GLOSSMART WEBSITE FOR DEPARTMENT STORE

A PROJECT REPORT

Submitted by

SIVAKUMAR P

(Reg. No: 24MCR102)

SIVAKUMARAN L

(Reg. No: 24MCR103)

SRI RAGHAVARDHINI M

(Reg. No: 24MCR107)

in partial fulfilment of the requirements

for the award of the degree

of

MASTER OF COMPUTER APPLICATIONS

DEPARTMENT OF COMPUTER APPLICATIONS

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE – 638 052

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 **"GLOSSMART WEBSITE FOR DEPARTMENT STORE"** is the Bonafide record of project work done by **SIVAKUMAR P (Reg. No: 24MCR102), SIVAKUMARAN L (Reg. No: 24MCR103), SRIRAGHAVARDHINI M (Reg. No: 24MCR107)** 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**

(Signature with seal)

Date:

Submitted for end semester viva voce examination held on ________________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We affirm that the project report entitled **"GLOSSMART WEBSITE FOR DEPARTMENT STORE"** 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 candidates.

SIVAKUMAR P (REG. NO: 24MCR102)

SIVAKUMARAN L (REG. NO: 24MCR103)

SRI RAGHAVARDHINI M (REG. NO: 24MCR107)

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

Date: Name and Signature of the Supervisor

[Mr.S.B.KARTHIKEYAN]

ABSTRACT

The Department Glossary Shop website is an innovative e-commerce platform designed to streamline the shopping experience for departmental goods. Built using the MERN stack (MongoDB, Express, React, and Node.js), this full-stack application offers a user-friendly interface and robust backend architecture, catering to diverse categories such as groceries, vegetables, fruits, and stationery. The platform enables users to browse products, manage a shopping cart, and complete secure purchases. By incorporating advanced features like product search, filtering, and order tracking, the website ensures a seamless and personalized shopping journey for its users.

This application prioritizes scalability, performance, and security. The front-end, developed in React, provides a dynamic and responsive user experience, while the back-end, powered by Node.js and Express, handles business logic and API services. MongoDB serves as the database backbone, offering flexibility to manage product inventories, user accounts, and order histories efficiently. Key functionalities such as user authentication, role-based access, and a secure checkout process with multiple payment options enhance user trust and satisfaction.The inclusion of an admin dashboard allows for efficient management of products, discounts, and order statuses.

Designed to adapt to evolving business needs, the platform can scale to accommodate additional features and a larger user base. By leveraging the MERN stack, the application ensures fast load times, easy maintenance, and modular development for future enhancements. Its intuitive interface, combined with robust functionality, makes it a reliable solution for departmental stores seeking to establish or enhance their online presence.The Department Glossary Shop website also emphasizes user engagement and convenience through features like order history review, personalized recommendations, and a loyalty rewards system, which can be integrated in future updates. With its focus on accessibility and inclusivity, the website ensures that users across different devices and regions have an optimized shopping experience. Its modular design allows for easy integration with third-party services, such as delivery partners and payment gateways, further enhancing the operational efficiency of the platform.

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved Correspondent **Thiru.A.K.ILANGO, B.Com, M.B.A, LLB.,** and other philanthropic trust members of Kongu Vellalar Institute of Technology Trust for having provided with necessary resources to complete this project.We are always grateful to our beloved visionary Principal **Dr.V.BALUSAMY, B.E.(Hons), M.Tech, Ph.D.,** and thank him for his motivation and moral support.

We express our deep sense of gratitude and profound thanks to **Dr.A.TAMILARASI, M.Sc, M.Phil, Ph.D, M.Tech.,** Professor and Head of the Department in Computer Applications for her invaluable commitment and guidance for this project.

We also wish to express my gratitude and sincere thanks to our project coordinator **Ms.S.HEMALATHA, MCA.,** Assistant Professor(Sr.G), Department of Computer Applications, Kongu Engineering College, who had motivated us in all aspects for completing the project in the scheduled time.

We would like to express our gratitude and sincere thanks to our project guide **Mr.S.B.KARTHIKEYAN, MCA.,** Assistant professor, Department of Computer Applications, Kongu Engineering College for giving his valuable guidance and suggestions which helped us in the successful completion of the project.

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

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JS	Java Script
JSX	Java Script XML
JSON	JavaScript Object Notation
API	Application Programming Interface
DOM	Document Object Model
JWT	JSON Web Token
NPM	Node Package Manager

CHAPTER 1

INTRODUCTION

1.1 ABOUT THE PROJECT

The Gloss Mart web-based application is designed to revolutionize the online shopping experience, making it simple, efficient, and enjoyable for users. As technology continues to shape our daily lives, the ability to shop from the comfort of our own homes has become an essential part of modern life. Gloss Mart aims to bring a seamless, user-friendly shopping experience that allows customers to browse a wide range of products, make secure purchases, and track their orders—all from the convenience of their device.

The core goal of this project is to develop an intuitive and accessible e-commerce platform, where users can easily search for products, view detailed product information, and place orders with minimal effort. The website will feature an advanced search engine that allows users to find exactly what they need by entering specific terms, while also enabling them to filter results based on categories, price range, ratings, and other attributes. This ensures that the shopping process is tailored to the unique preferences of each user.

Once the user selects a product, it will be added to a virtual shopping cart, from which they can review their selections before proceeding to checkout. After completing the purchase, the user will receive instant confirmation and be able to track the status of their order, ensuring complete transparency. The order information will be sent to the system administrator, who will manage the delivery process and keep users updated on the progress of their purchases.

The Gloss Mart platform aims to provide a flexible and secure online shopping environment. It incorporates modern web security practices to ensure that user data is protected, and features such as customer accounts and order history will streamline the shopping experience. The system is designed to handle a wide range of products, from electronics to clothing to home goods, making it a one-stop shop for all customer needs.

In essence, Gloss Mart is a dynamic and customer-focused e-commerce platform that delivers convenience, security, and an engaging online shopping experience. Whether users are browsing casually or making a specific purchase, this platform makes it easier than ever to shop online with confidence.

1.2 OBJECTIVE

The Gloss Mart web-based platform is designed to serve as a comprehensive solution for buying, selling, and distributing products and services across the internet and other networks. The main objective of this project is to create an accessible and efficient e-commerce platform that enables users to shop online 24/7, overcoming the limitations of traditional brick-and-mortar stores that operate only during business hours.

The platform will provide customers with the ability to compare products from different sellers, ensuring they can make well-informed decisions before purchasing. Each product will be accompanied by a detailed description, including specifications, images, and customer reviews, which will aid users in selecting the best product that suits their needs.

The Gloss Mart system aims to significantly reduce clerical work by automating time-consuming and repetitive tasks such as data entry, order processing, and report generation. This automation will allow the staff to focus on more strategic tasks, boosting overall productivity and efficiency. By streamlining back-end processes, the system will also ensure faster order fulfilment, improving the overall customer experience.

Another key objective is enhancing communication between users and administrators. The system will feature messaging and notification tools that will enable customers to stay updated on their orders, stock availability, and any promotional offers. In addition, users will have access to real-time information on the status of their orders, providing transparency and improving customer satisfaction.

The platform will also facilitate quicker and more efficient report generation, giving businesses valuable insights into critical operational data, such as sales volumes, order trends, and inventory levels. This data will be updated in real-time, empowering administrators to make informed decisions quickly and efficiently.

Overall, Gloss Mart aims to provide a seamless online shopping experience, increase business productivity through automation, and offer real-time analytics for better decision-making. By improving operational efficiency and enhancing user communication, this system will position Gloss Mart as a reliable, convenient, and competitive online shopping platform.

1.3 PROBLEM STATEMENT

Since the shop is associated with the lives of ordinary people and their day-to-day routines, it requires an online shopping website. The manual handling of the record is time-consuming and highly prone to error. To automate, the admin can easily add the products and retrieve customer data. This system makes the complicated process as simple as possible using Structured & Modular techniques & Menu-oriented interface. Even though it cannot claim that this work is entirely exhaustive, the main purpose of this system is to perform shop activities in a computerized way rather than manually, which is time-consuming.

The existing system have lots of defects due to rapid growth of technology. business organizations have switched over from the traditional method of selling products to online method of selling products. Customers can purchase from the comfort of their own homes or work place Selling items in online is to be made easier and convenient for the customer through internet. The internet is being used to make selling products easier and more convenient for customers. It's also simple to reverse the transactions.

Through the duration of selecting, buying and paying for an online product may not take more than 15 minutes, the delivery of the product to customer's doorstep takes about 1-3 days. This frustrates the customer and prevents them from shopping online. Physical designs for products stores allow price negotiations between buyers and the sellers.

1.4 BACKGROUND

The current operational processes of Gloss Mart, a department store, are primarily handled manually, which presents several challenges as the business grows. With an increasing volume of customer orders, maintaining accurate and up-to-date records becomes increasingly difficult, and the risk of human error escalates. As the store expands its offerings and customer base, traditional manual systems struggle to keep up, leading to inefficiencies and data management issues.

1.5 EXISTING SYSTEM

In the existing system of Gloss Mart, all operations are handled manually by the admin. Customers make purchases directly in the store, and the admin manually records each order, manages customer details, and updates the product Catalog. Inventory and order information are tracked manually, and the checkout process is also handled without automation, leading to potential errors and delays. If the store operates online, it features a basic catalog of products with descriptions and prices, but lacks advanced e-commerce functionalities like real-time stock tracking or automated order processing. This manual approach becomes increasingly inefficient and prone to mistakes as the number of customers and orders grows, making it difficult to scale the business effectively.

1.5.1 DRAWBACKS OF EXISTING SYSTEM

The current system at GLOSSmart operates manually, leading to several challenges and inefficiencies:

1. Manual Data Entry: All customer, order, and product information is recorded manually, making the system prone to human errors, inaccuracies, and inconsistencies.
2. Time-Consuming Processes: Tasks such as calculating prices, finalizing bills, and generating reports are done manually, leading to slow processing and delays in order fulfillment and customer service.
3. Inventory Tracking Issues: Inventory management is manual, which makes it difficult to track stock levels in real-time, leading to overselling out-of-stock products and inaccurate stock data.
4. Difficulty in Monitoring Deliveries: The status of delivered products is hard to track, leading to potential confusion and delayed responses to customer inquiries about order status.
5. Lack of Transaction Grouping: Transactions cannot be easily grouped or viewed in a daily or weekly format, making it difficult to analyze sales data or assess business performance efficiently.

1.6 PROPOSED WORK

The proposed system for Glossmart Department Store is designed to enhance business performance, improve customer satisfaction, and boost competitive advantage. The shift to an online platform provides customers with the convenience of shopping from home, offering a wide range of products at competitive prices. Customers will be able to browse and search for products across various categories, making it easy for them to find exactly what they need.

