USER CENTRIC E-COMMERCE PLATFORM FOR DAIRY PRODUCTS

A PROJECT REPORT

Submitted by

MEGASHRI N
(Reg. No: 24MCR064)

MOHAN NIVASH C
(Reg. No: 24MCR067)

SELYA VARSNEE M
(Reg. No: 24MCR098)

in partial fulfillment of the
requirements for the award of
the degree of

MASTER OF COMPUTER APPLICATIONS

DEPARTMENT OF COMPUTER APPLICATIONS

Estd : 1984

KONGU ENGINEERING COLLEGE
(Autonomous)

PERUNDURAI, ERODE– 638 060

DECEMBER 2024

DEPARTMENT OF COMPUTER APPLICATIONS

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI, ERODE – 638 060

DECEMBER 2024

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **“USER CENTRIC E-COMMERCE PLATFORM FOR DAIRY PRODUCTS”** is the bonafide record of project work done by **MEGASHRI N (24MCR064), MOHAN NIVASH C (24MCR067)** and **SELYA VARSNEE M (24MCR098)** in partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications of Anna University, Chennai during the year 2024-2025.

SUPERVISOR	**HEAD OF THE DEPARTMENT**
Date:	**(Signature with seal)**

Submitted for the end semester viva voce examination held on ________________________

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

DECLARATION

We affirm that the project entitled **“USER CENTRIC E-COMMERCE PLATFORM FOR DAIRY PRODUCTS”** being submitted in partial fulfilment of the requirements for the award of Master of Computer Applications is the original work carried out by us. It has not formed the part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

MEGASHRI M
(Reg. No: 24MCR064)

MOHAN NIVASH V
(Reg. No: 24MCR067)

SELYA VARSNEE M
(Reg. No: 24MCR098)

I certify that the declaration made by the above candidates is true to the best of my knowledge.

Date: **Name and Signature of the supervisor**

(Dr.P.VIJAYAKUMAR)

ABSTRACT

This project entitled as "**USER CENTRIC E-COMMERCE PLATFORM FOR DAIRY PRODUCTS**" aimed to create an intuitive and efficient digital marketplace connecting consumers with local dairy producers. This platform focuses on delivering a seamless user experience for exploring, selecting, and purchasing fresh dairy products. By employing Node.js for the frontend, the system offers a dynamic and responsive interface, ensuring users can effortlessly navigate through product categories, view real-time availability, and complete transactions with ease. The design prioritizes user-centric features, including personalized recommendations and easy account management, to enhance customer satisfaction and engagement.

The backend of the platform is powered by MongoDB, chosen for its flexibility and scalability in managing diverse datasets such as product catalogs, user profiles, and transaction histories. This NoSQL database facilitates quick and reliable data retrieval, ensuring that users experience minimal latency while browsing or placing orders. MongoDB's structure allows the system to handle real-time updates, such as inventory changes and order processing, making it a robust solution for dynamic e-commerce operations. Integration between the Node.js frontend and MongoDB backend ensures a seamless flow of information, enabling efficient order management and secure data handling.

Overall, this project combines modern web technologies to address the growing demand for convenient and sustainable access to fresh dairy products. By fostering direct interaction between producers and consumers, it supports local businesses while providing users with high-quality products. The platform is designed to scale as demand increases, ensuring long-term adaptability and relevance in the competitive e-commerce landscape.

ACKNOWLEDGEMENT

We respect and thank our correspondent, **THIRU.A.K. ILANGO B.Com., MBA., LLB.,** and our principal, **Dr. V. BALUSAMY BE (Hons.)., M.Tech, Ph.D.** Kongu Engineering College, Perundurai for providing us with the facilities offered.

We convey our gratitude and heartfelt thanks to our Head of Department, Professor **Dr.A.TAMILARASI, MSc., MPhil., Ph.D., M.Tech.,** Department of Computer Applications, Kongu Engineering College, for her perfect guidance and support that made this work to be completed successfully.

We also wish to express our gratitude and sincere thanks to our project coordinators, **Mrs. S. HEMALATHA, MCA., Assistant Professor (Sr.G),** Department of Computer Applications, Kongu Engineering College, who have motivated us in all aspects to complete the project within the scheduled time.

We would like to express our gratitude and sincere thanks to our project guide, **Dr. P. VIJAYAKUMAR, MCA, M.Phil., Ph.D., MBA,** Department of Computer Applications, Kongu Engineering College, for giving his valuable guidance and suggestions that helped us in the successful completion of the project.

We owe a great deal of gratitude to our parents for helping us to overwhelm in all proceedings. We bow our hearts and heads with heartfelt thanks to all those who gave us their warm services to succeed and achieve our work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS	**FULLFORM**
API	Application Programming Interface
DBMS	Database Management System
UI	User Interface
JSON	JavaScript Object Notation
HTML	HyperText Markup Language
CSS	Cascading Style Sheets

CHAPTER 1

INTRODUCTION

1.1 ABOUT THE PROJECT

An online laundry management system is a software application designed to streamline and automate various aspects of laundry business operations. Whether it is a small local laundry service or a larger commercial laundry facility, such a system can help manage day-to-day tasks efficiently. Here are some key features and functionalities typically found in an online laundry management system.

The project's primary backend database, MongoDB, ensures effective administration of dynamic data, including user accounts, transaction histories, and product catalogues. For managing inventory changes, order processing, and user interactions, MongoDB's NoSQL design provides great scalability and real-time updates.The goal of this platform is to empower local dairy enterprises and give consumers dependable, convenient access to fresh, locally produced products.

1.2 EXISTING SYSTEM

The current system for buying dairy goods mostly uses conventional techniques, such going to nearby stores or relying on door-to-door delivery services, which can be difficult and time-consuming for customers. These approaches frequently result in inefficiencies and discontent since they don't provide real-time information on product availability, pricing, or order tracking. Dairy goods are included in the inventory of various e-commerce platforms, but they frequently lack a user-centric strategy designed especially for the particular needs of dairy consumption, such as freshness and prompt delivery.

1.3 DRAWBACKS OF EXISTING SYSTEM

Numerous issues with the current system limit its effectiveness and customer happiness. The absence of real-time inventory updates is a significant problem that can result in instances where clients place purchases only to discover that certain products are not available. The buying procedure is further made more difficult and time-consuming by the lack of a centralised platform, which limits the alternatives for evaluating costs, quality, or delivery schedules. Manual record-keeping is a common component of traditional systems, which raises the possibility of mistakes in inventory tracking and order management. The inability of local producers to reach consumers directly restricts their capacity to grow and contend with larger suppliers.

1.4 PROPOSED SYSTEM

"User Centric E-Commerce Platform for Dairy Products," the suggested system, is a specialised digital platform designed to revolutionise the distribution and purchase of dairy goods. The system offers a complete solution that serves both producers and consumers by incorporating cutting-edge web technology. Customers can check product specifications, browse a wide variety of fresh dairy products, and make well-informed judgements about what to buy. While tools like order monitoring and alerts improve the user experience by informing customers of their purchase progress, the use of secure payment methods guarantees seamless and trouble-free transactions.

The platform offers producers an easy-to-use interface for tracking sales performance, updating product availability, and managing inventories. While the MongoDB backend guarantees dependable storage and retrieval of dynamic data, including customer profiles, product details, and order histories, the Node.js frontend enables a very responsive user experience

1.5 ADVANTAGES OF THE PROPOSED SYSTEM

The inadequacies of the current system are addressed by the many benefits of the suggested approach. The opportunity to establish direct connections between customers and regional producers, which promotes confidence and guarantees access to fresh, premium dairy products, is one important advantage. Users are always aware of product availability because to the platform's real-time updates, which reduce errors in inventory and order management. By encouraging local companies and lowering dependency on middlemen, it also encourages sustainability by resulting in shorter supply chains and less of an adverse effect on the environment.

1.6 SUMMARY

The proposed system, "User-Centric E-Commerce Platform for Dairy Products," aims to modernize the traditional methods of purchasing and distributing dairy goods. By leveraging MongoDB's dynamic data management and Node.js for a responsive frontend, the platform ensures real-time inventory updates, secure transactions, and seamless order tracking. It addresses the inefficiencies of conventional systems, such as the lack of centralized platforms and real-time information, while fostering direct connections between consumers and local producers. This innovative approach not only enhances customer satisfaction by providing fresh, high-quality products but also promotes sustainability by reducing supply chain dependency and supporting local businesses.

CHAPTER 2

SYSTEM ANALYSIS

2.1 IDENTIFICATION OF NEED

The increasing demand for fresh and high-quality dairy products, coupled with the growing preference for convenient online shopping, highlights the necessity for a dedicated e-commerce platform tailored to the dairy industry. Traditional supply chains often face challenges such as limited accessibility, lack of transparency, inconsistent product quality, and inefficiencies in delivery logistics. Consumers require a reliable platform that not only provides easy access to fresh dairy products but also ensures trust through real-time tracking, secure transactions, and user-focused services. Similarly, dairy producers and vendors need a digital space to expand their reach, streamline operations, and connect directly with their customers.

