E-Commerce Website

A Project Report for Industrial Training and Internship

submitted by

Ankita Nath
Arpita Nath
Rahit Saha
Suvajit Biswas

In the partial fulfillment of the award of the degree of

B.Tech

in the

Computer Science and Engineering

Of

Heritage Institute Of Technology

At

Ardent Computech Pvt. Ltd.

ARDENT
COMPUTECH PVT. LTD.
High-End Technology Training and Project

ARDENT®
GROUP OF COMPANIES
INTERNSHIP | DEVELOPMENT | PROJECT
An ISO 9001 : 2015 Certified Company

CERTIFICATE FROM SUPERVISOR

This is to certify that **"Ankita Nath,Arpita Nath,Rahit Saha,Suvajit Biswas, 12622001018,12622001035,12623001210,12622020053**" have completed the project titled "**E-Commerce Website**" under my supervision during the period from "6/6/2025" to "16/7/2025 " which is in partial fulfillment of requirements for the award of the **B.Tech** degree and submitted to the Department of "**Computer Science and Engineering**" of **"Heritage Institute Of Technology"**.

_____

Signature of the Supervisor

Date: dd/mm/yy

Name of the Project Supervisor:

BONAFIDE CERTIFICATE

Certified that this project work was carried out under my supervision

"E-Commerce Website" is the bonafide work of

Name of the student: Ankita Nath ***Signature:***

Name of the student: Arpita Nath ***Signature:***

Name of the student: Rahit Saha ***Signature:***

Name of the student: Suvajit Biswas ***Signature:***

SIGNATURE

Name :

PROJECT MENTOR

SIGNATURE

Name:

EXAMINERS

Ardent Original Seal

ACKNOWLEDGEMENT

The achievement that is associated with the successful completion of any task would be incomplete without mentioning the names of those people whose endless cooperation made it possible. Their constant guidance and encouragement made all our efforts successful.

We take this opportunity to express our deep gratitude towards our project mentor, ***Sourav Goswami*** for giving such valuable suggestions, guidance and encouragement during the development of this project work.

Last but not the least we are grateful to all the faculty members of **Ardent Computech Pvt. Ltd.** for their support.

(Note: All entries of the proforma of approval should be filled up with appropriate and complete information or approval in any respect will be summarily rejected.)

Name of the Student With Group:

1. Ankita Nath
2. Arpita Nath
3. Rahit Saha
4. Suvajit Biswas

1. Title of the Project: **E-Commerce Website**

2. Name and Address of the Guide: **Mr. *Sourav Goswami***
Sr. Subject Matter Expert &
Technical Head (C Programming), Module #132
Ground Floor, SDF Building, GP Block, Sector V
Bidhannagar, Kolkata, West Bengal – 700091

3. Educational Qualification of the Guide: Ph.d* ☐ M.Tech ☐ B.Tech*/B.E* ☐ MCA* ☐ M.Sc* ☐

4. Working and Teaching Experience of the Guide: _____ Years

5. Software Used in the Project: a. Visual Studio Code

1.
2.
3.
4.

Signature of the Guide:
Date:
Name: *Sourav Goswami*,
Subject Matter Expert,
Signature, Designation, Stamp of the Project Proposal Evaluator

Signature of the Student:
Date:
For the office use only

Approved	Not Approved

PROJECT RESPONSIBILITY FORM

Serial No	Name Of Member	Responsibility
1.	Ankita Nath	Project Leader, CodingDatabase Manage, Backend Developer, Frontend Developer, Deployment
2.	Arpita Nath	CodingDatabase Manage, Backend Developer, Frontend Developer, Project Report
3.	Rahit Saha	Frontend Developer, PPt, Project Documentation
4.	Suvajit Biswas	Frontend Developer, PPt, Project Documentation

SELF CERTIFICATE

This is to certify that the dissertation/project proposal entitled " **E-Commerce Website**" Done by us, is an authentic work carried out for the partial fulfillment of the requirements for the award of the certificate of Bachelor of Computer Application under the guidance of Mr. Subhojit Santra. The matter embodied in this project work has not been submitted earlier for the award of any certificate to the best of our knowledge and belief.

Name of the Student

- Ankita Nath
- Arpita Nath
- Rahit Saha
- Suvajit Biswas

Signature of the students

a.
b.
c.
d.

CERTIFICATE BY GUIDE

This is to certify that this project entitled "**E-Commerce Website**" submitted in partial fulfillment of the certificate of Bachelor of Computer Application through **Ardent Computech Pvt Ltd**, done by the Group Members

1. **Ankita Nath**
2. **Arpita Nath**
3. **Rahit Saha**
4. **Suvajit Biswas**

Is an authentic work carried out under my guidance & best of our knowledge and belief.

a.

b.

Signature of the students

Signature of the Guide

Date:

Date:

CERTIFICATE OF APPROVAL

This is to certify that this proposal of Minor project, entitled "**E-Commerce Website**" is a record of bona-fide work, carried out by**:1. Ankita Nath,2. Arpita Nath,3. Rahit Saha,4. Suvajit Biswas,** under my supervision and guidance through the Ardent Computech Pvt. Ltd. In my opinion, the report in its present form partially fulfills all the requirements, as specified by the **Heritage Institute Of Technology** as per regulations of the **Ardent**. It has attained the standard necessary for submission. To the best of my knowledge, the results embodied in this report, are original and worthy of incorporation in the present version of the report for Bachelor of Technology.

Guide/Supervisor

Mr. *Sourav Goswami*

Subject Matter Expert & Technical Head (C Programming & Data Structure)
Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)
Module #132, Ground Floor, SDF Building, GP Block, Sector V, Bidhannagar,
Kolkata, West Bengal, Kolkata – 700091

External Examiner(s)

Head of the Department
Of Computer Application Midnapore
(Affiliated to Vidyasagar University,WB)

Table of Contents

1. Introduction

1.1 Purpose

*This Software Requirements Specification (SRS) document outlines the functional, non-functional, and technical requirements for the **Commerce** platform—an end-to-end e-commerce solution designed to provide a seamless shopping experience for users and efficient product and order management for administrators.*

*The product is identified as **Commerce v1.0**, with its initial release focusing on essential features such as product browsing, cart management, order placement, and secure payment integration using Stripe. This SRS defines the complete scope of the platform, including user-facing functionalities, backend services, administrative tools, and third-party payment integration.*

Specifically, this document addresses the following core subsystems and functionalities:

1. ***Product Catalog and Filtering**: Allows users to browse and search products with filters for a tailored shopping experience.*
2. ***Cart and Checkout System**: Facilitates adding items to cart, updating quantities, and placing orders.*
3. ***Admin Dashboard**: Provides tools for managing products, viewing orders, and overseeing platform activity.*
4. ***Payment Gateway Integration**: Enables secure and real-time payment processing through Stripe.*

*The **Commerce** platform aims to streamline the online shopping and selling process while maintaining scalability, security, and a user-friendly interface. This SRS ensures that all development efforts align with the goals of creating a reliable and intuitive e-commerce experience.*

1.2 Document Conventions

This document has been developed in accordance with the IEEE standards for Software Requirements Specification (SRS) Documents.
***Commerce** will also be referred to as the **"web-based platform"** or **"e-commerce platform"** throughout this document.*

*This section outlines the typographical, structural, and formatting standards followed in preparing the SRS for **Commerce v1.0**, which aims to provide a robust and user-friendly online shopping experience.*

1. Fonts and Styles

- ***Section Titles**: **Bold**, size **18**, Times New Roman font*
- ***Subsection Titles**: **Bold**, size **14**, Times New Roman font*
- ***Body Text**: Italicized, size **11**, Arial font*

- ***Key Terms or Concepts**: **Bold** within the body text for emphasis*

2. Requirements Prioritization

- *High-level requirements inherit the priority of their detailed counterparts unless explicitly reclassified.*
- *Each requirement is categorized as:*
    - ***High**: Essential for the core operation of the platform (e.g., product listing, cart, checkout).*
    - ***Medium**: Important features that enhance usability or scalability (e.g., advanced filters, wishlists).*
    - ***Low**: Optional or future-scope enhancements (e.g., product reviews, loyalty programs).*

3. Terminology and Definitions

- *Consistent usage of standard industry terms such as **product catalog**, **shopping cart**, **order tracking**, and **payment gateway integration**.*
- *Platform-specific terms (e.g., **Admin Dashboard**, **Stripe Payment**, **Cart Management**) are used consistently throughout the document.*

4. Consistency Across the Document

Each section adheres to a structured format to maintain clarity and ensure traceability:

- ***Purpose**: Explains the goal of the section or feature.*
- ***Scope**: Describes the boundaries or limitations of the feature.*
- ***Details**: Provides a clear, actionable description of the functionality or requirement.*

*These conventions ensure that the SRS for the **Commerce** platform is well-structured, easy to navigate, and facilitates clear communication among stakeholders and developers.*

1.3 Intended Audience and Reading Suggestions

1.3.1 Intended Audience

*This Software Requirements Specification (SRS) document is intended for the following stakeholders involved in the development and deployment of the **Commerce** e-commerce platform:*

1. ***Developers***
   ○ *Responsible for building the platform based on the functional and technical requirements defined in this document.*
   ○ ***Focus Areas**: Functional Requirements, System Features, Technical Architecture.*

2. ***Project Managers***
   — *Oversee the development lifecycle, ensuring timely delivery and alignment with business goals.*
   ○ ***Focus Areas**: Purpose, Scope, Requirements Prioritization.*

3. ***Marketing Teams***
   ○ *Understand the platform's key features and value proposition for marketing and promotional activities.*
   — ***Focus Areas**: Introduction, Core Features, and User Benefits.*

4. ***Testers (QA Engineers)***
   — *Validate that the implemented features meet all functional and non-functional specifications.*
   ○ ***Focus Areas**: Functional Requirements, Non-Functional Requirements.*

5. ***End-Users (Buyers and Sellers)***
   — *Gain clarity on how the platform facilitates product browsing, purchasing, and selling.*
   ○ ***Focus Areas**: Product Catalog, Cart System, Checkout Process, and Payment Features.*

6. ***Documentation Writers***
   ○ *Create user manuals, API documentation, and system usage guides based on the platform's structure.*
   — ***Focus Areas**: Functional Requirements, User Interaction Details, Technical Architecture.*

1.3.2 Document Organization

The SRS is structured into the following main sections to promote clarity and ease of access:

- ***Section 1: Introduction***
  ○ *Describes the purpose, document conventions, intended audience, and scope.*

- ***Section 2: System Overview***
  ○ *Provides a summary of platform architecture, core features, and high-level workflow.*

- ***Section 3: Functional Requirements***
  — *Defines the core functionalities the platform must support, such as browsing, cart management, order placement, and payment.*

- ***Section 4: Non-Functional Requirements***
  二 *Specifies system quality attributes like performance, security, scalability, and usability.*

- ***Section 5: Technical Architecture***
  三 *Outlines the technology stack, backend services, database schema, and third-party integrations (e.g., Stripe).*

- ***Section 6: Appendices***
  ○ *Includes glossary, diagrams, and other reference materials.*

1.3.3 Reading Suggestions

To efficiently navigate and understand the SRS, readers are advised to follow a role-specific reading sequence:

1. ***General Readers***
   ○ *Begin with **Section 1 (Introduction)** and **Section 2 (System Overview)** to understand the platform's objective and key components.*

2. ***Developers***
   — *Focus on **Section 3 (Functional Requirements)** and **Section 5 (Technical Architecture)** for implementation guidance.*

3. ***Testers***
   — *Refer to **Section 3 (Functional Requirements)** and **Section 4 (Non-Functional Requirements)** to plan and execute test cases.*

4. ***Marketing Teams***
   — *Review **Section 2 (System Overview)** and **Section 6 (Appendices)** to extract messaging and platform strengths.*

5. ***Documentation Writers***
   — *Concentrate on **Section 3 (Functional Requirements)** and **Section 6 (Appendices)** for preparing end-user documentation and technical references.*

1.4 Product Scope

*The **Commerce** platform is a full-featured e-commerce solution designed to offer a seamless digital shopping experience for consumers and streamlined management tools for administrators. The primary objective of Commerce is to enable users to browse products, manage a cart, place orders, and complete secure payments—while offering sellers a user-friendly interface .*

Benefits, Objectives, and Goals

- ***Benefits**:*
  - *Provides a fast and intuitive online shopping experience.*
  - *Simplifies product and order management for both users and admins.*
  - *Ensures secure transactions through integrated payment gateways like **Stripe**.*
- ***Objectives**:*
  - *Deliver a responsive and user-centric platform with features such as product browsing, filtering, cart handling, and secure checkout.*

    - *Equip admins with tools to manage product listings, view orders, and maintain operational efficiency.*
    - *Integrate third-party services (e.g., payment processing) for real-time, reliable transactions.*
- ***Goals**:*
    - *Establish a scalable e-commerce foundation for future feature expansion (e.g., reviews, wishlists, analytics).*
    - *Empower sellers to effectively manage their online store presence.*
    - *Create a trusted digital marketplace with a strong focus on usability, performance, and security.*

This software supports broader business strategies by enabling digital commerce, improving customer satisfaction, and leveraging modern web technologies to ensure operational efficiency and platform growth.

1.5 References

IEEE Software Requirement Specifications Template:

https://dspmuranchi.ac.in/pdf/Blog/srs_template-ieee.pdf

2. Overall Description

2.1 Product Perspective

*The **Commerce** platform is a modular, web-based e-commerce application designed to provide a seamless shopping and order management experience for users and administrators. It integrates effectively with external services such as **Stripe** for secure payment processing and can be extended to support third-party logistics or inventory systems.*

The platform is composed of several core subsystems that function cohesively to deliver a complete digital commerce solution:

- ***Product Catalog & Filtering**: Enables users to browse and filter products based on categories, price, and other attributes.*

- ***Shopping Cart & Checkout****: Allows users to add items to a cart, update quantities, and place orders.*

- ***Admin Dashboard****: Offers backend capabilities for administrators to manage product listings, monitor orders, and track user activity.*

- ***Payment Gateway Integration****: Uses* ***Stripe*** *to handle secure, real-time payments.*

The system architecture follows a modern MERN stack (MongoDB, Express, React, Node.js) with:

- ***React*** *powering the user interface, enhanced with* ***Redux or Context API*** *for state management.*

- ***Express*** *handling backend API routes for products and orders.*

- ***MongoDB*** *storing structured data across collections for products, users, and orders.*

Overall, ***Commerce*** *delivers a scalable, efficient, and secure platform for conducting online sales while providing intuitive features for both customers and administrators.*

2.2 Product Functions

The ***Commerce*** *platform provides the following major functions to support a full-featured e-commerce experience:*

- ***Product Catalog and Filtering***
  - *Enables users to browse a wide range of products.*
  - *Provides search and filtering options based on category, price, and availability.*

- ***Shopping Cart Management***
  - *Allows users to add, remove, or update items in their shopping cart.*
  - *Reflects real-time price calculations including quantity and subtotal updates.*

- ***Order Placement and Checkout***
  - *Guides users through a secure and streamlined checkout process.*
  - *Supports address entry, order review, and payment confirmation.*

- ***Admin Dashboard***
  - *Allows administrators to add, update, or delete product listings.*
  - *Provides order tracking, user management, and inventory monitoring capabilities.*

- ***Payment Gateway Integration (Stripe)***
  - *Facilitates secure, real-time payment processing.*
  - *Supports credit/debit cards.*

- ***User Account Management***
  - *Enables account registration, login/logout, and session management.*
  - *Allows users to view their order history and manage profile settings.*

2.3 User Classes and Characteristics

User Class	Description	Frequency of Use	Key Functions Used	Technical Expertise	Importance
Buyers	General users browsing and purchasing products through the platform.	Frequent	Product Catalog, Cart Management, Checkout, Payment, Order Tracking	Basic	High
Sellers	Individuals or businesses listing and managing products for sale.	Frequent	Product Upload/Update, Inventory Management, Order Tracking	Basic to Moderate	High
Administrators	Platform managers overseeing operations, system data, and user activity.	Frequent	Admin Dashboard, Product/Order Moderation, User Management	Advanced	High
Logistics/Support Staff	Staff handling backend tasks such as order fulfillment or customer support.	Moderate	Order Details, Delivery Updates, Issue Resolution Tools	Moderate	Medium
Guest Users	Unregistered users browsing products without placing orders.	Occasional	Product Catalog, Basic Filters	Basic	Medium
Developers	Technical team responsible for maintaining and expanding the platform.	Occasional	Technical Architecture, APIs, System Logs	Advanced	Medium

2.4 Operating Environment

*The **Commerce** platform is designed to operate seamlessly across multiple devices and environments, providing a consistent and responsive experience for both users and administrators. The following environments and configurations are supported:*

- ***Hardware Platform:***
  - *Accessible on **desktop**, **laptop**, **tablet**, and **smartphone** devices.*
  - ***Minimum Requirements**:*
    - ***Mobile Devices**: 4GB RAM, dual-core processor, and active internet connection.*
    - ***Desktops/Laptops**: 8GB RAM, dual-core processor, and support for modern web browsers.*

- ***Operating Systems:***
  - ***Mobile**:*
    - *Android 8.0 (Oreo) and above*
    - *iOS 12.0 and above*
  - ***Desktop**:*
    - *Windows 10 and above*
    - *macOS Mojave and above*
    - *Linux distributions (Ubuntu 18.04+ recommended)*

- ***Web Browsers:***
  - *Fully compatible with the latest versions of:*
    - ***Google Chrome***
    - ***Mozilla Firefox***
    - ***Microsoft Edge***
    - ***Apple Safari***

- ***Third-Party Software and Services:***
  - ***Payment Gateway Integration**:*

    - ***Stripe** is used for secure and real-time transaction processing.*

- ***Authentication (optional)**:*
    - *OAuth 2.0 (e.g., Google login) for secure user access.*
- ***Hosting & Deployment (optional)**:*
    - *Supports deployment on **Vercel**, **Render**, **Heroku**, or **AWS**.*

● ***Network:***

- *A **stable internet connection** is required for:*
    - *Real-time cart and checkout updates*
    - *Secure payment processing via Stripe*
    - *Admin dashboard operations and backend API communication*

2.5 Design and Implementation Constraints

- ***Technology Stack**: Frontend in **React.js**, backend in **Node.js (Express)**, and **MongoDB** for database management.*
- ***Third-Party Integration**: Must integrate securely with **Stripe** for payments.*
- ***Security**: Enforce **HTTPS**, data encryption, and secure authentication (e.g., JWT).*
- ***Performance**: Ensure **low-latency** cart updates, checkout, and order processing.*
- ***Multilingual Support**: Platform should support **multiple languages** for broader accessibility.*
- ***Standards Compliance**: Follows **IEEE SRS** and software design best practices for maintainability and scalability.*

2.6 User Documentation

*The **Commerce** platform will include the following user documentation:*

- ***User Manual**: Comprehensive PDF and web-based guide covering all features (browsing, cart, checkout, admin tools).*
- ***Tutorials**: Step-by-step video and text guides for tasks like product listing, cart usage, and order tracking.*
- ***FAQs**: A section addressing common questions related to shopping, payments, and account management.*

2.7 Assumptions and Dependencies

- ***Assumptions**:*
  - *Users have stable internet access.*
  - *Stripe and other third-party services will function reliably.*
  - *Multilingual support meets user accessibility needs.*

- ***Dependencies**:*
  - *Platform depends on **React.js**, **Node.js**, and **MongoDB**.*
  - *Payment processing is dependent on **Stripe integration**.*

3. External Interface Requirements

3.1 User Interfaces

*The **Commerce** platform provides distinct user interfaces tailored for Admins and Buyers, ensuring role-specific functionality and usability:*

3.1.1.1 ***Admin Interface***

- *Key Components:*
  - ***Sell Management**: Tools to oversee product listings.*
  - ***Admin Dashboard**: Centralized analytics and user activity management.*

3.1.1.2 ***Buyer Interface***

- *Key Components:*
  - ***Product Listings**: Seamless navigation and eco-friendly filters for buying products.*
  - ***Payment Gateway** : Facilitates secure, real-time payment processing.*

3.2 Hardware Interfaces

*The **Commerce** platform supports the following hardware interfaces:*

- ***Device Types**: Smartphones, tablets, desktops, and laptops.*
- ***Hardware Requirements**: Minimum 4GB RAM for mobile and 8GB RAM for desktops.*
- ***Protocols**: Supports standard communication protocols like HTTPS for secure data transmission.*

3.3 Software Interfaces

- ***Database**: Uses **MongoDB** to store products, users, and order data.*
- ***Operating Systems**: Compatible with **Windows**, **macOS**, **Linux**, **Android**, and **iOS**.*
- ***APIs**:*
  - ***Stripe API** for secure payment processing.*
  - *REST APIs for handling products, orders, and user data.*
- ***Libraries & Tools**:*
  - ***React.js** for frontend, **Node.js (Express)** for backend.*
  - *Optional: Integration with cloud services for scalability.*
- ***Data Flow**:*
  - ***Incoming**: User actions (e.g., add to cart, order placement).*
  - ***Outgoing**: Order confirmations, payment statuses, user notifications.*

3.4 Communications Interfaces

*The **Commerce** platform relies on the following communication interfaces:*

- ***Protocols**: Uses HTTPS for secure web communications and API calls.*
- ***Web Browser Support**: Compatible with Chrome, Firefox, Safari, and Edge.*
- ***Network Requirements**: Stable internet connection for real-time updates*

These interfaces ensure secure, efficient, and synchronized communication between users and the platform.