Once customers select products, they can add them to a virtual shopping cart, allowing for easy management of their purchases. The system enhances the overall management of the store by offering greater flexibility, improved security, and better usability, ensuring a seamless shopping experience for both customers and administrators.

The system is equipped with robust features to ensure accurate billing and timely processing of orders. It tracks the billing details, including product prices, taxes, and discounts, to ensure accurate invoicing. The order details are sent directly to the administrator, who can then update the delivery status and monitor the progress of orders. This enables customers to track their orders in real-time, improving transparency and customer trust.

By automating processes like inventory management, order tracking, and billing, the system streamlines operations, reduces human error, and enhances the overall user experience. The proposed system also includes strong security measures to protect customer data, ensuring secure transactions and safe online shopping.

1.6.1 ADVANTAGES OF PROPOSED SYSTEM

- Convenience for Customers
- Wide Product Range
- Improved Efficiency
- Accurate Billing
- Real-Time Order Tracking

- Secure Transactions
- Improved Customer Experience
- Scalability
- Reduced Human Errors
- Efficient Inventory Management
- Faster Order Processing
- Better Business Insights
- Cost-Effective
- 24/7 Accessibility

SUMMARY

Chapter 1 introduces the GlossMart Website for Department Store project, which aims to create an efficient, user-friendly e-commerce platform to enhance the online shopping experience. The project addresses the current system's limitations, which rely heavily on manual processes, leading to errors, inefficiencies, and difficulty in managing orders, inventory, and customer information. The proposed system seeks to automate these tasks, offering benefits like accurate billing, real-time order tracking, secure transactions, and improved efficiency. By streamlining operations and reducing human error, the new system will provide a more convenient shopping experience for customers while enabling better business management and scalability. The overall goal is to improve customer satisfaction, increase productivity, and support the business's growth through automation and enhanced online features.

CHAPTER 2

SYSTEM ANALYSIS

2.1 OBJECTIVE

System analysis is a process of collecting and interpreting facts, identifying the problems and decomposition of a system into its components. System analysis is conducted for the purpose of studying a system or its parts, in order to identify its objectives. It is a problem-solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose. Analysis specifies what the system should do.

2.2 IDENTIFICATION OF NEED

The need for an online department store website like Glossmart arises from the increasing demand for convenient shopping experiences that allow customers to purchase products without leaving the comfort of their homes. With more consumers shifting towards e-commerce, there is a need for an intuitive and user-friendly platform that not only allows customers to browse a wide variety of products but also provides secure and easy online transactions. Customers expect to have access to a broad range of products, such as electronics, clothing, and groceries, with features like filtering, sorting, and detailed product descriptions to facilitate decision-making. The Glossmart website aims to fulfill this need by offering a seamless online shopping experience that caters to these customer demands.

Furthermore, there is a growing expectation for businesses to provide a personalized experience. Customers want to manage their accounts, track their orders, and have access to tailored product recommendations. As a result, a secure user authentication system and easy account management features become essential. The ability to save products in a shopping cart, review purchase history, and track deliveries will not only make the shopping experience more convenient but also increase customer retention. Glossmart addresses this need by integrating features such as a secure login system, order history, and a customer-friendly interface for managing accounts and shopping preferences.

2.3 FEASIBILITY STUDY

A feasibility study is crucial in determining the practicality and efficiency of the proposed system. This study aims to assess whether the proposed Glossmart Department Store Website system is technically, economically, and operationally viable. The system will focus on managing product bookings, transactions, and customer interactions through an online platform, and this analysis will help identify the necessary processing activities and address potential challenges during development.

The study evaluates the following aspects:

- Technical Feasibility
- Operational Feasibility
- Economic Feasibility

2.3.1 Technical Feasibility

Technical feasibility assesses the hardware, software, and resources required to build and run the proposed system. In this case, Glossmart will be a web application developed using the MERN stack (MongoDB, Express.js, React.js, Node.js), which provides a robust platform for building scalable and responsive websites.

Software Requirements: The system will utilize the following technologies:

Frontend: **React.js**, which is ideal for building interactive UIs, and **React Router** for routing.

Backend: **Node.js** with **Express.js**, which handles HTTP requests and manages server-side operations.

Database: **MongoDB** to store customer data, product details, orders, and more.

Authentication: **JWT (JSON Web Tokens)** for secure user authentication.

Hardware Requirements: The system will be hosted on cloud platforms such as **AWS** or **Heroku**, minimizing the need for high-end local hardware. The project will use existing resources for development and testing, such as a local development environment with an **RYZEN5** processor, **8GB RAM**, and **512TB Hard Disk**.

Scalability and Reliability: The system will be designed to handle an increasing number of users and transactions, making use of MongoDB's scalability and cloud hosting solutions. The use of Node.js will ensure the application can handle multiple requests concurrently without performance issues.

2.3.2 Operational Feasibility

Operational feasibility evaluates how practical and efficient the proposed system will be in real-world operations. The goal is to ensure that the **Glossmart** system integrates seamlessly into the organization's operations while meeting the needs of both the business and end-users.

- **System Integration**: The proposed system will integrate with existing business processes, including product management, customer account management, and order fulfillment. The admin panel will allow store managers to easily update product listings, track stock levels, and process customer orders.
- **User Adoption and Impact**: The system's design focuses on ease of use for both customers and administrators. The user interface will be intuitive, allowing customers to quickly browse and purchase products, while administrators will benefit from a streamlined dashboard for managing products and orders.
- **Employee and Management Support**: Since the proposed system will automate several manual tasks such as inventory management and order processing, it will reduce the workload on staff and improve efficiency. Furthermore, since the system is designed to be easy to use, there will be little resistance to its adoption from employees or administrators.
- **Security and Backup**: The system will be built with security in mind, using secure authentication and encryption to protect sensitive customer data. Regular backups will be implemented to ensure data integrity and reliability.

2.3.3 Economic Feasibility

Economic feasibility evaluates whether the project is financially viable, taking into account development costs, ongoing operational costs, and potential revenue generation.

- **Development Costs**: The initial development costs include the cost of designing the website, coding the application, setting up hosting, and integrating payment gateways. These costs will primarily involve the development team and hosting services, which can be estimated based on hourly development rates and subscription fees for cloud services.
- **Operational Costs**: Once the website is live, there will be ongoing costs for hosting, domain registration, and maintaining the database. Cloud hosting on platforms like **AWS** or **Heroku** provides cost flexibility based on usage. Other operational costs include regular updates, bug fixes, and potential system expansions as the business grows.
- **Revenue Generation**: The **Glossmart** website will generate revenue primarily through product sales, with additional income from possible advertising and promotional features. As traffic increases, the site can also implement features like subscription models or exclusive offers for premium users.
- **Return on Investment (ROI)**: The initial ROI is expected to be realized within the first year through increased sales, with significant profit margins anticipated as customer retention increases and operational efficiencies are realized.

2.4 SYSTEM REQUIREMENTS

System requirements are essential for describing the behaviour and features of the **Glossmart Department Store Website** system. These requirements define the software and hardware necessary to ensure the system's functionality and performance.

2.4.1 Software Specification

- **Frontend**: **React.js** — Used for building dynamic user interfaces, providing a smooth and interactive user experience for customers.

- **Backend**: **Node.js** with **Express.js** — A server-side framework that allows the handling of API requests and server-side operations effectively.
- **Database**: **MongoDB** — A NoSQL database ideal for storing flexible and scalable product and customer data.
- **Editor**: **Visual Studio Code** — The code editor used for development, equipped with various extensions and tools to support JavaScript development.

2.4.2 Hardware Specification

- **Processor**: RYZEN 5
- **RAM**: 8GB
- **Hard Disk Capacity**: 512GB
- **Operating System**: Windows 11

The system will operate in a cloud environment for hosting, and the development and testing can be done using mid-range personal computers with the above specifications.

2.5 SOFTWARE DESCRIPTION

This section provides a brief description of the key software technologies used in the **Glossmart** project.

React.js

React.js is a powerful JavaScript library developed by Facebook for building dynamic user interfaces (UIs). It uses a component-based architecture, allowing developers to create reusable UI components. React uses a virtual DOM, making the application faster by only re-rendering parts of the UI that have changed. This improves performance, especially in applications with dynamic content.

- **Advantages**:
  - Fast and efficient rendering with virtual DOM.
  - Component-based architecture promotes reusability and maintainability.

- Supports dynamic and responsive user interfaces.
- Extensive ecosystem with community-driven tools and libraries.

- **Limitations**:
  - Learning curve, especially for developers unfamiliar with JavaScript or SX.
  - Requires additional libraries for full-fledged application features (e.g., routing, state management).

MongoDB

MongoDB is a NoSQL database that stores data in flexible, JSON-like documents, making it ideal for unstructured or semi-structured data. It is widely used in modern web applications due to its scalability, high performance, and seamless integration with technologies like Node.js.

Key Features

- Schema-less: Stores data in dynamic, flexible documents.
- High Performance: Optimized for fast read/write operations.
- Scalable: Supports sharding for horizontal scaling.
- Rich Querying: Advanced query and aggregation capabilities.
- Cross-Platform: Compatible with Windows, Linux, and macOS.
- Geospatial Support: Handles location-based data effectively.

Node.js

Node.js is a runtime environment for executing JavaScript code on the server side. It is built on Chrome's V8 JavaScript engine and allows for asynchronous, non-blocking I/O operations, making it highly efficient for scalable, real-time applications. Node.js is widely used for building fast and scalable backend services.

- **Features**:
  - Event-driven, non-blocking I/O.
  - Extremely fast execution due to V8 engine.
  - Ideal for building scalable and real-time applications like web services and APIs.

SUMMARY

This chapter outlined the feasibility study for the Glossmart Department Store Website, addressing the technical, operational, and economic feasibility. The system's requirements were clearly defined, and the software and hardware specifications were provided. The key technologies like React.js, MySQL, and Node.js were discussed, highlighting their role in building the system. The next chapter will delve into the system design in further detail, providing insight into the structure and functionality of the system.

CHAPTER 3

SYSTEM DESIGN

3.1 OBJECTIVE

This chapter gives a complete overview of the modules that are used in this system. Dataflow and ER diagram are also included as well as a system architecture diagram that illustrates the front end and back-end tools. It also includes a number of designs such as database, table ,input and output design all of which will be thoroughly described.