2.2 FEASIBILITY STUDY

A feasibility study for an online laundry management system is a crucial step in assessing the viability and practicality of implementing such a system. This study typically involves an in-depth analysis of various aspects, including technical, economic, operational, and legal considerations. Here are the key components that would be part of a feasibility study for an online laundry management system:

- Technical Feasibility
- Operational Feasibility
- Economic Feasibility

2.2.1 Technical Feasibility

The price and gain evaluation can be concluded that automatic gadget is beneficial in today’s fast-transferring world. The evaluation of technical feasibility ought to be primarily based totally on a define layout of the gadget necessities in phrases of input, output, files, packages and procedure. The technical feasibility study basically centered on an alternative for hardware, software and design approach to determine the functional aspects of the system.

A useful way to present the results of the feasibility studies is using a feasibility study of technical, operational etc. A technical feasibility study helps organizations to make them idea more functional and design approach to support the systems.

2.2.2 Operational Feasibility

Operational feasibility is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. The operational feasibility assessment focuses on the degree to which the proposed development projects fits in with the existing business environment and objectives with regard to development schedule, delivery date, corporate culture, and existing business processes.

To ensure success, desired operational outcomes must be imparted during design and development. These include such design-dependent parameters such as reliability, usability, predictability, disposability, sustainability, affordability and others. These parameters required to be considered at the early stages of design if desired operational behavior are to be realized.

2.2.3 Economic Feasibility

Economic feasibility is the most important and frequently used method for evaluating the effectiveness of the proposed system. It is very essential because the main goal of the proposed system is to have economically result along with increased efficiency. Especially in the present scenario, where the objective is towards compatibility, reduced cost is weighted against the ultimate income or benefit derived from the developed system. It includes quantification and identification of all the benefits expected. This assessment typically involves a cost/ benefits analysis. The software used in its application is available as an open source and thereby it is cost effective. Cost incurred in debugging is relatively less than in other applications.

2.3 SOFTWARE REQUIREMENT SPECIFICATION

A declaration that describes the capability that a system needs to satisfy the user's requirements is known as a system requirement. The system requirements for specific machines, software, or business operations are general. Taking it all the way down to the hardware and coding that operates the software. System requirements are the most efficient way to address user needs while lowering implementation costs.

2.3.1 Hardware Requirements

The hardware for the system is selected considering factors such as CPU processing speed, memory access, peripheral channel access speed, printed speed; seek time& relational data of hard disk and communication speed, etc.

Processor	:	Intel Core i3
RAM	:	4.00 GB
Monitor	:	15”VGA MonitorHard
Disk	:	1 TB
Keyboard	:	ACCUTYPE keyboard

2.3.2 Software Requirements

The software for the project is selected considering factors such as working front-end environment, flexibility in the coding language, database knowledge of enhanced backend technology, etc.

Operating System	:	Windows 10
Front End	:	Reactjs
Back End	:	Nodejs
Database	:	MongoDB
Tools	:	Visual studio code

Front End

React JS

ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front-end library which is responsible only for the view layer of the application. The primary goal of ReactJS is to create User Interfaces (UI) that accelerate apps. It makes use of virtual DOM (JavaScript object), which enhances the app's performance. Faster than the standard DOM is the

Javascript virtual DOM. ReactJS integrates with different frameworks and can be used on the client and server sides. It makes use of components and data patterns to make larger apps more readable and maintainable. A ReactJS application is made up of multiple components,

each component responsible for outputting a small, reusable piece of HTML code. The components are the heart of all React applications. These Components can be nested with other components to allow complex applications to be built of simple building blocks. ReactJS uses a virtual DOM-based mechanism to fill data in HTML DOM. The virtual DOM works fast as it only changes individual DOM elements instead of reloading the complete DOM every time. NodeJS and NPM are the platforms needed to develop any ReactJS application. React-Bootstrap replaces the Bootstrap JavaScript. Each component has been built from scratch as a true React component.

Server

Node JS

Node.js is a server-side JavaScript runtime built on Chrome's V8 JavaScript engine. It allows you to run JavaScript on the server, which can be used to build backend services, such as webservers. Node.js is commonly used for building server-side applications, as it provides an easy way to create a network application that can handle a large number of concurrent connections.

Node.js is an event-driven runtime, meaning that it allows you to build applications that can handle events and perform certain actions in response to those events. It is designed to be lightweight and efficient, making it well-suited for building scalable network applications. One of the main benefits of using Node.js is that it allows you to use JavaScript on both the frontend and back end of an application. This can make it easier to develop full-stack applications, as you can use the same language throughout the entire application.

Database

MongoDB is a popular NoSQL database system that provides a flexible and scalable approach to storing and handling data. Unlike traditional relational databases, MongoDB is schema-less, which means developers can store and retrieve data in a more dynamic and agile manner. MongoDB is a document-oriented database in which data is stored in BSON (Binary JSON)-like documents. Each document is a JSON-like object with field-value pairs, which allows for nested structures and arrays.

This flexibility makes it ideal for handling complex and dynamic data. Data is organized into collections, which are similar to tables in relational databases. Each collection consists of documents, and each document represents a record in the collection. Collections and documents can be created and modified on-the-fly without requiring a predefined schema. MongoDB uses a rich query language that supports a wide range of queries, including filtering, sorting, and aggregation. It allows developers to perform complex queries and retrieve specific data from large datasets efficiently. Indexing is a crucial feature in MongoDB for optimizing query performance. Developers can create indexes on fields to speed up data retrieval, particularly for frequently queried fields. MongoDB uses a JSON-like syntax for queries, making it easy for developers to work with and understand. This aligns well with modern web development practices, where JSON is a common data interchange format. MongoDB includes a powerful aggregation framework that enables developers to perform data transformations, calculations, and aggregations directly within the database

2.4 SUMMARY

The proposed system addresses the growing need for a dedicated e-commerce platform for dairy products, emphasizing accessibility, transparency, and efficiency. A feasibility study ensures the system's technical, operational, and economic viability, highlighting benefits such as cost-effectiveness and improved functionality. The platform employs ReactJS for a dynamic front-end, Node.js for scalable server-side applications, and MongoDB for a flexible, schema-less database. These technologies ensure seamless real-time updates, efficient data handling, and a user-friendly experience. By integrating modern web development practices, the system enhances user satisfaction, optimizes delivery logistics, and supports both consumers and producers in the dairy industry.

CHAPTER 3
SYSTEM DESIGN

3.1 MODULE DESCRIPTION

A module description provides detailed information about the module and its supported components, which is accessible in different manners.

The project contains the following modules:

- Login
- Product Search Module
- Cart Module
- Product Management Module
- Order Management Module
- Payment Integration Module

Login Module:

This module controls user registration, login, and role-based permissions to guarantee safe platform access. It enables both new and existing users to establish accounts and log in with safe credentials. For added protection, it offers password encryption (e.g., bcrypt). In order to provide stateless and secure communication between the client and server, authentication is handled utilising technologies such as JWT (JSON Web Token). Customers can browse and buy things, while Admins can manage the platform. Role-based access control is in place to differentiate between the two groups. This module serves as the cornerstone for data security, session management, and user identity maintenance.

Product Search Module:

This module offers consumers a well-structured and intuitive interface for perusing the dairy goods that are available. In an organised catalogue, it shows product details like name, price, category, and pictures. It has strong search capabilities to improve user experience, enabling users to locate products by price ranges, categories, or keywords. Furthermore, filtering features that allow customers to easily traverse the catalogue include category-specific views and price sorting (low-to-high or high-to-low). In order to manage lengthy product listings with ease and guarantee consistent platform performance and usability, pagination is used.

Cart Module:

This module offers consumers a well-structured and intuitive interface for perusing the dairy goods that are available. In an organised catalogue, it shows product details like name, price, category, and pictures. It has strong search capabilities to improve user experience, enabling users to locate products by price ranges, categories, or keywords. Furthermore, filtering features that allow customers to easily traverse the catalogue include category-specific views and price sorting (low-to-high or high-to-low). In order to manage lengthy product listings with ease and guarantee consistent platform performance and usability, pagination is used

Product Management Module:

Through the Product Management module, the administrator can effectively oversee the platform's dairy product catalogue. With information like name, price, category, number, and product photos, it enables the entry of new products. Admins have the ability to change the prices and stock levels of existing products or take them down from the listing as necessary. By keeping the product catalogue current and well-structured, this module makes sure that users are given correct and pertinent information. It provides the framework for a seamless consumer buying experience by preserving product availability and data.