4. System Use Cases

*This section organizes the functional requirements of the **Commerce** platform by system features, detailing the major services provided by the product.*

4.1 Admin Interface

4.1.1 Admin Dashboard

Description and Priority:

Centralized interface for managing platform analytics and user activities.

- **Priority**: High.

Stimulus/Response Sequences:

- **S**: Admin views platform analytics.
- **R**: Displays data such as user activity, sales trends.

Functional Requirements:

- **REQ-1**: Display user activity and sales analytics.
- **REQ-2**: Tools for managing user permissions and activities.

4.1.2 Log In and Sign Up System

Description and Priority

Provides secure access for admin accounts with authentication and account management capabilities.

- **Priority**: High.

Stimulus/Response Sequences

- **S**: Admin enters credentials for login.
- **R**: System validates credentials and grants access to the admin dashboard.
- **S**: Admin attempts to sign up.
- **R**: System registers the admin account after verifying required information.

Functional Requirements

- **REQ-1**: Support admin account creation with email and secure password.
- **REQ-2**: Provide functionality for password recovery and updates.

- **REQ-3**: Restrict access to admin-specific features based on authentication status.

4.1.3 Sell Management

Description and Priority:

Tools to oversee and manage product listings for selling.

- **Priority**: High.

Stimulus/Response Sequences:

- **S**: Admin edits a product listing.
- **R**: System updates the listing in real-time.
- **S**: Admin reviews flagged listings.
- **R**: System notifies the seller and updates the listing's status.

Functional Requirements:

- **REQ-1**: Enable adding, editing, and removing product listings.
- **REQ-2**: Provide approval workflows for seller-submitted listings.

4.2 Buyer Interface

4.2.1 Product Listings

Description and Priority:

Seamless navigation with eco-friendly filters for product discovery.

- **Priority**: High.

Stimulus/Response Sequences:

- **S**: Buyer searches for a product.

- **R**: Displays results filtered by eco-friendly options.

Functional Requirements:

- **REQ-1**: Support product search with eco-friendly filters.
- **REQ-2**: Allow buyers to sort products by sustainability ratings.

4.2.2 Log In and Sign Up System

Description and Priority

Allows buyers to create accounts, securely log in, and access personalized features like EcoPoints and purchase history.

- **Priority**: High.

Stimulus/Response Sequences

- **S**: Buyer attempts to sign up with email and password.
- **R**: System registers the account and confirms via email verification.
- **S**: Buyer logs in with credentials.
- **R**: System validates credentials and redirects to the buyer's dashboard.

Functional Requirements

- **REQ-1**: Enable buyer account creation with email and secure password.
- **REQ-2**: Provide social login options (e.g., Google, Facebook).
- **REQ-3**: Support password recovery and updates.
- **REQ-4**: Authenticate users before granting access to personalized features like EcoPoints and order tracking.

4.2.3 Payment Gateway

Description and Priority

Secure and seamless integration of payment gateways to process transactions for buying products with a small amount of transportation fees being charged.

- **Priority**: High.

Stimulus/Response Sequences

- **S**: Buyer selects a product and proceeds to checkout.
- **R**: Displays payment options (e.g., Stripe) and securely processes the transaction.
- **S**: Buyer cancels a transaction mid-way.
- **R**: Payment process is terminated, and the cart remains unchanged.

Functional Requirements

- **REQ-1**: Support multiple payment options.
- **REQ-2**: Secure payment processing with SSL/TLS encryption.
- **REQ-3**: Provide real-time confirmation of successful transactions and receipts.
- **REQ-4**: Handle payment failures with error messages and retry options.
- **REQ-5**: Ensure compliance with global payment standards like PCI-DSS.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- ***Response Time****: The system must respond to user actions (e.g., searching products, calculating carbon footprints) within* ***2 seconds*** *under normal conditions.*
- ***Transaction Handling****: The payment gateway must process transactions and provide confirmation within* ***5 seconds****.*

- ***Concurrent Users****: The platform must support up to* ***10,000 concurrent users*** *without performance degradation.*
- ***Scalability****: The system must handle a* ***30% increase in user activity*** *during peak times (e.g., seasonal sales).*
- ***Uptime****: The platform must maintain* ***99.9% availability*** *to ensure reliability.*

5.2 Safety Requirements

- ***Data Protection****: Ensure user data, including payment details, is encrypted to prevent unauthorized access.*
- ***Fraud Prevention****: Implement safeguards to detect and block fraudulent transactions.*
- ***User Safeguards****: Prevent harmful content by moderating blogs and user-generated content.*
- ***Compliance****: Adhere to global safety regulations like GDPR, CCPA, and PCI-DSS for secure transactions and data handling.*
- ***System Monitoring****: Enable real-time monitoring to detect and mitigate potential threats or breaches.*

5.3 Security Requirements

- ***Authentication****: Implement authentication for user accounts.*
- ***Data Encryption****: Encrypt sensitive data, including personal information and payment details.*
- ***Role-Based Access****: Restrict access to admin features based on user roles.*
- ***Privacy Protection****: Provide users with options to control and delete their data.*

5.4 Software Quality Attributes

- ***SQA1 – Usability****: The interface should be intuitive and task-oriented, ensuring ease of use for non-technical users without superfluous elements.*
- ***SQA2 – Availability****: The platform must maintain* ***99.9% uptime****, ensuring constant accessibility with a stable internet connection.*
- ***SQA3 – Reliability****: All critical functions, including payment processing and carbon footprint calculations, must operate correctly under normal conditions.*
- ***SQA4 – Scalability****: The system should handle up to* ***10,000 concurrent users*** *and adapt to traffic spikes during peak times.*

- ***SQA5 – Security****: Data, including personal and payment information, must be encrypted using AES-256, ensuring user privacy and protection.*

5.5 Business Rules

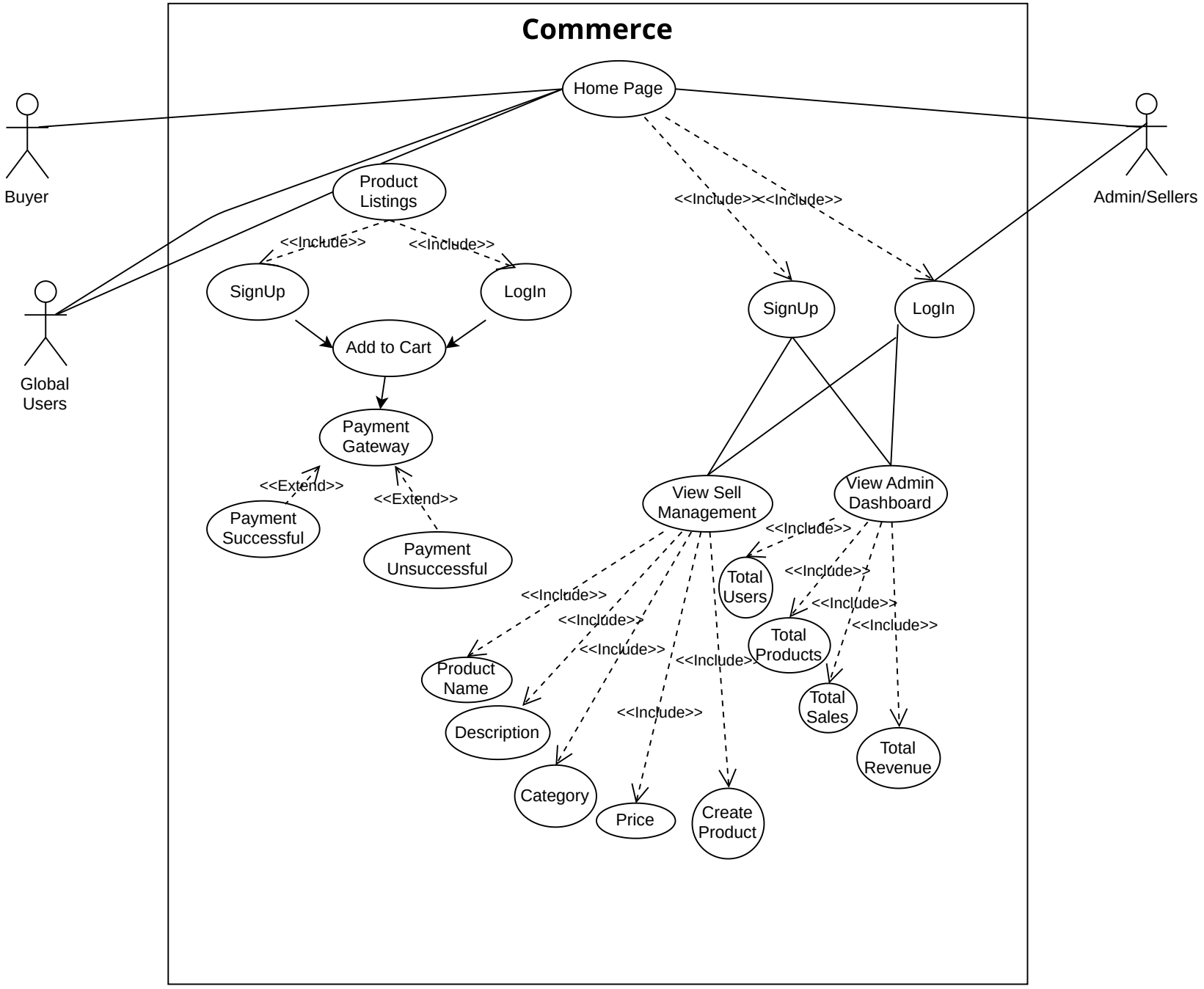
- ***BR1****: Only admins can manage and moderate product listings, blogs, and user-generated content.*
- ***BR2****: Buyers can earn EcoPoints only for purchases with verified low-carbon shipping options.*
- ***BR3****: Blockchain-based sourcing details must be accessible for all listed products.*
- ***BR4****: Users must authenticate their accounts before performing any transactions.*

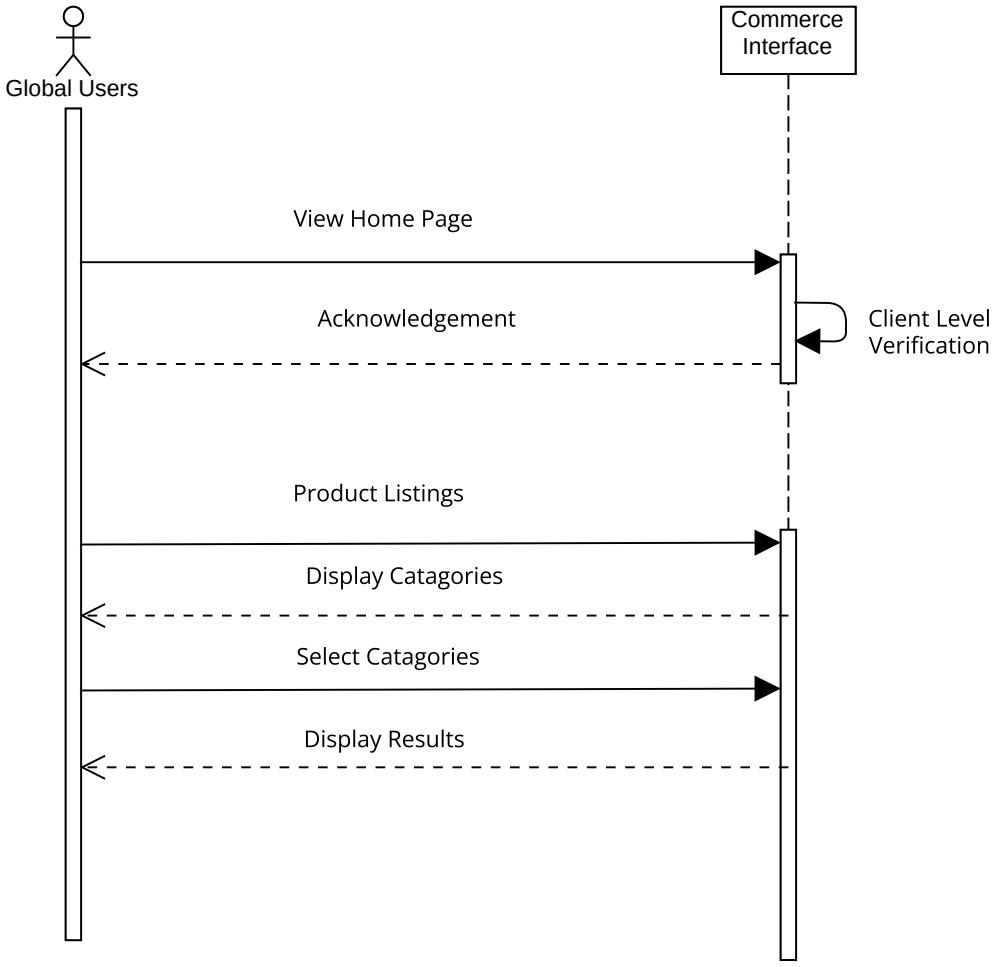
6. Other Requirements

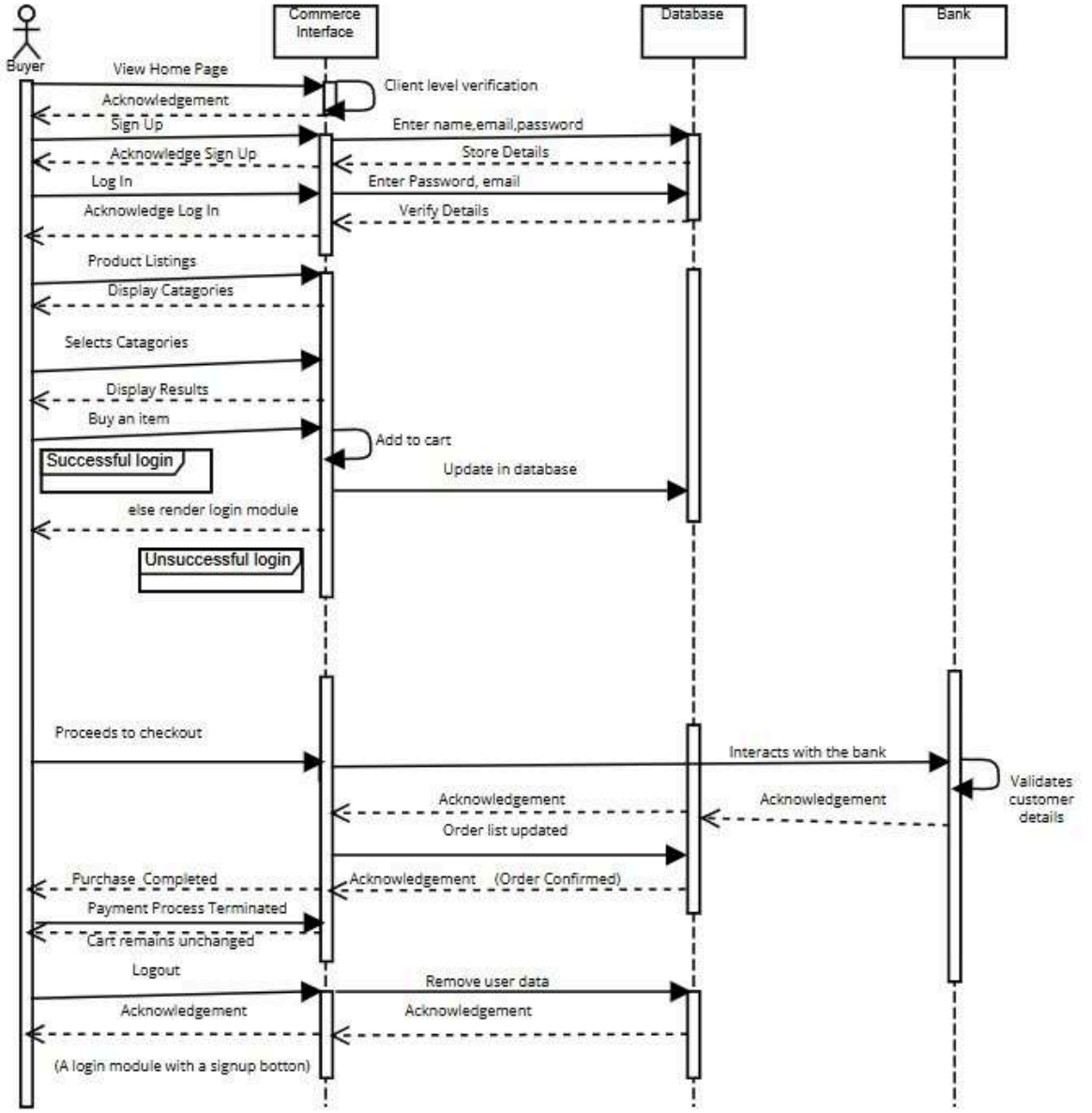
- ***OR1 – Database Requirements****: The system must use MongoDB, ensuring scalability and secure storage of user, product, and transaction data.*
- ***OR4 – Reusability****: The codebase must follow modular design principles to enable future feature additions.*
- ***OR5 – Accessibility****: Ensure the platform complies with WCAG 2.1 standards for users with disabilities.*

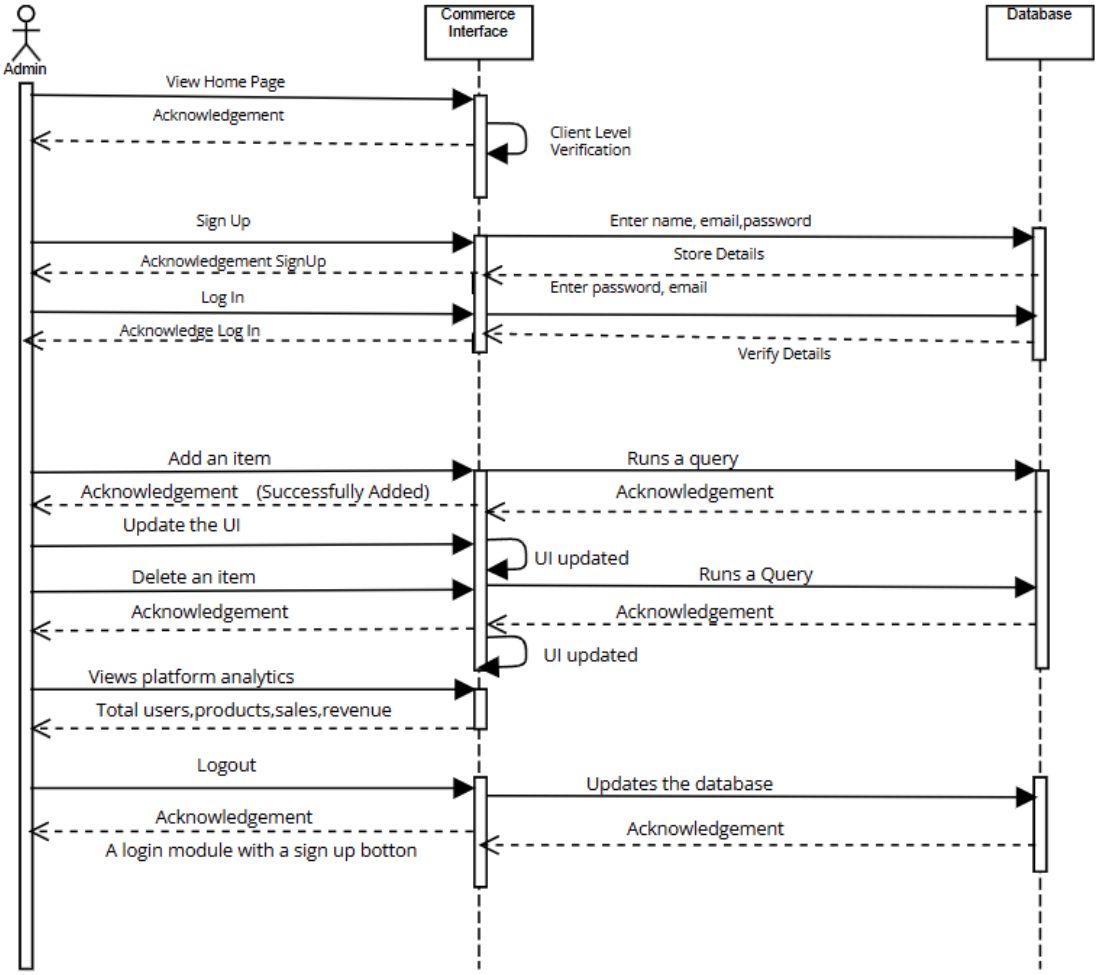
Appendix A: Glossary

- ***S****: Stimulus*
- ***R****: Response*
- ***REQ****: Requirements*
- ***BR****: Business Rules*
- ***SQA****: Software Quality Attributes*
- ***FAQ****: Frequently Asked Questions*
- ***COD****: Cash on Delivery*
- ***GUI****: Graphical User Interface*
- ***OR****: Other Requirements*