3.2 OVERVIEW OF THE PROJECT

The following modules are provided in the project:

- ➢ Administrator module
    - Admin Entry
    - Add category
    - Manage Product Details
    - Manage Orders

- ➢ User Module
    - Registration
    - Login
    - View products
    - Add to cart
    - Wishlist
    - Order product

- Order history
- View profile
- Rating & Feedback
- Payment

3.3 MODULE DESCRIPTION

3.3.1 Administrator Entry

The administrator is the super user of this application. Only admin have access into this admin page. Admin may be the owner of the shop. The administrator has all the information about all the users and about all products. The administrator logins to the web site and adds items and stock details. The product details include product name, description, category, price, image and discount. The administrator views all the items, stock, user and order details. The customer registers their details along with customer mail id and password. They enter into the site and view all the item and stock details. They can generate orders for any items with quantity less than the stock quantity.

Add Category

The category section specifies what type of product this is. The product can be added to the appropriate category by the administrator. It assists the user in quickly locating the product. The admin can also create a new category for the product to be added to, as well as add the product to the newly created category. The category is based entirely on the product category, not the brand.

Manage Product Details

- Add Products – The pottery hub contains different kind of products. The products can be classified into different categories by name. Admin can add new products into system with all its details including an image.
- Edit Products – Administrator can edit the products based on the stock of the particular products.
- Delete Products - Administrator can delete the products based on the stock of that products.

Manage Orders

- View Order -Administrator can view the orders which is generated by the users which contains bill number, date, number of items, Total price, status of the order, details of the order.

3.3.2 User Module

User Registration

A new user will have to register in this website by providing essential details in order to view the products in the system.

Login

A user must login with the mail id and password to the website after registration and then they will be directed to home page.

View Products

User can view the list of products based on their needs after successful login. A detailed description of a particular product with product name, products details, product image, price can be viewed by users.

Search Product

Users can search for a particular product in the list by name for the easy purpose.

Add to cart

The user can add the desired product into the cart by clicking add to cart option on the product. They can view the cart by clicking on the cart button. All products added by cart can be viewed in the cart. User can remove an item from the cart by clicking remove.

Wish List

The user will be able to add products to their wish list that they like the most. The user will be able to see the desire list of products that have been added to the wish list, as well as delete them from the wish list.

Order Product

After confirming the items in the cart, the user can submit the cart by providing a delivery address. On successful submitting the cart will become empty.

Order History

User can check the status of the order using my orders. Admin will update the status of the order.

Rating & Feedback

The user will be able to write a review for the product and also rate it on a scale of one to five stars. Users will also provide feedback on the writing commands.

Payment

Before making a payment, the user information is automatically filled by their registration details. The user can pay the cash through Stripe. In the Cash on Delivery the customer makes payment once the good is delivered. The terms and accepted forms of payment depends on payment provisions of a purchase agreement.

3.4 USE CASE DIAGRAM

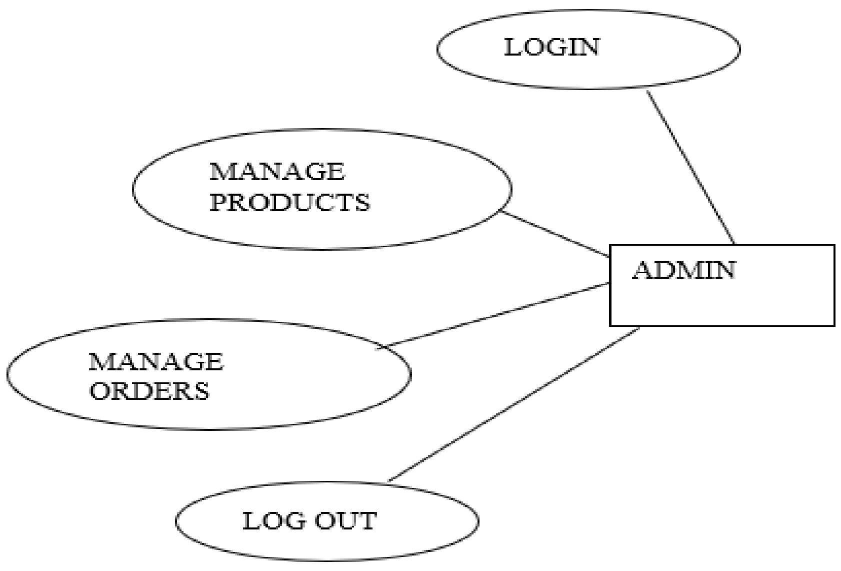


Figure 3.4.1 Admin use case Diagram

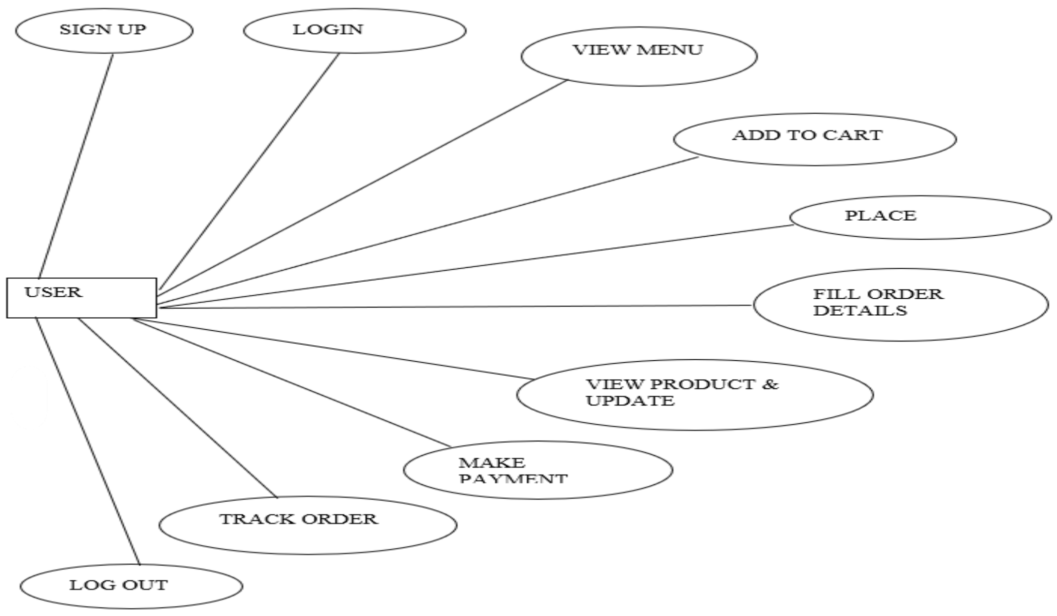


Figure 3.4.2 User use case Diagram

DESCRIPTION:

Use Cases:

- Login : For both Admin and User.
- Register : For Users or Customers to create an account.
- Add Products : For Admin to add items to the system.
- Manage Products : Admin oversees products.
- Manage Orders : Admin processes orders.
- Maintain Stocks : Admin updates inventory levels.
- View Users : Admin can view user profiles.
- Browse Products : Customers or Users browse the catalog.
- View Product : Details of a specific product.
- Add to Cart : Customers add items to their cart.
- Make Order : Customers place an order.
- Update Cart : Customers modify their cart contents.

- View Profile : Users or Customers can view their personal details.
- Logout : Ends the session for all actors.

Relationships:

- Lines connect the actors to the use cases they interact with.
- Admin has more system management-related use cases.
- Customers are mainly focused on shopping activities.

3.5 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically It can be manual, automated, or a combination of both.

LEVEL 0

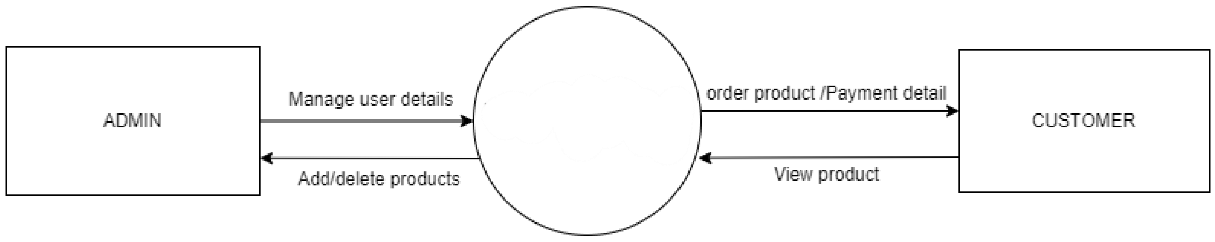


Figure 3.5.1 Dataflow diagram level 0

In Figure 3.5.1 Admin can maintain stock and also manage the user. User can purchase through this website.

LEVEL 1

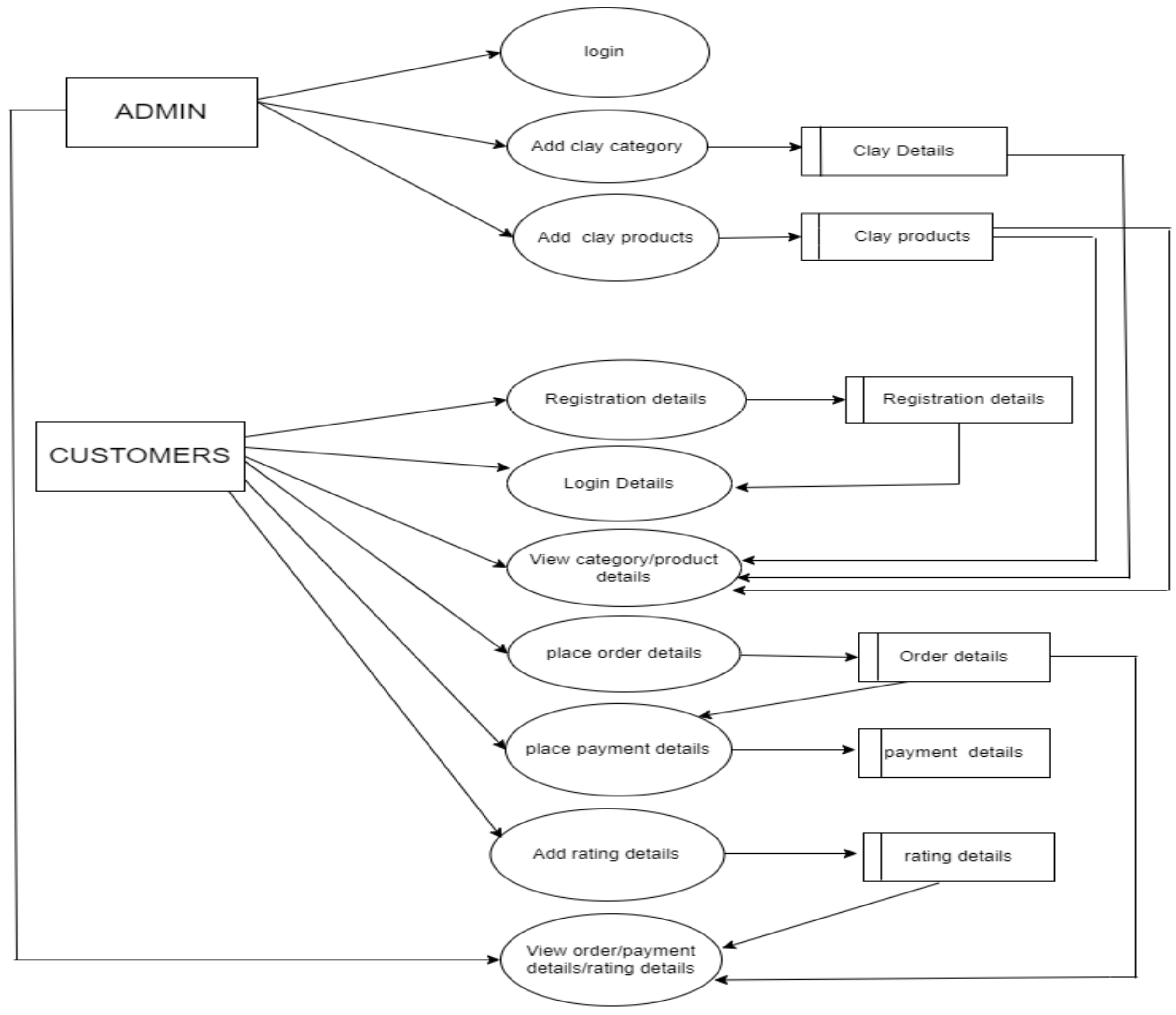


Figure 3.5.2 Dataflow diagram level 1

In Figure 3.5.2 Admin can login and manage the products, category, order, visitors count respectively. User can register, search product, manage account, add the products to cart, place the order and finally checkout the order.

