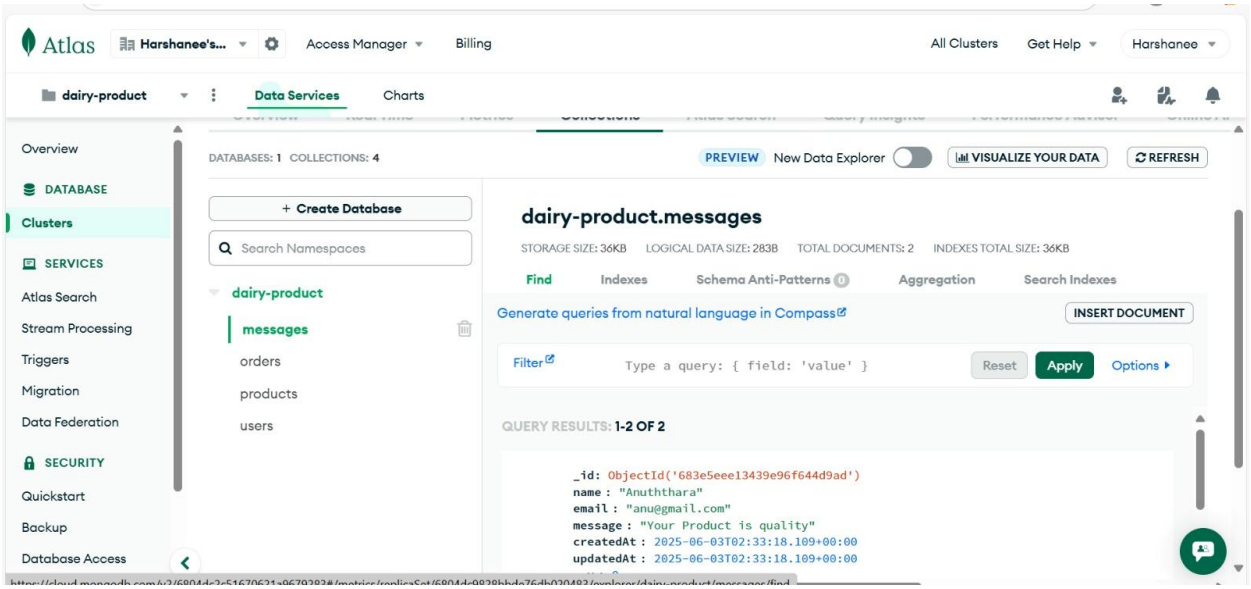

Figure 03 -: Message Query

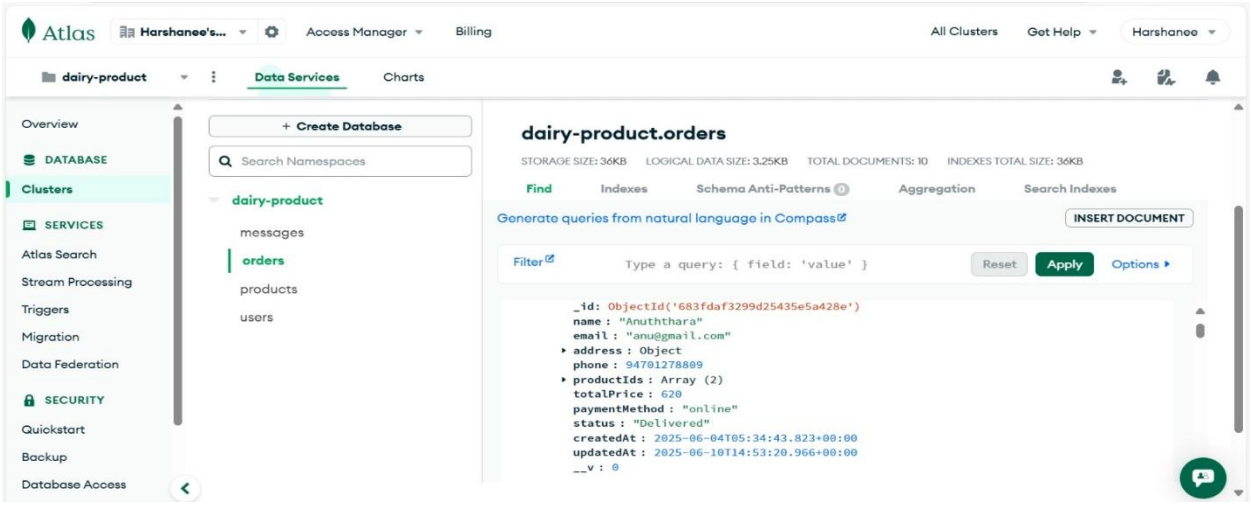

Figure 04 - : Orders Query

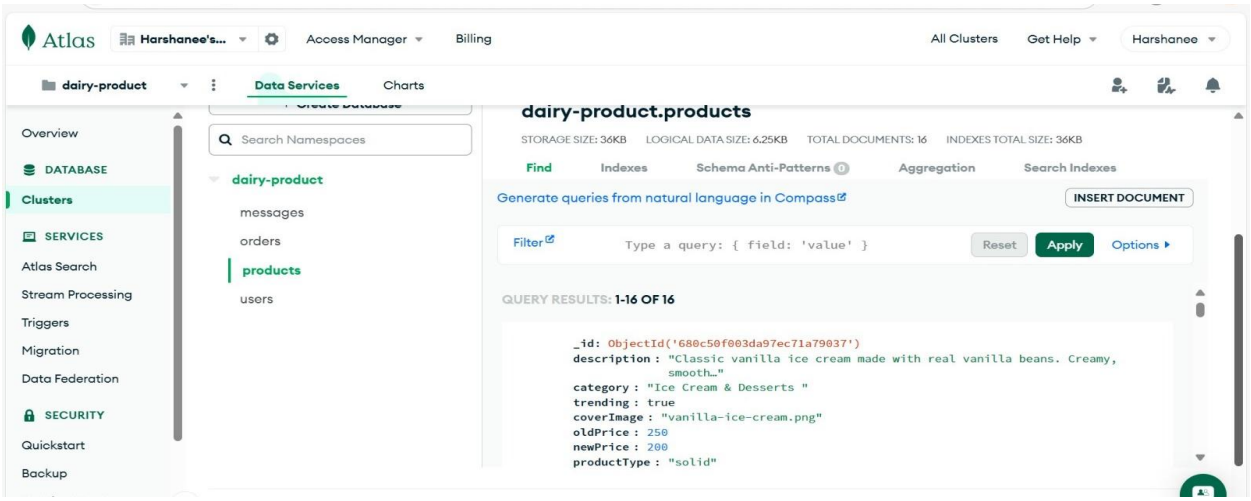

Figure 05 -: Products Query

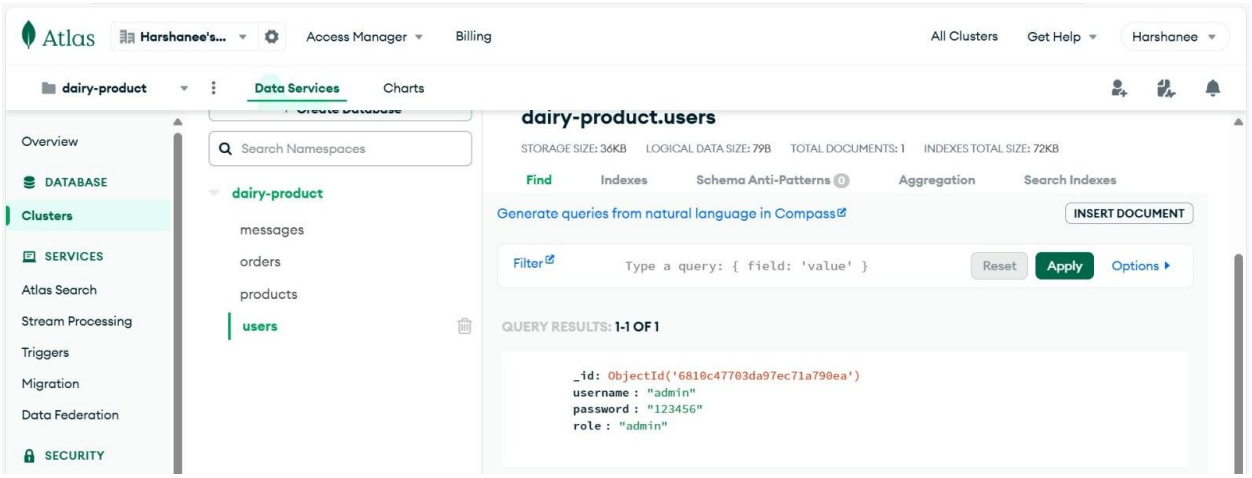