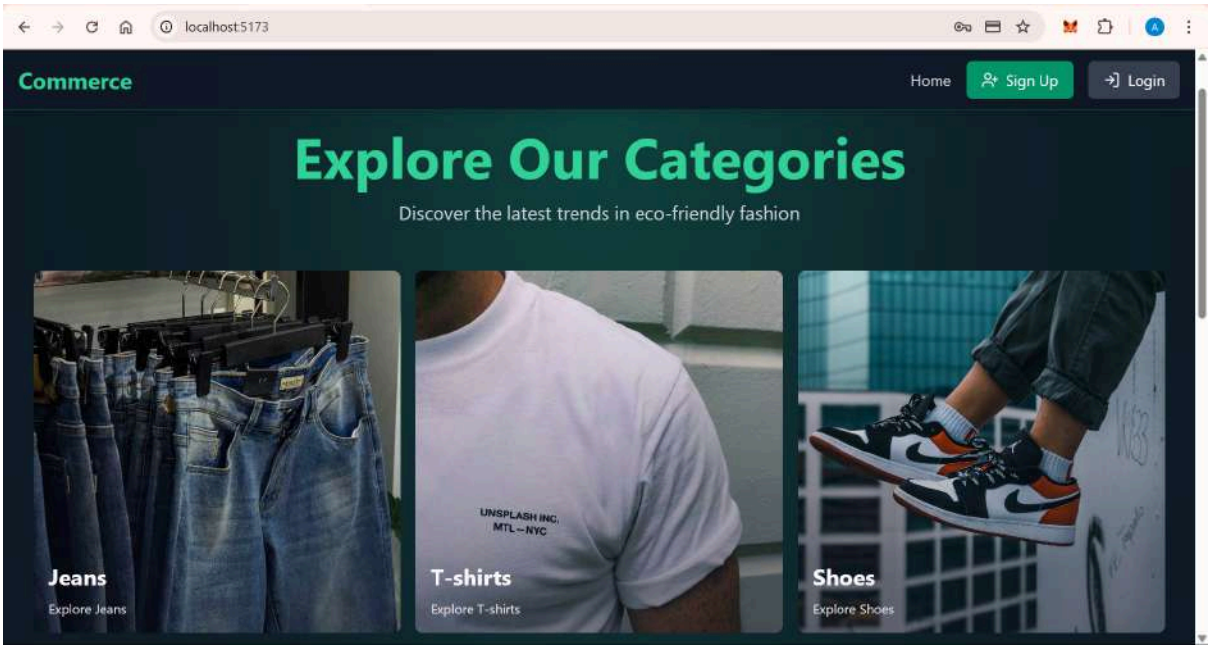

Buyer
Commerce Interface
Database
Bank
View Home Page
Client level verification
Acknowledgement
Sign Up
Enter name,email,password
Acknowledge Sign Up
Store Details
Log In
Enter Password, email
Acknowledge Log In
Verify Details
Product Listings
Display Catagories
Selects Catagories
Display Results
Buy an item
Add to cart
Successful login
Update in database
else render login module
Unsuccessful login
Proceeds to checkout
Interacts with the bank
Validates customer details
Acknowledgement
Acknowledgement
Order list updated
Purchase Completed
Acknowledgement (Order Confirmed)
Payment Process Terminated
Cart remains unchanged
Logout
Remove user data
Acknowledgement
Acknowledgement
(A login module with a signup botton)


Admin
Commerce Interface
Database
View Home Page
Acknowledgement
Client Level Verification
Sign Up
Enter name, email,password
Acknowledgement SignUp
Store Details
Log In
Enter password, email
Acknowledge Log In
Verify Details
Add an item
Runs a query
Acknowledgement (Successfully Added)
Acknowledgement
Update the UI
UI updated
Delete an item
Runs a Query
Acknowledgement
Acknowledgement
UI updated
Views platform analytics
Total users,products,sales,revenue
Logout
Updates the database
Acknowledgement
Acknowledgement
A login module with a sign up botton

UI SNAPSHOT

Frontend:

1.Home Page



HomePage.jsx

```
import { useEffect } from "react";
import CategoryItem from "../components/CategoryItem";
import { useProductStore } from "../stores/useProductStore";
import FeaturedProducts from "../components/FeaturedProducts";

const categories = [
	{ href: "/jeans", name: "Jeans", imageUrl: "/jeans.jpg" },
	{ href: "/t-shirts", name: "T-shirts", imageUrl: "/tshirts.jpg" },
	{ href: "/shoes", name: "Shoes", imageUrl: "/shoes.jpg" },
	{ href: "/glasses", name: "Glasses", imageUrl: "/glasses.png" },
	{ href: "/jackets", name: "Jackets", imageUrl: "/jackets.jpg" },
	{ href: "/suits", name: "Suits", imageUrl: "/suits.jpg" },
	{ href: "/bags", name: "Bags", imageUrl: "/bags.jpg" },
];

const HomePage = () => {
	const { fetchFeaturedProducts, products, isLoading } = useProductStore();

	useEffect(() => {
		fetchFeaturedProducts();
```

```
	}, [fetchFeaturedProducts]);

	return (
		<div className='relative min-h-screen text-white overflow-hidden'>
			<div className='relative z-10 max-w-7xl mx-auto px-4 sm:px-6
lg:px-8 py-16'>
				<h1 className='text-center text-5xl sm:text-6xl font-bold
text-emerald-400 mb-4'>
					Explore Our Categories
				</h1>
				<p className='text-center text-xl text-gray-300 mb-12'>
					Discover the latest trends in eco-friendly fashion
				</p>

				<div className='grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3
gap-4'>
					{categories.map((category) => (
						<CategoryItem category={category}
key={category.name} />
					))}
				</div>

				{!isLoading && products.length > 0 && <FeaturedProducts
featuredProducts={products} />}
			</div>
		</div>
	);
};
export default HomePage;
```

CategoryItem.jsx

```
import { Link } from "react-router-dom";

const CategoryItem = ({ category }) => {
	return (
		<div className='relative overflow-hidden h-96 w-full rounded-lg
group'>
			<Link to={"/category" + category.href}>
				<div className='w-full h-full cursor-pointer'>
					<div className='absolute inset-0 bg-gradient-to-b
from-transparent to-gray-900 opacity-50 z-10' />
					<img
						src={category.imageUrl}
						alt={category.name}
```

```
                                        className='w-full h-full object-cover
transition-transform duration-500 ease-out group-hover:scale-110'
                                        loading='lazy'
                                    />
                                    <div className='absolute bottom-0 left-0 right-0
p-4 z-20'>
                                        <h3 className='text-white text-2xl
font-bold mb-2'>{category.name}</h3>
                                        <p className='text-gray-200
text-sm'>Explore {category.name}</p>
                                    </div>
                                </div>
                            </Link>
                        </div>
                );
};

export default CategoryItem;
```

FeaturedProducts.jsx

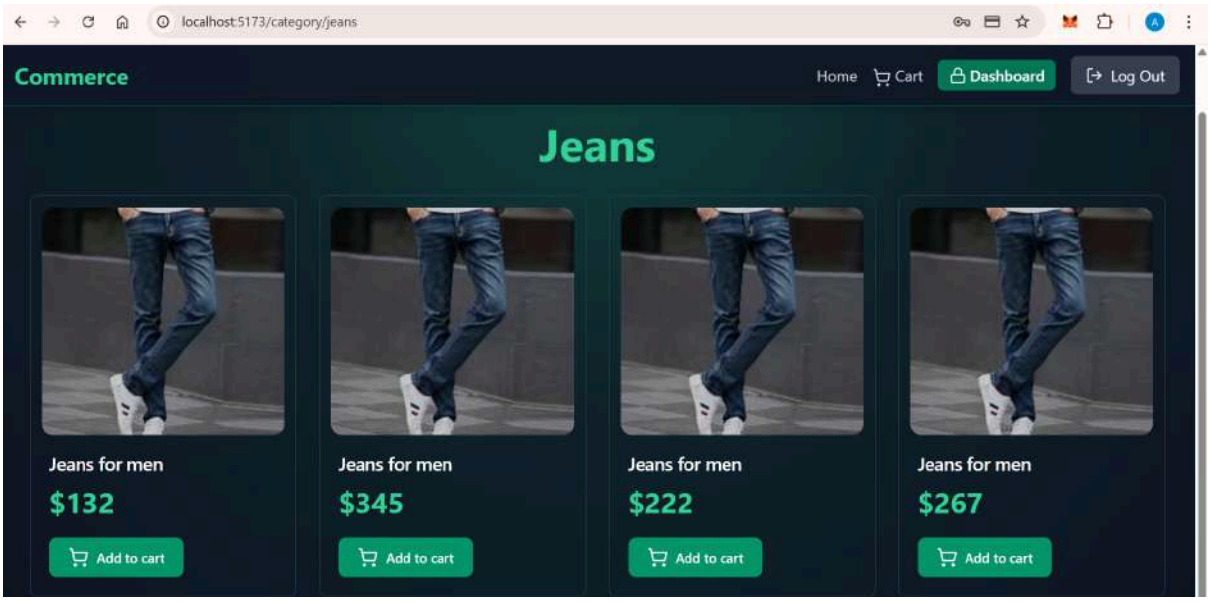
```
import { useEffect, useState } from "react";
import { ShoppingCart, ChevronLeft, ChevronRight } from "lucide-react";
import { useCartStore } from "../stores/useCartStore";

const FeaturedProducts = ({ featuredProducts }) => {
        const [currentIndex, setCurrentIndex] = useState(0);
        const [itemsPerPage, setItemsPerPage] = useState(4);

        const { addToCart } = useCartStore();

        useEffect(() => {
                const handleResize = () => {
                        if (window.innerWidth < 640) setItemsPerPage(1);
                        else if (window.innerWidth < 1024) setItemsPerPage(2);
                        else if (window.innerWidth < 1280) setItemsPerPage(3);
                        else setItemsPerPage(4);
                };

                handleResize();
                window.addEventListener("resize", handleResize);
                return () => window.removeEventListener("resize", handleResize);
        }, []);
```

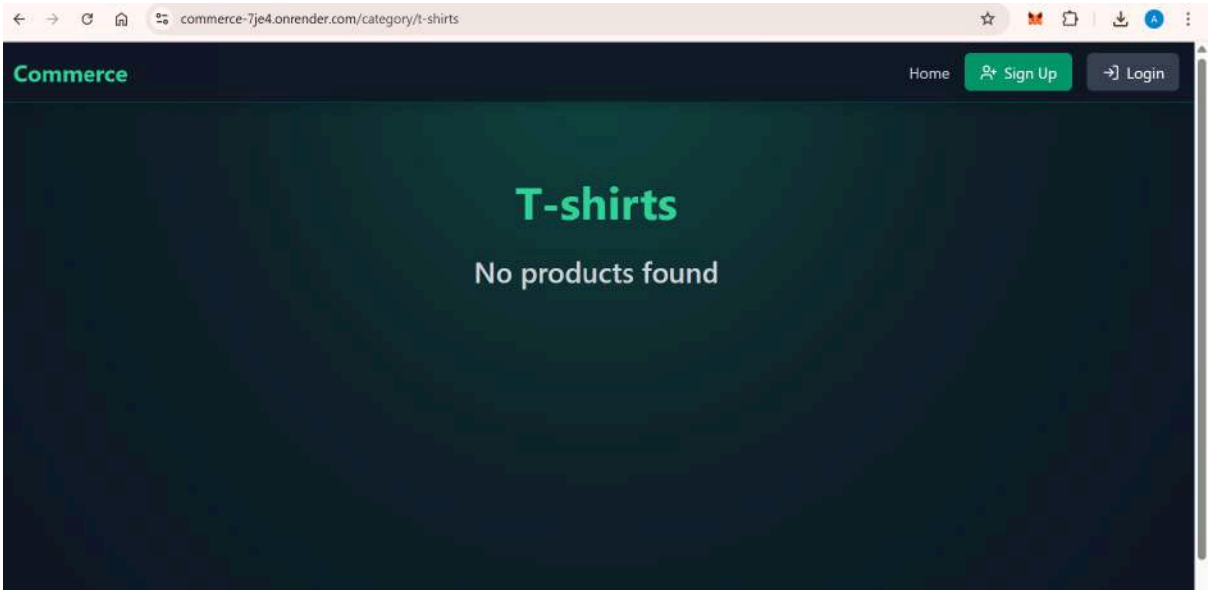
```
	const nextSlide = () => {
		setCurrentIndex((prevIndex) => prevIndex + itemsPerPage);
	};

	const prevSlide = () => {
		setCurrentIndex((prevIndex) => prevIndex - itemsPerPage);
	};

	const isStartDisabled = currentIndex === 0;
	const isEndDisabled = currentIndex >= featuredProducts.length -
itemsPerPage;

	return (
		<div className='py-12'>
			<div className='container mx-auto px-4'>
				<h2 className='text-center text-5xl sm:text-6xl font-bold
text-emerald-400 mb-4'>Featured</h2>
				<div className='relative'>
					<div className='overflow-hidden'>
						<div
							className='flex transition-transform
duration-300 ease-in-out'
							style={{ transform:
`translateX(-${currentIndex * (100 / itemsPerPage)}%)` }}
						>
							{featuredProducts?.map((product) =>
(
								<div key={product._id}
className='w-full sm:w-1/2 lg:w-1/3 xl:w-1/4 flex-shrink-0 px-2'>
<div
className='bg-white bg-opacity-10 backdrop-blur-sm rounded-lg shadow-lg
overflow-hidden h-full transition-all duration-300 hover:shadow-xl border
border-emerald-500/30'>
<div
className='overflow-hidden'>
<img

src={product.image}

alt={product.name}

className='w-full h-48 object-cover transition-transform duration-300 ease-in-out
hover:scale-110'
/>
```

```
                                                                    </div>
                                                                    <div
                                                                    <h3
                                                                    <p
                                                                    </p>
                                                                    <button
                                                                    flex
                                                                    >
                                                                    </button>
                                                                </div>
                                                            </div>
                                                        </div>
                                                    ))}
                                                </div>
                                            </div>
                                            <button
                                                onClick={prevSlide}
                                                disabled={isStartDisabled}
                                                >
                                                <ChevronLeft />
                                            </button>

                                            <button
                                                onClick={nextSlide}
```

```
                                        disabled={isEndDisabled}
                                        className={`absolute top-1/2 -right-4
transform -translate-y-1/2 p-2 rounded-full transition-colors duration-300 ${
                                                isEndDisabled ? "bg-gray-400
cursor-not-allowed" : "bg-emerald-600 hover:bg-emerald-500"
                                        }`}
                                >
                                        <ChevronRight className='w-6 h-6' />
                                </button>
                        </div>
                </div>
        </div>
    );
};
export default FeaturedProducts;
```

2.Product List





CategoryPage.jsx

```
import { useEffect } from "react";
import { useProductStore } from "../stores/useProductStore";
import { useParams } from "react-router-dom";
import { motion } from "framer-motion";
import ProductCard from "../components/ProductCard";

const CategoryPage = () => {
	const { fetchProductsByCategory, products } = useProductStore();

	const { category } = useParams();

	useEffect(() => {
		fetchProductsByCategory(category);
	}, [fetchProductsByCategory, category]);

	console.log("products:", products);
	return (
		<div className='min-h-screen'>
			<div className='relative z-10 max-w-screen-xl mx-auto px-4
sm:px-6 lg:px-8 py-16'>
				<motion.h1
					className='text-center text-4xl sm:text-5xl
font-bold text-emerald-400 mb-8'
					initial={{ opacity: 0, y: -20 }}
					animate={{ opacity: 1, y: 0 }}
					transition={{ duration: 0.8 }}
				>
```

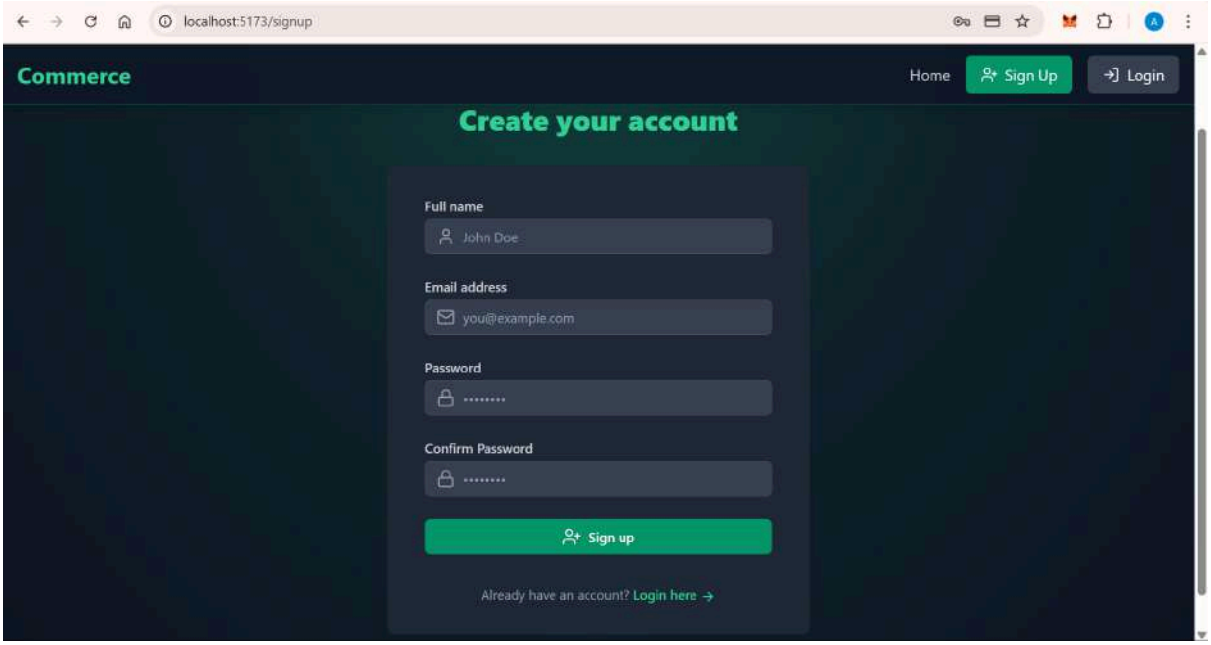
```
                                        {category.charAt(0).toUpperCase() +
category.slice(1)}
                                </motion.h1>

                                <motion.div
                                        className='grid grid-cols-1 sm:grid-cols-2
lg:grid-cols-3 xl:grid-cols-4 gap-6 justify-items-center'
                                        initial={{ opacity: 0, y: 20 }}
                                        animate={{ opacity: 1, y: 0 }}
                                        transition={{ duration: 0.8, delay: 0.2 }}
                                >
                                        {products?.length === 0 && (
                                                <h2 className='text-3xl font-semibold
text-gray-300 text-center col-span-full'>
                                                        No products found
                                                </h2>
                                        )}

                                        {products?.map((product) => (
                                                <ProductCard key={product._id}
product={product} />
                                        ))}
                                </motion.div>
                        </div>
                </div>
        );
};
export default CategoryPage;
```

ProductCard.jsx

```
import toast from "react-hot-toast";
import { ShoppingCart } from "lucide-react";
import { useUserStore } from "../stores/useUserStore";
import { useCartStore } from "../stores/useCartStore";

const ProductCard = ({ product }) => {
        const { user } = useUserStore();
        const { addToCart } = useCartStore();
        const handleAddToCart = () => {
                if (!user) {
                        toast.error("Please login to add products to cart", { id: "login" });
                        return;
                } else {
```

```
                // add to cart
                addToCart(product);
            }
        };

        return (
            <div className='flex w-full relative flex-col overflow-hidden rounded-lg
border border-gray-700 shadow-lg'>
                <div className='relative mx-3 mt-3 flex h-60 overflow-hidden
rounded-xl'>
                    <img className='object-cover w-full'
src={product.image} alt='product image' />
                    <div className='absolute inset-0 bg-black bg-opacity-20'
/>
                </div>

                <div className='mt-4 px-5 pb-5'>
                    <h5 className='text-xl font-semibold tracking-tight
text-white'>{product.name}</h5>
                    <div className='mt-2 mb-5 flex items-center
justify-between'>
                        <p>
                            <span className='text-3xl font-bold
text-emerald-400'>${product.price}</span>
                        </p>
                    </div>
                    <button
                        className='flex items-center justify-center
rounded-lg bg-emerald-600 px-5 py-2.5 text-center text-sm font-medium
                         text-white hover:bg-emerald-700
focus:outline-none focus:ring-4 focus:ring-emerald-300'
                        onClick={handleAddToCart}
                    >
                        <ShoppingCart size={22} className='mr-2' />
                        Add to cart
                    </button>
                </div>
            </div>
        );
};
export default ProductCard;
```