DESCRIPTION:

User flow:

1. Login: The user can log in to the application using their credentials if they have an existing account.
2. Register: If they don't have an account, they can register a new account by providing necessary details.

3. View Product: The user can view the available products.
4. Add to Cart: They can add desired products to their cart.
5. Order Confirmation: After selecting their items, they can confirm the order.
6. Payment Method: The user can choose their preferred payment method is online payment.
7. Order Details: Once the order is placed, the user receives a confirmation with order details.

Admin Flow:

1. Add Product: The admin can add new products to the application.
2. Product Information: The admin manages product details, including information and data.
3. Confirm Order: The admin confirms the order placed by the user.
4. Order Details: The admin receives order details.

3.6 DATABASE DESIGN

Database design in MongoDB involves organizing data into collections of documents structured in a way that ensures efficiency and scalability. Instead of predefined schemas like relational databases, MongoDB uses a flexible, schema-less approach, allowing for dynamic and evolving data structures. The design process involves identifying the necessary data, defining relationships, and structuring the documents and collections to meet the application's requirements.

In MongoDB, relationships are often embedded directly into documents for one-to-many or denormalized structures. For more complex relationships, references between documents in different collections can be used.

Data Integration

In MongoDB, information from various collections can be logically related and accessed seamlessly using embedded documents or references. This approach minimizes redundancy by centralizing related data within a single document or efficiently linking it across collections. The flexible document model allows for logical data centralization while physically distributing data across clusters, ensuring performance and reliability.

Data Independence

MongoDB achieves data independence through its schema-less design, allowing applications to function without being tightly coupled to the underlying data structure. Changes to the document structure or indexes can be made without impacting application logic. This insulation ensures that physical data organization can evolve without requiring reprogramming of dependent applications.

Data Security

MongoDB provides robust security measures to protect data from unauthorized access and corruption:

- **Authentication and Authorization**: Role-based access control ensures that users only have permissions for their specific roles.
- **Encryption**: Data is secured through encryption at rest and in transit using TLS/SSL protocols.
- **Backup and Restore**: MongoDB provides tools to create secure backups and ensure data recovery.
- **Access Control**: Restricted access through IP whitelists, passwords, and tokens prevents unauthorized users from accessing sensitive data.
- **Data Lifecycle Protection**: Encryption, hashing, and other secure key management techniques safeguard data throughout its lifecycle, protecting it against cyber-attacks and breaches.

3.7 TABLE DESIGN

Table 3.7.1 User Collection

Field	**Type**	**Description**
_id	ObjectId	Unique identifier for each user (auto-generated).
name	String	Name of the user.
email	String	Email address of the user (unique).
password	String	Encrypted password for user authentication.
role	String	Role of the user (e.g., 'user' or 'admin').

Stores user details, including authentication and role management.

Table 3.7.2 Product Collection

Field	**Type**	**Description**
_id	ObjectId	Unique identifier for each product (auto-generated).
productName	String	Name of the product.
category	String	Product category (e.g., Electronics, Apparel).
price	Number	Price of the product.
stock	Number	Quantity of the product in stock.
description	String	Description of the product.

Stores details of the products available in the system.

Table 3.7.3 Order Collection

Field	**Type**	**Description**
_id	ObjectId	Unique identifier for each order (auto-generated).
userId	ObjectId	Reference to the user who placed the order.
products	Array	List of products in the order (with quantity and price).
totalAmount	Number	Total cost of the order.
status	String	Status of the order (e.g., Pending, Shipped, Delivered).

Stores information about user orders and associated details.

Table 3.7.4 Cart Collection

Field	Type	Description
_id	ObjectId	Unique identifier for the cart (auto-generated).
userId	ObjectId	Reference to the user who owns the cart.
products	Array	List of products added to the cart (with quantity).
totalPrice	Number	Total price of all items in the cart.
updatedAt	Date	Timestamp of the last cart update.

Stores temporary user-specific cart data before placing an order.

3.8 INPUT DESIGN

Input design is the process of converting user-originated inputs to ap understandable format. Input design is one of the most expensive phases of the operation of 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 system should be user friendly to gain appropriate information to the user. The decisions made during the input design are the project gives the low time consumption to make sensitive application made simple. The coding is being done such that proper validation are made to get the perfect input. No error inputs are accepted.

User Login Form

In the user login form, the customer can login with their registered email id and password to access the portal

User Register Form

In the user register form, the customer should register themselves for further notification about their order details.

Address Form

In the address form, customer should enter their proper address for their shipment process and billing process.

3.9 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 most important direct source of information to the user Output design deals with form design. Efficient output design should improve the interfacing with user.

The term output applies to any information produced by an information system in terms of data displayed. When analyst design system output, they identify the specific output that is needed to meet the requirements of end user. The output is designed in such a way that is attractive, convenient and informative. As the outputs are the most important sources of information to the users, better design should improve the systems relationships with user and also will help in decision-making.

SUMMARY

This chapter contains a detailed description of the module. Login, manage use, product module, user module, and payment module are all mentioned in the module description. Dataflow and ER diagrams. It also includes several designs, such as database table, input, and output designs, which are all covered in detail. The coding will be discussed in Chapter 4 of the document.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 OBJECTIVE

System implementation refers to the process of ensuring that an information system becomes fully functional and ready for use. In the context of the GlossMart department store website, this involved building a new system from the ground up while also integrating elements inspired by existing solutions. The implementation phase enabled users to adopt and evaluate the system effectively, ensuring smooth operation.

4.2 CODE DESCRIPTION

Code descriptions play a crucial role in summarizing the purpose of the code and conveying the programmer's intent. Well-written comments do not merely restate what the code does but provide clarity on why it is implemented in a specific way. These comments can also be leveraged by documentation tools to create external references or integrate seamlessly with other systems and programming tools.

4.3 STANDARDIZATION OF THE CODE

Code standardization is essential for maintaining consistency, readability, and ease of maintenance throughout a project. For the Glossmart department store website, adopting standardized coding practices was crucial to creating a well-organized and efficient codebase. These practices enhanced collaboration, reduced potential errors, and ensured the system was easy to update and scale.

4.4 EXCEPTION HANDLING

Exception handling is a method used to manage unexpected errors or abnormal behavior in the code, ensuring smooth execution of the program. For the Glossmart department store project, exception handling was crucial for managing errors and controlling the flow of the application. While **if-else** blocks were primarily used to handle predictable errors by checking specific conditions, they required manually written logic tailored to each task. However, in many cases, especially during server operations such as database queries or API interactions, errors only became apparent when an operation was attempted.

To address this, the system employed custom error-handling mechanisms to monitor server connections in the background, attempt reconnections, and handle issues like timeouts or authentication failures. **Try-catch** blocks in Node.js, middleware in **Express.js**, and logging tools like **Winston** ensured that both server and client-side errors were caught and managed efficiently. This approach allowed the system to remain resilient, ensuring minimal disruption and a smooth experience for users..

4.5 IMPLEMENTATION

The implementation phase of the Glossmart project involves testing the developed software with sample data and correcting any identified errors. This phase also includes setting up the system fields with sample data, making necessary adjustments to detect and resolve issues, and providing training for end-users. It is the stage where the theoretical design of the system is transformed into a fully functional application. Consequently, it is a critical phase for ensuring the success of the new system and instilling confidence in users that the system will be reliable and effective. Successful implementation requires careful planning, an in-depth analysis of the existing system, consideration of any constraints, and the design of strategies to facilitate the transition. Evaluating various changeover methods is also crucial for a smooth system rollout.

SUMMARY

In this chapter shows explained that the purpose of a code description is to summarise the code or to clarify the programmer's intent. Good comments don't repeat or explain the code. A programming style is defined by coding standards. A coding standard isn't usually concerned with what's proper or bad in a broader sense. Exception handling is a method or process for dealing with and executing anomalous statements in code. In Next Chapter 5 shown about the Testing.

CHAPTER 5

TESTING AND RESULTS

5.1 OBJECTIVE

To design and develop an interactive and user-friendly online shopping platform for departmental store goods, enabling customers to browse, search, and purchase products conveniently. The platform will integrate features such as secure user authentication, real-time inventory updates, efficient product filtering, and seamless checkout processes. Additionally, an admin dashboard will allow for streamlined product management and data analysis, enhancing operational efficiency and customer satisfaction.

5.2 SYSTEM TESTING

Testing is an integral part of any system development life cycle. Insufficient and untested applications may trend to crash and result in loss of economic and manpower investment besides user's dissatisfaction and downfall of reputation, Software testing can be looked upon as one among the many processes, an organization performs, and that provides the last opportunity to correct any flaws in the developed system. Software testing includes selecting test data that have more probability of giving errors

The first step in system testing is to develop a plan that tests all aspects of the system. Completeness, correctness, reliability and maintainability of the software are to be tested for the best quality assurance that the system meets the specification and requirements for its intended use and performance. System testing is the most useful practical process of executing a program with the implicit intention of finding errors that make the program fails.

System testing is done in three phases

- Unit testing
- Integration testing
- Validation testing

5.2.1 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 therequirements of the system. In unit checking out, we have to check the applications makingup the device. For this reason, Unit checking out once in a while referred to as a program checking out. The software program in a device are the modules and workouts whichmight 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.