Figure 06 -: Users Query

4.2.2User Interface Design

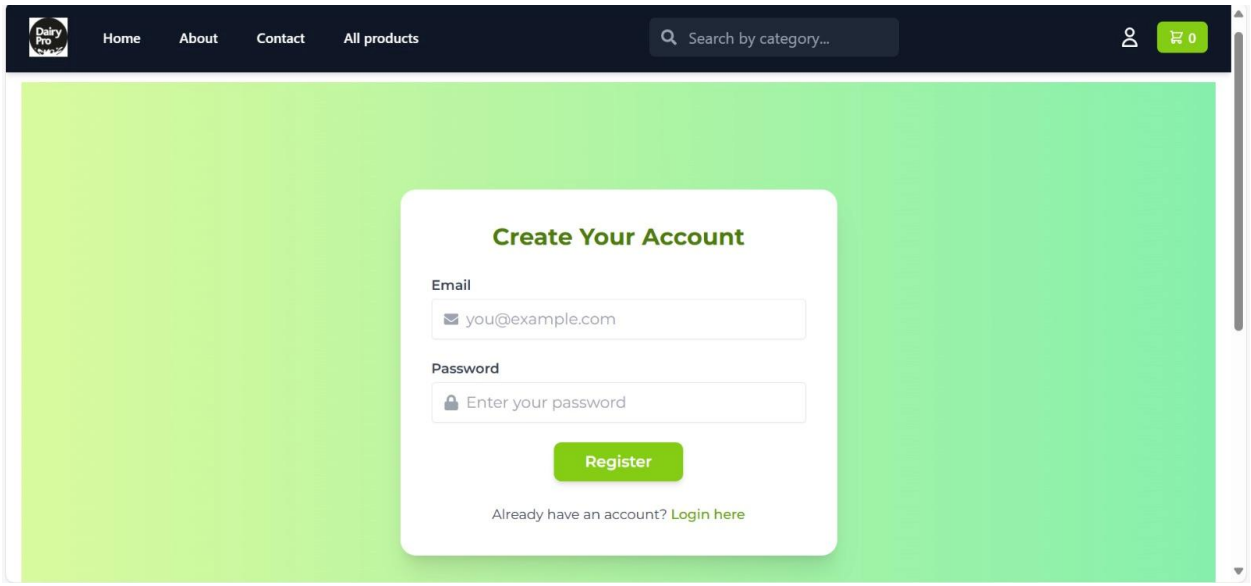

Figure 07 -: Customer Register page

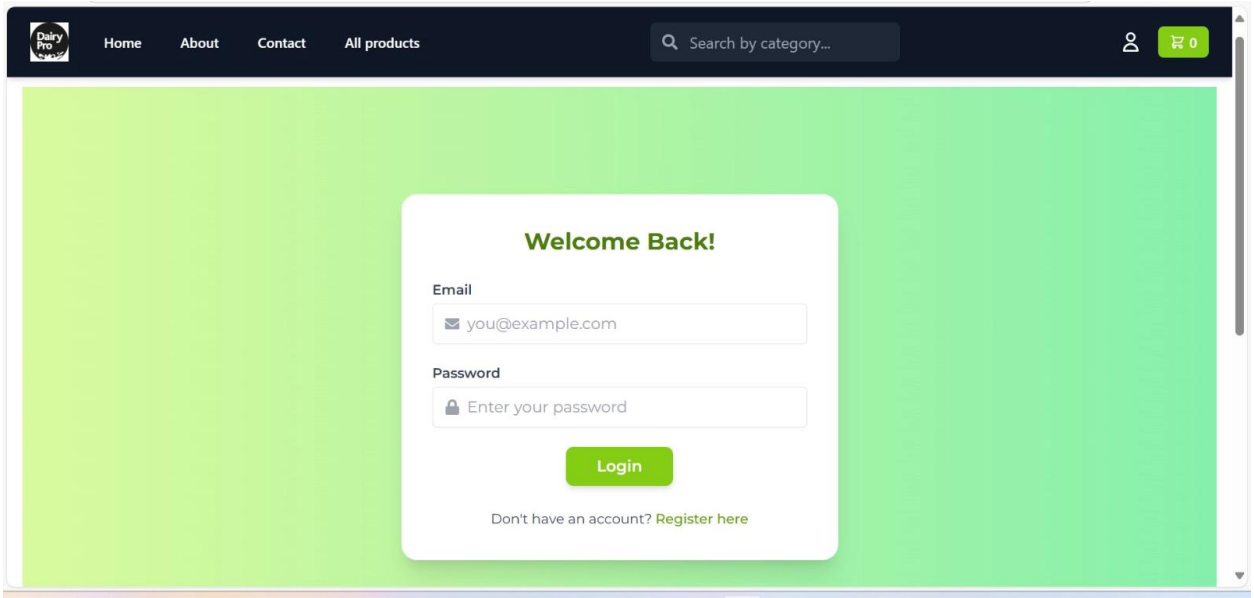

Figure 08 -: Customer Login page

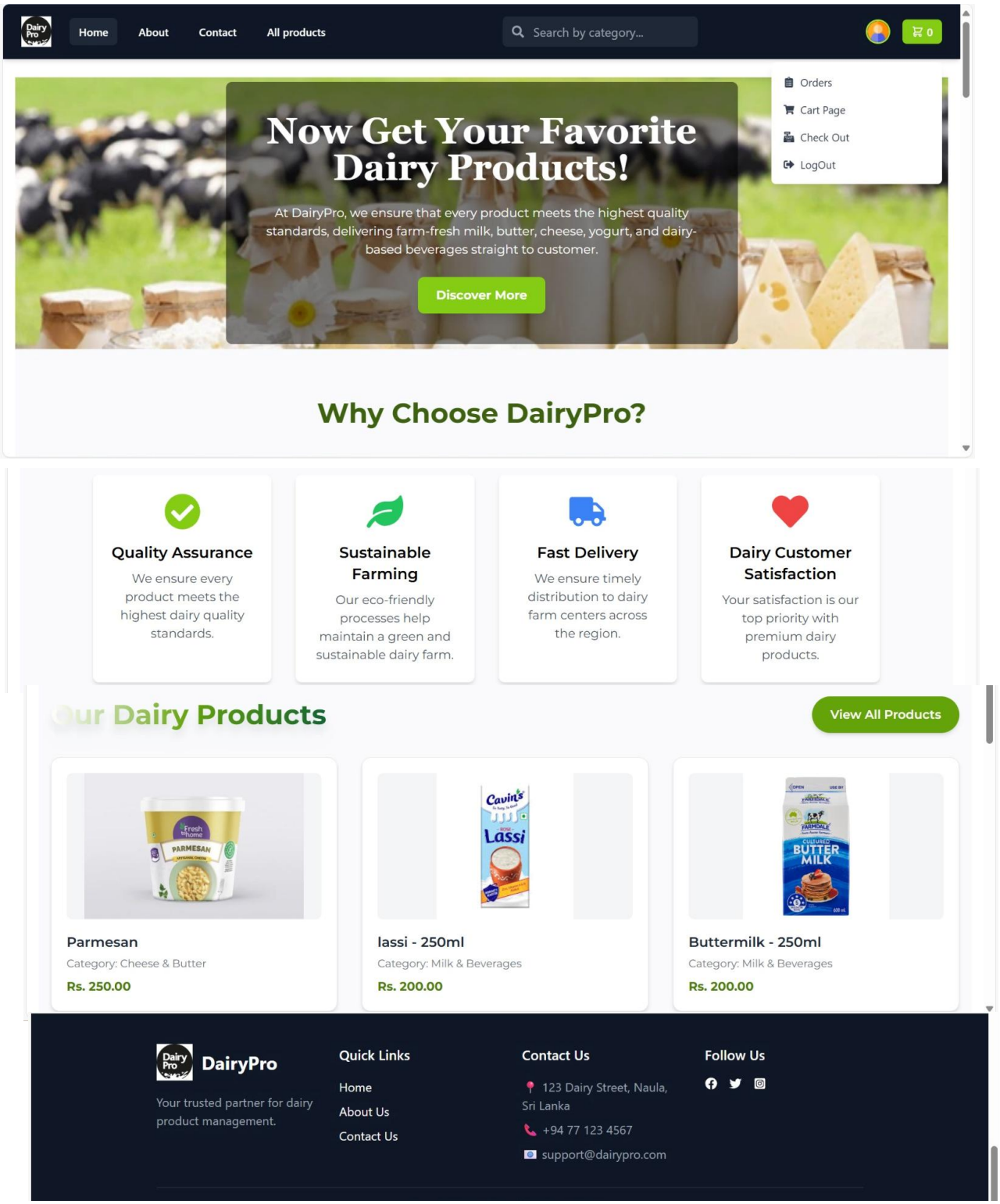