3.Sign Up



SignUpPage.jsx

```
import { useState } from "react";
import { Link } from "react-router-dom";
import { UserPlus, Mail, Lock, User, ArrowRight, Loader } from "lucide-react";
import { motion } from "framer-motion";
import { useUserStore } from "../stores/useUserStore";

const SignUpPage = () => {
	const [formData, setFormData] = useState({
		name: "",
		email: "",
		password: "",
		confirmPassword: "",
	});

	const { signup, loading } = useUserStore();

	const handleSubmit = (e) => {
		e.preventDefault();
		signup(formData);
	};

	return (
		<div className='flex flex-col justify-center py-12 sm:px-6 lg:px-8'>
			<motion.div
```

```
                        className='sm:mx-auto sm:w-full sm:max-w-md'
                        initial={{ opacity: 0, y: -20 }}
                        animate={{ opacity: 1, y: 0 }}
                        transition={{ duration: 0.8 }}
                    >
                        <h2 className='mt-6 text-center text-3xl font-extrabold
text-emerald-400'>Create your account</h2>
                    </motion.div>

                    <motion.div
                        className='mt-8 sm:mx-auto sm:w-full sm:max-w-md'
                        initial={{ opacity: 0, y: 20 }}
                        animate={{ opacity: 1, y: 0 }}
                        transition={{ duration: 0.8, delay: 0.2 }}
                    >
                        <div className='bg-gray-800 py-8 px-4 shadow
sm:rounded-lg sm:px-10'>
                            <form onSubmit={handleSubmit}
className='space-y-6'>
                                <div>
                                    <label htmlFor='name'
className='block text-sm font-medium text-gray-300'>
                                        Full name
                                    </label>
                                    <div className='mt-1 relative
rounded-md shadow-sm'>
                                        <div className='absolute
inset-y-0 left-0 pl-3 flex items-center pointer-events-none'>
                                            <User className='h-5
w-5 text-gray-400' aria-hidden='true' />
                                        </div>
                                        <input
                                            id='name'
                                            type='text'
                                            required
                                            value={formData.name}
                                            onChange={(e) =>
setFormData({ ...formData, name: e.target.value })}
                                            className='block
w-full px-3 py-2 pl-10 bg-gray-700 border border-gray-600 rounded-md shadow-sm
                                             placeholder-gray-400
focus:outline-none focus:ring-emerald-500 focus:border-emerald-500 sm:text-sm'
                                            placeholder='John Doe'
                                        />
```

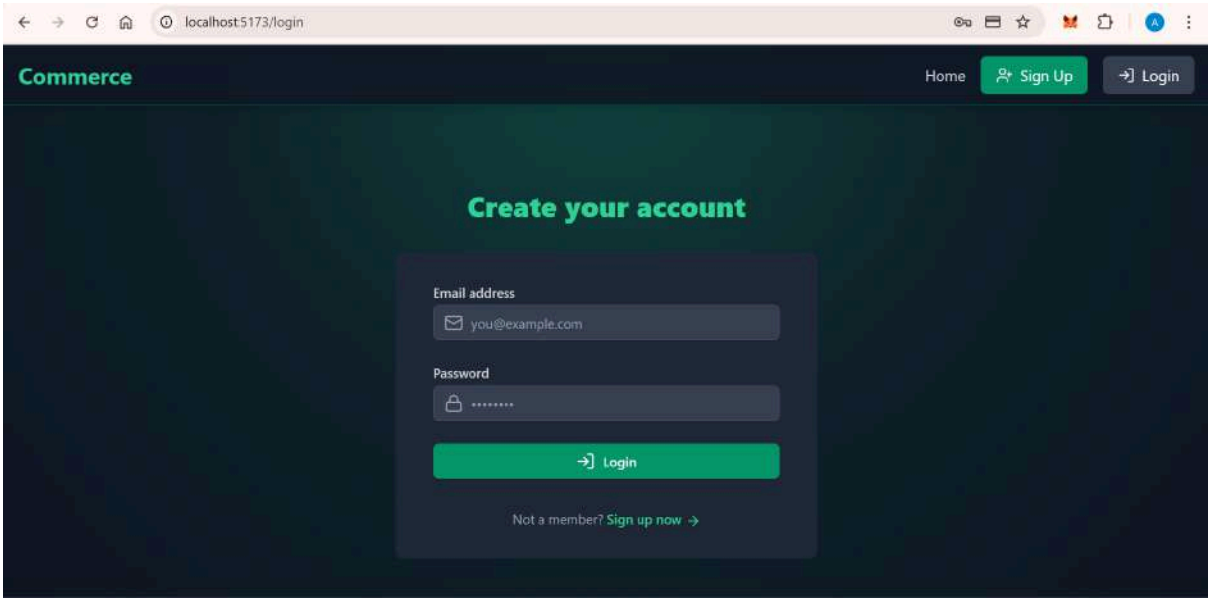
```
                    </div>
                </div>

                <div>
                    <label htmlFor='email'
classNam e='block text-sm font-medium text-gray-300'>
                        Email address
                    </label>
                    <div className='mt-1 relative
rounded-md shadow-sm'>
                        <div className='absolute
inset-y-0 left-0 pl-3 flex items-center pointer-events-none'>
                            <Mail className='h-5
w-5 text-gray-400' aria-hidden='true' />
                        </div>
                        <input
                            id='email'
                            type='email'
                            required
                            value={formData.email}
                            onChange={(e) =>
setFormData({ ...formData, email: e.target.value })}
                            className=' block
w-full px-3 py-2 pl-10 bg-gray-700 border border-gray-600
                            rounded-md
shadow-sm
                             placeholder-gray-400
focus:outline-none focus:ring-emerald-500

focus:border-emerald-500 sm:text-sm'

placeholder='you@example.com'
                        />
                    </div>
                </div>

                <div>
                    <label htmlFor='password'
classNam e='block text-sm font-medium text-gray-300'>
                        Password
                    </label>
                    <div className='mt-1 relative
rounded-md shadow-sm'>
```

```
                                                    <div className='absolute
inset-y-0 left-0 pl-3 flex items-center pointer-events-none'>
                                                        <Lock className='h-5
w-5 text-gray-400' aria-hidden='true' />
                                                    </div>
                                                    <input
                                                        id='password'
                                                        type='password'
                                                        required

value={formData.password}
                                                        onChange={(e) =>
setFormData({ ...formData, password: e.target.value })}
                                                        className=' block
w-full px-3 py-2 pl-10 bg-gray-700 border border-gray-600
                                                        rounded-md
shadow-sm placeholder-gray-400 focus:outline-none focus:ring-emerald-500
focus:border-emerald-500 sm:text-sm'
                                                        placeholder='••••••••'
                                                    />
                                                </div>
                                            </div>

                                            <div>
                                                <label htmlFor='confirmPassword'
className='block text-sm font-medium text-gray-300'>
                                                    Confirm Password
                                                </label>
                                                <div className='mt-1 relative
rounded-md shadow-sm'>
                                                    <div className='absolute
inset-y-0 left-0 pl-3 flex items-center pointer-events-none'>
                                                        <Lock className='h-5
w-5 text-gray-400' aria-hidden='true' />
                                                    </div>
                                                    <input
                                                        id='confirmPassword'
                                                        type='password'
                                                        required

value={formData.confirmPassword}
                                                        onChange={(e) =>
setFormData({ ...formData, confirmPassword: e.target.value })}
```

```
                                                    className=' block
w-full px-3 py-2 pl-10 bg-gray-700 border
                                                     border-gray-600
rounded-md shadow-sm placeholder-gray-400 focus:outline-none
focus:ring-emerald-500 focus:border-emerald-500 sm:text-sm'
                                                    placeholder='••••••••'
                                                />
                                            </div>
                                        </div>

                                        <button
                                            type='submit'
                                            className='w-full flex justify-center
py-2 px-4 border border-transparent
                                            rounded-md shadow-sm text-sm
font-medium text-white bg-emerald-600
                                             hover:bg-emerald-700
focus:outline-none focus:ring-2 focus:ring-offset-2
                                              focus:ring-emerald-500 transition
duration-150 ease-in-out disabled:opacity-50'
                                            disabled={loading}
                                        >
                                            {loading ? (
                                                <>
                                                    <Loader
className='mr-2 h-5 w-5 animate-spin' aria-hidden='true' />
                                                    Loading...
                                                </>
                                            ) : (
                                                <>
                                                    <UserPlus
className='mr-2 h-5 w-5' aria-hidden='true' />
                                                    Sign up
                                                </>
                                            )}
                                        </button>
                                    </form>

                                    <p className='mt-8 text-center text-sm
text-gray-400'>
                                        Already have an account?{" "}
                                        <Link to='/login' className='font-medium
text-emerald-400 hover:text-emerald-300'>
```

```
                                                    Login here <ArrowRight
className='inline h-4 w-4' />
                                                </Link>
                                            </p>
                                        </div>
                                    </motion.div>
                                </div>
                            );
};
export default SignUpPage;
```

4.Log In



LoginPage.jsx

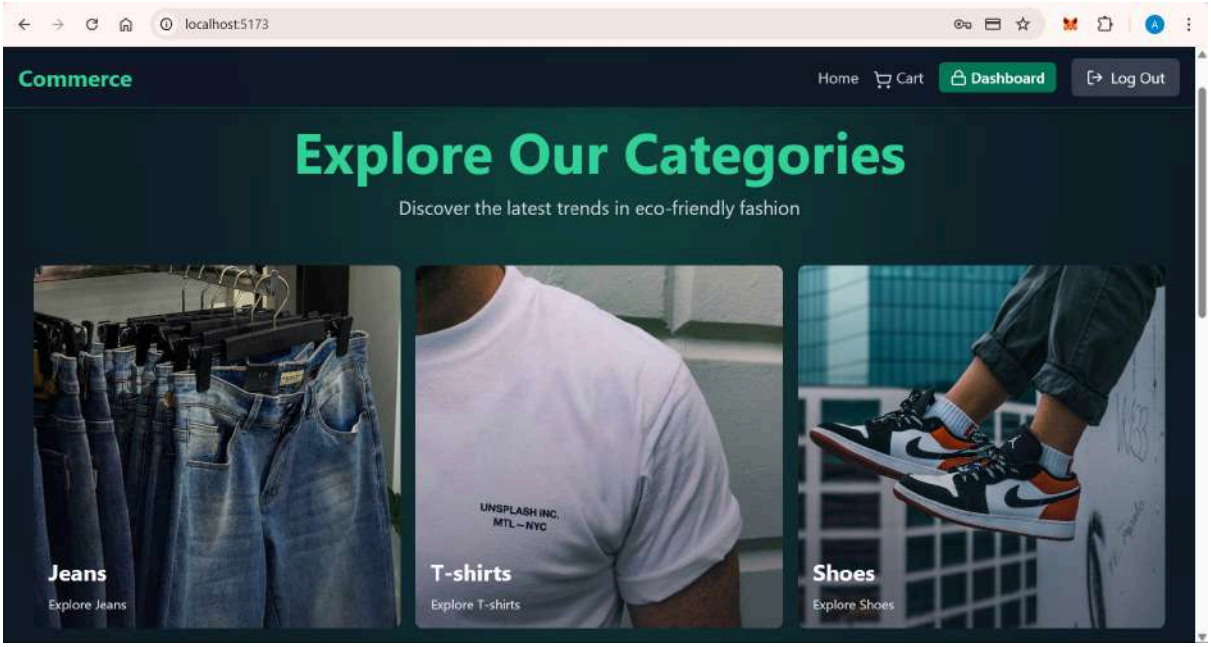
```
import { useState } from "react";
import { motion } from "framer-motion";
import { Link } from "react-router-dom";
import { LogIn, Mail, Lock, ArrowRight, Loader } from "lucide-react";
import { useUserStore } from "../stores/useUserStore";

const LoginPage = () => {
	const [email, setEmail] = useState("");
	const [password, setPassword] = useState("");

	const { login, loading } = useUserStore();

	const handleSubmit = (e) => {
```

```
            e.preventDefault();
            console.log(email, password);
            login(email, password);
        };

        return (
            <div className='flex flex-col justify-center py-12 sm:px-6 lg:px-8'>
                <motion.div
                    className='sm:mx-auto sm:w-full sm:max-w-md'
                    initial={{ opacity: 0, y: -20 }}
                    animate={{ opacity: 1, y: 0 }}
                    transition={{ duration: 0.8 }}
                >
                    <h2 className='mt-6 text-center text-3xl font-extrabold
text-emerald-400'>Create your account</h2>
                </motion.div>

                <motion.div
                    className='mt-8 sm:mx-auto sm:w-full sm:max-w-md'
                    initial={{ opacity: 0, y: 20 }}
                    animate={{ opacity: 1, y: 0 }}
                    transition={{ duration: 0.8, delay: 0.2 }}
                >
                    <div className='bg-gray-800 py-8 px-4 shadow
sm:rounded-lg sm:px-10'>
                        <form onSubmit={handleSubmit}
className='space-y-6'>
                            <div>
                                <label htmlFor='email'
className='block text-sm font-medium text-gray-300'>
                                    Email address
                                </label>
                                <div className='mt-1 relative
rounded-md shadow-sm'>
                                    <div className='absolute
inset-y-0 left-0 pl-3 flex items-center pointer-events-none'>
                                        <Mail className='h-5
w-5 text-gray-400' aria-hidden='true' />
                                    </div>
                                    <input
                                        id='email'
                                        type='email'
                                        required
                                        value={email}
```

```
                                                    onChange={(e) =>
setEmail(e.target.value)}
                                                    className=' block
w-full px-3 py-2 pl-10 bg-gray-700 border border-gray-600
                                                    rounded-md
shadow-sm
                                                     placeholder-gray-400
focus:outline-none focus:ring-emerald-500

focus:border-emerald-500 sm:text-sm'

placeholder='you@example.com'
                                                        />
                                                </div>
                                            </div>

                                            <div>
                                                <label htmlFor='password'
className='block text-sm font-medium text-gray-300'>
                                                    Password
                                                </label>
                                                <div className='mt-1 relative
rounded-md shadow-sm'>
                                                    <div className='absolute
inset-y-0 left-0 pl-3 flex items-center pointer-events-none'>
                                                        <Lock className='h-5
w-5 text-gray-400' aria-hidden='true' />
                                                    </div>
                                                    <input
                                                        id='password'
                                                        type='password'
                                                        required
                                                        value={password}
                                                        onChange={(e) =>
setPassword(e.target.value)}
                                                        className=' block
w-full px-3 py-2 pl-10 bg-gray-700 border border-gray-600
                                                        rounded-md
shadow-sm placeholder-gray-400 focus:outline-none focus:ring-emerald-500
focus:border-emerald-500 sm:text-sm'
                                                        placeholder='••••••••'
                                                    />
                                                </div>
                                            </div>
```

```
                                        <button
                                                type='submit'
                                                className='w-full flex justify-center
py-2 px-4 border border-transparent
                                                rounded-md shadow-sm text-sm
font-medium text-white bg-emerald-600
                                                 hover:bg-emerald-700
focus:outline-none focus:ring-2 focus:ring-offset-2
                                                  focus:ring-emerald-500 transition
duration-150 ease-in-out disabled:opacity-50'
                                                disabled={loading}
                                        >
                                                {loading ? (
                                                        <>
                                                                <Loader
className='mr-2 h-5 w-5 animate-spin' aria-hidden='true' />
                                                                Loading...
                                                        </>
                                                ) : (
                                                        <>
                                                                <LogIn
className='mr-2 h-5 w-5' aria-hidden='true' />
                                                                Login
                                                        </>
                                                )}
                                        </button>
                                </form>

                                <p className='mt-8 text-center text-sm
text-gray-400'>
                                        Not a member?{" "}
                                        <Link to='/signup' className='font-medium
text-emerald-400 hover:text-emerald-300'>
                                                Sign up now <ArrowRight
className='inline h-4 w-4' />
                                        </Link>
                                </p>
                        </div>
                </motion.div>
        </div>
    );
};
export default LoginPage;
```

5.After login and signup Admin page



AdminPage.jsx

```
import { BarChart, PlusCircle, ShoppingBasket } from "lucide-react";
import { useEffect, useState } from "react";
import { motion } from "framer-motion";

import AnalyticsTab from "../components/AnalyticsTab";
import CreateProductForm from "../components/CreateProductForm";
import ProductsList from "../components/ProductsList";
import { useProductStore } from "../stores/useProductStore";

const tabs = [
	{ id: "create", label: "Create Product", icon: PlusCircle },
	{ id: "products", label: "Products", icon: ShoppingBasket },
	{ id: "analytics", label: "Analytics", icon: BarChart },
];

const AdminPage = () => {
	const [activeTab, setActiveTab] = useState("create");
	const { fetchAllProducts } = useProductStore();

	useEffect(() => {
		fetchAllProducts();
	}, [fetchAllProducts]);
```

```
	return (
		<div className='min-h-screen relative overflow-hidden'>
			<div className='relative z-10 container mx-auto px-4 py-16'>
				<motion.h1
					className='text-4xl font-bold mb-8
text-emerald-400 text-center'
					initial={{ opacity: 0, y: -20 }}
					animate={{ opacity: 1, y: 0 }}
					transition={{ duration: 0.8 }}
				>
					Admin Dashboard
				</motion.h1>

				<div className='flex justify-center mb-8'>
					{tabs.map((tab) => (
						<button
							key={tab.id}
							onClick={() => setActiveTab(tab.id)}
							className={`flex items-center px-4
py-2 mx-2 rounded-md transition-colors duration-200 ${
								activeTab === tab.id
									? "bg-emerald-600
text-white"
									: "bg-gray-700
text-gray-300 hover:bg-gray-600"
							}`}
						>
							<tab.icon className='mr-2 h-5 w-5'
/>
							{tab.label}
						</button>
					))}
				</div>
				{activeTab === "create" && <CreateProductForm />}
				{activeTab === "products" && <ProductsList />}
				{activeTab === "analytics" && <AnalyticsTab />}
			</div>
		</div>
	);
};
export default AdminPage;
```

ProductsList.jsx

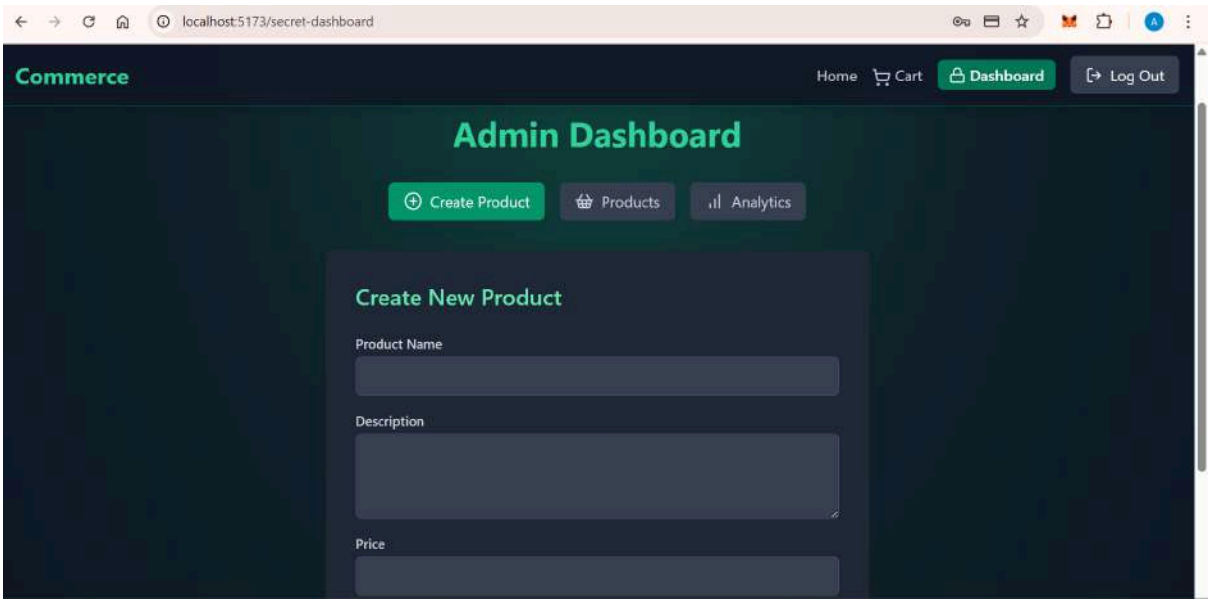
```
import { motion } from "framer-motion";
import { Trash, Star } from "lucide-react";
import { useProductStore } from "../stores/useProductStore";

const ProductsList = () => {
	const { deleteProduct, toggleFeaturedProduct, products } =
useProductStore();

	console.log("products", products);

	return (
		<motion.div
			className='bg-gray-800 shadow-lg rounded-lg overflow-hidden
max-w-4xl mx-auto'
			initial={{ opacity: 0, y: 20 }}
			animate={{ opacity: 1, y: 0 }}
			transition={{ duration: 0.8 }}
		>
			<table className=' min-w-full divide-y divide-gray-700'>
				<thead className='bg-gray-700'>
					<tr>
						<th
							scope='col'
							className='px-6 py-3 text-left
text-xs font-medium text-gray-300 uppercase tracking-wider'
						>
							Product
						</th>
						<th
							scope='col'
							className='px-6 py-3 text-left
text-xs font-medium text-gray-300 uppercase tracking-wider'
						>
							Price
						</th>
						<th
							scope='col'
							className='px-6 py-3 text-left
text-xs font-medium text-gray-300 uppercase tracking-wider'
						>
							Category
						</th>
```