Order Management Module:

From placing to delivery, the Order Management module manages the whole order lifecycle. Customers can place purchases by verifying the items in their cart and entering their delivery information at the time of checkout. Order statuses, including Processing, Shipped, and Delivered, are tracked after an order is placed, and an order summary is generated. To notify clients in real time, administrators have the ability to monitor and modify order statuses. By ensuring a smooth order processing flow and offering transparency and effective platform buy management, this module raises user happiness.

Payment Integration Module:

Customers can complete payments securely and conveniently at checkout thanks to the Payment Integration module. It accepts a number of payment methods, including Cash on Delivery (COD) and the incorporation of online payment platforms like Stripe or Razorpay. By using encryption techniques and verifying payment information prior to order confirmation, the module guarantees transaction security. An order confirmation and a purchase summary are sent to users upon successful payment. By offering dependable and smooth payment options, this module improves user experience while guaranteeing consumer ease.

3.2 DATAFLOW DIAGRAM

A data flow diagram shows the way information flows through a process or system. It includes data inputs and outputs, data stores, and the various subprocesses the data moves through.

LEVEL 0

It initiates the laundry service by submitting clothes for washing and Processes laundry requests and manages washing, drying. Admin receives and processes the order, checking for any special instructions and assigning a unique order ID. The following process is described in the below figure 3.1

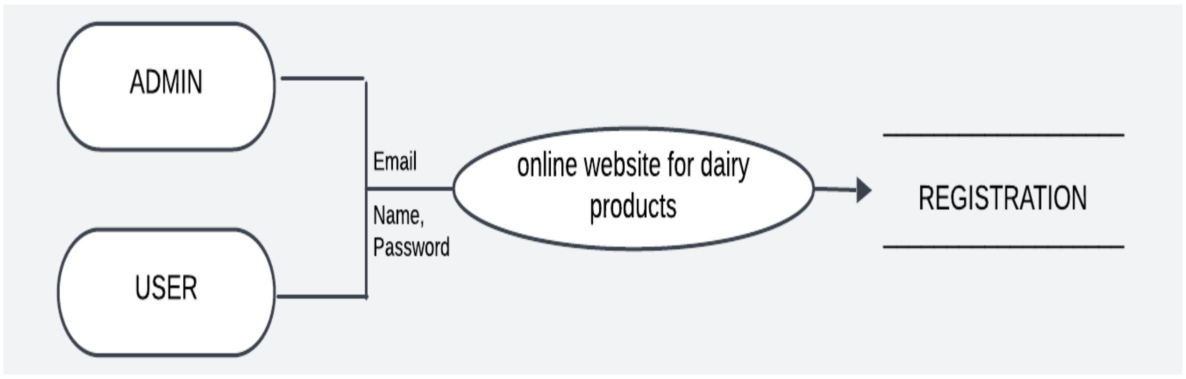


Figure 3.1 Dataflow Diagram Level 0

LEVEL 1

Records details of each laundry order, including the order ID, customer details, items to be laundered, and processing status. The current status of the order is communicated back to the customer through notifications or updates. the system may collect feedback from customers, updating the customer database with reviews and ratings. The order details and issues are explained in the following diagram figure 3.2

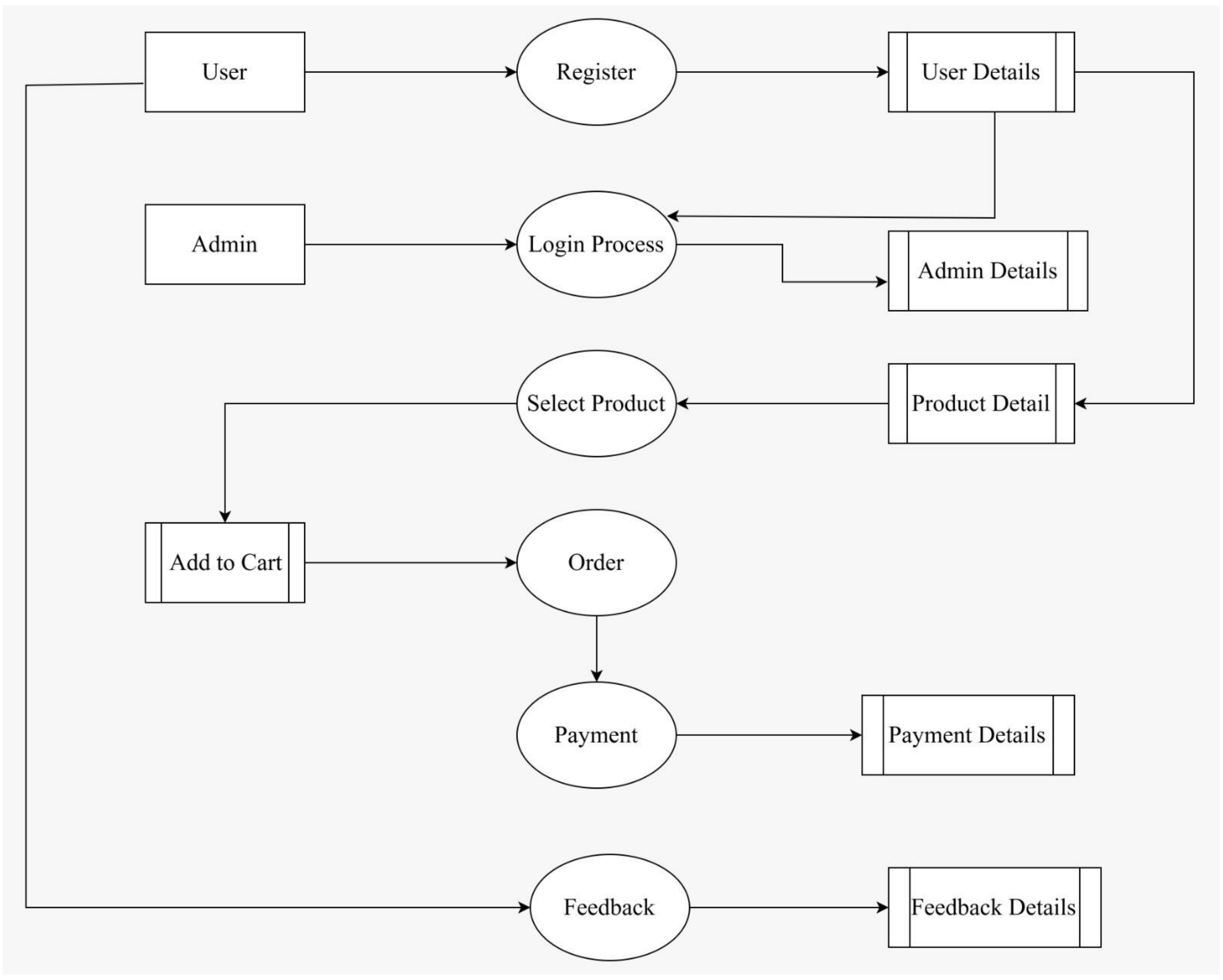


Figure 3.2 Dataflow Diagram Level 1

3.3 DATABASE DESIGN

Database design is the organization of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. With this information, they can begin to fit the data into the database model. Database management system manages the data accordingly. The term database design can be used to describe many different parts of the design of an overall database system.

Principally, and most correctly, it can be thought of as the logical design of the base data structure used to store the data. In an object database, the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structure, but also the forms and queries used as part of the overall database application within the database management system.

3.4 INPUT DESIGN

Input design is the process of converting user-originated inputs to a computer-understandable format. Input design is one of the most expensive phases of the operation of a computerized system and is often the major problem of a system. A large number of problems with a system can usually be tracked back to fault input design and method.

Every moment of input design should be analyzed and designed with utmost care. The decisions made during the input design are the project gives the low time consumption to make sensitive application made simple. Thus, the developed system is well within the budget. This was achieved because most of the technologies used are freely available. Only the customized product had to be purchased. In the project, the forms are designed with easy- to-use options. The coding is being done such that proper validation is made to get the perfect input. No error inputs are accepted.

3.5 OUTPUT DESIGN

Output design generally refers to the results and information that are generated by the system for many end-users; it should be understandable with the enhanced format. Computer output is the most important direct source of information to the user. Output design deals with form design. Efficient output design should improve the interfacing with the user. The term output applies to any information produced by an information system in terms of data displayed. When analysts design system output, they identify the specific output that is needed to meet the requirements of the end user.

Previewing the output reports by the user is crucial because they are the ultimate judge of the quality of the output and the success of the system. System analysis helps to determine which applications, websites or documents are allowed or blocked. The output should be designed in an attractive, convenient, and informative manner. The following figure 3.6 shows the data storage of feedbacks given by the customer.