Figure 09 -: Home Page

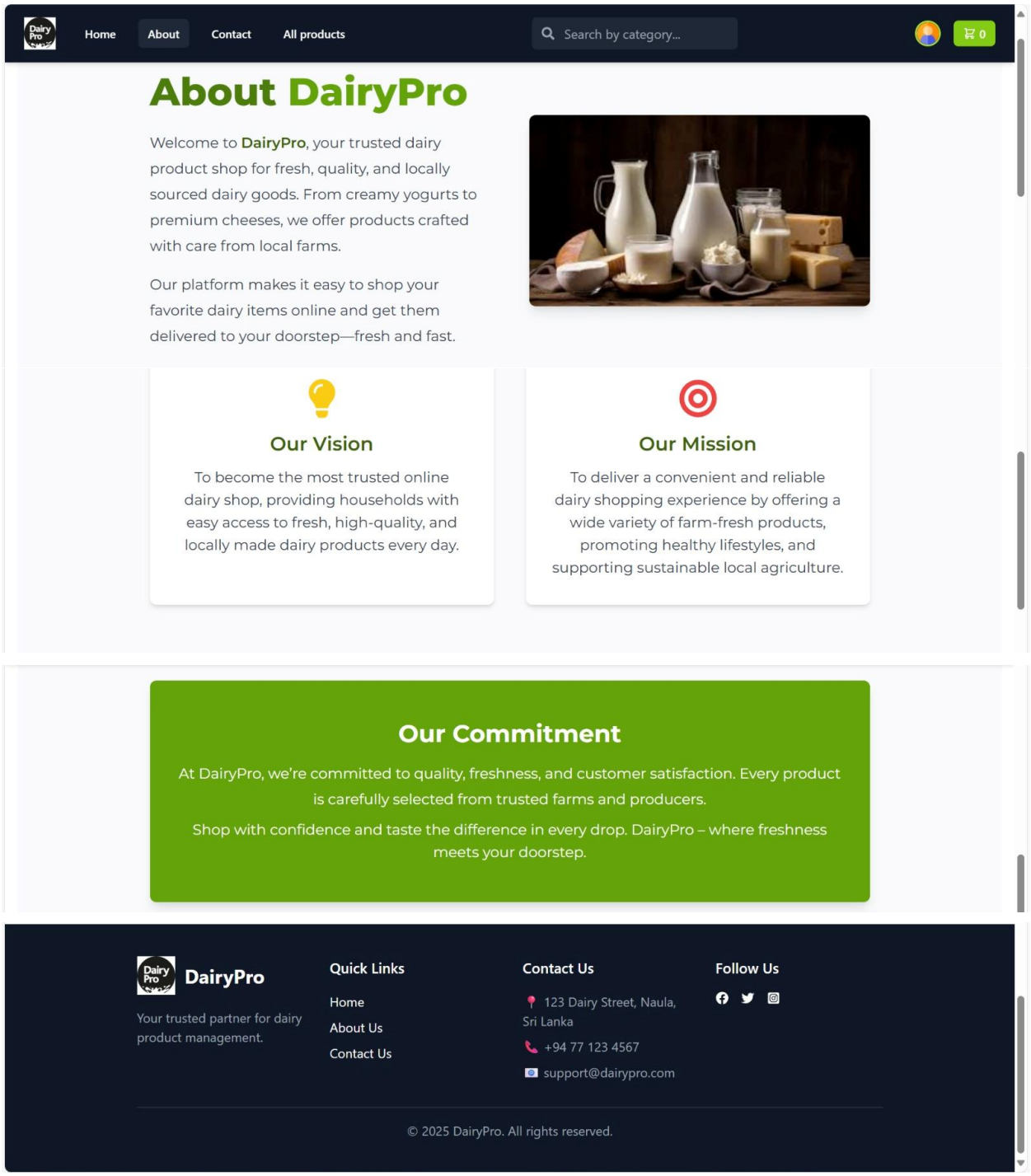

Figure 10-: About Page

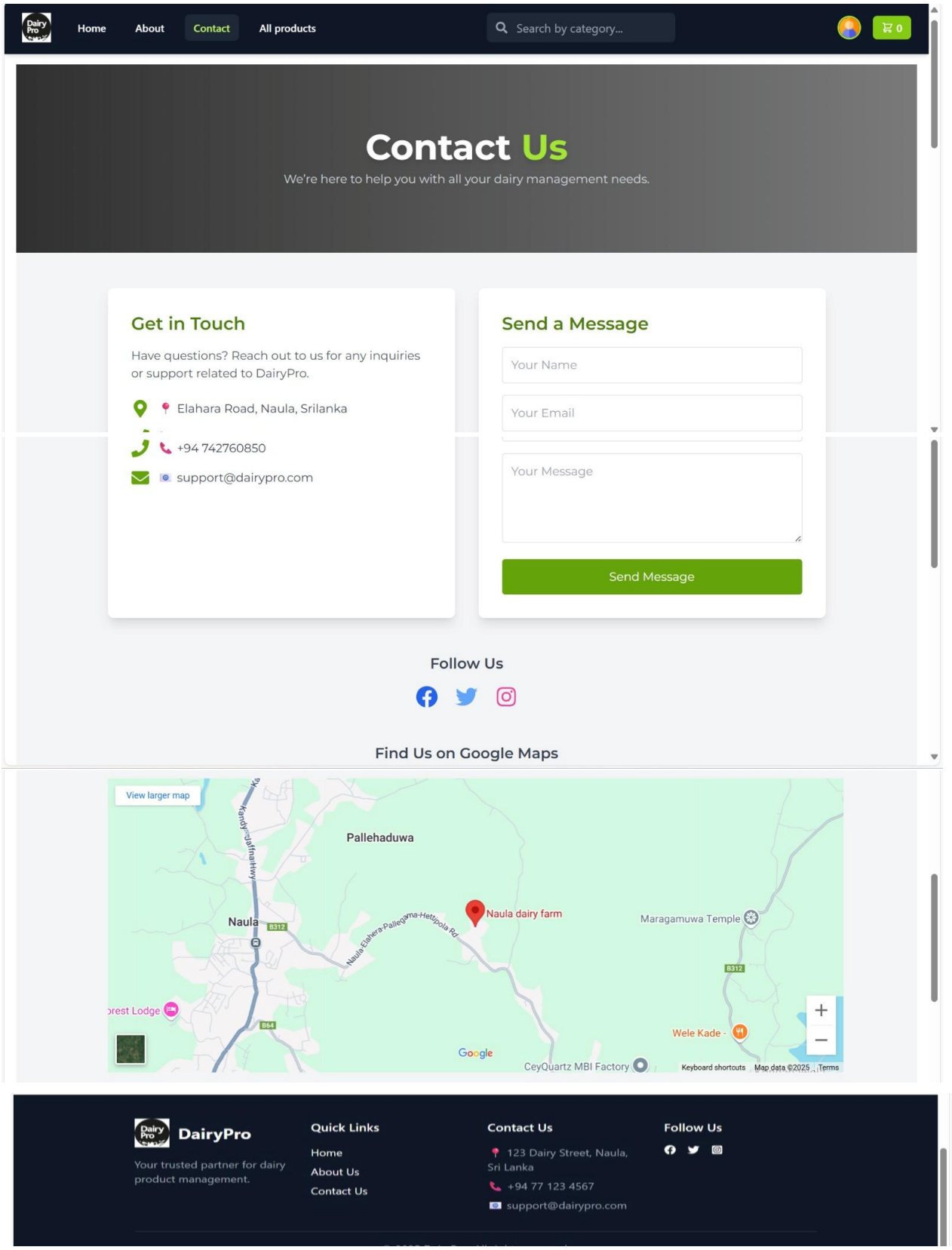

Figure 11 -: ContactUs Page

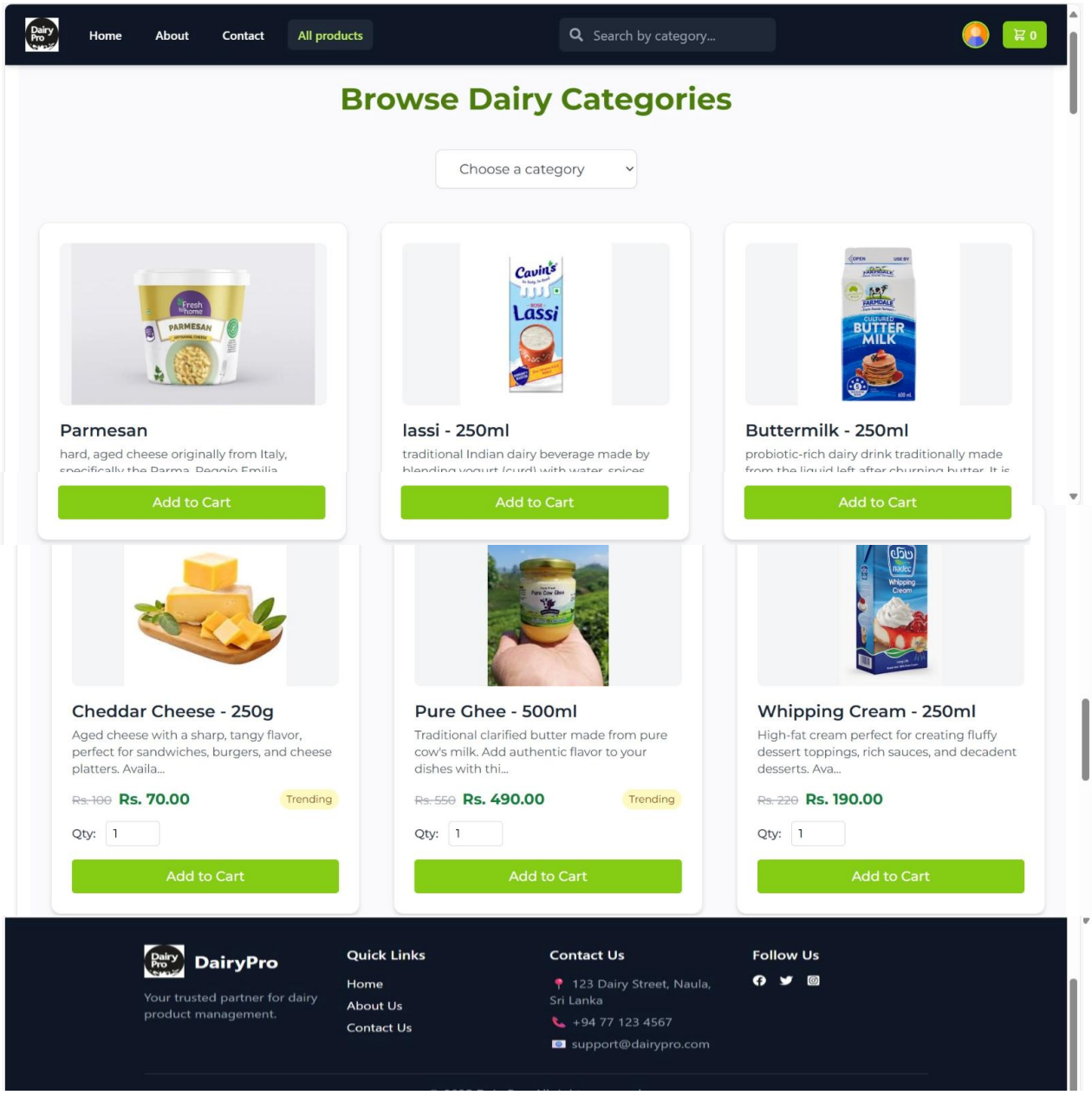

Figure 12 -: All Products Page(Browse categories of products)