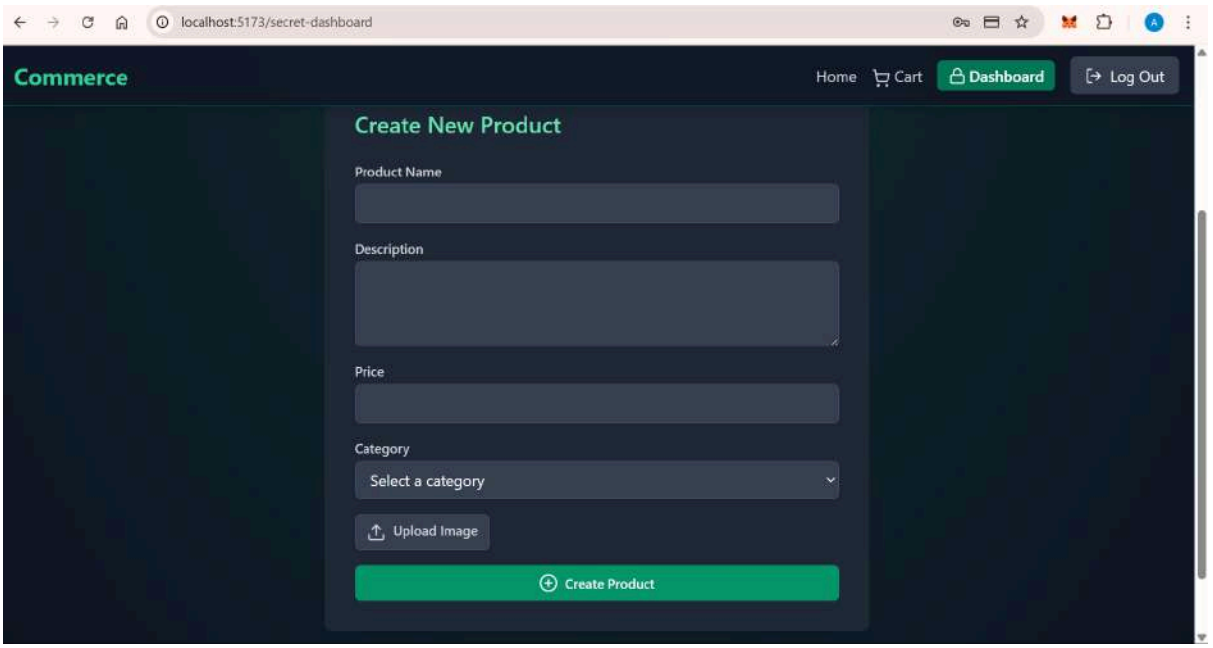
```
                                        <th
                                            scope='col'
                                            className='px-6 py-3 text-left
text-xs font-medium text-gray-300 uppercase tracking-wider'
                                        >
                                            Featured
                                        </th>
                                        <th
                                            scope='col'
                                            className='px-6 py-3 text-left
text-xs font-medium text-gray-300 uppercase tracking-wider'
                                        >
                                            Actions
                                        </th>
                                    </tr>
                                </thead>

                                <tbody className='bg-gray-800 divide-y
divide-gray-700'>
                                    {products?.map((product) => (
                                        <tr key={product._id}
className='hover:bg-gray-700'>
                                            <td className='px-6 py-4
whitespace-nowrap'>
                                                <div className='flex
items-center'>
                                                    <div
className='flex-shrink-0 h-10 w-10'>
                                                        <img

className='h-10 w-10 rounded-full object-cover'

src={product.image}

alt={product.name}
                                                        />
                                                    </div>
                                                    <div className='ml-4'>
                                                        <div
className='text-sm font-medium text-white'>{product.name}</div>
                                                    </div>
                                                </div>
                                            </td>
```

```
                                                <td className='px-6 py-4
whitespace-nowrap'>
                                                    <div className='text-sm
text-gray-300'>${product.price.toFixed(2)}</div>
                                                </td>
                                                <td className='px-6 py-4
whitespace-nowrap'>
                                                    <div className='text-sm
text-gray-300'>{product.category}</div>
                                                </td>
                                                <td className='px-6 py-4
whitespace-nowrap'>
                                                    <button
                                                        onClick={() =>
toggleFeaturedProduct(product._id)}
                                                        className={`p-1
rounded-full ${

product.isFeatured ? "bg-yellow-400 text-gray-900" : "bg-gray-600 text-gray-300"
                                                        } hover:bg-yellow-500
transition-colors duration-200`}
                                                    >
                                                        <Star className='h-5
w-5' />
                                                    </button>
                                                </td>
                                                <td className='px-6 py-4
whitespace-nowrap text-sm font-medium'>
                                                    <button
                                                        onClick={() =>
deleteProduct(product._id)}

className='text-red-400 hover:text-red-300'
                                                    >
                                                        <Trash className='h-5
w-5' />
                                                    </button>
                                                </td>
                                            </tr>
                                        ))}
                                    </tbody>
                                </table>
                            </motion.div>
                        );
```

};
export default ProductsList;

6.Create Product





CreateProductForm.jsx

```
import { useState } from "react";
import { motion } from "framer-motion";
import { PlusCircle, Upload, Loader } from "lucide-react";
import { useProductStore } from "../stores/useProductStore";
```

```
const categories = ["jeans", "t-shirts", "shoes", "glasses", "jackets", "suits", "bags"];

const CreateProductForm = () => {
	const [newProduct, setNewProduct] = useState({
		name: "",
		description: "",
		price: "",
		category: "",
		image: "",
	});

	const { createProduct, loading } = useProductStore();

	const handleSubmit = async (e) => {
		e.preventDefault();
		try {
			await createProduct(newProduct);
			setNewProduct({ name: "", description: "", price: "", category: "",
image: "" });
		} catch {
			console.log("error creating a product");
		}
	};

	const handleImageChange = (e) => {
		const file = e.target.files[0];
		if (file) {
			const reader = new FileReader();

			reader.onloadend = () => {
				setNewProduct({ ...newProduct, image: reader.result });
			};

			reader.readAsDataURL(file); // base64
		}
	};

	return (
		<motion.div
			className='bg-gray-800 shadow-lg rounded-lg p-8 mb-8
max-w-xl mx-auto'
			initial={{ opacity: 0, y: 20 }}
			animate={{ opacity: 1, y: 0 }}
```

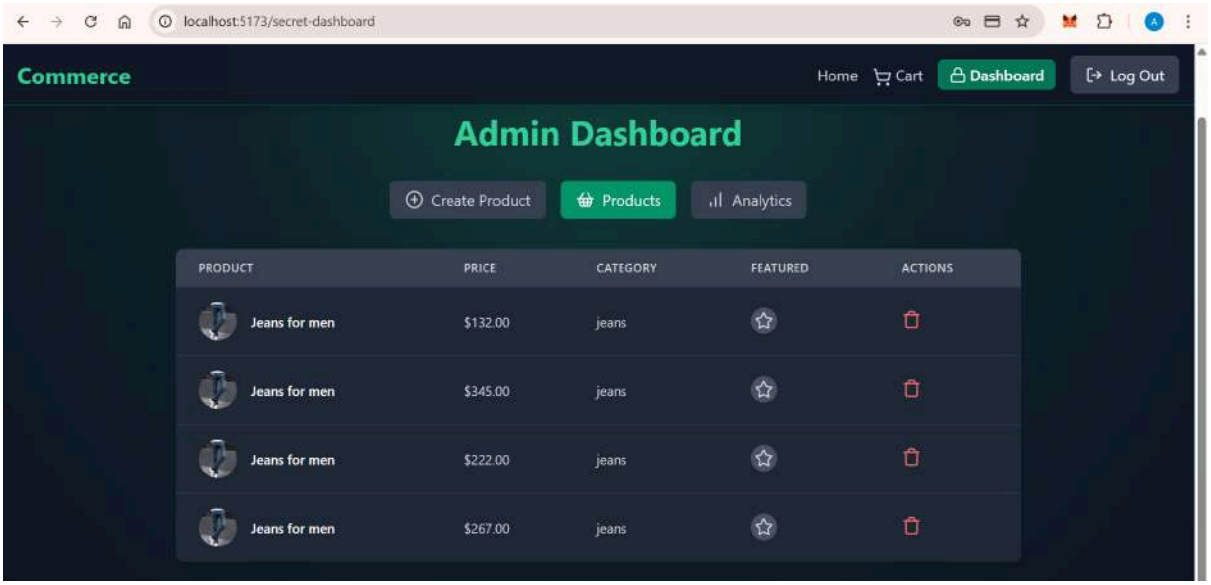
```
                transition={{ duration: 0.8 }}
            >
                <h2 className='text-2xl font-semibold mb-6
text-emerald-300'>Create New Product</h2>

                <form onSubmit={handleSubmit} className='space-y-4'>
                    <div>
                        <label htmlFor='name' className='block text-sm
font-medium text-gray-300'>
                            Product Name
                        </label>
                        <input
                            type='text'
                            id='name'
                            name='name'
                            value={newProduct.name}
                            onChange={(e) => setNewProduct({
...newProduct, name: e.target.value })}
                            className='mt-1 block w-full bg-gray-700
border border-gray-600 rounded-md shadow-sm py-2
                             px-3 text-white focus:outline-none
focus:ring-2
                            focus:ring-emerald-500
focus:border-emerald-500'
                            required
                        />
                    </div>

                    <div>
                        <label htmlFor='description' className='block
text-sm font-medium text-gray-300'>
                            Description
                        </label>
                        <textarea
                            id='description'
                            name='description'
                            value={newProduct.description}
                            onChange={(e) => setNewProduct({
...newProduct, description: e.target.value })}
                            rows='3'
                            className='mt-1 block w-full bg-gray-700
border border-gray-600 rounded-md shadow-sm
                             py-2 px-3 text-white focus:outline-none
focus:ring-2 focus:ring-emerald-500
```

```
                                        focus:border-emerald-500'
                                        required
                                    />
                                </div>

                                <div>
                                    <label htmlFor='price' className='block text-sm
font-medium text-gray-300'>
                                        Price
                                    </label>
                                    <input
                                        type='number'
                                        id='price'
                                        name='price'
                                        value={newProduct.price}
                                        onChange={(e) => setNewProduct({
...newProduct, price: e.target.value })}
                                        step='0.01'
                                        className='mt-1 block w-full bg-gray-700
border border-gray-600 rounded-md shadow-sm
                                        py-2 px-3 text-white focus:outline-none
focus:ring-2 focus:ring-emerald-500
                                        focus:border-emerald-500'
                                        required
                                    />
                                </div>

                                <div>
                                    <label htmlFor='category' className='block
text-sm font-medium text-gray-300'>
                                        Category
                                    </label>
                                    <select
                                        id='category'
                                        name='category'
                                        value={newProduct.category}
                                        onChange={(e) => setNewProduct({
...newProduct, category: e.target.value })}
                                        className='mt-1 block w-full bg-gray-700
border border-gray-600 rounded-md
                                        shadow-sm py-2 px-3 text-white
focus:outline-none
                                        focus:ring-2 focus:ring-emerald-500
focus:border-emerald-500'
```

```
                                        required
                                >
                                        <option value="">Select a category</option>
                                        {categories.map((category) => (
                                                <option key={category}
value={category}>
                                                        {category}
                                                </option>
                                        ))}
                                </select>
                        </div>

                        <div className='mt-1 flex items-center'>
                                <input type='file' id='image' className='sr-only'
accept='image/*' onChange={handleImageChange} />
                                <label
                                        htmlFor='image'
                                        className='cursor-pointer bg-gray-700
py-2 px-3 border border-gray-600 rounded-md shadow-sm text-sm leading-4
font-medium text-gray-300 hover:bg-gray-600 focus:outline-none focus:ring-2
focus:ring-offset-2 focus:ring-emerald-500'
                                >
                                        <Upload className='h-5 w-5 inline-block
mr-2' />
                                        Upload Image
                                </label>
                                {newProduct.image && <span className='ml-3
text-sm text-gray-400'>Image uploaded </span>}
                        </div>

                        <button
                                type='submit'
                                className='w-full flex justify-center py-2 px-4
border border-transparent rounded-md
                                shadow-sm text-sm font-medium text-white
bg-emerald-600 hover:bg-emerald-700
                                focus:outline-none focus:ring-2 focus:ring-offset-2
focus:ring-emerald-500 disabled:opacity-50'
                                disabled={loading}
                        >
                                {loading ? (
                                        <>
                                                <Loader className='mr-2 h-5 w-5
animate-spin' aria-hidden='true' />
```

```
                                        Loading...
                                    </>
                                ) : (
                                    <>
                                        <PlusCircle className='mr-2 h-5
w-5' />
                                        Create Product
                                    </>
                                )}
                            </button>
                        </form>
                    </motion.div>
                );
};
export default CreateProductForm;
```

7.Featured Product



FeaturedProducts.jsx

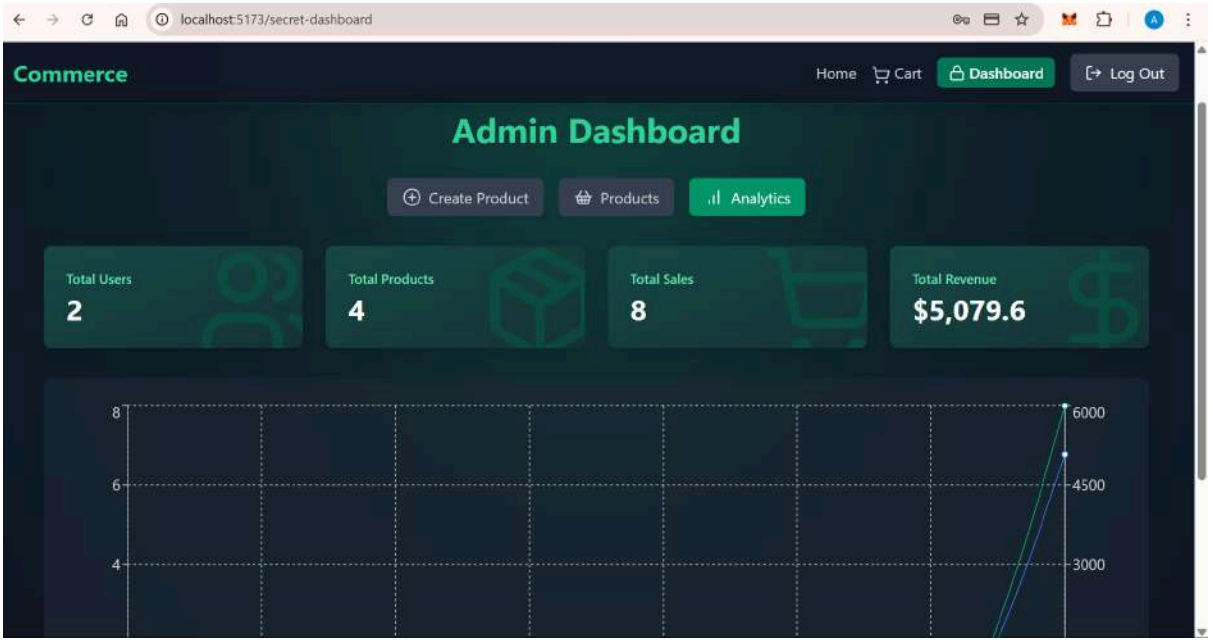
```
import { useEffect, useState } from "react";
import { ShoppingCart, ChevronLeft, ChevronRight } from "lucide-react";
import { useCartStore } from "../stores/useCartStore";

const FeaturedProducts = ({ featuredProducts }) => {
        const [currentIndex, setCurrentIndex] = useState(0);
        const [itemsPerPage, setItemsPerPage] = useState(4);

        const { addToCart } = useCartStore();
```

```
	useEffect(() => {
		const handleResize = () => {
			if (window.innerWidth < 640) setItemsPerPage(1);
			else if (window.innerWidth < 1024) setItemsPerPage(2);
			else if (window.innerWidth < 1280) setItemsPerPage(3);
			else setItemsPerPage(4);
		};

		handleResize();
		window.addEventListener("resize", handleResize);
		return () => window.removeEventListener("resize", handleResize);
	}, []);

	const nextSlide = () => {
		setCurrentIndex((prevIndex) => prevIndex + itemsPerPage);
	};

	const prevSlide = () => {
		setCurrentIndex((prevIndex) => prevIndex - itemsPerPage);
	};

	const isStartDisabled = currentIndex === 0;
	const isEndDisabled = currentIndex >= featuredProducts.length -
itemsPerPage;

	return (
		<div className='py-12'>
			<div className='container mx-auto px-4'>
				<h2 className='text-center text-5xl sm:text-6xl font-bold
text-emerald-400 mb-4'>Featured</h2>
				<div className='relative'>
					<div className='overflow-hidden'>
						<div
							className='flex transition-transform
duration-300 ease-in-out'
							style={{ transform:
`translateX(-${currentIndex * (100 / itemsPerPage)}%)` }}
						>
							{featuredProducts?.map((product) =>
(
								<div key={product._id}
className='w-full sm:w-1/2 lg:w-1/3 xl:w-1/4 flex-shrink-0 px-2'>
```

```
                                                    <div
className='bg-white bg-opacity-10 backdrop-blur-sm rounded-lg shadow-lg
overflow-hidden h-full transition-all duration-300 hover:shadow-xl border
border-emerald-500/30'>
                                                        <div
className='overflow-hidden'>
                                                            <img

src={product.image}

alt={product.name}

className='w-full h-48 object-cover transition-transform duration-300 ease-in-out
hover:scale-110'
                                                            />
                                                        </div>
                                                        <div
className='p-4'>
                                                            <h3
className='text-lg font-semibold mb-2 text-white'>{product.name}</h3>
                                                            <p
className='text-emerald-300 font-medium mb-4'>

${product.price.toFixed(2)}
                                                            </p>
                                                            <button

onClick={() => addToCart(product)}

className='w-full bg-emerald-600 hover:bg-emerald-500 text-white font-semibold
py-2 px-4 rounded transition-colors duration-300
                                                                flex
items-center justify-center'
                                                            >

<ShoppingCart className='w-5 h-5 mr-2' />

Add to Cart
                                                            </button>
                                                        </div>
                                                    </div>
                                                </div>
                                            ))}
                                        </div>
```

```
                                        </div>
                                        <button
                                                onClick={prevSlide}
                                                disabled={isStartDisabled}
                                                className={`absolute top-1/2 -left-4
transform -translate-y-1/2 p-2 rounded-full transition-colors duration-300 ${
                                                        isStartDisabled ? "bg-gray-400
cursor-not-allowed" : "bg-emerald-600 hover:bg-emerald-500"
                                                }`}
                                        >
                                                <ChevronLeft className='w-6 h-6' />
                                        </button>

                                        <button
                                                onClick={nextSlide}
                                                disabled={isEndDisabled}
                                                className={`absolute top-1/2 -right-4
transform -translate-y-1/2 p-2 rounded-full transition-colors duration-300 ${
                                                        isEndDisabled ? "bg-gray-400
cursor-not-allowed" : "bg-emerald-600 hover:bg-emerald-500"
                                                }`}
                                        >
                                                <ChevronRight className='w-6 h-6' />
                                        </button>
                                </div>
                        </div>
                </div>
        );
};
export default FeaturedProducts;
```

8.Admin Dashboard



AnalyticsTab.jsx

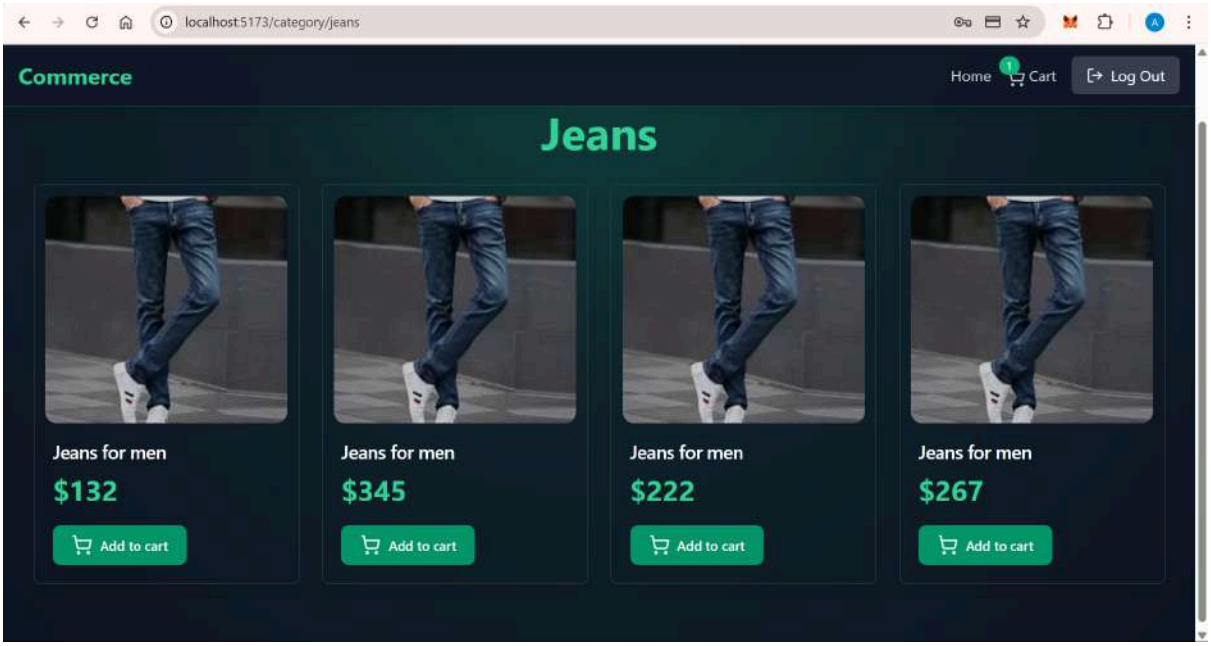
```
import { motion } from "framer-motion";
import { useEffect, useState } from "react";
import axios from "../lib/axios";
import { Users, Package, ShoppingCart, DollarSign } from "lucide-react";
import { LineChart, Line, XAxis, YAxis, CartesianGrid, Tooltip, Legend,
ResponsiveContainer } from "recharts";

const AnalyticsTab = () => {
	const [analyticsData, setAnalyticsData] = useState({
```

```
		users: 0,
		products: 0,
		totalSales: 0,
		totalRevenue: 0,
	});
	const [isLoading, setIsLoading] = useState(true);
	const [dailySalesData, setDailySalesData] = useState([]);

	useEffect(() => {
		const fetchAnalyticsData = async () => {
			try {
				const response = await axios.get("/analytics");
				setAnalyticsData(response.data.analyticsData);
				setDailySalesData(response.data.dailySalesData);
			} catch (error) {
				console.error("Error fetching analytics data:", error);
			} finally {
				setIsLoading(false);
			}
		};

		fetchAnalyticsData();
	}, []);

	if (isLoading) {
		return <div>Loading...</div>;
	}

	return (
		<div className='max-w-7xl mx-auto px-4 sm:px-6 lg:px-8'>
			<div className='grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4
gap-6 mb-8'>
				<AnalyticsCard
					title='Total Users'
					value={analyticsData.users.toLocaleString()}
					icon={Users}
					color='from-emerald-500 to-teal-700'
				/>
				<AnalyticsCard
					title='Total Products'
					value={analyticsData.products.toLocaleString()}
					icon={Package}
					color='from-emerald-500 to-green-700'
				/>
```