3.6 SUMMARY

The system design encompasses key modules such as Login, Product Search, Cart, Product Management, Order Management, and Payment Integration, each contributing to a seamless and secure user experience. These modules handle essential functions like user authentication, product browsing, order tracking, inventory management, and secure payment processing. Data flow diagrams outline the system's operations, showing how data moves through various processes, while database design ensures efficient data storage and retrieval. Input design focuses on capturing accurate user inputs with validation mechanisms, and output design emphasizes generating clear, user-friendly information to enhance interaction and system usability.

CHAPTER 4

IMPLEMENTATION

4.1 CODE DESCRIPTION

Code description can be used to summarize code or to explain the programmer's intent. Good comments don't repeat the code or explain it. They clarify its intent. Comments are sometimes processed in various ways to generate documentation external to the source code itself by document generator or used for integration with systems and other kinds of external programming tools. I have chosen to react JS as the front end and MongoDB as the back end.

4.2 STANDARDIZATION OF THE CODING

Coding standards define a programming style. A coding standard does not usually concern itself with wrong or right in a more abstract sense. It is simply a set of rules and guidelines for the formatting of source code. The other common type of coding standard is the one used in or between development teams. Professional code performs a job in such a way that is easy to maintain and debug. All the coding standards are followed while creating this project. Coding standards become easier, the earlier you start. It is better to do a neat job than cleaning up after all is done. Every coder will have a unique pattern that he adheres to. Such a style might include the conventions he uses to name variables and functions and how he comments his work. When the said pattern and style are standardized, it pays off the effort well in the long.

4.3 ERROR HANDLING

Exception handling is a process or method used for handling abnormal statements in the code and executing them. It also enables to handle the flow control of the code/program. For handling the code, various handlers are used that process the exception and execute the code. Mainly if-else block is used to handle errors using condition checking.

In many cases many corner cases must be checked during an execution but "if-else" can only handle the defined conditions. In if-else, conditions are manually generated based on the task. An error is a serious problem and an application does not usually get passed without incident. Errors cause an application to crash, and ideally send an error message offering some suggestion to resolve the problem and return to a normal operating state, There is no way to deal with errors “live” or in production the only solution is to detect them via error monitoring and bug tracking and dispatch a developer or two to sort out the code.

Exceptions, on the other hand, are exceptional conditions an application should reasonably be accepted to handle. A programming language allows developers to include try…catch statements to handle exceptions and apply a sequence of logic to deal with the situation instead of crashing. And when an application encounters an exception that there’s no workaround for, that’s called an unhandled exception, which is the same thing as an error.

4.4 SUMMARY

The implementation chapter emphasizes the importance of clear code documentation, adherence to coding standards, and robust error handling to ensure maintainable and reliable software development. ReactJS is utilized for the front end, and MongoDB serves as the back end, with coding standards ensuring consistency and ease of debugging. Error handling employs techniques like *if-else* blocks for predefined conditions and *try-catch* statements for managing exceptions, preventing crashes and enabling logical recovery during runtime. This structured approach minimizes disruptions, improves maintainability, and enhances the overall reliability of the system.

CHAPTER 5

TESTING AND RESULTS

5.1 TESTING

Software testing serves as the final assessment of specifications, designs, and coding and is a crucial component of software quality assurance. The system is tested throughout the testing phase utilizing diverse test data. The preparation of test data is essential to the system testing process. The system under study is tested after the test data preparation. Once the source code is complete, relevant data structures should be documented. The finished project must go through testing and validation when errors are explicitly targeted and attempted to be found.

TYPES OF TESTING

The various types of testing

- Unit testing
- Integration Testing
- Validation Testing

5.1.1 Unit Testing

A testing strategy known as unit and integration testing has been used to check that the system behaves as expected. The testing strategy was based on the functionality and the requirements of the system. In unit checking out, we have to check the applications making up the device. For this reason, Unit checking out once in a while is referred to as a program checking out. The software programs in a device are the modules and workouts that might be assembled and included to carry out a selected function, Unit checking out the first at the modules independently of 1 another, to find mistakes.

This enables, to stumble on mistakes in coding and common sense which might be contained with the module alone. The checking out became completed at some stage in programming level itself

Test Case 1

Module	: User Login
Input	: Username and Password
Output	: Redirect to home page

Test 1

Module	: User Login
Email	: mega@gmail.com
Password	: 123456
Output	: Redirected to approval page
Analysis	: Username and Password has been verified

Test 2

Module	: User Login
Email	: selya@gmail.com
Password	: 123456
Output	: Enter the correct username
Analysis	: Username and Password has been checked and error shown

5.1.2 Integration Testing

Integration testing is done to test if the individual modules work together as one single unit. In integration testing, the individual modules that are to be integrated are available for testing. Thus, the manual test data that is used to test the interfaces is replaced by that which is generated automatically from the various modules. It can be used for testing how the module would interact with the proposed system. The modules are integrated and tested to reveal the problem interfaces.

Test Case 1

Module	: User Login
Input	: Username and Password
Output	: Redirect to home page

Test 1

Module	: User Login
Email	: selya@gmail.com
Password	: 123456
Output	: Redirected to approval page
Analysis	: Username and Password has been verified

Test 2

Module	: User Login
Email	: mega@gmail.com
Password	: 123456
Output	: Enter the correct username
Analysis	: Username and Password has been checked and error shown

5.1.3 Validation Testing

Verification and validation checking out are critical tests, which might be achieved earlier than the product has been surpassed over to the customer. This makes sure, that the software program checking out lifestyles cycle begins off evolved early. Each verification and validation intends to make certain that the product is made in step with the necessities of the customer and does certainly fulfil the supposed purpose.

Test case 1

Module	: User Registration
Input	: Password

Test 1

Module	: User Registration
Password	: 123456
Output	: Registered successfully

5.2 SUMMARY

This Chapter focuses on the testing and validation of the software to ensure quality and reliability. It involves different types of testing—Unit, Integration, and Validation Testing. Unit testing verifies individual modules to detect coding and logical errors, while integration testing checks how modules interact and function as a single unit. Validation testing ensures the software meets customer requirements and performs as intended. Test cases for user login and registration illustrate the testing process, highlighting scenarios where inputs like username and password are verified for correctness, ensuring proper redirection or error handling as required.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

The “User centric E-Commerce platform for Dairy Products” successfully bridges the gap between dairy product suppliers and consumers, offering a seamless, efficient, and user-friendly experience. By incorporating features like personalized product recommendations, secure payment gateways, and real-time order tracking, the platform prioritizes user convenience and satisfaction. This project demonstrates the potential of technology in transforming traditional dairy supply chains into a modern digital ecosystem, enhancing accessibility and transparency for both producers and consumers.

This project demonstrates how technology can revolutionize traditional supply chains, empowering both consumers and producers while fostering growth in the dairy sector. The successful implementation of this platform underscores the importance of digital transformation in building efficient, scalable, and user-focused e-commerce ecosystems.

6.2 FUTURE ENHANCEMENT

Looking ahead, The platform can be further enhanced by integrating advanced technologies like artificial intelligence for better inventory management and predictive analytics to forecast demand. Incorporating IoT devices for real-time monitoring of dairy product quality during transit can ensure freshness and safety. Additionally, expanding the platform to include multilingual support and a wider range of payment options will make it more accessible to a diverse user base. Partnering with local farmers and implementing a subscription-based delivery system can also create a more sustainable and scalable solution.

Furthermore, establishing partnerships with local farmers and dairy cooperatives can strengthen the supply chain, and introducing subscription-based delivery models can cater to recurring customer needs. Adding features like blockchain for supply chain transparency and eco-friendly packaging options can enhance trust and align the platform with sustainable practices, paving the way for a robust and future-ready e-commerce solution.

6.3 SUMMARY

This Chapter concludes that the "User-Centric E-Commerce Platform for Dairy Products" successfully modernizes traditional dairy supply chains, offering features like secure payments, real-time tracking, and personalized recommendations to enhance user satisfaction. It highlights the transformative role of technology in creating efficient and transparent digital ecosystems. Future enhancements include integrating AI for inventory management, IoT for quality monitoring, blockchain for transparency, and multilingual support. Expanding partnerships with local farmers, adding eco-friendly practices, and introducing subscription-based delivery systems are proposed to ensure scalability, sustainability, and a robust user experience.