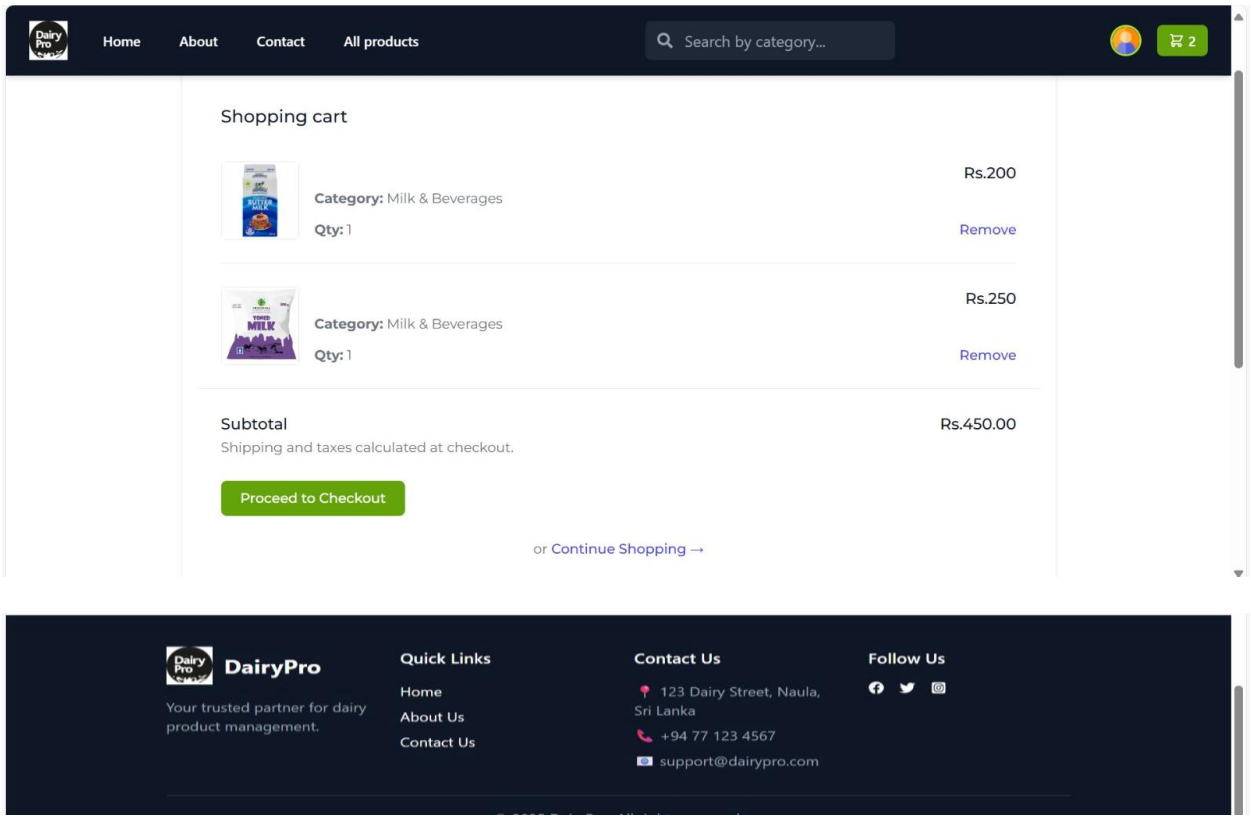

Figure 13 -: Cart Page

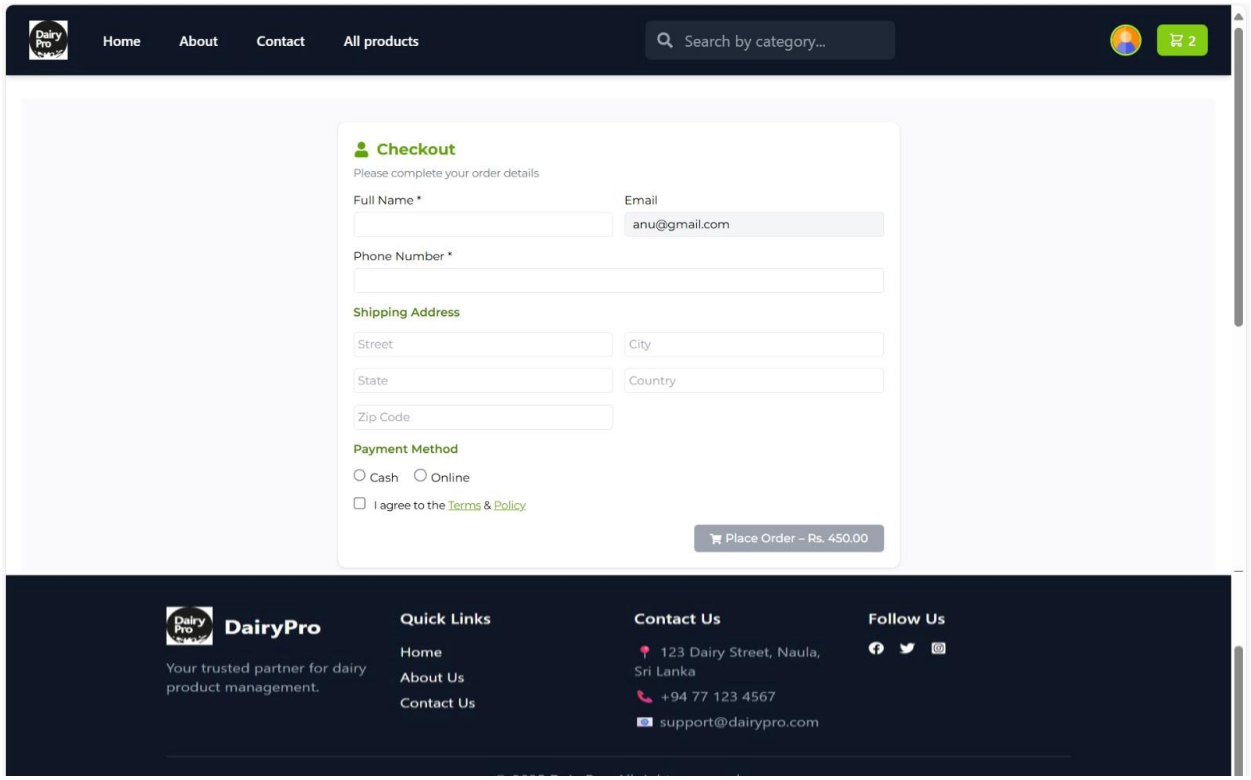

Figure 14 -: Checkout Page

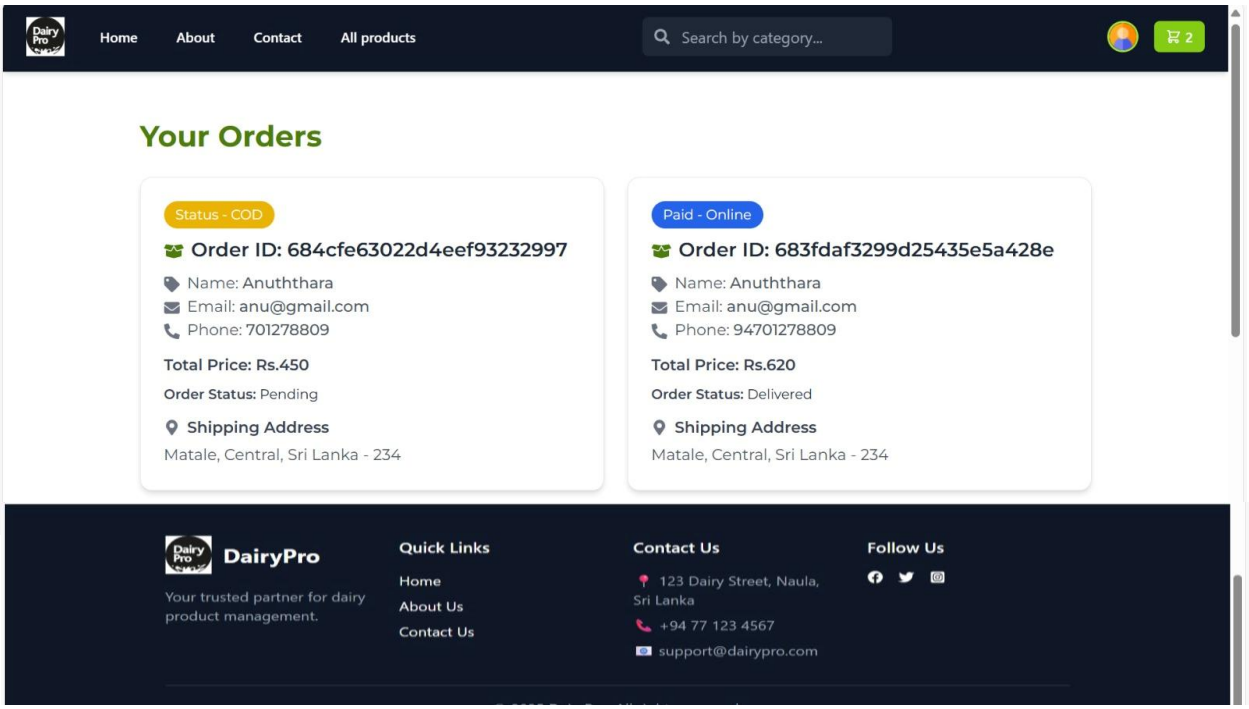

Figure 15 -: Order Page

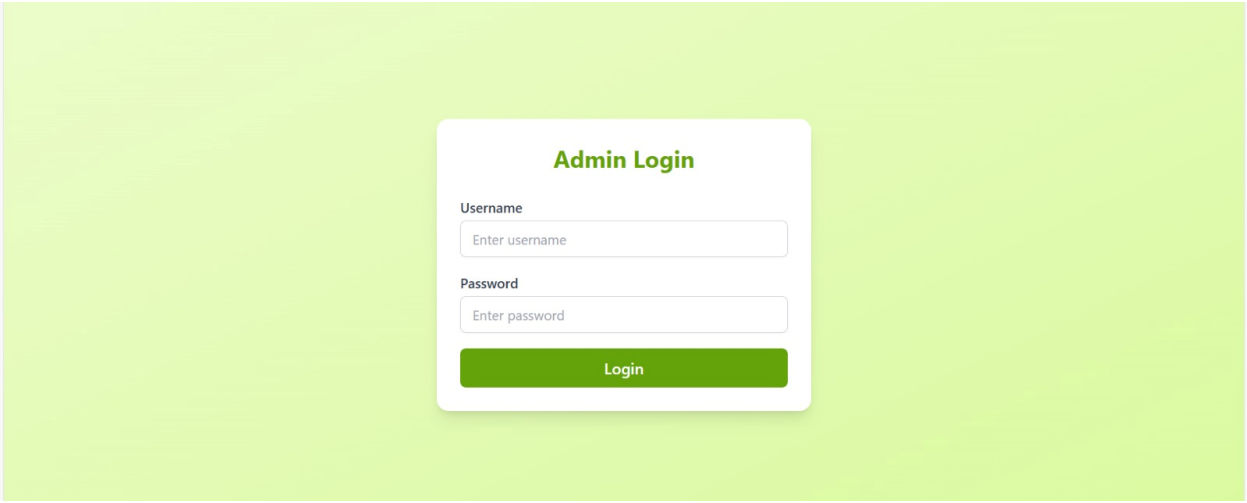

Figure 16 -: Admin Login Page

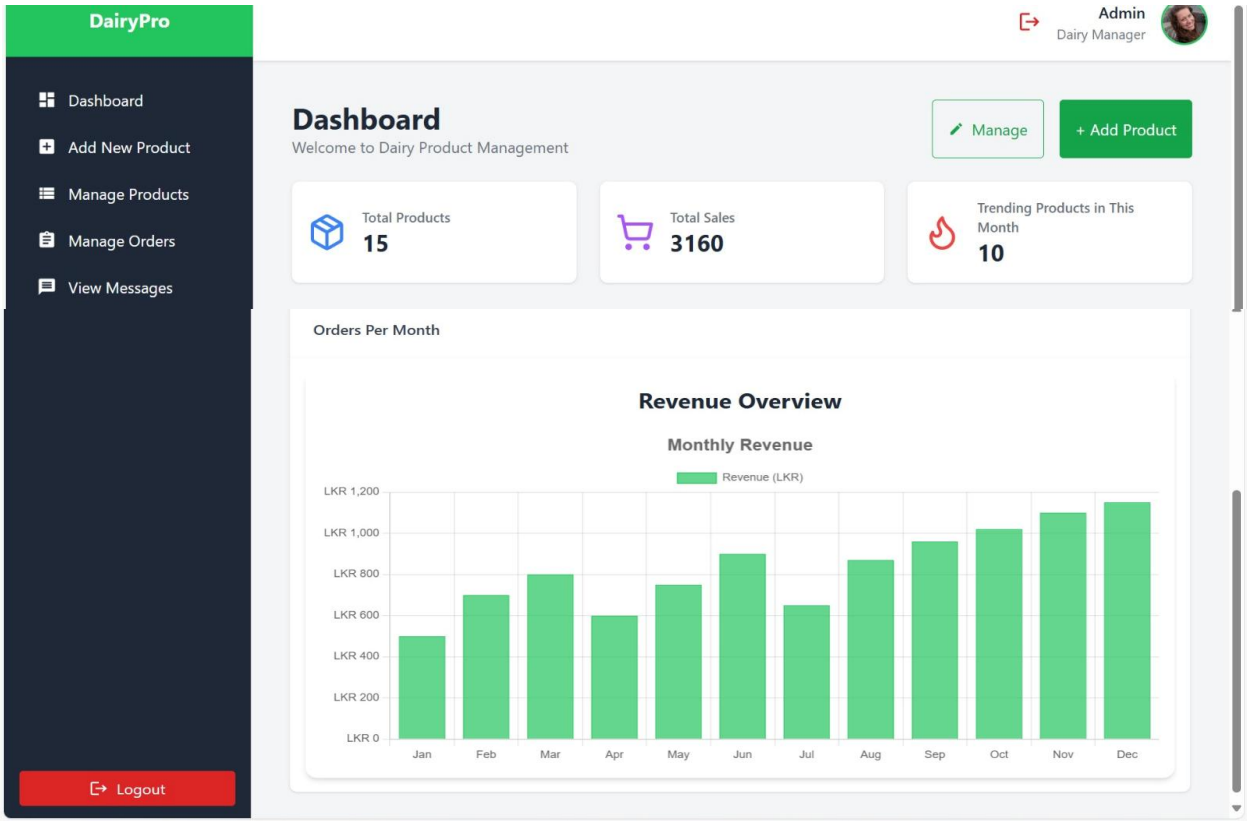

Figure 17 -: Admin Dashboard page

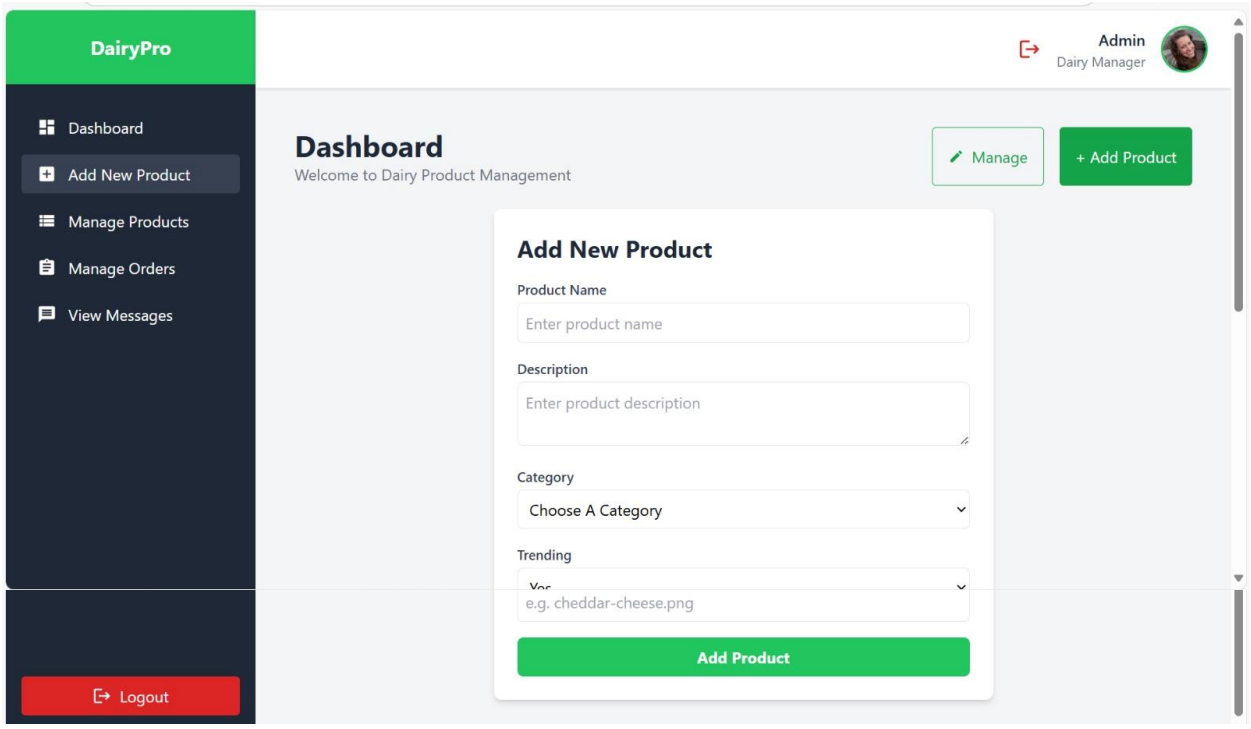

Figure 18 -: Add product page

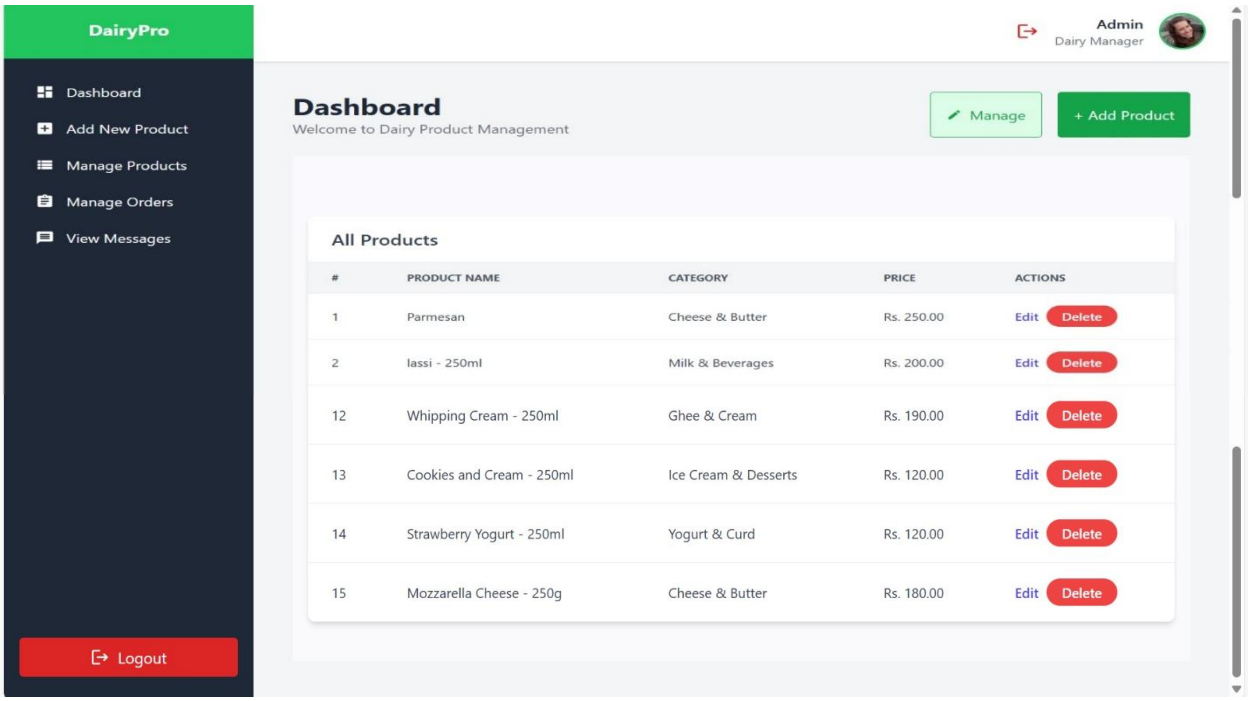

Figure 19 -: Manage Product Page

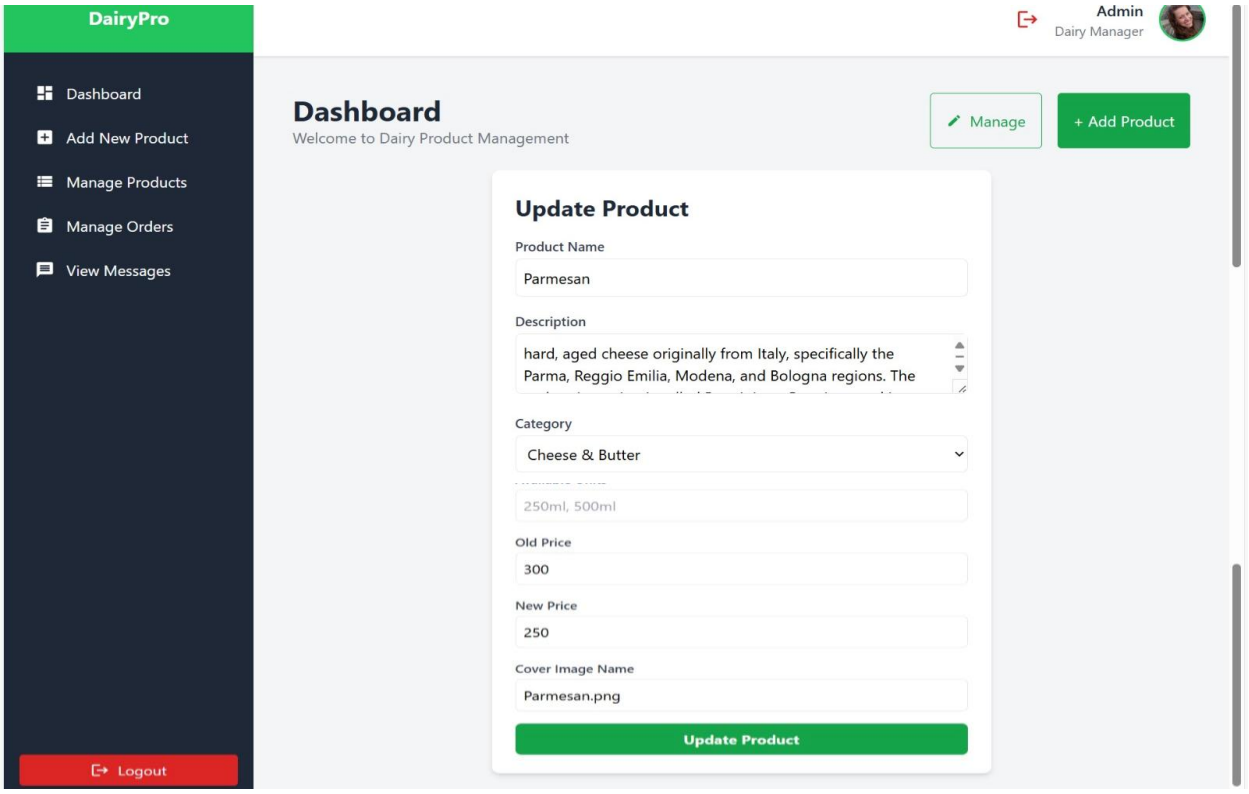

Figure 20 -: Update Product Page

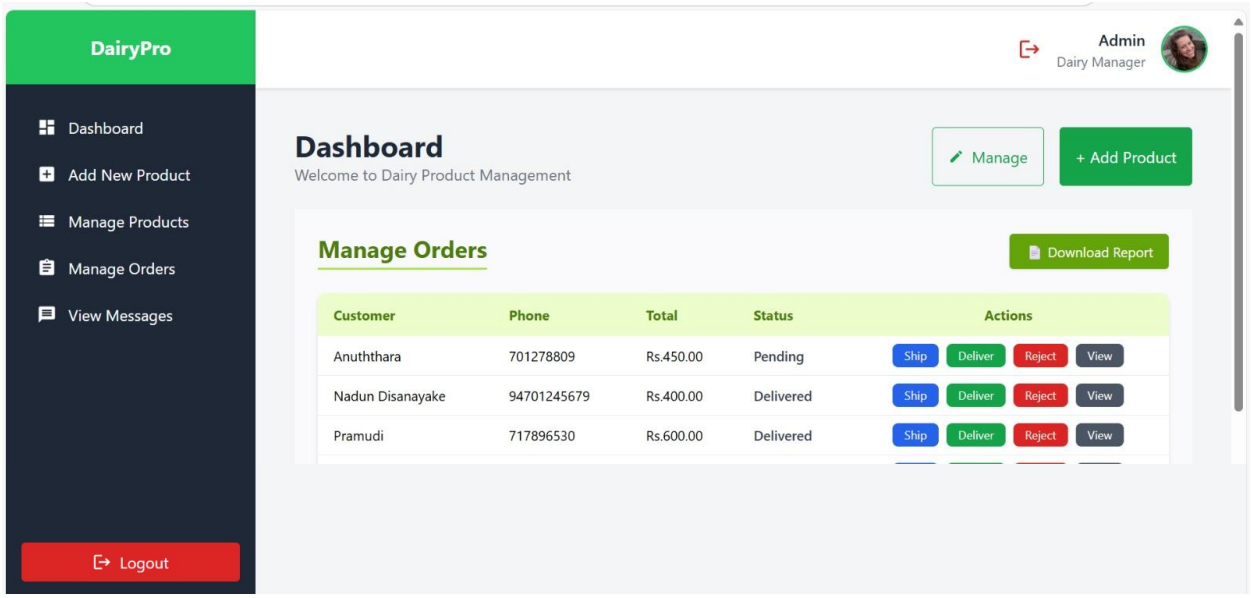

Figure 21 -: Order Management Page

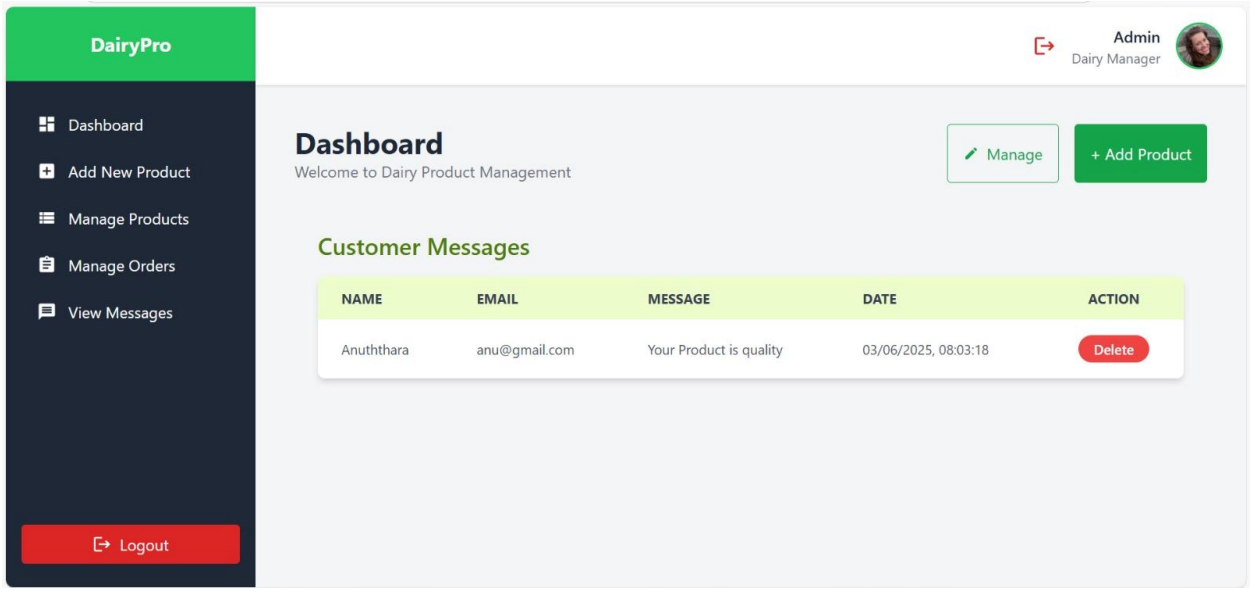

Figure 22 -: View Message Page

4.3 Design Consideration

The following important considerations guided the design of the DairyPro Management System:

- Role-Based Access: To maintain security, the admin and the customer have varying levels of access.
- Scalability: The system can readily accommodate additional features as it expands to meet the needs of the company.
- Security: Login is secured, and user passwords are encrypted.
- Performance: For a seamless experience, load and access data quickly.
- Internet Dependency: For complete functionality, such as order updates, internet access is necessary.
- User-Friendly Interface: A straightforward layout that makes it easy for users to comprehend and operate the system.
- Data Migration: Enables the system to import current manual records.

05.Implementation

5.1 Development environment

The DairyPro Management System was developed using a full-stack JavaScript environment. The development was done on Visual Studio Code (VS Code) with Node.js runtime, and version control managed using Git. Testing and API requests were handled using Postman during development.