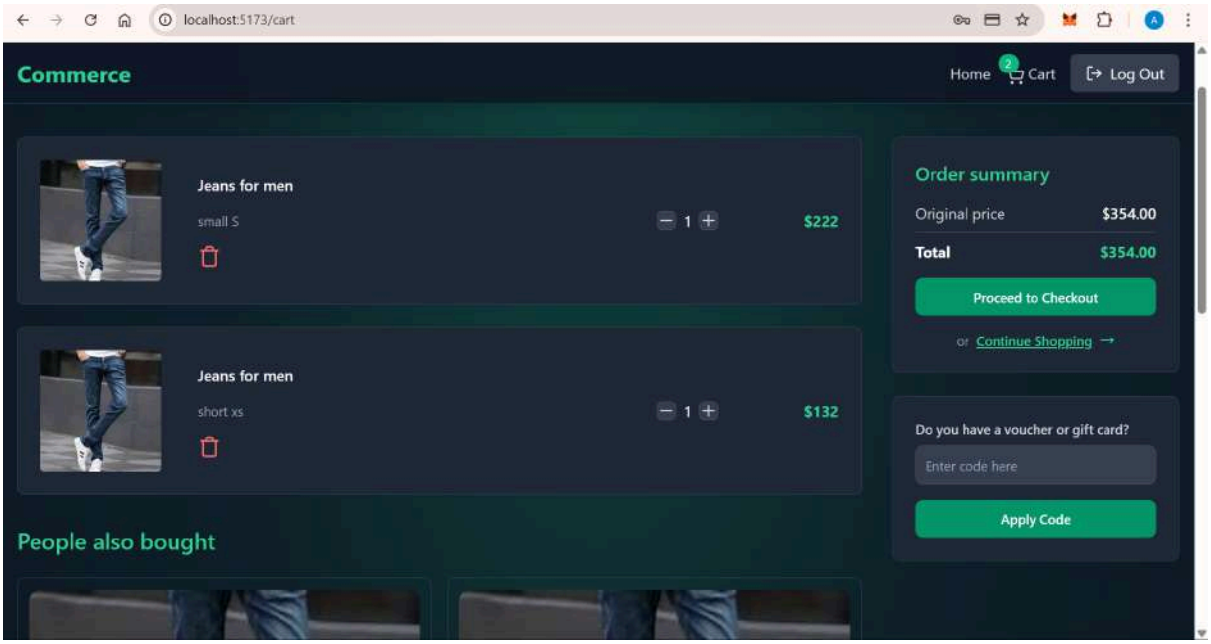
```
                        <AnalyticsCard
                                title='Total Sales'
                                value={analyticsData.totalSales.toLocaleString()}
                                icon={ShoppingCart}
                                color='from-emerald-500 to-cyan-700'
                        />
                        <AnalyticsCard
                                title='Total Revenue'

value={`$${analyticsData.totalRevenue.toLocaleString()}`}
                                icon={DollarSign}
                                color='from-emerald-500 to-lime-700'
                        />
                </div>
                <motion.div
                        className='bg-gray-800/60 rounded-lg p-6 shadow-lg'
                        initial={{ opacity: 0, y: 20 }}
                        animate={{ opacity: 1, y: 0 }}
                        transition={{ duration: 0.5, delay: 0.25 }}
                >
                        <ResponsiveContainer width='100%' height={400}>
                                <LineChart data={dailySalesData}>
                                        <CartesianGrid strokeDasharray='3 3' />
                                        <XAxis dataKey='name' stroke='#D1D5DB'
/>
                                        <YAxis yAxisId='left' stroke='#D1D5DB' />
                                        <YAxis yAxisId='right' orientation='right'
stroke='#D1D5DB' />
                                        <Tooltip />
                                        <Legend />
                                        <Line
                                                yAxisId='left'
                                                type='monotone'
                                                dataKey='sales'
                                                stroke='#10B981'
                                                activeDot={{ r: 8 }}
                                                name='Sales'
                                        />
                                        <Line
                                                yAxisId='right'
                                                type='monotone'
                                                dataKey='revenue'
                                                stroke='#3B82F6'
                                                activeDot={{ r: 8 }}
```

```
                                                name='Revenue'
                                            />
                                        </LineChart>
                                    </ResponsiveContainer>
                                </motion.div>
                            </div>
                        );
};
export default AnalyticsTab;

const AnalyticsCard = ({ title, value, icon: Icon, color }) => (
                        <motion.div
                            className={`bg-gray-800 rounded-lg p-6 shadow-lg overflow-hidden
relative ${color}`}
                            initial={{ opacity: 0, y: 20 }}
                            animate={{ opacity: 1, y: 0 }}
                            transition={{ duration: 0.5 }}
                        >
                            <div className='flex justify-between items-center'>
                                <div className='z-10'>
                                    <p className='text-emerald-300 text-sm mb-1
font-semibold'>{title}</p>
                                    <h3 className='text-white text-3xl
font-bold'>{value}</h3>
                                </div>
                            </div>
                            <div className='absolute inset-0 bg-gradient-to-br from-emerald-600
to-emerald-900 opacity-30' />
                            <div className='absolute -bottom-4 -right-4 text-emerald-800
opacity-50'>
                                <Icon className='h-32 w-32' />
                            </div>
                        </motion.div>
);
```

9.Add to cart



Proceed to Checkout



CartPage.jsx

```
import { Link } from "react-router-dom";
import { useCartStore } from "../stores/useCartStore";
import { motion } from "framer-motion";
import { ShoppingCart } from "lucide-react";
import CartItem from "../components/CartItem";
import PeopleAlsoBought from "../components/PeopleAlsoBought";
import OrderSummary from "../components/OrderSummary";
```

```
import GiftCouponCard from "../components/GiftCouponCard";

const CartPage = () => {
	const { cart } = useCartStore();

	return (
		<div className='py-8 md:py-16'>
			<div className='mx-auto max-w-screen-xl px-4 2xl:px-0'>
				<div className='mt-6 sm:mt-8 md:gap-6 lg:flex
lg:items-start xl:gap-8'>
					<motion.div
						className='mx-auto w-full flex-none
lg:max-w-2xl xl:max-w-4xl'
						initial={{ opacity: 0, x: -20 }}
						animate={{ opacity: 1, x: 0 }}
						transition={{ duration: 0.5, delay: 0.2 }}
					>
						{cart.length === 0 ? (
							<EmptyCartUI />
						) : (
							<div className='space-y-6'>
								{cart.map((item) => (
									<CartItem
key={item._id} item={item} />
								))}
							</div>
						)}
						{cart.length > 0 && <PeopleAlsoBought />}
					</motion.div>

					{cart.length > 0 && (
						<motion.div
							className='mx-auto mt-6 max-w-4xl
flex-1 space-y-6 lg:mt-0 lg:w-full'
							initial={{ opacity: 0, x: 20 }}
							animate={{ opacity: 1, x: 0 }}
							transition={{ duration: 0.5, delay: 0.4
}}
						>
							<OrderSummary />
							<GiftCouponCard />
						</motion.div>
					)}
				</div>
```

```
                        </div>
                </div>
        );
};
export default CartPage;

const EmptyCartUI = () => (
        <motion.div
                className='flex flex-col items-center justify-center space-y-4 py-16'
                initial={{ opacity: 0, y: 20 }}
                animate={{ opacity: 1, y: 0 }}
                transition={{ duration: 0.5 }}
        >
                <ShoppingCart className='h-24 w-24 text-gray-300' />
                <h3 className='text-2xl font-semibold '>Your cart is empty</h3>
                <p className='text-gray-400'>Looks like you {"haven't"} added
anything to your cart yet.</p>
                <Link
                        className='mt-4 rounded-md bg-emerald-500 px-6 py-2
text-white transition-colors hover:bg-emerald-600'
                        to='/'
                >
                        Start Shopping
                </Link>
        </motion.div>
);
```

CartItem.jsx

```
import { Minus, Plus, Trash } from "lucide-react";
import { useCartStore } from "../stores/useCartStore";

const CartItem = ({ item }) => {
        const { removeFromCart, updateQuantity } = useCartStore();

        return (
                <div className='rounded-lg border p-4 shadow-sm border-gray-700
bg-gray-800 md:p-6'>
                        <div className='space-y-4 md:flex md:items-center
md:justify-between md:gap-6 md:space-y-0'>
                                <div className='shrink-0 md:order-1'>
                                        <img className='h-20 md:h-32 rounded
object-cover' src={item.image} />
                                </div>
```

```
                        <label className='sr-only'>Choose quantity:</label>

                        <div className='flex items-center justify-between
md:order-3 md:justify-end'>
                            <div className='flex items-center gap-2'>
                                <button
                                    className='inline-flex h-5 w-5
shrink-0 items-center justify-center rounded-md border
                                     border-gray-600 bg-gray-700
hover:bg-gray-600 focus:outline-none focus:ring-2
                                      focus:ring-emerald-500'
                                    onClick={() =>
updateQuantity(item._id, item.quantity - 1)}
                                    >
                                    <Minus className='text-gray-300' />
                                </button>
                                <p>{item.quantity}</p>
                                <button
                                    className='inline-flex h-5 w-5
shrink-0 items-center justify-center rounded-md border
                                     border-gray-600 bg-gray-700
hover:bg-gray-600 focus:outline-none
                                focus:ring-2 focus:ring-emerald-500'
                                    onClick={() =>
updateQuantity(item._id, item.quantity + 1)}
                                    >
                                    <Plus className='text-gray-300' />
                                </button>
                            </div>

                            <div className='text-end md:order-4 md:w-32'>
                                <p className='text-base font-bold
text-emerald-400'>${item.price}</p>
                            </div>
                        </div>

                        <div className='w-full min-w-0 flex-1 space-y-4
md:order-2 md:max-w-md'>
                            <p className='text-base font-medium text-white
hover:text-emerald-400 hover:underline'>
                                {item.name}
                            </p>
                            <p className='text-sm
text-gray-400'>{item.description}</p>
```

```
                                    <div className='flex items-center gap-4'>
                                        <button
                                            className='inline-flex items-center
text-sm font-medium text-red-400
                                             hover:text-red-300 hover:underline'
                                            onClick={() =>
removeFromCart(item._id)}
                                        >
                                            <Trash />
                                        </button>
                                    </div>
                                </div>
                            </div>
                        </div>
                    );
};
export default CartItem;
```

OrderSummary.jsx

```
import { motion } from "framer-motion";
import { useCartStore } from "../stores/useCartStore";
import { Link } from "react-router-dom";
import { MoveRight } from "lucide-react";
import { loadStripe } from "@stripe/stripe-js";
import axios from "../lib/axios";

const stripePromise = loadStripe(

"pk_test_51RfhJv2aPyHFPcblNNYCjIl4vVbKZOkTW8DnIOPMA9CY5mDQDI9uYS1
krLu0W6EVdYbu92wdZi49O0YwUEqoLZC700jFhd0Jmr"
);

/*const OrderSummary = () => {
        const { total, subtotal, coupon, isCouponApplied, cart } = useCartStore();

        const savings = subtotal - total;
        const formattedSubtotal = subtotal.toFixed(2);
        const formattedTotal = total.toFixed(2);
        const formattedSavings = savings.toFixed(2);

        const handlePayment = async () => {
                const stripe = await stripePromise;
```

```
		const res = await axios.post("/payments/create-checkout-session", {
			products: cart,
			couponCode: coupon ? coupon.code : null,
		});

		const session = res.data;
		const result = await stripe.redirectToCheckout({
			sessionId: session.id,
		});

		if (result.error) {
			console.error("Error:", result.error);
		}
	};*/
	const OrderSummary = () => {
		const { total, subtotal, coupon, isCouponApplied, cart } =
useCartStore();
		const savings = subtotal - total;
		const formattedSubtotal = subtotal.toFixed(2);
		const formattedTotal = total.toFixed(2);
		const formattedSavings = savings.toFixed(2);

		const handlePayment = async () => {
		  try {
			// Create a new Stripe checkout session
			const res = await
axios.post("/payments/create-checkout-session", {
			  products: cart,
			  couponCode: coupon ? coupon.code : null,
			});

			// Verify that the session ID is valid and not expired
			if (!res.data || !res.data.id) {
			  throw new Error("Invalid checkout session");
			}

			const stripe = await stripePromise;
			const session = res.data;

			// Redirect to Stripe checkout page
			const result = await stripe.redirectToCheckout({
				sessionId: session.id,
    });
```

```
    if (result.error) {
      console.error("Error:", result.error);
    }
  } catch (error) {
    console.error("Error:", error);
  }
};



        return (
                <motion.div
                        className='space-y-4 rounded-lg border border-gray-700
bg-gray-800 p-4 shadow-sm sm:p-6'
                        initial={{ opacity: 0, y: 20 }}
                        animate={{ opacity: 1, y: 0 }}
                        transition={{ duration: 0.5 }}
                >
                        <p className='text-xl font-semibold text-emerald-400'>Order
summary</p>

                        <div className='space-y-4'>
                                <div className='space-y-2'>
                                        <dl className='flex items-center justify-between
gap-4'>
                                                <dt className='text-base font-normal
text-gray-300'>Original price</dt>
                                                <dd className='text-base font-medium
text-white'>${formattedSubtotal}</dd>
                                        </dl>

                                        {savings > 0 && (
                                                <dl className='flex items-center
justify-between gap-4'>
                                                        <dt className='text-base
font-normal text-gray-300'>Savings</dt>
                                                        <dd className='text-base
font-medium text-emerald-400'>-${formattedSavings}</dd>
                                                </dl>
                                        )}

                                        {coupon && isCouponApplied && (
                                                <dl className='flex items-center
justify-between gap-4'>
```

```
                                                <dt className='text-base
font-normal text-gray-300'>Coupon ({coupon.code})</dt>
                                                <dd className='text-base
font-medium text-emerald-400'>-{coupon.discountPercentage}%</dd>
                                            </dl>
                                        )}
                                        <dl className='flex items-center justify-between
gap-4 border-t border-gray-600 pt-2'>
                                            <dt className='text-base font-bold
text-white'>Total</dt>
                                            <dd className='text-base font-bold
text-emerald-400'>${formattedTotal}</dd>
                                        </dl>
                                    </div>

                                    <motion.button
                                        className='flex w-full items-center justify-center
rounded-lg bg-emerald-600 px-5 py-2.5 text-sm font-medium text-white
hover:bg-emerald-700 focus:outline-none focus:ring-4 focus:ring-emerald-300'
                                        whileHover={{ scale: 1.05 }}
                                        whileTap={{ scale: 0.95 }}
                                        onClick={handlePayment}
                                    >
                                        Proceed to Checkout
                                    </motion.button>

                                    <div className='flex items-center justify-center gap-2'>
                                        <span className='text-sm font-normal
text-gray-400'>or</span>
                                        <Link
                                            to='/'
                                            className='inline-flex items-center gap-2
text-sm font-medium text-emerald-400 underline hover:text-emerald-300
hover:no-underline'
                                        >
                                            Continue Shopping
                                            <MoveRight size={16} />
                                        </Link>
                                    </div>
                                </div>
                            </motion.div>
        );
};
export default OrderSummary;
```

GiftCouponCard.jsx

```
import { motion } from "framer-motion";
import { useEffect, useState } from "react";
import { useCartStore } from "../stores/useCartStore";

const GiftCouponCard = () => {
	const [userInputCode, setUserInputCode] = useState("");
	const { coupon, isCouponApplied, applyCoupon, getMyCoupon,
removeCoupon } = useCartStore();

	useEffect(() => {
		getMyCoupon();
	}, [getMyCoupon]);

	useEffect(() => {
		if (coupon) setUserInputCode(coupon.code);
	}, [coupon]);

	const handleApplyCoupon = () => {
		if (!userInputCode) return;
		applyCoupon(userInputCode);
	};

	const handleRemoveCoupon = async () => {
		await removeCoupon();
		setUserInputCode("");
	};

	return (
		<motion.div
			className='space-y-4 rounded-lg border border-gray-700
bg-gray-800 p-4 shadow-sm sm:p-6'
			initial={{ opacity: 0, y: 20 }}
			animate={{ opacity: 1, y: 0 }}
			transition={{ duration: 0.5, delay: 0.2 }}
		>
			<div className='space-y-4'>
				<div>
					<label htmlFor='voucher' className='mb-2 block
text-sm font-medium text-gray-300'>
						Do you have a voucher or gift card?
					</label>
```

```
                                    <input
                                        type='text'
                                        id='voucher'
                                        className='block w-full rounded-lg border
border-gray-600 bg-gray-700
        p-2.5 text-sm text-white placeholder-gray-400 focus:border-emerald-500
        focus:ring-emerald-500'
                                        placeholder='Enter code here'
                                        value={userInputCode}
                                        onChange={(e) =>
setUserInputCode(e.target.value)}
                                        required
                                    />
                                </div>

                                <motion.button
                                    type='button'
                                    className='flex w-full items-center justify-center
rounded-lg bg-emerald-600 px-5 py-2.5 text-sm font-medium text-white
hover:bg-emerald-700 focus:outline-none focus:ring-4 focus:ring-emerald-300'
                                    whileHover={{ scale: 1.05 }}
                                    whileTap={{ scale: 0.95 }}
                                    onClick={handleApplyCoupon}
                                >
                                    Apply Code
                                </motion.button>
                            </div>
                            {isCouponApplied && coupon && (
                                <div className='mt-4'>
                                    <h3 className='text-lg font-medium
text-gray-300'>Applied Coupon</h3>

                                    <p className='mt-2 text-sm text-gray-400'>
                                        {coupon.code} -
{coupon.discountPercentage}% off
                                    </p>

                                    <motion.button
                                        type='button'
                                        className='mt-2 flex w-full items-center
justify-center rounded-lg bg-red-600
        px-5 py-2.5 text-sm font-medium text-white hover:bg-red-700
focus:outline-none
         focus:ring-4 focus:ring-red-300'
```

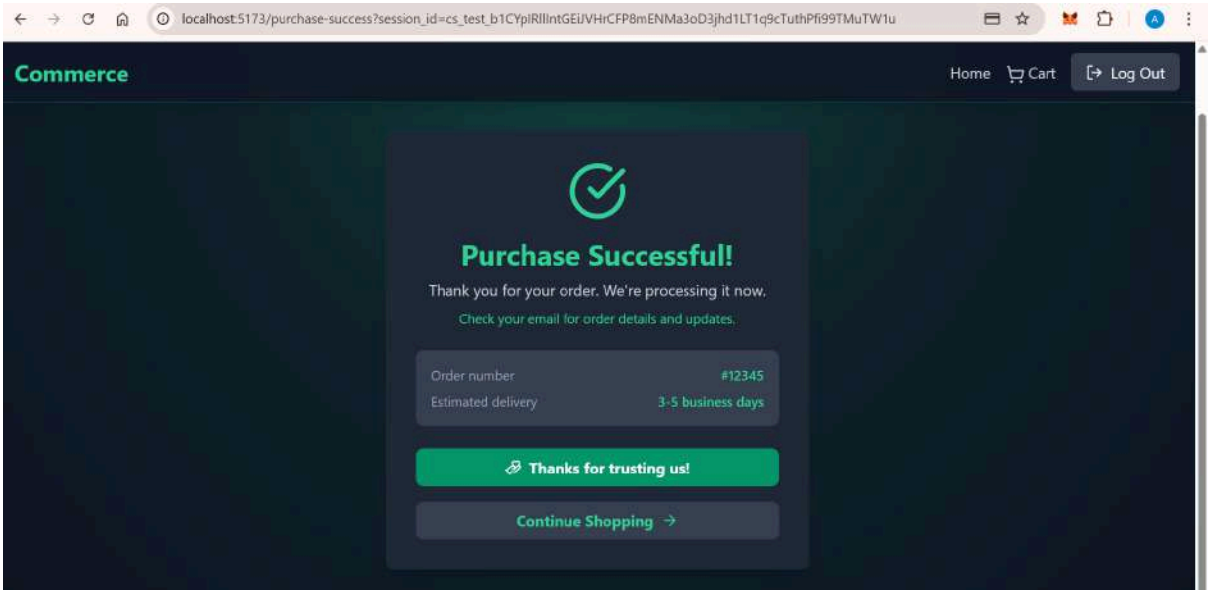
```
                                        whileHover={{ scale: 1.05 }}
                                        whileTap={{ scale: 0.95 }}
                                        onClick={handleRemoveCoupon}
                                    >
                                        Remove Coupon
                                    </motion.button>
                                </div>
                            )}

                            {coupon && (
                                <div className='mt-4'>
                                    <h3 className='text-lg font-medium
text-gray-300'>Your Available Coupon:</h3>
                                    <p className='mt-2 text-sm text-gray-400'>
                                        {coupon.code} -
{coupon.discountPercentage}% off
                                    </p>
                                </div>
                            )}
                        </motion.div>
                    );
};
export default GiftCouponCard;
```