CASE 1

Module	:	Admin login module
Test type	:	Loading of admin
Input	:	email-id and password
Expected Output	:	Logged in successfully. Admin Page opens

TEST

Username	:	siva@gmail.com
Password	:	siva@123
Output	:	Admin page opens
Analysis	:	email-id and Password has been verified

CASE 2

Module	:	User module
Test type	:	User login
Input	:	email-id and password
Expected Output	:	Main page opens

TEST

Email	:	ragavarthini@gmail.com
Password	:	Ragavarthini@2002
Output	:	Main page opens
Analysis	:	email-id and Password has been verified
Result	:	Pass

5.2.2 Integration Testing

Integration testing is done to test itself 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 used to test the interfaces replaced by that which in generated automatically from the various modules

CASE 1

Module	:	User Login module
Test Type	:	Working of login
Input	:	On clicking respective button
Expected Output	:	Navigation between modules is completed

TEST

Input	:	On clicking login, signup, view service details
Output	:	Respective forms open accordingly
Analysis	:	The expected output is same
Result	:	Pass

CASE 2

Module	:	Admin module
Test Type	:	Add, edit, delete products
Input	:	Navigation for getting reports
Expected Output	:	Reports must be generated

TEST

Input	:	Add, edit, delete products
Output	:	Reports are generated correctly
Analysis	:	The expected output is same
Result	:	Pass

5.2.3 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. The intention of each verification and validation is to make certain that the product is made in step with the necessities of the customer and does certainly fulfil the purpose.

CASE 1

Module	:	User Registration module
Test Type	:	User registration details form
Input	:	Input to all fields
Expected Output	:	fields should be validated

TEST

Input	:	Enter password
Output	:	The password must appear as ****
Analysis	:	The expected output is same
Result	:	Pass

SUMMARY

The preceding chapter delves into various types of testing, such as system testing, unit testing, and user acceptability testing. After completing the source code, during the system evaluation, associated data structures are documented. The final project undergoes testing and validation, encompassing both subtle and overt attempts to uncover problems. Each unit of the system is evaluated, and its functionality is confirmed during unit testing to ensure it performs as expected. This chapter presents testing and its results. System testing entails testing the system as a whole, while end-to-end testing verifies that all scenarios function as intended. As a result, the challenges encountered have been overcome, achieving online shopping for the advertisement product. Chapter 6 outlines the conclusion and future work for the project.

CHAPTER 6

RESULT AND DISCUSSION

6.1 OBJECTIVE

The result and discussion are explained in full in this chapter. This issue includes the use of the application, the benefits of the admin, the result reached, and the technology utilised to achieve it. The way this application will work and the technologies that will be needed to make it work.

6.2 USAGE OF THE APPICATION

A web-based shopping platform has been developed to facilitate the ordering of products and services from a store that serves both in-person and online customers. Currently, all operations within the store are conducted manually, encompassing order processing, payment handling, and customer data management, which leads to inefficiencies in both time and financial resources. Additionally, customers are required to visit the store to purchase disposable items, which further consumes their time and money. To mitigate these issues, a web application has been created for the store, with the owner acting as the administrator. This administrator has the capability to access the web application, allowing for the addition and removal of products from the site, the management of product reviews, and the oversight of customer payment information.

6.3 BENEFIT OF THE WEBSITE

The GlossMart Website presents numerous advantages, such as the convenience for customers to shop at any time and from any location, supported by an intuitive interface that facilitates effortless product navigation and secure payment options. It improves operational efficiency by utilizing real-time inventory management, thereby minimizing manual errors in order fulfilment. For store administrators, the application offers robust tools for managing products, tracking sales, and engaging with customers, which fosters enhanced decision-making and superior service quality. This platform not only conserves time but also enhances customer satisfaction and promotes business expansion.

6.4 RESULT ACHIEVED

To address the shortcomings of conventional manual systems, the "Gloss Mart Website" has been developed. This platform is designed to eliminate or significantly reduce the challenges associated with manual shopping processes. It is specifically crafted to cater to the unique requirements of departmental stores, facilitating seamless and efficient operations.

The system prioritizes user-friendliness to minimize data entry mistakes and provides alerts for any invalid data input. It is accessible to all users, as it does not necessitate formal expertise. As an e-commerce platform, it guarantees secure, dependable, and swift management of product browsing, order processing, and payment transactions. This system empowers organizations to optimize their resource utilization effectively. Every business, irrespective of its size, encounters difficulties in managing data related to categories, products, orders, payments, and customer confirmations. The Gloss Mart Website effectively tackles these issues by providing a customizable solution that aligns with specific managerial needs. This approach aids in strategic planning and ensures that the organization is equipped with the necessary tools to fulfil its long-term objectives.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

In conclusion, the Gloss Mart Website offers a contemporary, effective, and trustworthy alternative to the difficulties encountered by conventional manual systems in department stores. By optimizing processes, minimizing mistakes, and improving user satisfaction, it guarantees a secure and seamless shopping experience for customers, while equipping administrators with powerful management resources. This platform not only enhances operational efficiency but also fosters strategic development, rendering it an essential resource for businesses seeking to attain sustained success in the competitive retail environment.

7.2 FUTURE SCOPE

- **Enhanced User Experience**: Implement AI-driven personalized recommendations, advanced search filters, offline mode, app trials, dark mode, multilingual support, and parental controls to make the platform more inclusive and user-friendly.
- **Improved Developer Interaction and Analytics**: Provide integrated feedback systems, detailed analytics dashboards, and seamless update mechanisms to foster better communication and insights for developers.
- **Increased Engagement and Security**: Introduce gamification rewards, subscription management, social media integration, stricter app security measures, and wearable device compatibility to boost engagement and ensure a safe and innovative user environment.

APPENDICES

A.SAMPLE CODE

Home.jsx

```
import React, { useState } from 'react'
import './Home.css'
import Header from '../../Component/Header/Header'
import ExploreMenu from '../../Component/ExploreMenu/ExploreMenu'
import FoodDisplay from '../../Component/FoodDisplay/FoodDisplay'
import AppDownload from '../../Component/AppDownload/AppDownload'
const Home = () => {
  const [category,setCategory] = useState("All");
  return (
    <div>
        <Header/>
        <ExploreMenu category={category} setCategory={setCategory}/>
        <FoodDisplay category={category}/>
        <AppDownload/>
    </div>
  )
}
export default Home
```

Cart.jsx

```
import React, { useContext } from "react";
import "./Cart.css";
import { StoreContext } from "../../context/StoreContext";
import { useNavigate } from "react-router-dom";
const Cart = () => {
    const { cartItems, food_list, removeFromCart, getTotalCartAmount,url } =
useContext(StoreContext);
  const navigate = useNavigate();
  return (
    <div className="cart">
      <div className="cart-items">
        <div className="cart-items-title">
          <p>Items</p>
          <p>Title</p>
          <p>Price</p>
          <p>Quantity</p>
          <p>Total</p>
          <p>Remove</p>
        </div>
        <br />
        <hr />
        {food_list.map((item, index) => {
          if (cartItems[item._id] > 0) {
```

```
      return (
        <div>
          <div className="cart-items-title cart-items-item">
            <img src={url+"/images/"+item.image} alt="" />
            <p>{item.name}</p>
            <p>${item.price}</p>
            <p>{cartItems[item._id]}</p>
            <p>${item.price * cartItems[item._id]}</p>
            <p onClick={()=>removeFromCart(item._id)} className="cross">x</p>
          </div>
          <hr />
        </div>
      );
    }
  })}
</div>
<div className="cart-bottom">
  <div className="cart-total">
    <h2>Cart Totals</h2>
    <div>
      <div className="cart-total-details">
        <p>Subtotal</p>
        <p>${getTotalCartAmount()}</p>
      </div>
```

```
        <hr />
        <div className="cart-total-details">
          <p>Delivery fee</p>
          <p>${getTotalCartAmount()===0?0:2}</p>
        </div>
        <hr />
        <div className="cart-total-details">
          <b>Total</b>
          <b>${getTotalCartAmount()===0?0:getTotalCartAmount()+2}</b>
        </div>
      </div>
      <button onClick={()=>navigate('/order')}>PRROCEED TO CHECKOUT</button>
    </div>
    <div className="cart-promocode">
      <div>
        <p>If you have a promo code, Enter it here</p>
        <div className="cart-promocode-input">
          <input type="text" placeholder="promo code" />
          <button>Submit</button>
        </div>
      </div>
    </div>
  </div>
</div>
```

```
  );
};
export default Cart;
```

MyOrders.jsx

```
import React, { useContext, useEffect, useState } from 'react'
import './MyOrders.css'
import { StoreContext } from '../../context/StoreContext';
import axios from 'axios';
import { assets } from '../../assets/assets';
const MyOrders = () => {
    const {url,token} = useContext(StoreContext);
    const [data,setData] = useState([]);
    const fetchOrders = async () => {
        const response = await axios.post(url+"/api/order/userorders",{},{headers:{token}});
        setData(response.data.data);
        console.log(response.data.data);
    }
    useEffect(() => {
        if (token) {
            fetchOrders();
        }
    },[token])
  return (
    <div className="my-orders">
```

```
      <h2>My Orders</h2>
      <div className="container">
        {data.map((order,index) => {
          return(
           <div key={index} className="my-orders-order">
              <img src={assets.parcel_icon} alt="not" />
              <p>{order.items.map((item,index) => {
                if(index === order.items.length-1) {
                  return item.name+" x "+item.quantity
                }
                else{
                  return item.name+" x "+item.quantity+", "
                }
              })}</p>
              <p>Rs: {order.amount}.00</p>
              <p>Items: {order.items.length}</p>
              <p><span>&#x25cf;</span><b>{order.status}</b></p>
              <button onClick={fetchOrders}>Track Order</button>
           </div>
          )
        })}
      </div>
    </div>
)}export default MyOrders
```