APPENDICES

A.SAMPLE CODING

LOGIN PAGE

```
import {Fragment, useEffect, useState } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { clearAuthError, login } from '../../actions/userActions';
import MetaData from '../layouts/MetaData';
import { toast } from 'react-toastify';
import { Link, useLocation, useNavigate } from 'react-router-dom';
export default function Login() {
const [email, setEmail] = useState("")
const [password, setPassword] = useState("")
const dispatch = useDispatch();
const navigate = useNavigate();
const location = useLocation();
const { loading, error, isAuthenticated } = useSelector(state => state.authState)
const redirect = location.search?'/'+location.search.split('=')[1]:'/';
        const  submitHandler = (e) => {
  e.preventDefault();
  dispatch(login(email, password))
}
useEffect(() => {
if(isAuthenticated) {
   navigate(redirect)
}
 if(error)  {
   toast(error, {
      position: toast.POSITION.BOTTOM_CENTER,
      type: 'error',
      onOpen: ()=> { dispatch(clearAuthError) }
```

```
        return

    }
  },[error, isAuthenticated, dispatch, navigate]
return (
<Fragment>
    <MetaData title={Login} />
    <div className="row wrapper">
        <div className="col-10 col-lg-5">
            <form onSubmit={submitHandler} className="shadow-lg">
                <h1 className="mb-3">Login</h1>
                <div className="form-group">
                <label htmlFor="email_field">Email</label>
                <input
                    type="email"
                    id="email_field"
                    className="form-control"
                    value={email}
                    onChange={e =>setEmail(e.target.value)}
                />
                </div>
                <div className="form-group">
                <label htmlFor="password_field">Password</label>
                <input
                    type="password"
                    id="password_field"
                    className="form-control"
                    value={password}
                    onChange={e =>setPassword(e.target.value)}
                />
                </div>
                 <Link to="/password/forgot" className="float-right mb-4">Forgot Password?</Link>
                <button
                id="login_button"
                type="submit"
                className="btn btn-block py-3"
                disabled={loading}
                >
                LOGIN
                </button>
                <Link to="/register" className="float-right mt-3">New User?</Link>
            </form>
        </div>
    </div>
```

FRONT PAGE

```
import { Fragment, useEffect, useState } from "react";
import { useDispatch, useSelector } from "react-redux";
import { getProducts } from "../actions/productActions";
import Loader from "./layouts/Loader";
import MetaData from "./layouts/MetaData";
import Product from "./product/Product";
import  {toast} from 'react-toastify';
import Pagination from 'react-js-pagination';
export  default function Home(){
const dispatch = useDispatch();
 const {products, loading, error, productsCount, resPerPage} =    useSelector((state) =>
state.productsState)
const [currentPage, setCurrentPage] = useState(1);
const setCurrentPageNo = (pageNo) =>
   setCurrentPage(pageNo)
}
useEffect(()=>{
  if(error) {
      return toast.error(error,{
         position: toast.POSITION.BOTTOM_CENTER
      })
  }
  dispatch(getProducts(null, null, null, null, currentPage))
 }, [error, dispatch, currentPage]
  return (
  <Fragment>
      {loading ? <Loader/>:
         <Fragment>
            <MetaData title={'Buy Best Products'} />
            <h1 id="products_heading">Latest Products</h1>
            <section id="products" className="container mt-5">
               <div className="row">
                  { products && products.map(product => (
                     <Product col={4} key={product._id}  product={product}/>
                  ))}
               </div>
         </section>
            {productsCount > 0 && productsCount > resPerPage?
                  <div className="d-flex justify-content-center mt-5">
                 <Pagination
                     activePage={currentPage}
                     onChange={setCurrentPageNo}
                     totalItemsCount={productsCount}
                     itemsCountPerPage={resPerPage}
                     nextPageText={'Next'}
 firstPageText={'Firs
lastPageText={'Last'}
itemClass={'page-item'}
                        linkClass={'page-link'}

            />
               </div> : null }
            </Fragment>
      }
   </Fragment>
    )
    }
```

APP.JS

```
import './App.css';
import Home from './components/Home';
import Footer from './components/layouts/Footer';
import Header from './components/layouts/Header';
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom'
import { HelmetProvider } from 'react-helmet-async'
import { ToastContainer } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import ProductDetail from './components/product/ProductDetail';
import ProductSearch from './components/product/ProductSearch';
import Login from './components/user/Login';
import Register from './components/user/Register';
import { useEffect, useState } from 'react';
import store from './store';
import { loadUser } from './actions/userActions';
import Profile from './components/user/Profile';
import ProtectedRoute from './components/route/ProtectedRoute';
import UpdateProfile from './components/user/UpdateProfile';
import UpdatePassword from './components/user/UpdatePassword';
import ForgotPassword from './components/user/ForgotPassword';
import ResetPassword from './components/user/ResetPassword';
import Cart from './components/cart/Cart';
import Shipping from './components/cart/Shipping';
import ConfirmOrder from './components/cart/ConfirmOrder';
import Payment from './components/cart/Payment';
import axios from 'axios';
import { Elements } from '@stripe/react-stripe-js';
import { loadStripe } from '@stripe/stripe-js';
import OrderSuccess from './components/cart/OrderSuccess';
import UserOrders from './components/order/UserOrders';
import OrderDetail from './components/order/OrderDetail';
import Dashboard from './components/admin/Dashboard';
import ProductList from './components/admin/ProductList';
import NewProduct from './components/admin/NewProduct';
import UpdateProduct from './components/admin/UpdateProduct';
import OrderList from './components/admin/OrderList';
import UpdateOrder from './components/admin/UpdateOrder';
import UserList from './components/admin/UserList';
import UpdateUser from './components/admin/UpdateUser';
import ReviewList from './components/admin/ReviewList';
function App() {
  const [stripeApiKey, setStripeApiKey] = useState("")
  useEffect(() => {
    store.dispatch(loadUser)
    async function getStripeApiKey(){
      const {data} = await axios.get('/api/v1/stripeapi')
      setStripeApiKey(data.stripeApiKey)
    }
    getStripeApiKey()
  },[])
return (
    <Router>
      <div className="App">
        <HelmetProvider>
            <Header/>
                <div className='container container-fluid'>
                  <ToastContainer theme='dark' />
                  <Routes>
                      <Route path='/' element={<Home/>} />
                      <Route path='/search/:keyword' element={<ProductSearch/>} />
                      <Route path='/product/:id' element={<ProductDetail/>} />
                      <Route path='/login' element={<Login/>} />
                      <Route path='/register' element={<Register/>} />
                      <Route path='/myprofile' element={<ProtectedRoute><Profile/></ProtectedRoute> } />
                      <Route path='/myprofile/update'
element={<ProtectedRoute><UpdateProfile/></ProtectedRoute> } />
 <Route path='/myprofile/update/password'
element={<ProtectedRoute><UpdatePassword/></ProtectedRoute> } />
  <Route path='/password/forgot' element={<ForgotPassword/> } />
                      <Route path='/password/reset/:token' element={<ResetPassword/> } />
                      <Route path='/cart' element={<Cart/> } />
                      <Route path='/shipping' element={<ProtectedRoute><Shipping/></ProtectedRoute> } />
 <Route path='/order/confirm' element={<ProtectedRoute><ConfirmOrder/></ProtectedRoute> } />
                      <Route path='/order/success'
element={<ProtectedRoute><OrderSuccess/></ProtectedRoute> } />
                      <Route path='/orders' element={<ProtectedRoute><UserOrders/></ProtectedRoute> } />
<Route path='/order/:id' element={<ProtectedRoute><OrderDetail/></ProtectedRoute> } />
{stripeApiKey && <Route path='/payment' element={<ProtectedRoute><Elements
stripe={loadStripe(stripeApiKey)}><Payment/></Elements></ProtectedRoute> } />
}
 </Routes>
        </div>
        {/* Admin Routes */}
        <Routes>
              <Route path='/admin/dashboard' element={ <ProtectedRoute
 isAdmin={true}><Dashboard/></ProtectedRoute> } />
 <Route path='/admin/products' element={ <ProtectedRoute
 isAdmin={true}><ProductList/></ProtectedRoute> } />
                <Route path='/admin/products/create' element={ <ProtectedRoute
 isAdmin={true}><NewProduct/></ProtectedRoute> } />
                <Route path='/admin/product/:id' element={ <ProtectedRoute
 isAdmin={true}><UpdateProduct/></ProtectedRoute> } />
                <Route path='/admin/orders' element={ <ProtectedRoute
 isAdmin={true}><OrderList/></ProtectedRoute> } />
                <Route path='/admin/order/:id' element={ <ProtectedRoute
 isAdmin={true}><UpdateOrder/></ProtectedRoute> } />
                <Route path='/admin/users' element={ <ProtectedRoute
 isAdmin={true}><UserList/></ProtectedRoute> } />
                <Route path='/admin/user/:id' element={ <ProtectedRoute
 isAdmin={true}><UpdateUser/></ProtectedRoute> } />
                <Route path='/admin/reviews' element={ <ProtectedRoute
 isAdmin={true}><ReviewList/></ProtectedRoute> } />
              </Routes>
           <Footer/>
        </HelmetProvider>
      </div>
    </Router>
  );
 }
 export default App;
```