PeopleAlsoBought.jsx

```
import { useEffect, useState } from "react";
import ProductCard from "./ProductCard";
import axios from "../lib/axios";
import toast from "react-hot-toast";
import LoadingSpinner from "./LoadingSpinner";

const PeopleAlsoBought = () => {
    const [recommendations, setRecommendations] = useState([]);
    const [isLoading, setIsLoading] = useState(true);

    useEffect(() => {
        const fetchRecommendations = async () => {
            try {
                const res = await
axios.get("/products/recommendations");
                setRecommendations(res.data);
            } catch (error) {
```

```
                        toast.error(error.response.data.message || "An error
occurred while fetching recommendations");
                } finally {
                        setIsLoading(false);
                }
        };

        fetchRecommendations();
}, []);

if (isLoading) return <LoadingSpinner />;

return (
        <div className='mt-8'>
                <h3 className='text-2xl font-semibold
text-emerald-400'>People also bought</h3>
                <div className='mt-6 grid grid-cols-1 gap-4 sm:grid-cols-2 lg:
grid-col-3'>
                        {recommendations.map((product) => (
                                <ProductCard key={product._id} product={product}
/>
                        ))}
                </div>
        </div>
);
};
export default PeopleAlsoBought;
```

10.Payment Successful



PurchaseSuccessPage.jsx

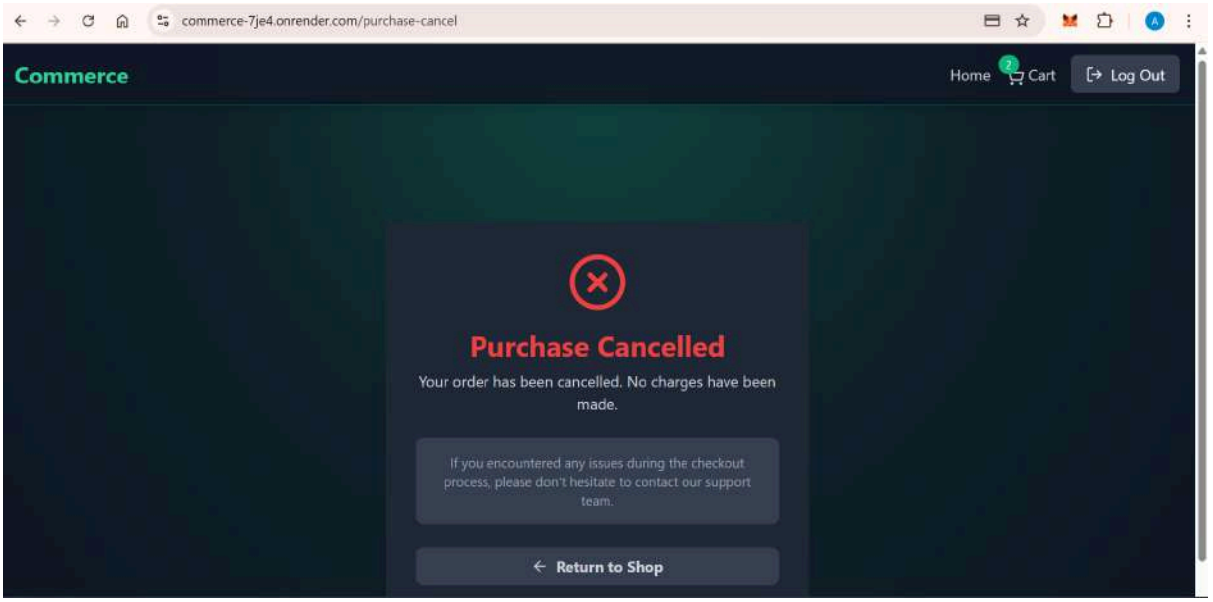
```
import { ArrowRight, CheckCircle, HandHeart } from "lucide-react";
import { useEffect, useState } from "react";
import { Link } from "react-router-dom";
import { useCartStore } from "../stores/useCartStore";
import axios from "../lib/axios";
import Confetti from "react-confetti";

const PurchaseSuccessPage = () => {
	const [isProcessing, setIsProcessing] = useState(true);
	const { clearCart } = useCartStore();
	const [error, setError] = useState(null);

	useEffect(() => {
		const handleCheckoutSuccess = async (sessionId) => {
			try {
				await axios.post("/payments/checkout-success", {
					sessionId,
				});
				clearCart();
			} catch (error) {
				console.log(error);
			} finally {
				setIsProcessing(false);
			}
		};
```

```
                const sessionId = new
URLSearchParams(window.location.search).get("session_id");
                if (sessionId) {
                        handleCheckoutSuccess(sessionId);
                } else {
                        setIsProcessing(false);
                        setError("No session ID found in the URL");
                }
        }, [clearCart]);

        if (isProcessing) return "Processing...";

        if (error) return `Error: ${error}`;

        return (
                <div className='h-screen flex items-center justify-center px-4'>
                        <Confetti
                                width={window.innerWidth}
                                height={window.innerHeight}
                                gravity={0.1}
                                style={{ zIndex: 99 }}
                                numberOfPieces={700}
                                recycle={false}
                        />

                        <div className='max-w-md w-full bg-gray-800 rounded-lg
shadow-xl overflow-hidden relative z-10'>
                                <div className='p-6 sm:p-8'>
                                        <div className='flex justify-center'>
                                                <CheckCircle className='text-emerald-400
w-16 h-16 mb-4' />
                                        </div>
                                        <h1 className='text-2xl sm:text-3xl font-bold
text-center text-emerald-400 mb-2'>
                                                Purchase Successful!
                                        </h1>

                                        <p className='text-gray-300 text-center mb-2'>
                                                Thank you for your order. {"We're"}
processing it now.
                                        </p>
                                        <p className='text-emerald-400 text-center
text-sm mb-6'>
```

```
                                        Check your email for order details and
updates.
                                    </p>
                                    <div className='bg-gray-700 rounded-lg p-4
mb-6'>
                                        <div className='flex items-center
justify-between mb-2'>
                                            <span className='text-sm
text-gray-400'>Order number</span>
                                            <span className='text-sm
font-semibold text-emerald-400'>#12345</span>
                                        </div>
                                        <div className='flex items-center
justify-between'>
                                            <span className='text-sm
text-gray-400'>Estimated delivery</span>
                                            <span className='text-sm
font-semibold text-emerald-400'>3-5 business days</span>
                                        </div>
                                    </div>

                                    <div className='space-y-4'>
                                        <button
                                            className='w-full bg-emerald-600
hover:bg-emerald-700 text-white font-bold py-2 px-4
          rounded-lg transition duration-300 flex items-center justify-center'
                                        >
                                            <HandHeart className='mr-2'
size={18} />
                                            Thanks for trusting us!
                                        </button>
                                        <Link
                                            to={"/"}
                                            className='w-full bg-gray-700
hover:bg-gray-600 text-emerald-400 font-bold py-2 px-4
          rounded-lg transition duration-300 flex items-center justify-center'
                                        >
                                            Continue Shopping
                                            <ArrowRight className='ml-2'
size={18} />
                                        </Link>
                                    </div>
                                </div>
                            </div>
```

```
                </div>
        );
};
export default PurchaseSuccessPage;
```

11. Purchase Cancel Page



PurchaseCancelPage.jsx

```
import { XCircle, ArrowLeft } from "lucide-react";
import { motion } from "framer-motion";
import { Link } from "react-router-dom";

const PurchaseCancelPage = () => {
        return (
                <div className='min-h-screen flex items-center justify-center px-4'>
                        <motion.div
                                initial={{ opacity: 0, y: 20 }}
                                animate={{ opacity: 1, y: 0 }}
                                transition={{ duration: 0.5 }}
                                className='max-w-md w-full bg-gray-800 rounded-lg
shadow-xl overflow-hidden relative z-10'
                        >
                                <div className='p-6 sm:p-8'>
                                        <div className='flex justify-center'>
                                                <XCircle className='text-red-500 w-16
h-16 mb-4' />
```

```
                                        </div>
                                        <h1 className='text-2xl sm:text-3xl font-bold
text-center text-red-500 mb-2'>Purchase Cancelled</h1>
                                        <p className='text-gray-300 text-center mb-6'>
                                                Your order has been cancelled. No charges
have been made.
                                        </p>
                                        <div className='bg-gray-700 rounded-lg p-4
mb-6'>
                                                <p className='text-sm text-gray-400
text-center'>
                                                        If you encountered any issues during
the checkout process, please don&apos;t hesitate to
                                                        contact our support team.
                                                </p>
                                        </div>
                                        <div className='space-y-4'>
                                                <Link
                                                        to={"/"}
                                                        className='w-full bg-gray-700
hover:bg-gray-600 text-gray-300 font-bold py-2 px-4 rounded-lg transition
duration-300 flex items-center justify-center'
                                                >
                                                        <ArrowLeft className='mr-2'
size={18} />
                                                        Return to Shop
                                                </Link>
                                        </div>
                                </div>
                        </motion.div>
                </div>
        );
};

export default PurchaseCancelPage;
```

Backend :

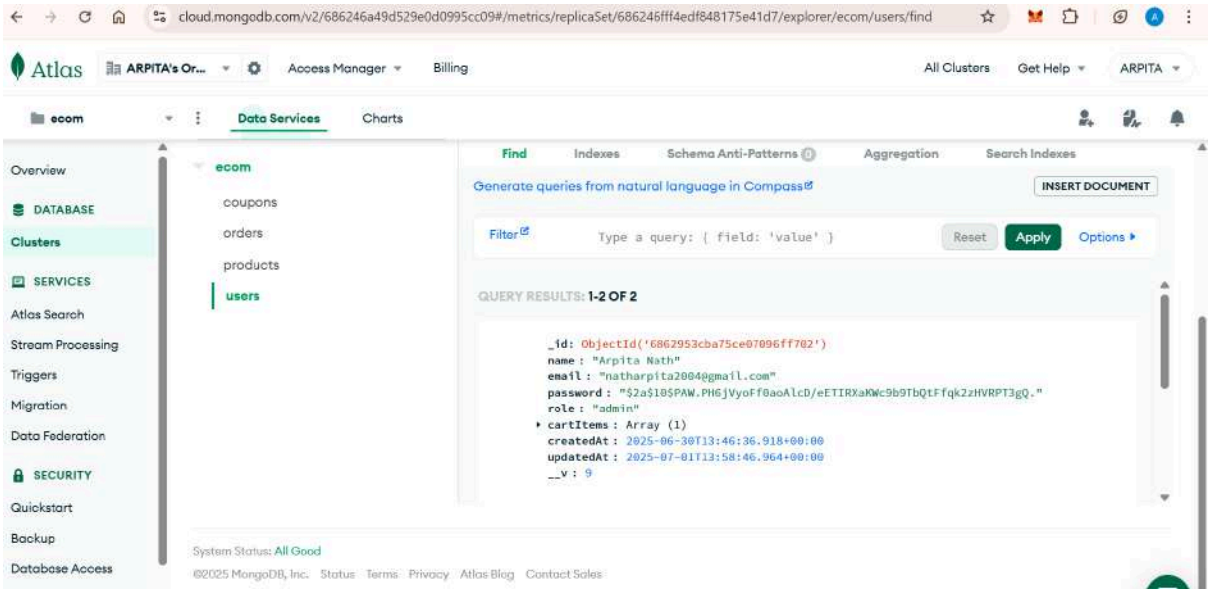
12.Database -

db.js

```
import mongoose from "mongoose";

export const connectDB = async () => {
	try {
		const conn = await mongoose.connect(process.env.MONGO_URI);
		console.log(`MongoDB connected: ${conn.connection.host}`);
	} catch (error) {
		console.log("Error connecting to MONGODB", error.message);
		process.exit(1);
	}
};
```

Authentication-

user.model.js




```
import mongoose from "mongoose";
import bcrypt from "bcryptjs";

const userSchema = new mongoose.Schema(
	{
		name: {
			type: String,
			required: [true, "Name is required"],
		},
		email: {
			type: String,
			required: [true, "Email is required"],
			unique: true,
```

```
                lowercase: true,
                trim: true,
            },
            password: {
                type: String,
                required: [true, "Password is required"],
                minlength: [6, "Password must be at least 6 characters long"],
            },
            cartItems: [
                {
                    quantity: {
                        type: Number,
                        default: 1,
                    },
                    product: {
                        type: mongoose.Schema.Types.ObjectId,
                        ref: "Product",
                    },
                },
            ],
            role: {
                type: String,
                enum: ["customer", "admin"],
                default: "customer",
            },
        },
        {
            timestamps: true,
        }
);

// Pre-save hook to hash password before saving to database
userSchema.pre("save", async function (next) {
    if (!this.isModified("password")) return next();

    try {
        const salt = await bcrypt.genSalt(10);
        this.password = await bcrypt.hash(this.password, salt);
        next();
    } catch (error) {
        next(error);
    }
});
```

```
userSchema.methods.comparePassword = async function (password) {
	return bcrypt.compare(password, this.password);
};

const User = mongoose.model("User", userSchema);

export default User;
```

Controllers -

analytics.controller.js

```
import Order from "../models/order.model.js";
import Product from "../models/product.model.js";
import User from "../models/user.model.js";

export const getAnalyticsData = async () => {
	const totalUsers = await User.countDocuments();
	const totalProducts = await Product.countDocuments();

	const salesData = await Order.aggregate([
		{
			$group: {
				_id: null, // it groups all documents together,
				totalSales: { $sum: 1 },
				totalRevenue: { $sum: "$totalAmount" },
			},
		},
	]);

	const { totalSales, totalRevenue } = salesData[0] || { totalSales: 0,
totalRevenue: 0 };

	return {
		users: totalUsers,
		products: totalProducts,
		totalSales,
		totalRevenue,
	};
};

export const getDailySalesData = async (startDate, endDate) => {
	try {
```

```
		const dailySalesData = await Order.aggregate([
			{
				$match: {
					createdAt: {
						$gte: startDate,
						$lte: endDate,
					},
				},
			},
			{
				$group: {
					_id: { $dateToString: { format: "%Y-%m-%d", date:
"$createdAt" } },
					sales: { $sum: 1 },
					revenue: { $sum: "$totalAmount" },
				},
			},
			{ $sort: { _id: 1 } },
		]);

		// example of dailySalesData
		// [
		//	{
		//		_id: "2024-08-18",
		//		sales: 12,
		//		revenue: 1450.75
		//	},
		// ]

		const dateArray = getDatesInRange(startDate, endDate);
		// console.log(dateArray) // ['2024-08-18', '2024-08-19', ... ]

		return dateArray.map((date) => {
			const foundData = dailySalesData.find((item) => item._id ===
date);

			return {
				date,
				sales: foundData?.sales || 0,
				revenue: foundData?.revenue || 0,
			};
		});
	} catch (error) {
		throw error;
```

```
	}
};

function getDatesInRange(startDate, endDate) {
	const dates = [];
	let currentDate = new Date(startDate);

	while (currentDate <= endDate) {
		dates.push(currentDate.toISOString().split("T")[0]);
		currentDate.setDate(currentDate.getDate() + 1);
	}

	return dates;
}
```

auth.controller.js

```
import { redis } from "../lib/redis.js";
import User from "../models/user.model.js";
import jwt from "jsonwebtoken";

const generateTokens = (userId) => {
	const accessToken = jwt.sign({ userId },
process.env.ACCESS_TOKEN_SECRET, {
		expiresIn: "15m",
	});

	const refreshToken = jwt.sign({ userId },
process.env.REFRESH_TOKEN_SECRET, {
		expiresIn: "7d",
	});

	return { accessToken, refreshToken };
};

const storeRefreshToken = async (userId, refreshToken) => {
	await redis.set(`refresh_token:${userId}`, refreshToken, "EX", 7 * 24 * 60 *
60); // 7days
};

const setCookies = (res, accessToken, refreshToken) => {
	res.cookie("accessToken", accessToken, {
```

```
		httpOnly: true, // prevent XSS attacks, cross site scripting attack
		secure: process.env.NODE_ENV === "production",
		sameSite: "strict", // prevents CSRF attack, cross-site request forgery
attack
		maxAge: 15 * 60 * 1000, // 15 minutes
	});
	res.cookie("refreshToken", refreshToken, {
		httpOnly: true, // prevent XSS attacks, cross site scripting attack
		secure: process.env.NODE_ENV === "production",
		sameSite: "strict", // prevents CSRF attack, cross-site request forgery
attack
		maxAge: 7 * 24 * 60 * 60 * 1000, // 7 days
	});
};

export const signup = async (req, res) => {
	const { email, password, name } = req.body;
	try {
		const userExists = await User.findOne({ email });

		if (userExists) {
			return res.status(400).json({ message: "User already exists" });
		}
		const user = await User.create({ name, email, password });

		// authenticate
		const { accessToken, refreshToken } = generateTokens(user._id);
		await storeRefreshToken(user._id, refreshToken);

		setCookies(res, accessToken, refreshToken);

		res.status(201).json({
			_id: user._id,
			name: user.name,
			email: user.email,
			role: user.role,
		});
	} catch (error) {
		console.log("Error in signup controller", error.message);
		res.status(500).json({ message: error.message });
	}
};

export const login = async (req, res) => {
```

```
	try {
		const { email, password } = req.body;
		const user = await User.findOne({ email });

		if (user && (await user.comparePassword(password))) {
			const { accessToken, refreshToken } =
generateTokens(user._id);
			await storeRefreshToken(user._id, refreshToken);
			setCookies(res, accessToken, refreshToken);

			res.json({
				_id: user._id,
				name: user.name,
				email: user.email,
				role: user.role,
			});
		} else {
			res.status(400).json({ message: "Invalid email or password" });
		}
	} catch (error) {
		console.log("Error in login controller", error.message);
		res.status(500).json({ message: error.message });
	}
};

export const logout = async (req, res) => {
	try {
		const refreshToken = req.cookies.refreshToken;
		if (refreshToken) {
			const decoded = jwt.verify(refreshToken,
process.env.REFRESH_TOKEN_SECRET);
			await redis.del(`refresh_token:${decoded.userId}`);
		}

		res.clearCookie("accessToken");
		res.clearCookie("refreshToken");
		res.json({ message: "Logged out successfully" });
	} catch (error) {
		console.log("Error in logout controller", error.message);
		res.status(500).json({ message: "Server error", error: error.message });
	}
};

// this will refresh the access token
```

```
export const refreshToken = async (req, res) => {
	try {
		const refreshToken = req.cookies.refreshToken;

		if (!refreshToken) {
			return res.status(401).json({ message: "No refresh token
provided" });
		}

		const decoded = jwt.verify(refreshToken,
process.env.REFRESH_TOKEN_SECRET);
		const storedToken = await
redis.get(`refresh_token:${decoded.userId}`);

		if (storedToken !== refreshToken) {
			return res.status(401).json({ message: "Invalid refresh token" });
		}

		const accessToken = jwt.sign({ userId: decoded.userId },
process.env.ACCESS_TOKEN_SECRET, { expiresIn: "15m" });

		res.cookie("accessToken", accessToken, {
			httpOnly: true,
			secure: process.env.NODE_ENV === "production",
			sameSite: "strict",
			maxAge: 15 * 60 * 1000,
		});

		res.json({ message: "Token refreshed successfully" });
	} catch (error) {
		console.log("Error in refreshToken controller", error.message);
		res.status(500).json({ message: "Server error", error: error.message });
	}
};

export const getProfile = async (req, res) => {
	try {
		res.json(req.user);
	} catch (error) {
		res.status(500).json({ message: "Server error", error: error.message });
	}
};
```

Middleware-

auth.middleware.js

```
import jwt from "jsonwebtoken";
import User from "../models/user.model.js";

export const protectRoute = async (req, res, next) => {
	try {
		const accessToken = req.cookies.accessToken;

		if (!accessToken) {
			return res.status(401).json({ message: "Unauthorized - No
access token provided" });
		}

		try {
			const decoded = jwt.verify(accessToken,
process.env.ACCESS_TOKEN_SECRET);
			const user = await
User.findById(decoded.userId).select("-password");

			if (!user) {
				return res.status(401).json({ message: "User not found"
});
			}

			req.user = user;

			next();
		} catch (error) {
			if (error.name === "TokenExpiredError") {
				return res.status(401).json({ message: "Unauthorized -
Access token expired" });
			}
			throw error;
		}
	} catch (error) {
		console.log("Error in protectRoute middleware", error.message);
		return res.status(401).json({ message: "Unauthorized - Invalid access
token" });
	}
};
```

```
export const adminRoute = (req, res, next) => {
	if (req.user && req.user.role === "admin") {
		next();
	} else {
		return res.status(403).json({ message: "Access denied - Admin only" });
	}
};
```