PlaceOrder.jsx

```
import React, { useContext, useEffect, useState } from 'react'
import './PlaceOrder.css'
import { StoreContext } from '../../context/StoreContext'
import axios from 'axios';
import { Navigate, useNavigate } from 'react-router-dom';
const PlaceOrder = () => {
  const {getTotalCartAmount,token,food_list,cartItems,url} = useContext(StoreContext);
  const [data,setData] = useState({
    firstName:"",
    lastName:"",
    email:"",
    street:"",
    city:"",
    state:"",
    zipcode:"",
    country:"",
    phone:""
  })
    const onChangeHandler = (event) => {
      const name = event.target.name;
      const value = event.target.value;
      setData(data=>({...data,[name]:value}))
```

```
}
const placeOrder = async(event) => {
  event.preventDefault();
  let orderItems = [];
  food_list.map((item) => {
    if (cartItems[item._id]>0) {
      let itemInfo = item;
      itemInfo["quantity"] = cartItems[item._id];
      orderItems.push(itemInfo)
    }
  })
  let orderData = {
    address:data,
    items:orderItems,
    amount:getTotalCartAmount()+2,
  }
  let response = await axios.post(url+"/api/order/place",orderData,{headers:{token}})
  if (response.data.success) {
    const {session_url} = response.data;
    window.location.replace(session_url);
  }
  else {
    alert("Error");
  }
  }
  const navigate = useNavigate();
  useEffect(() => {
    if (!token) {
      navigate('/cart')
    }
    else if(getTotalCartAmount()===0){
      navigate('/cart')
    }
  },[token])
 return (
  <form onSubmit={placeOrder} className="place-order">
     <div className="place-order-left">
        <p className="title">Delivery Information</p>
        <div className="multi-fields">
                    <input required name='firstName' onChange={onChangeHandler}
value={data.firstName} type="text" placeholder='First name'/>
                    <input required name='lastName' onChange={onChangeHandler}
value={data.lastName} type="text" placeholder='Last name'/>
        </div>
          <input required name='email' onChange={onChangeHandler} value={data.email}
type="email" placeholder='Email address'/>
          <input required name='street' onChange={onChangeHandler} value={data.street}
type="text" placeholder='Street'/>
```

```
          <div className="multi-fields">
              <input required name='city' onChange={onChangeHandler} value={data.city} type="text" placeholder='City'/>
          <div className="multi-fields">
                      <input required name='zipcode' onChange={onChangeHandler} value={data.zipcode} type="text" placeholder='Zip code'/>
                      <input required name='country' onChange={onChangeHandler} value={data.country} type="text" placeholder='Country'/>
          </div>
           <input required name='phone' onChange={onChangeHandler} value={data.phone} type="text" placeholder='Phone' />
        </div>
        <div className="place-order-right">
        <div className="cart-total">
          <h2>Cart Totals</h2>
          <div>
          <div className="cart-total-details">
             <p>Subtotal</p>
             <p>${getTotalCartAmount()}</p>
           </div>
           <hr />
           <div className="cart-total-details">
             <p>Delivery fee</p>
             <p>${getTotalCartAmount()===0?0:2}</p>
           </div>
```

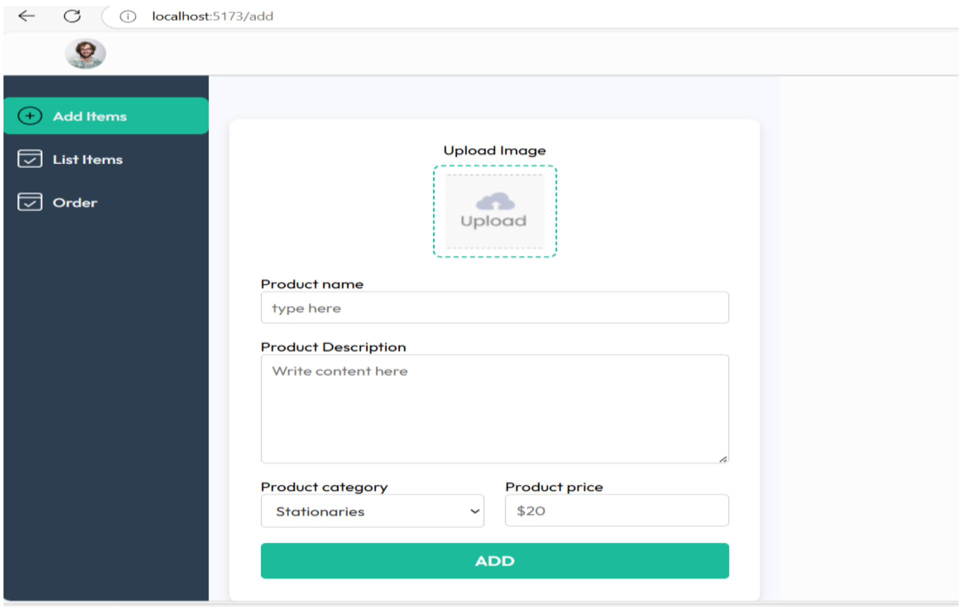
```
            <hr />
            <div className="cart-total-details">
              <b>Total</b>
              <b>${getTotalCartAmount()===0?0:getTotalCartAmount()+2}</b>
            </div>
          </div>
          <button  type='submit'>PURCHASE </button>
        </div>
       </div>
    </form>
  )
}
export default PlaceOrder
```

App.jsx

```
import React, { useState } from "react";
import { Route, Routes } from "react-router-dom";
import Home from "./pages/Home/Home";
import Cart from "./pages/Cart/Cart";
import Footer from "./Component/Footer/Footer";
import LoginPopup from "./Component/LoginPopup/LoginPopup";
import Verify from "./pages/Verify/Verify";
import MyOrders from "./pages/MyOrders/MyOrders";
const App = () => {
  const[showLogin,setShowLogin] = useState(false)
  return (
    <>
    {showLogin?<LoginPopup setShowLogin={setShowLogin}/>:<></>}
      <div className="app">
        <NavBar setShowLogin={setShowLogin} />
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/cart" element={<Cart />} />
          <Route path="/order" element={<PlaceOrder />} />
          <Route path="/verify" element={<Verify/>}/>
          <Route path="/myorders" element={<MyOrders/>}/>
        </Routes>
      </div>
      <Footer />
    </>
  );
};
export default App;
import React from 'react'
import Navbar from './components/Navbar/Navbar'
import Sidebar from './components/Sidebar/Sidebar'
import { Route, Routes } from 'react-router-dom'
import Add from './pages/Add/Add'
import List from './pages/List/List'

import Orders from './pages/Orders/Orders'

import { ToastContainer} from 'react-toastify';

import 'react-toastify/dist/ReactToastify.css';

const App = () => {

  const url = "http://localhost:4000"

    return (

    <div>

      <ToastContainer/>

      <Navbar/>

      <hr/>

      <div className="app-content">

        <Sidebar/>

        <Routes>

          <Route path="/add" element={<Add url={url}/>}/>

          <Route path="/list" element={<List url={url}/>}/>

          <Route path="/orders" element={<Orders url={url}/>}/>

        </Routes>

      </div>

    </div>

  )

}

export default App
```

APPENDIX-B SCREENSHOTS

ADMIN DASHBOARD



Product name: type here

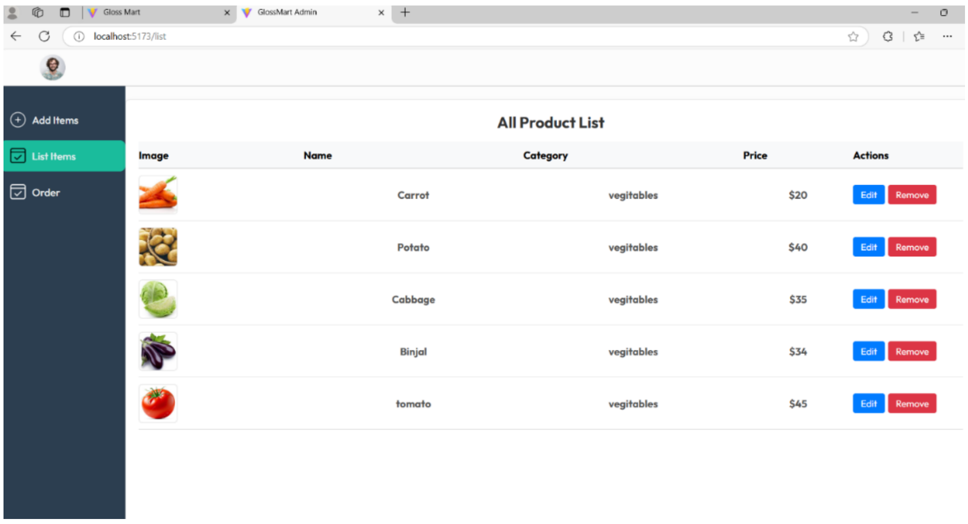
Product Description: Write content here

Product category: Stationaries

Product price: $20

FigureB.1 Admin Add Items

About page, Admin dashboard for adding the cart items to the database.

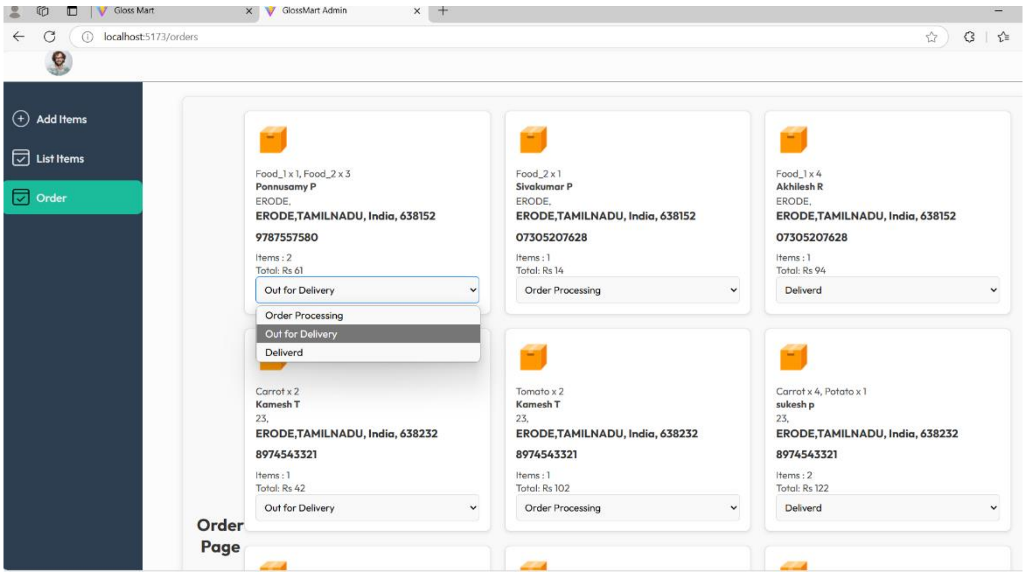


All Product List

Image	Name	Category	Price	Actions
	Carrot	vegitables	$20	Edit Remove
	Potato	vegitables	$40	Edit Remove
	Cabbage	vegitables	$35	Edit Remove
	Binjal	vegitables	$34	Edit Remove
	tomato	vegitables	$45	Edit Remove

FigureB.2 Admin List Items

About page, Admin dashboard for listing the cart items to the user on the daily basis we can update the products.