B. SCREENSHOTS

FRONT PAGE

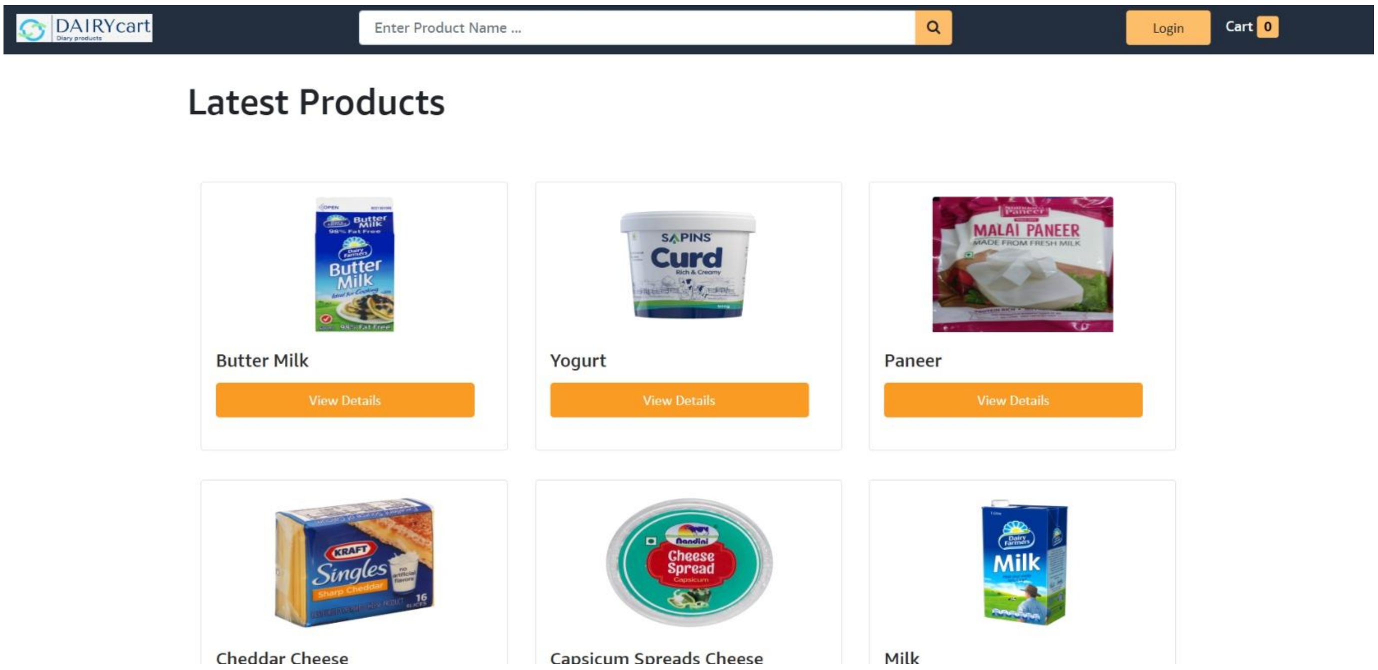
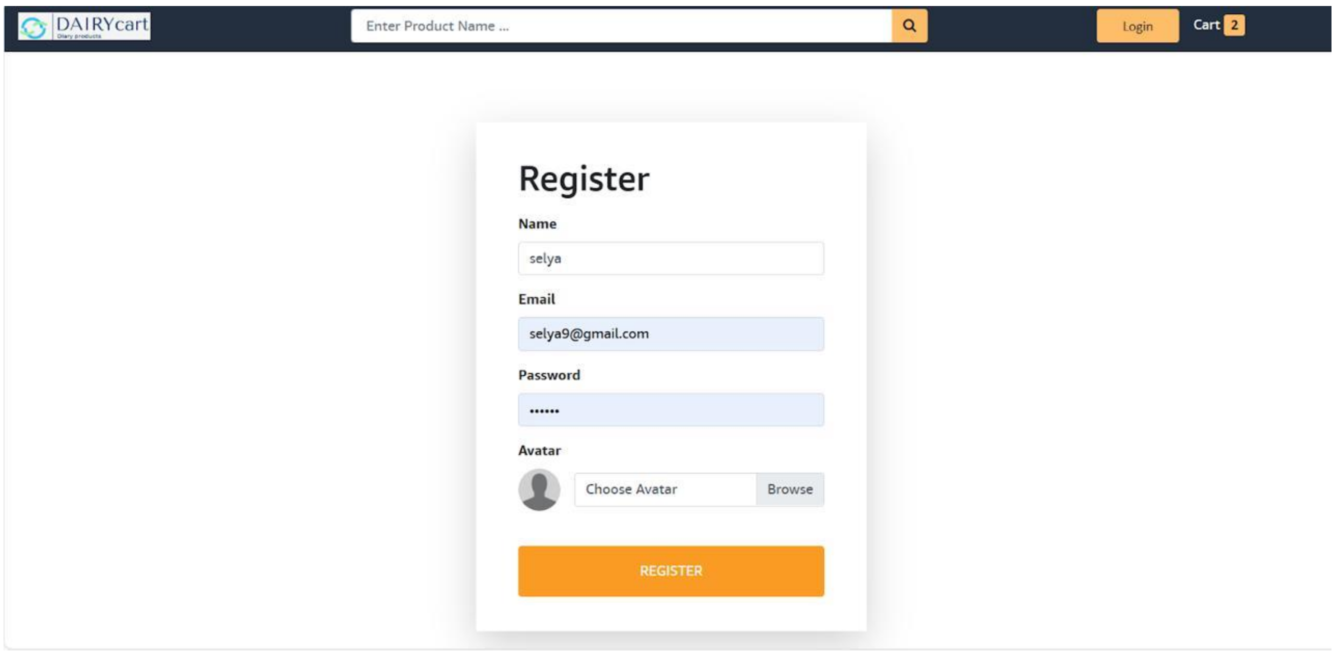