- Operating System: Windows 11
- IDE: Visual Studio Code
- Package Manager: npm (Node Package Manager)
- Version Control: Git & GitHub
- Browser Testing: Microsoft Edge and Google Chrome

5.2 Technologies used

The system uses the MERN Stack, which includes:

- Frontend: React.js + Vite(for user interface and client-side rendering)
- Backend: Node.js + Express.js (server-side logic and APIs)
- Database: MongoDB (NoSQL database for storing users, products, and orders,messages)
- Others:
    - Axios for HTTP requests
    - React Router for navigation
    - Mongoose for MongoDB schema modeling
    - JSON Web Token (JWT) for authentication
    - Bcrypt.js for password encryption

5.3 Implementation details

The system is divided into modules for easier management:

- **Frontend Modules:**

- LoginPage.jsx: Handles login for admin and customer
- ProductList.jsx: Displays products by category
- CartPage.jsx: Allows users to update and place orders
- AdminDashboard.jsx: Admin overview and access to reports

- **Backend Modules:**

- authController.js: Handles login and registration
- productController.js: Manages CRUD operations for products
- orderController.js: Handles order creation, updates, and status changes
- messageController.js: Receives and displays contact messages

- **Database Collections:**

- Users, Products, Orders, Messages

Role-based access ensures that admins and users interact with the system through separate dashboards and permissions.

5.4 Code snippets

1. User Registration (Backend – Node.js):

```
const bcrypt = require("bcryptjs");
const newUser = new User({
  name: req.body.name,
  email: req.body.email,
  password: await bcrypt.hash(req.body.password, 10),
  role: "customer",
});
await newUser.save();
```

2. Product Retrieval (Frontend – React + Axios):

```
useEffect(() => {