Food_1 x 1, Food_2 x 3
Ponnusamy P
ERODE,
ERODE,TAMILNADU, India, 638152
9787557580
Items : 2
Total: Rs 61
Out for Delivery

Order Processing
Out for Delivery
Deliverd

Food_2 x 1
Sivakumar P
ERODE,
ERODE,TAMILNADU, India, 638152
07305207628
Items : 1
Total: Rs 14
Order Processing

Food_1 x 4
Akhilesh R
ERODE,
ERODE,TAMILNADU, India, 638152
07305207628
Items : 1
Total: Rs 94
Deliverd

Carrot x 2
Kamesh T
23,
ERODE,TAMILNADU, India, 638232
8974543321
Items : 1
Total: Rs 42
Out for Delivery

Tomato x 2
Kamesh T
23,
ERODE,TAMILNADU, India, 638232
8974543321
Items : 1
Total: Rs 102
Order Processing

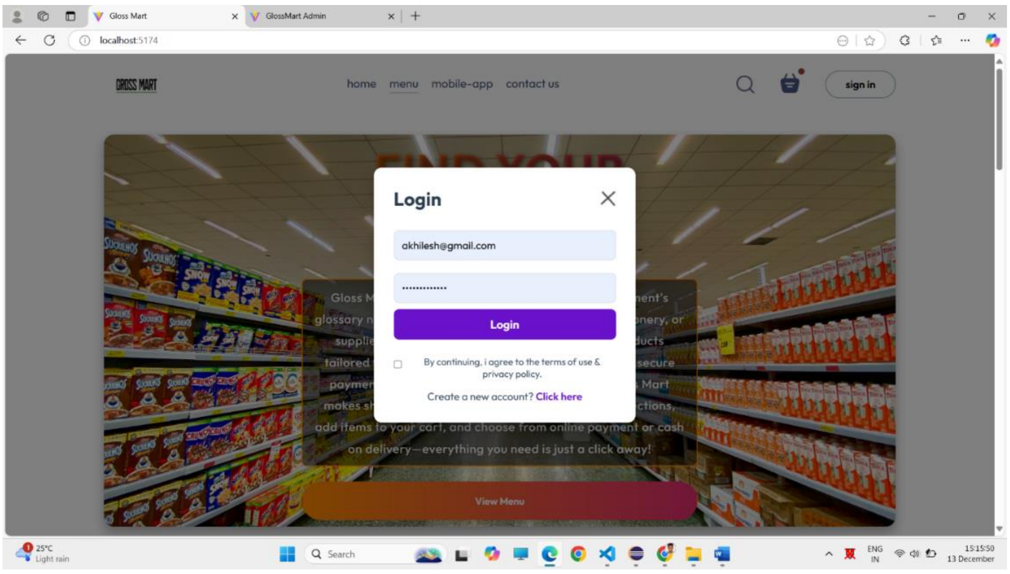
Carrot x 4, Potato x 1
sukesh p
23,
ERODE,TAMILNADU, India, 638232
8974543321
Items : 2
Total: Rs 122
Deliverd

Order Page

Figure B.3 Admin Manage Orders

About page,displays details of a user's order, including the items purchased, their quantities, prices, and the total cost. It might also show order status, such as "processing" "out for delivery" or "delivered," and could include tracking information. This visual summary helps users quickly understand their order details and track its progress.

USER SIGN-UP PAGE



Login

akhilesh@gmail.com

••••••••••••••

Login

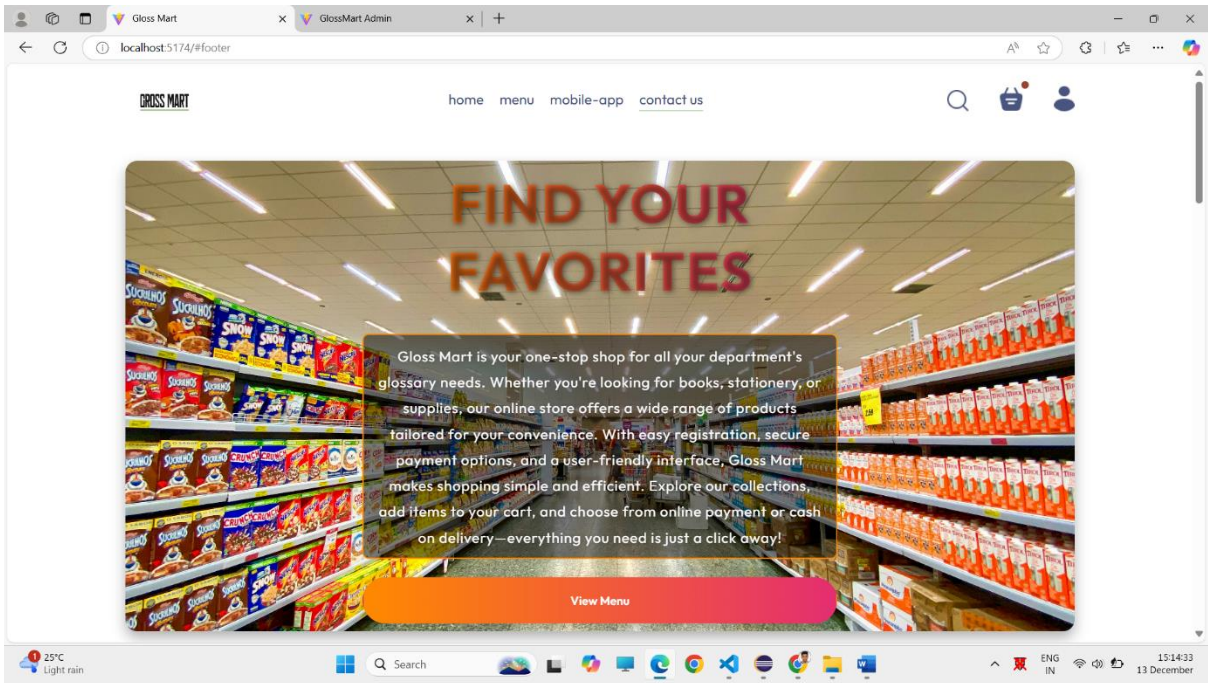
By continuing, i agree to the terms of use & privacy policy.

Create a new account? Click here

Figure B.4 User Login Page

The user login allows customers to securely access their account by entering their registered email and password. The sign-up process enables new users to create an account by providing essential details like name, email, and password for future login access.

HOME PAGE

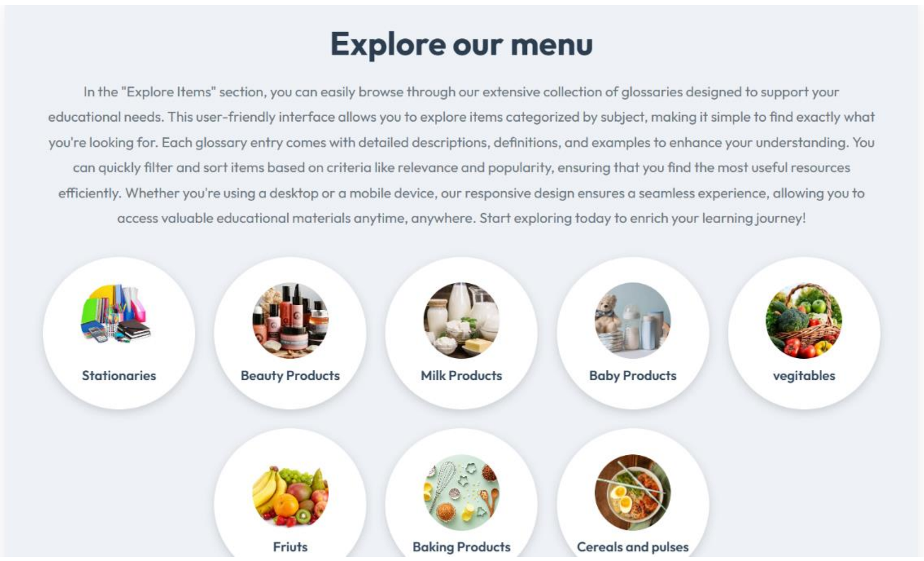


Gloss Mart is your one-stop shop for all your department's glossary needs. Whether you're looking for books, stationery, or supplies, our online store offers a wide range of products tailored for your convenience. With easy registration, secure payment options, and a user-friendly interface, Gloss Mart makes shopping simple and efficient. Explore our collections, add items to your cart, and choose from online payment or cash on delivery—everything you need is just a click away!

Figure B.5 User Home Page

The home page serves as the main entry point of the application, showcasing featured products, categories, and promotions. It provides users with easy navigation to explore items, view their cart, and access various sections of the site.

EXPLORE PRODUCT



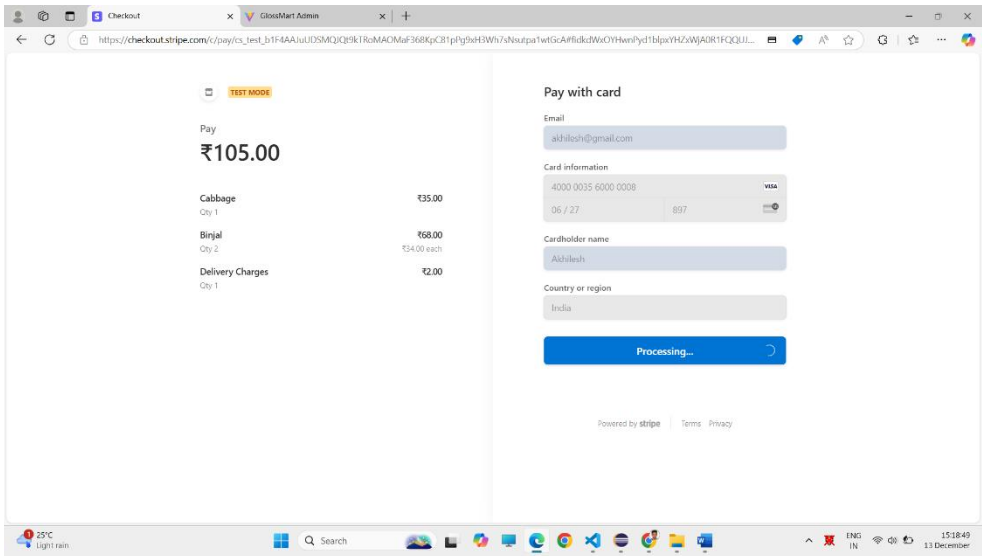
Explore our menu

In the "Explore Items" section, you can easily browse through our extensive collection of glossaries designed to support your educational needs. This user-friendly interface allows you to explore items categorized by subject, making it simple to find exactly what you're looking for. Each glossary entry comes with detailed descriptions, definitions, and examples to enhance your understanding. You can quickly filter and sort items based on criteria like relevance and popularity, ensuring that you find the most useful resources efficiently. Whether you're using a desktop or a mobile device, our responsive design ensures a seamless experience, allowing you to access valuable educational materials anytime, anywhere. Start exploring today to enrich your learning journey!