FIGURE B.1 FRONT PAGE OF THE WEBSITE

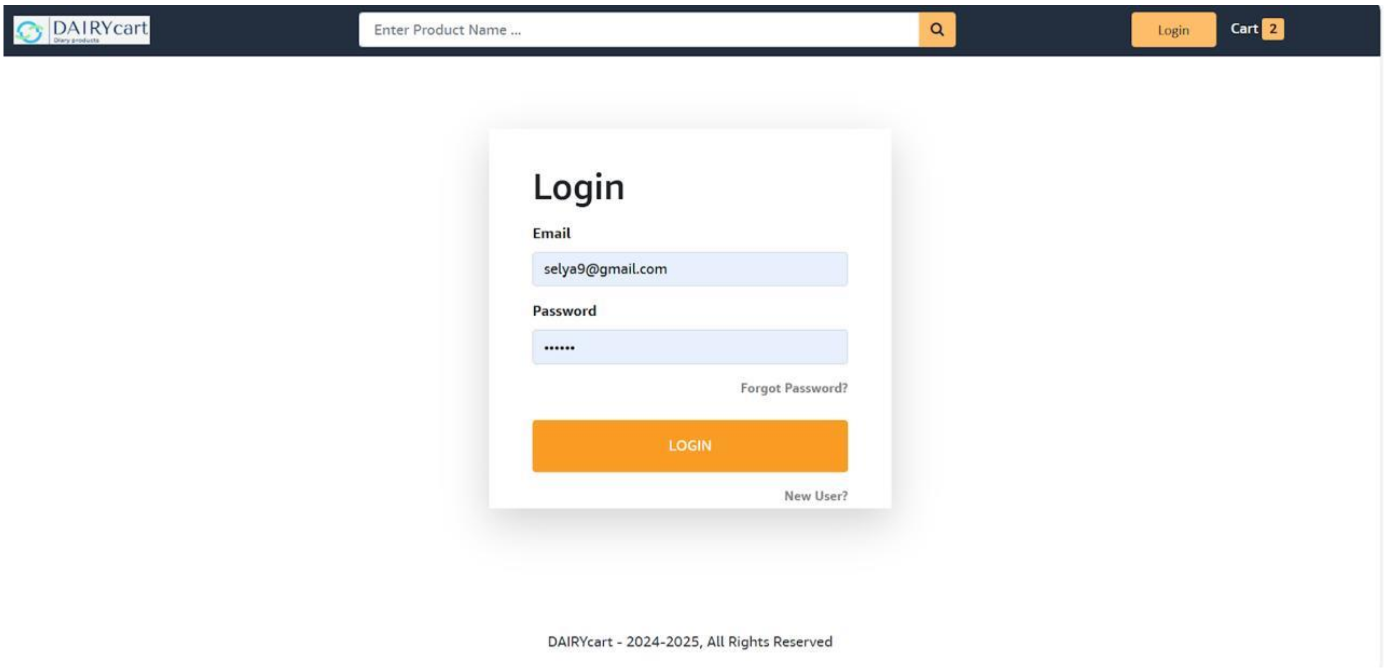
SIGNUP PAGE



Name: selya
Email: selya9@gmail.com
Password: ••••••
Avatar

FIGURE B.2 REGISTER PAGE FOR USER

LOGIN PAGE



Login

Email

selya9@gmail.com

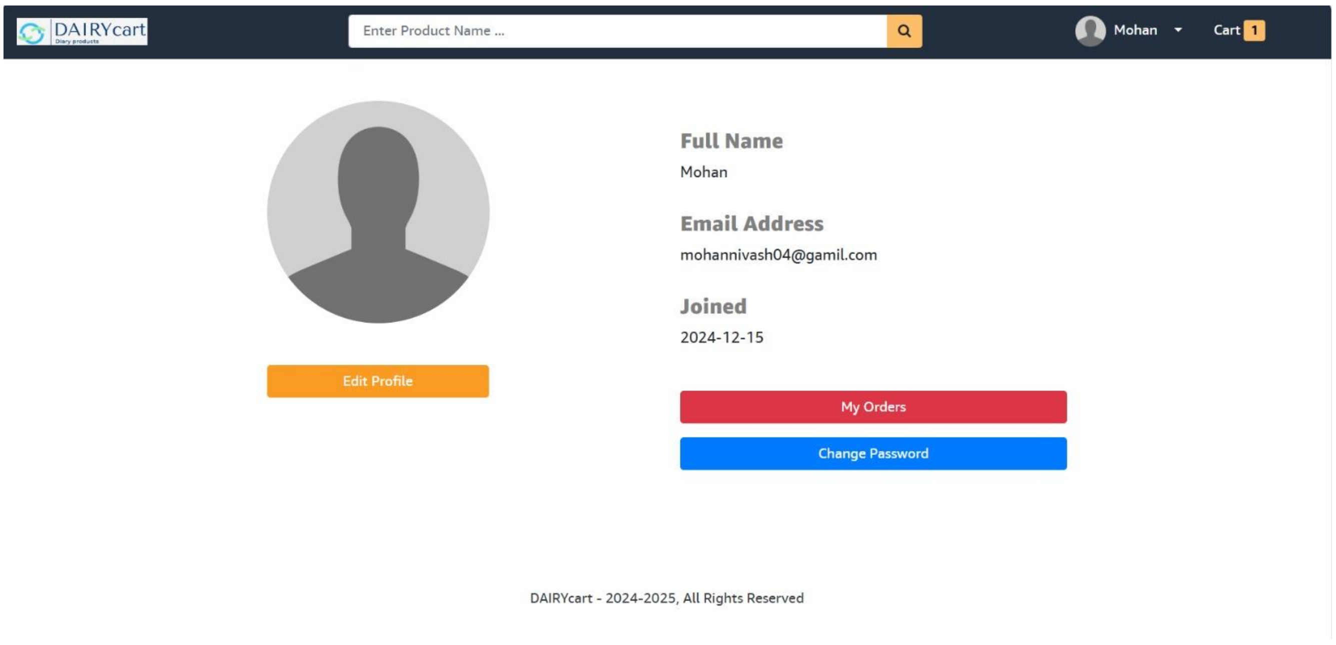
Password

••••••

DAIRYcart - 2024-2025, All Rights Reserved

FIGURE B.3 LOGIN PAGE FOR ADMIN

PROFILE PAGE



Full Name

Mohan

Email Address

mohannivash04@gamil.com

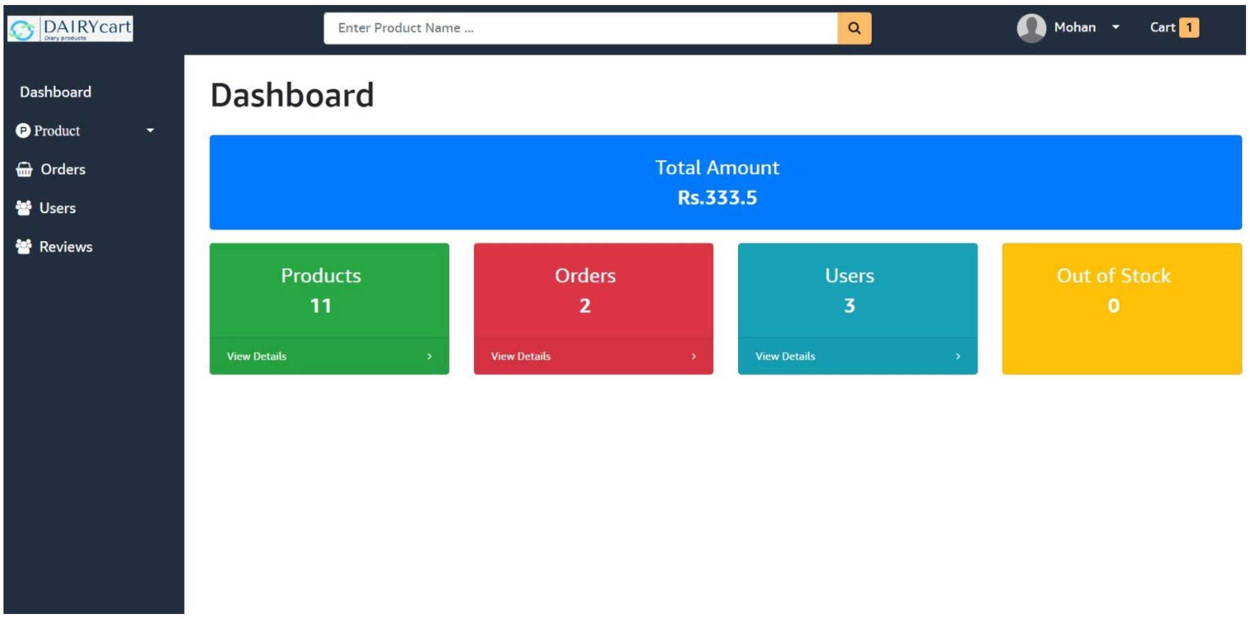
Joined

2024-12-15

DAIRYcart - 2024-2025, All Rights Reserved

FIGURE B.4 PROFILE PAGE FOR USERS

DASHBOARD PAGE



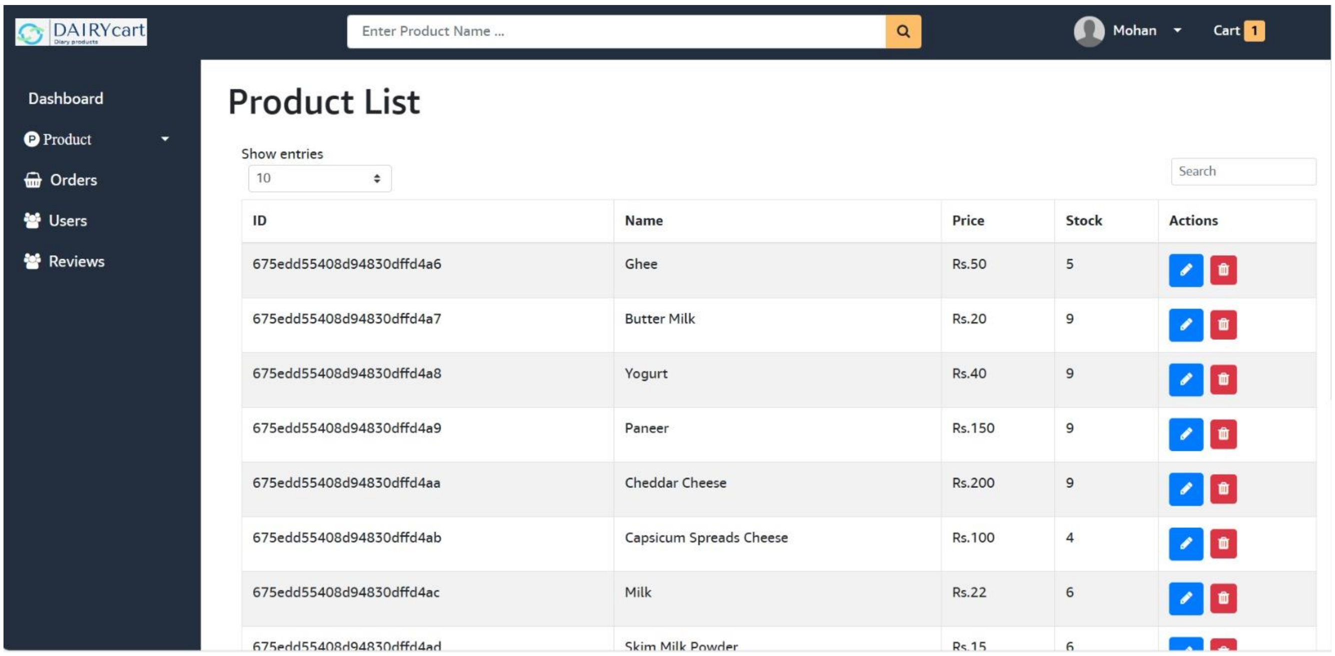
Dashboard

Total Amount
Rs.333.5

Products	Orders	Users	Out of Stock
11	2	3	0

FIGURE B.5 ADMIN DASHBOARD

PRODUCT LIST PAGE

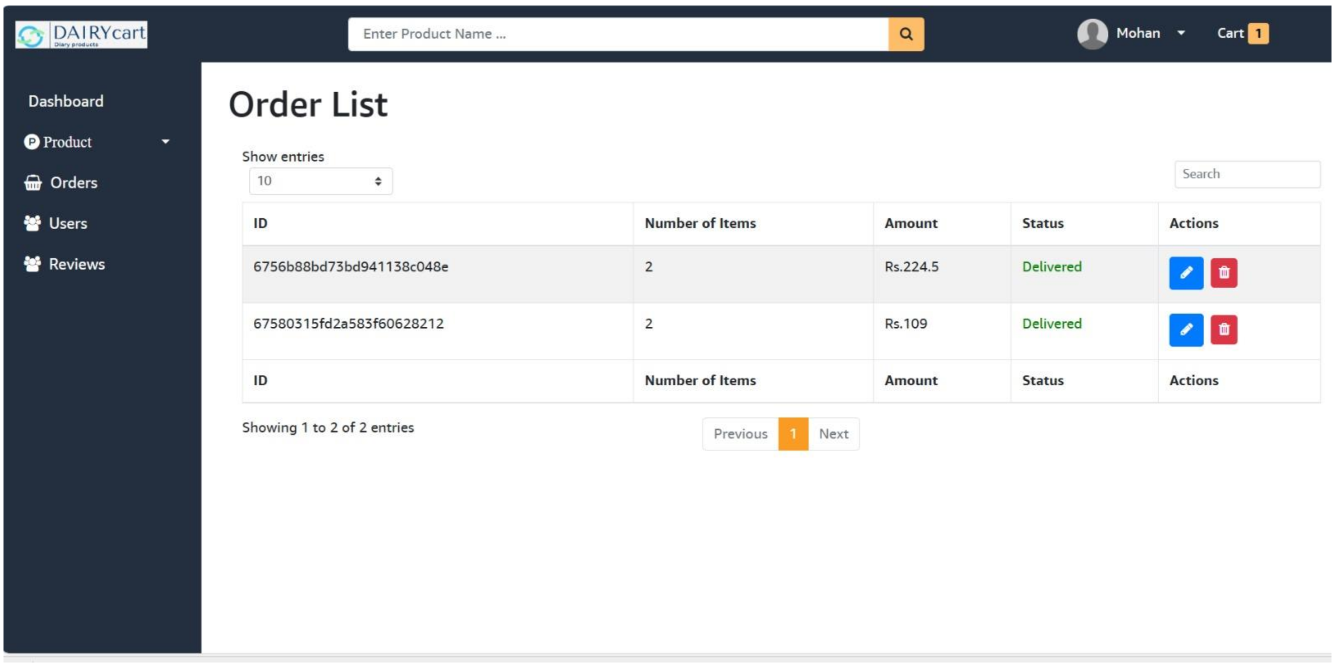


Product List

ID	Name	Price	Stock	Actions
675edd55408d94830dffd4a6	Ghee	Rs.50	5	
675edd55408d94830dffd4a7	Butter Milk	Rs.20	9	
675edd55408d94830dffd4a8	Yogurt	Rs.40	9	
675edd55408d94830dffd4a9	Paneer	Rs.150	9	
675edd55408d94830dffd4aa	Cheddar Cheese	Rs.200	9	
675edd55408d94830dffd4ab	Capsicum Spreads Cheese	Rs.100	4	
675edd55408d94830dffd4ac	Milk	Rs.22	6	
675edd55408d94830dffd4ad	Skim Milk Powder	Rs.15	6	

FIGURE B.6 PRODUCT LIST PAGE

ORDER LIST PAGE



Order List

Show entries 10

Search