  axios.get("/api/products").then((res) => {
    setProducts(res.data);
  });
}, []);
```

3. Order Status Update (Backend – Express.js):

```
router.put("/order/:id/status", async (req, res) => {
  const order = await Order.findById(req.params.id);
  order.status = req.body.status;
  await order.save();
  res.send("Order status updated");});
```

06.Testing

6.1 Types of testing conducted

To ensure the reliability, performance, and usability of the DairyPro Management System, several levels of testing were conducted throughout the development lifecycle:

1. Unit Testing
   - Individual components and functions were tested in isolation.
   - Example: Validating product add/update/delete operations and login validation logic.
2. Integration Testing
   - Tested the communication between frontend and backend systems.
   - Focused on:
     - API integration (e.g., user login, order placement),
     - MongoDB database operations,
     - Route and data flow accuracy.
3. System Testing
   - Evaluated the entire application as a whole to ensure all modules worked together as expected.
   - Covered complete user flow:
     - Registration → Login → Product Browsing → Cart → Checkout → Place order→Logout

6.2 Test Cases

Test Case ID	Functionality	Test Scenario	Expected Result	Status
TC01	User Authentication	User registers with valid credentials	User account is created, password encrypted	Passed
TC02	User Authentication	User logs in with correct credentials	User is redirected to user dashboard	Passed
TC03	Admin Authentication	Admin logs in with correct admin credentials	Admin accesses admin dashboard	Passed
TC04	Admin Authentication	Non-admin tries accessing admin dashboard	Access denied with error or redirect to login page	Passed
TC05	Product Browsing (User)	User selects category 'Milk & Beverages'	Only milk-related products are displayed	Passed