Figure B.6 Explore Product Page

The "Explore Products" section allows users to browse through various product categories, view detailed descriptions, and filter items based on preferences like price, ratings, and brand. It provides an interactive and seamless shopping experience to help users discover products easily.

PAYMENT PAGE



Checkout
GlossMart Admin
https://checkout.stripe.com/c/pay/cs_test_b1F4AAJuUDSMQJQt9kTRoMAOMaF368KpC81pPg9xH3Wh7sNsutpa1wtGcA#fidkdWxOYHwnPyd1blpxYHZxWjA0R1FQQUJ...

TEST MODE

Pay

₹105.00

Cabbage Qty 1	₹35.00
Binjal Qty 2	₹68.00 ₹34.00 each
Delivery Charges Qty 1	₹2.00

Pay with card

Email: akhilesh@gmail.com

Card information: 4000 0035 6000 0008 VISA, 06 / 27, 897

Cardholder name: Akhilesh

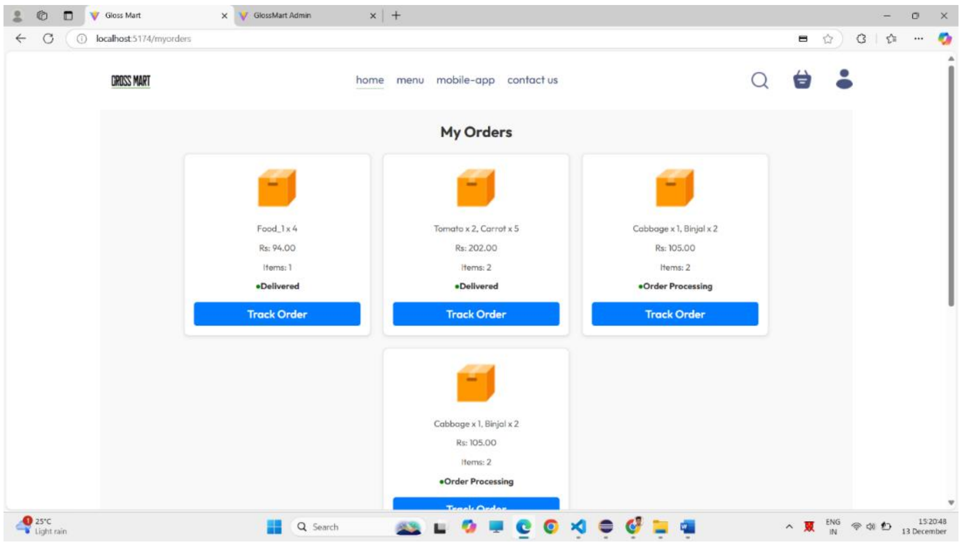
Country or region: India

Processing...
Powered by stripe
Terms
Privacy
25°C
Light rain
Search
ENG
IN
15:18:49
13 December

Figure B.7 Payment Page

The payment page using Stripe allows users to securely enter their payment information, such as credit card details, to complete their purchase. It provides a seamless checkout experience, including options for billing address entry and payment confirmation, ensuring secure transactions with Stripe's encryption and fraud protection features.

ORDER PAGE



My Orders

Food_1 x 4
Rs: 94.00
Items: 1
•Delivered

Tomato x 2, Carrot x 5
Rs: 202.00
Items: 2
•Delivered

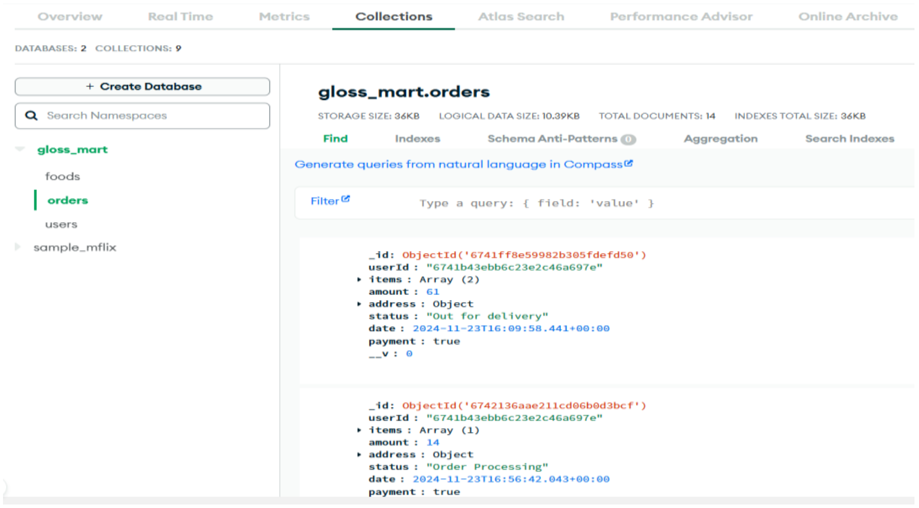
Cabbage x 1, Binjal x 2
Rs: 105.00
Items: 2
•Order Processing

Cabbage x 1, Binjal x 2
Rs: 105.00
Items: 2
•Order Processing

Figure B.8 Order Page

The order page displays a summary of the user's purchase, including product details, quantities, prices, and the total amount. It also shows the order status, delivery address, and tracking information, allowing users to monitor the progress of their order.

BACKEND ORDER COLLECTION



DATABASES: 2 COLLECTIONS: 9

gloss_mart.orders

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 10.39KB TOTAL DOCUMENTS: 14 INDEXES TOTAL SIZE: 36KB

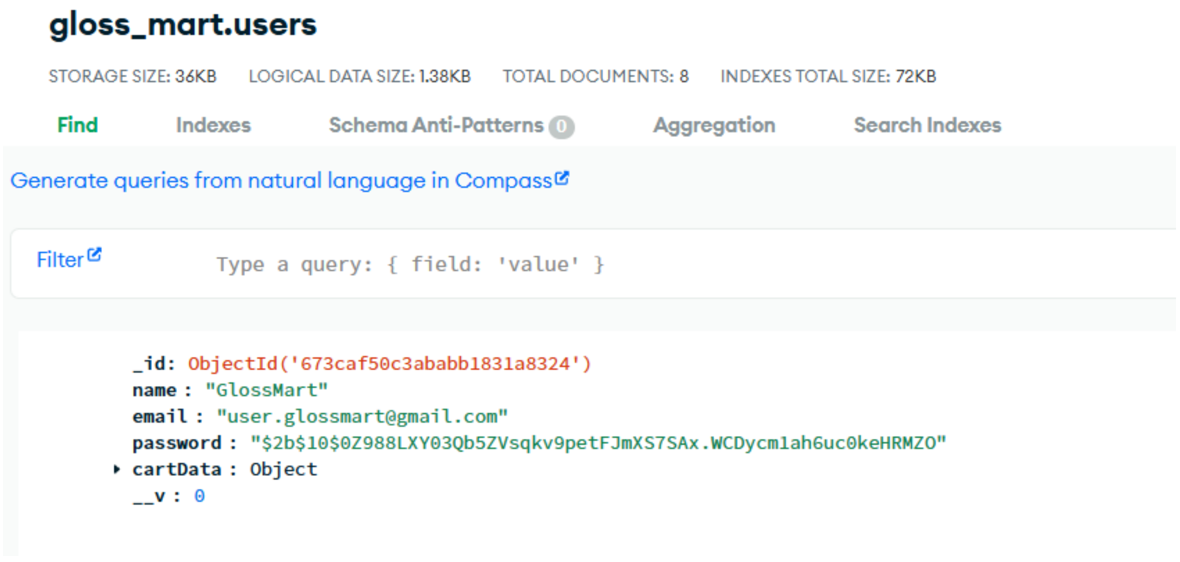
Type a query: { field: 'value' }

```
_id: ObjectId('6741ff8e59982b305fdefd50')
userId : "6741b43ebb6c23e2c46a697e"
▸ items : Array (2)
amount : 61
▸ address : Object
status : "Out for delivery"
date : 2024-11-23T16:09:58.441+00:00
payment : true
__v : 0

_id: ObjectId('6742136aae211cd06b0d3bcf')
userId : "6741b43ebb6c23e2c46a697e"
▸ items : Array (1)
amount : 14
▸ address : Object
status : "Order Processing"
date : 2024-11-23T16:56:42.043+00:00
payment : true
```

Figure B.9 Backend Order Collection

The order collection in the backend stores important information related to customer orders. Each order is uniquely identified by an _id and linked to a userId to associate it with a specific customer. The items array holds the details of the products purchased, while the amount reflects the total cost of the order. The address object contains shipping details, and the status field indicates the current stage of the order (e.g., "Out for delivery"). The date field records the order placement time, and payment is a boolean flag indicating whether the order has been paid. The __v field is used for internal version control by MongoDB.

BACKEND USER COLLECTION



```
_id: ObjectId('673caf50c3ababb1831a8324')
name : "GlossMart"
email : "user.glossmart@gmail.com"
password : "$2b$10$0Z988LXY03Qb5ZVsqkv9petFJmXS7SAx.WCDycm1ah6uc0keHRMZO"
▸ cartData : Object
__v : 0
```

Figure B.10 Backend User Collection

The user collection in the backend stores essential user details, with each user being uniquely identified by an _id (e.g., 673cb7f2e6bcc941b708e249). The name and email fields store the user's personal information, while the password field stores the hashed password for secure authentication. The cartData object contains information about the user's shopping cart, including items they have added but not yet purchased. The __v field is used by MongoDB for internal version control, ensuring data consistency during updates.

REFERENCES

1. MongoDB Official Documentation - A comprehensive guide to MongoDB features, schema design, and best practices. https://www.mongodb.com/docs/
2. React.js Official Documentation - In-depth information on React components, hooks, and state management. https://react.dev/
3. Express.js Handbook - Detailed insights on building backend applications using Express.js. https://expressjs.com/
4. TutorialsPoint MERN Stack Guide - Step-by-step guide to building full-stack applications using the MERN stack. https://www.tutorialspoint.com/mern_stack/index.html
5. Bootstrap Official Documentation - CSS framework for building responsive and visually appealing designs. https://getbootstrap.com/
6. Razorpay Payment Gateway Docs - Instructions for integrating online payment systems into web applications. https://razorpay.com/docs/
7. JSON Web Token (JWT) Documentation - Guidelines for implementing secure authentication in applications. https://jwt.io/introduction
8. Nodemailer Guide - For integrating email services like sending order confirmations or updates. https://nodemailer.com/about/
9. Stripe API Documentation - Simplified guide for setting up and managing payment processing. https://stripe.com/docs
10. Nielsen Norman Group - Essential principles for creating user-friendly and engaging e-commerce interfaces. https://www.nngroup.com/