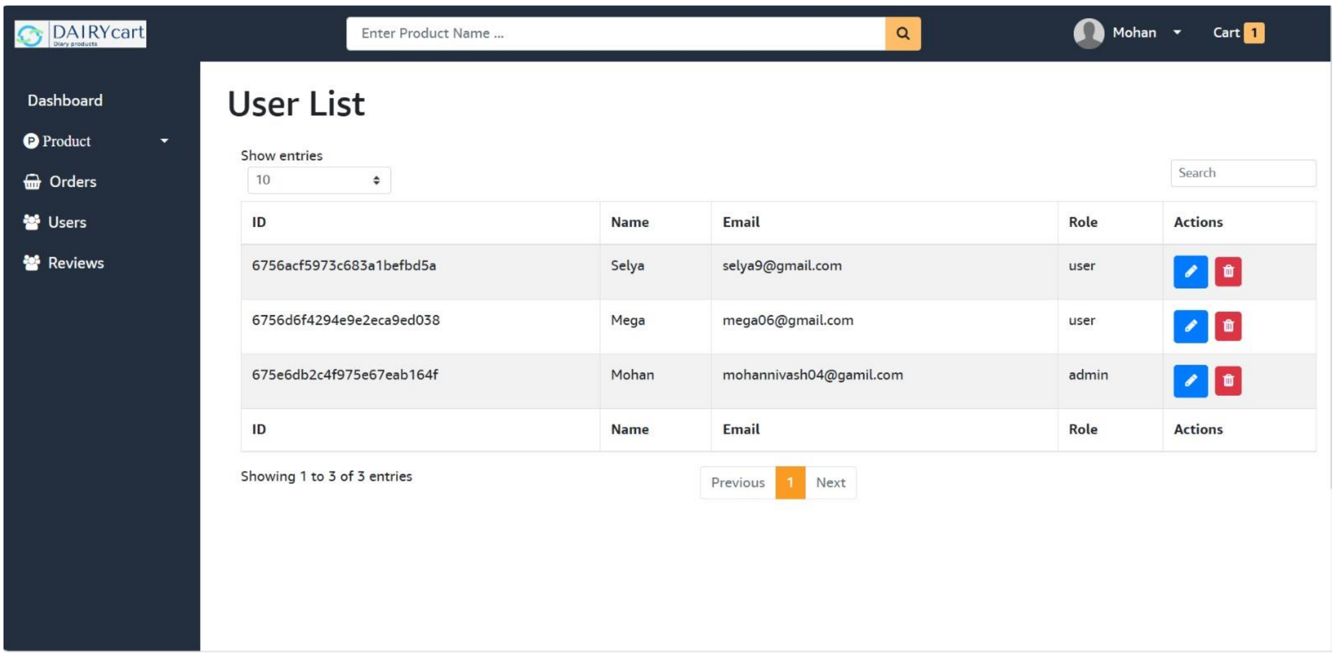
ID	Number of Items	Amount	Status	Actions
6756b88bd73bd941138c048e	2	Rs.224.5	Delivered	
67580315fd2a583f60628212	2	Rs.109	Delivered	
ID	Number of Items	Amount	Status	Actions

Showing 1 to 2 of 2 entries

Previous 1 Next

FIGURE B.7 ORDER LIST PAGE

USER LIST PAGE



User List

Show entries 10

Search

ID	Name	Email	Role	Actions
6756acf5973c683a1befbd5a	Selya	selya9@gmail.com	user	
6756d6f4294e9e2eca9ed038	Mega	mega06@gmail.com	user	
675e6db2c4f975e67eab164f	Mohan	mohannivash04@gamil.com	admin	
ID	Name	Email	Role	Actions

Showing 1 to 3 of 3 entries

Previous 1 Next

FIGURE B.8 USER LIST PAGE

REFERENCES

BOOK REFRENCES :

[1.] Kenneth C. Laudon and Carol Guercio Traver (2024). E-Commerce 2024: Business, Technology,Society.

[2.] Gavin Allanwood, Peter Beare. User Experience Design: A Practical Introduction.

[3.] Mario Casciaro, Luciano Mammino. Node.js Design Patterns.

[4.] Dave Chaffey. E-Business and E-Commerce Management.

[5.] Cal Henderson. Building Scalable Web Sites.

WEB REFERENCES :

[6.] https://www.w3.org/WAI/standards-guidelines/wcag/

[7.] https://www.shopify.com/

[8.] https://aws.amazon.com/personalize/

[9.] https://www.statista.com/

[10.] https://www.deloitte.com/

[11.] https://www.sciencedirect.com/topics/engineering/perishable-products