TC06	Product Browsing (User)	User views out-of-stock product	Product displayed as unavailable or 'Out of Stock' label shown	Passed
TC07	Product Management (Admin)	Admin creates a new product with valid details	Product is saved and listed	Passed
TC08	Product Management (Admin)	Admin edits an existing product	Updated info appears in product list	Passed
TC09	Product Management (Admin)	Admin deletes a product	Product is removed from database	Passed
TC10	Cart Management (User)	User removes item from cart	Cart updates and item disappears	Passed
TC11	Checkout and Place Order	User fills form and selects Cash On Delivery	Order placed and order confirmation shown	Passed
TC12	Order Management (Admin)	Admin marks a 'Pending' order as 'Shipped'	Status updates correctly	Passed
TC13	Order Management (Admin)	Admin views list of all orders	All orders are listed with status	Passed
TC14	Sales Analytics (Admin)	Admin opens dashboard layout page	Dashboard shows revenue chart, product stats	Passed
TC15	Role-Based Access Control	Logged-in user tries to access admin dashboard	Access denied due to insufficient permissions	Passed
TC16	Role-Based Access Control	Logged-in admin accesses admin-only section	Admin can access all management features	Passed

6.3 Testing tools

- **Postman:** API testing (GET, POST, PUT, DELETE routes)
- **Browser Developer Tools:** Frontend testing and debugging
- **Console and Logs:** Backend test logging for API requests and database interactions
- **Manual Testing Tools:** Checklist-based testing for frontend and UAT

6.4 Test results and analysis

Every test case that was specified was run on various DairyPro Management System components. The outcomes showed that the system satisfies its intended requirements and is functionally stable. The results are summarized as follows.

- A total of 16 test cases were executed.
- 16 was passed.
- 0 failed.

Key Takeaways: Authentication, product management, and order processing all worked as planned for users and administrators.

The successful implementation of role-based access prevented non-admin users from accessing admin features.

Real-time updates and precise filtering were displayed during product browsing and cart operations.

Both the frontend and the backend handled checkout and order status tracking flawlessly.

Neither manual nor API testing revealed any significant or critical bugs.

07.Results

7.1 Outcome of the Project

Both the administrator and user interfaces of the DairyPro Management System are operational, providing seamless product management, ordering, and reporting features. Admins can effectively manage inventory, view sales analytics, and manage customer orders, while users can browse dairy products, add items to their carts, and place orders with services like online payment and cash on delivery.

7.2 Performance metrics

- Average Page Load Time: Less than 2 seconds.
- For important operations (like login and product retrieval), the API response time is between 150 and 300 ms.
- System Uptime: 100% during each test session.
- User Acceptance Rate: All test cases passed with the anticipated outcomes.

7.3 Issues encountered and resolutions

Issue	Cause	Resolution
Admin dashboard access restricted for valid users	Role not correctly assigned in JWT	Updated backend middleware to verify roles properly
Images not displaying in product listings	Incorrect image path setup	Implemented file upload handler and stored proper paths in MongoDB
Product filter not updating dynamically	State management issue in Redux	Refactored reducer and added dependency to useEffect

08.Conclusion

8.1 Summary of the work done

The DairyPro Management System was developed to streamline dairy product management for both administrators and customers. The system includes features such as secure user authentication, product browsing, shopping cart functionality, order processing, and sales analytics. The backend was built with Express.js and MongoDB, while the frontend utilized React with Redux for state management.

8.2 Achievements

- created a fully functional admin dashboard to handle orders, sales reports, and products.
- JWT was used to integrate safe login and registration for administrators and users.
- made it possible to process orders and update product availability in real time.
- test cases passed across all functional modules.

8.3 Limitations of the project

- Does not yet support order update email or SMS notifications and online payment gateway.
- Barcode scanning is not supported; inventory updates are done manually.
- The system is not fully responsive on all screen sizes.
- Lack of support for multiple languages to increase accessibility

8.4 Future work and recommendations

- Set up email alerts and real-time notifications for system messages and order updates.
- Improve the mobile user interface for tablets and smartphones.
- Introduce barcode scanner integration for stock tracking.
- To reach a wider audience, include multilingual support.
- Set up online payment gateway (Payhere)integrate for the system

09.References

9.1 Books

- Sommerville, I. (2015). *Software Engineering* (10th Edition). Pearson Education.
- Pressman, R. S., & Maxim, B. R. (2014). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education

9.2 Articles

- John, D. (2022). "Best Practices in Full Stack Development." *Journal of Web Application Development*, Vol. 14(2), pp. 45–57.
- Silva, K. (2023). "Modern UI/UX for E-commerce Applications." *International Journal of Digital Design*, Vol. 7(1), pp. 25–34.

9.3 Websites

- React Documentation
- MongoDB Documentation
- Express.js Guide
- MDN Web Docs – JavaScript
- W3Schools – HTML, CSS, JavaScript

9.4 Other resources

- Project team notes and SRS document.
- GitHub repositories used for reference in backend structure.
- Postman collections used for API testing.

10.Appendices

10.1 Additional material that supports the main text

The appendices offer additional information to bolster the report's core ideas. Although they might not be necessary for the core body, these additions provide in-depth insights that are useful for reference.

Detailed Tables in Appendix A

- complete test case list, complete with IDs, scenarios, expected outcomes, and status.
- table structure of a database schema, comprising:
- Fields for collecting users
- Fields for collecting products
- Structure for collecting orders

❖ Raw Data Appendix B

- An example of raw JSON data
- object for user registration
- Structure of product items
- Order item with tracking of status

❖ API Endpoints, Appendix C

- A comprehensive list of backend API routes, such as /api/products, /api/orders, and /api/auth/login
- GET, POST, PUT, and DELETE are the methods used.
- A succinct explanation of the parameters for input and output

10.2 Additional diagrams

- ER -Diagram

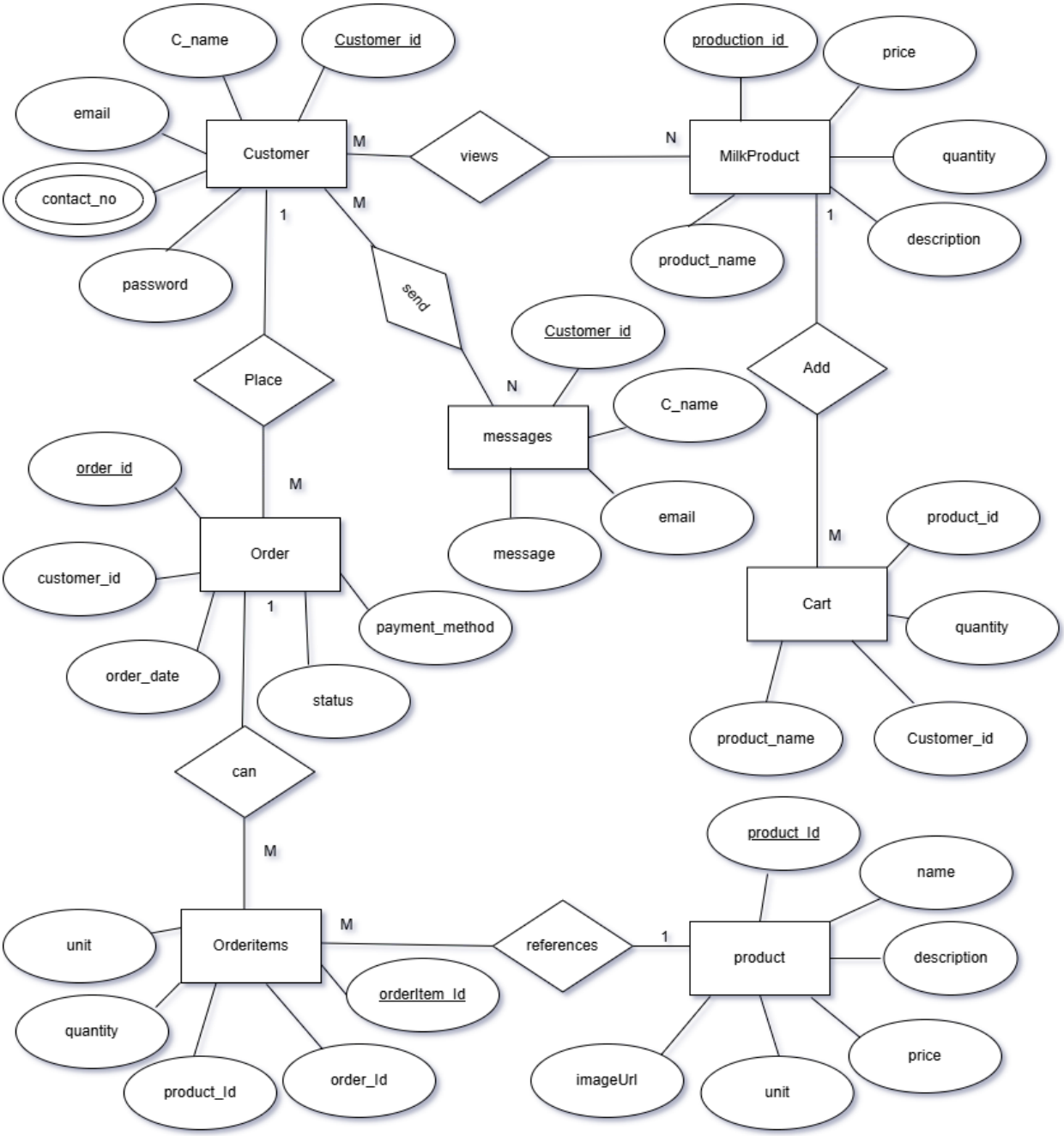

Figure 23 -: ER Diagram