Routes-

analytics.route.js

```
import express from "express";
import { adminRoute, protectRoute } from "../middleware/auth.middleware.js";
import { getAnalyticsData, getDailySalesData } from
"../controllers/analytics.controller.js";

const router = express.Router();

router.get("/", protectRoute, adminRoute, async (req, res) => {
	try {
		const analyticsData = await getAnalyticsData();

		const endDate = new Date();
		const startDate = new Date(endDate.getTime() - 7 * 24 * 60 * 60 *
1000);

		const dailySalesData = await getDailySalesData(startDate, endDate);

		res.json({
			analyticsData,
			dailySalesData,
		});
	} catch (error) {
		console.log("Error in analytics route", error.message);
		res.status(500).json({ message: "Server error", error: error.message });
	}
});

export default router;
```

auth.route.js

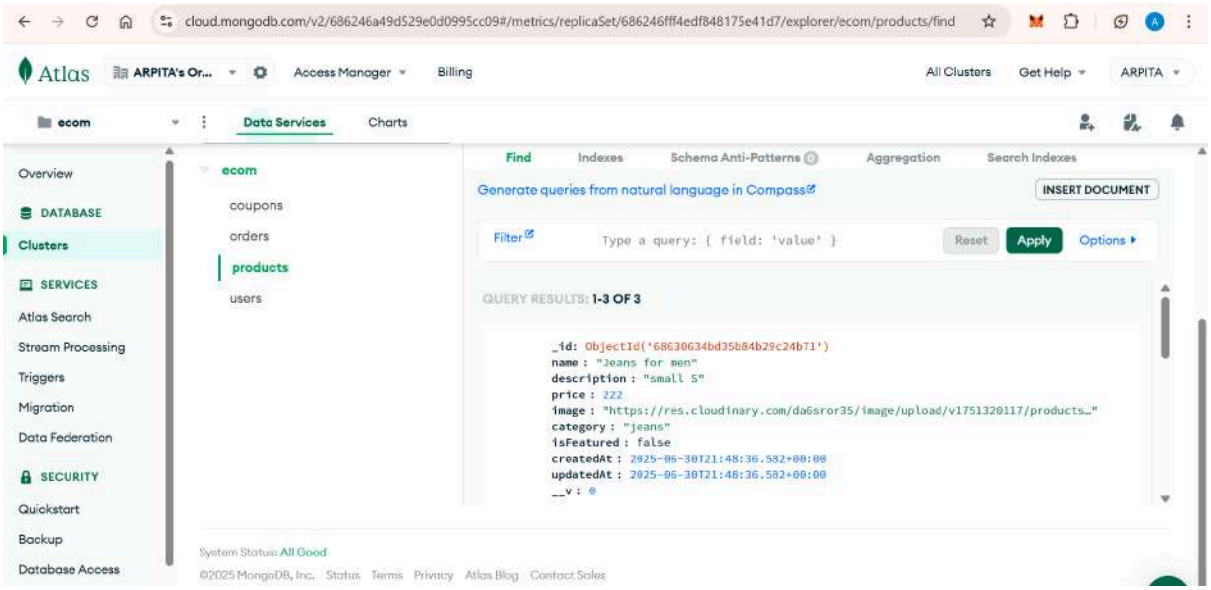
```
import express from "express";
import { login, logout, signup, refreshToken, getProfile } from
"../controllers/auth.controller.js";
import { protectRoute } from "../middleware/auth.middleware.js";

const router = express.Router();

router.post("/signup", signup);
router.post("/login", login);
router.post("/logout", logout);
router.post("/refresh-token", refreshToken);
router.get("/profile", protectRoute, getProfile);

export default router;
```

cart.route.js

```
import express from "express";
import { addToCart, getCartProducts, removeAllFromCart, updateQuantity } from
"../controllers/cart.controller.js";
import { protectRoute } from "../middleware/auth.middleware.js";

const router = express.Router();

router.get("/", protectRoute, getCartProducts);
router.post("/", protectRoute, addToCart);
router.delete("/", protectRoute, removeAllFromCart);
router.put("/:id", protectRoute, updateQuantity);

export default router;
```

Product -

product.model.js




```
import mongoose from "mongoose";

const productSchema = new mongoose.Schema(
	{
		name: {
			type: String,
			required: true,
		},
		description: {
			type: String,
			required: true,
		},
		price: {
			type: Number,
			min: 0,
			required: true,
		},
		image: {
			type: String,
			required: [true, "Image is required"],
		},
		category: {
			type: String,
			required: true,
		},
		isFeatured: {
			type: Boolean,
```

```
                default: false,
            },
        },
        { timestamps: true }
);

const Product = mongoose.model("Product", productSchema);

export default Product;
```

Controllers-

cart.controller.js

```
import Product from "../models/product.model.js";

export const getCartProducts = async (req, res) => {
    try {
        const products = await Product.find({ _id: { $in: req.user.cartItems } });

        // add quantity for each product
        const cartItems = products.map((product) => {
            const item = req.user.cartItems.find((cartItem) => cartItem.id
=== product.id);
            return { ...product.toJSON(), quantity: item.quantity };
        });

        res.json(cartItems);
    } catch (error) {
        console.log("Error in getCartProducts controller", error.message);
        res.status(500).json({ message: "Server error", error: error.message });
    }
};

export const addToCart = async (req, res) => {
    try {
        const { productId } = req.body;
        const user = req.user;

        const existingItem = user.cartItems.find((item) => item.id ===
productId);
        if (existingItem) {
            existingItem.quantity += 1;
```

```
		} else {
			user.cartItems.push(productId);
		}

		await user.save();
		res.json(user.cartItems);
	} catch (error) {
		console.log("Error in addToCart controller", error.message);
		res.status(500).json({ message: "Server error", error: error.message });
	}
};

export const removeAllFromCart = async (req, res) => {
	try {
		const { productId } = req.body;
		const user = req.user;
		if (!productId) {
			user.cartItems = [];
		} else {
			user.cartItems = user.cartItems.filter((item) => item.id !==
productId);
		}
		await user.save();
		res.json(user.cartItems);
	} catch (error) {
		res.status(500).json({ message: "Server error", error: error.message });
	}
};

export const updateQuantity = async (req, res) => {
	try {
		const { id: productId } = req.params;
		const { quantity } = req.body;
		const user = req.user;
		const existingItem = user.cartItems.find((item) => item.id ===
productId);

		if (existingItem) {
			if (quantity === 0) {
				user.cartItems = user.cartItems.filter((item) => item.id !==
productId);
				await user.save();
				return res.json(user.cartItems);
			}
```

```
                existingItem.quantity = quantity;
                await user.save();
                res.json(user.cartItems);
            } else {
                res.status(404).json({ message: "Product not found" });
            }
        } catch (error) {
            console.log("Error in updateQuantity controller", error.message);
            res.status(500).json({ message: "Server error", error: error.message });
        }
};
```

product.controller.js

```
import { redis } from "../lib/redis.js";
import cloudinary from "../lib/cloudinary.js";
import Product from "../models/product.model.js";

export const getAllProducts = async (req, res) => {
    try {
        const products = await Product.find({}); // find all products
        res.json({ products });
    } catch (error) {
        console.log("Error in getAllProducts controller", error.message);
        res.status(500).json({ message: "Server error", error: error.message });
    }
};

export const getFeaturedProducts = async (req, res) => {
    try {
        let featuredProducts = await redis.get("featured_products");
        if (featuredProducts) {
            return res.json(JSON.parse(featuredProducts));
        }

        // if not in redis, fetch from mongodb
        // .lean() is gonna return a plain javascript object instead of a mongodb
document
        // which is good for performance
        featuredProducts = await Product.find({ isFeatured: true }).lean();
```

```
		if (!featuredProducts) {
			return res.status(404).json({ message: "No featured products
found" });
		}

		// store in redis for future quick access

		await redis.set("featured_products", JSON.stringify(featuredProducts));

		res.json(featuredProducts);
	} catch (error) {
		console.log("Error in getFeaturedProducts controller", error.message);
		res.status(500).json({ message: "Server error", error: error.message });
	}
};

export const createProduct = async (req, res) => {
	try {
		const { name, description, price, image, category } = req.body;

		let cloudinaryResponse = null;

		if (image) {
			cloudinaryResponse = await cloudinary.uploader.upload(image,
{ folder: "products" });
		}

		const product = await Product.create({
			name,
			description,
			price,
			image: cloudinaryResponse?.secure_url ?
cloudinaryResponse.secure_url : "",
			category,
		});

		res.status(201).json(product);
	} catch (error) {
		console.log("Error in createProduct controller", error.message);
		res.status(500).json({ message: "Server error", error: error.message });
	}
};

export const deleteProduct = async (req, res) => {
```

```
	try {
		const product = await Product.findById(req.params.id);

		if (!product) {
			return res.status(404).json({ message: "Product not found" });
		}

		if (product.image) {
			const publicId = product.image.split("/").pop().split(".")[0];
			try {
				await cloudinary.uploader.destroy(`products/${publicId}`);
				console.log("deleted image from cloduinary");
			} catch (error) {
				console.log("error deleting image from cloduinary", error);
			}
		}

		await Product.findByIdAndDelete(req.params.id);

		res.json({ message: "Product deleted successfully" });
	} catch (error) {
		console.log("Error in deleteProduct controller", error.message);
		res.status(500).json({ message: "Server error", error: error.message });
	}
};

export const getRecommendedProducts = async (req, res) => {
	try {
		const products = await Product.aggregate([
			{
				$sample: { size: 4 },
			},
			{
				$project: {
					_id: 1,
					name: 1,
					description: 1,
					image: 1,
					price: 1,
				},
			},
		]);

		res.json(products);
```

```
	} catch (error) {
		console.log("Error in getRecommendedProducts controller",
error.message);
		res.status(500).json({ message: "Server error", error: error.message });
	}
};

export const getProductsByCategory = async (req, res) => {
	const { category } = req.params;
	try {
		const products = await Product.find({ category });
		res.json({ products });
	} catch (error) {
		console.log("Error in getProductsByCategory controller",
error.message);
		res.status(500).json({ message: "Server error", error: error.message });
	}
};

export const toggleFeaturedProduct = async (req, res) => {
	try {
		const product = await Product.findById(req.params.id);
		if (product) {
			product.isFeatured = !product.isFeatured;
			const updatedProduct = await product.save();
			await updateFeaturedProductsCache();
			res.json(updatedProduct);
		} else {
			res.status(404).json({ message: "Product not found" });
		}
	} catch (error) {
		console.log("Error in toggleFeaturedProduct controller",
error.message);
		res.status(500).json({ message: "Server error", error: error.message });
	}
};

async function updateFeaturedProductsCache() {
	try {
		// The lean() method  is used to return plain JavaScript objects instead
of full Mongoose documents. This can significantly improve performance

		const featuredProducts = await Product.find({ isFeatured: true }).lean();
		await redis.set("featured_products", JSON.stringify(featuredProducts));
```

```
	} catch (error) {
		console.log("error in update cache function");
	}
}
```

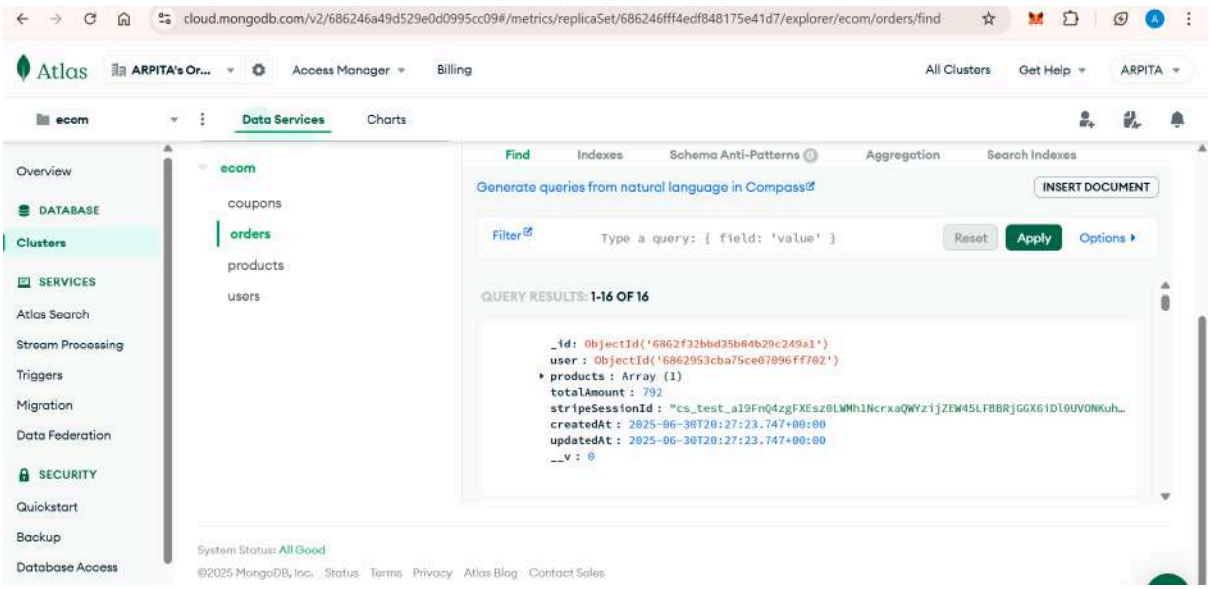
Routes-

product.route.js

```
import express from "express";
import {
	createProduct,
	deleteProduct,
	getAllProducts,
	getFeaturedProducts,
	getProductsByCategory,
	getRecommendedProducts,
	toggleFeaturedProduct,
} from "../controllers/product.controller.js";
import { adminRoute, protectRoute } from "../middleware/auth.middleware.js";

const router = express.Router();

router.get("/", protectRoute, adminRoute, getAllProducts);
router.get("/featured", getFeaturedProducts);
router.get("/category/:category", getProductsByCategory);
router.get("/recommendations", getRecommendedProducts);
router.post("/", protectRoute, adminRoute, createProduct);
router.patch("/:id", protectRoute, adminRoute, toggleFeaturedProduct);
router.delete("/:id", protectRoute, adminRoute, deleteProduct);

export default router;
```

Order-

order.model.js

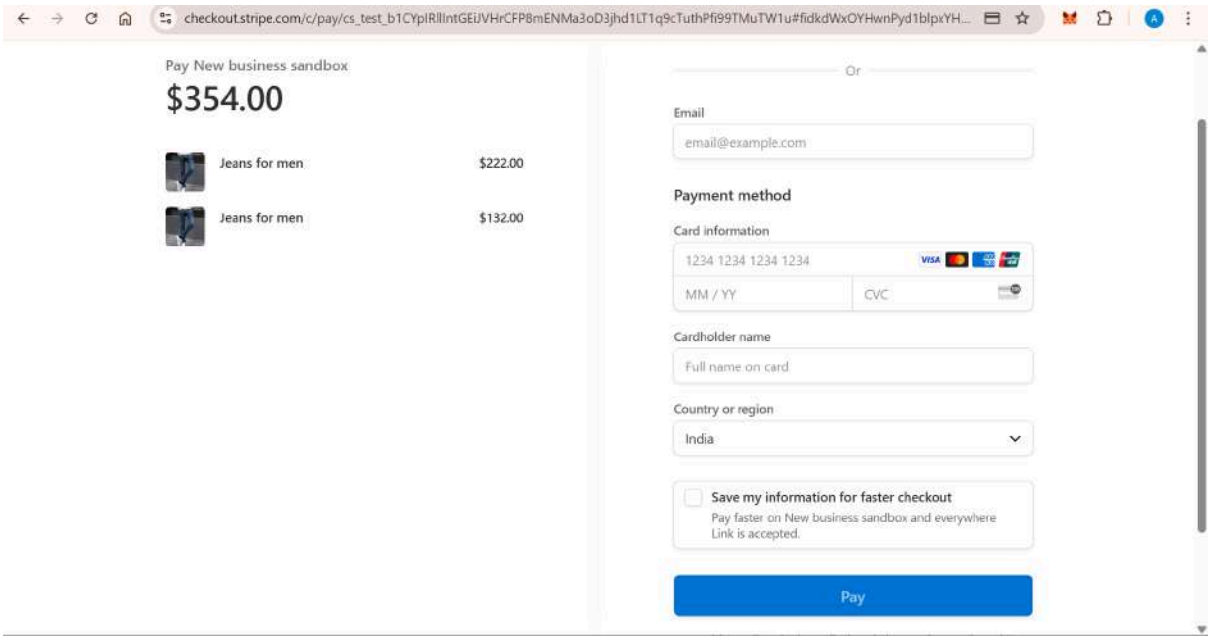



```
import mongoose from "mongoose";

const orderSchema = new mongoose.Schema(
	{
		user: {
			type: mongoose.Schema.Types.ObjectId,
			ref: "User",
			required: true,
		},
		products: [
			{
				product: {
					type: mongoose.Schema.Types.ObjectId,
					ref: "Product",
					required: true,
				},
				quantity: {
					type: Number,
					required: true,
					min: 1,
				},
				price: {
					type: Number,
					required: true,
					min: 0,
				},
			},
		],
		totalAmount: {
```

```
                type: Number,
                required: true,
                min: 0,
            },
            stripeSessionId: {
                type: String,
                unique: true,
            },
        },
        { timestamps: true }
);

const Order = mongoose.model("Order", orderSchema);

export default Order;
```

Controllers-

(Stripe Payment)



stripe.js

```
import Stripe from "stripe";
import dotenv from "dotenv";

dotenv.config();

export const stripe = new Stripe(process.env.STRIPE_SECRET_KEY);
```

payment.controller.js

```
import Coupon from "../models/coupon.model.js";
import Order from "../models/order.model.js";
import { stripe } from "../lib/stripe.js";

export const createCheckoutSession = async (req, res) => {
	try {
		const { products, couponCode } = req.body;

		if (!Array.isArray(products) || products.length === 0) {
			return res.status(400).json({ error: "Invalid or empty products array" });
		}

		let totalAmount = 0;

		const lineItems = products.map((product) => {
			const amount = Math.round(product.price * 100); // stripe wants u to send in the format of cents
			totalAmount += amount * product.quantity;

			return {
				price_data: {
					currency: "usd",
					product_data: {
						name: product.name,
						images: [product.image],
					},
					unit_amount: amount,
				},
				quantity: product.quantity || 1,
			};
		});

		let coupon = null;
		if (couponCode) {
			coupon = await Coupon.findOne({ code: couponCode, userId: req.user._id, isActive: true });
			if (coupon) {
				totalAmount -= Math.round((totalAmount * coupon.discountPercentage) / 100);
			}
```

```
                }

                const session = await stripe.checkout.sessions.create({
                        payment_method_types: ["card"],
                        line_items: lineItems,
                        mode: "payment",
                        success_url:
`${process.env.CLIENT_URL}/purchase-success?session_id={CHECKOUT_SESSI
ON_ID}`,
                        cancel_url: `${process.env.CLIENT_URL}/purchase-cancel`,
                        discounts: coupon
                                ? [
                                                {
                                                        coupon: await
createStripeCoupon(coupon.discountPercentage),
                                                },
                                  ]
                                : [],
                        metadata: {
                                userId: req.user._id.toString(),
                                couponCode: couponCode || "",
                                products: JSON.stringify(
                                        products.map((p) => ({
                                                id: p._id,
                                                quantity: p.quantity,
                                                price: p.price,
                                        }))
                                ),
                        },
                });

                if (totalAmount >= 20000) {
                        await createNewCoupon(req.user._id);
                }
                res.status(200).json({ id: session.id, totalAmount: totalAmount / 100 });
        } catch (error) {
                console.error("Error processing checkout:", error);
                res.status(500).json({ message: "Error processing checkout", error:
error.message });
        }
};

export const checkoutSuccess = async (req, res) => {
        try {
```

```
		const { sessionId } = req.body;
		const session = await stripe.checkout.sessions.retrieve(sessionId);

		if (session.payment_status === "paid") {
			if (session.metadata.couponCode) {
				await Coupon.findOneAndUpdate(
					{
						code: session.metadata.couponCode,
						userId: session.metadata.userId,
					},
					{
						isActive: false,
					}
				);
			}

			// create a new Order
			const products = JSON.parse(session.metadata.products);
			const newOrder = new Order({
				user: session.metadata.userId,
				products: products.map((product) => ({
					product: product.id,
					quantity: product.quantity,
					price: product.price,
				})),
				totalAmount: session.amount_total / 100, // convert from
cents to dollars,
				stripeSessionId: sessionId,
			});

			await newOrder.save();

			res.status(200).json({
				success: true,
				message: "Payment successful, order created, and
coupon deactivated if used.",
				orderId: newOrder._id,
			});
		}
	} catch (error) {
		console.error("Error processing successful checkout:", error);
		res.status(500).json({ message: "Error processing successful
checkout", error: error.message });
	}
```

```
};

async function createStripeCoupon(discountPercentage) {
	const coupon = await stripe.coupons.create({
		percent_off: discountPercentage,
		duration: "once",
	});

	return coupon.id;
}

async function createNewCoupon(userId) {
	await Coupon.findOneAndDelete({ userId });

	const newCoupon = new Coupon({
		code: "GIFT" + Math.random().toString(36).substring(2,
8).toUpperCase(),
		discountPercentage: 10,
		expirationDate: new Date(Date.now() + 30 * 24 * 60 * 60 * 1000), // 30
days from now
		userId: userId,
	});

	await newCoupon.save();

	return newCoupon;
}
```

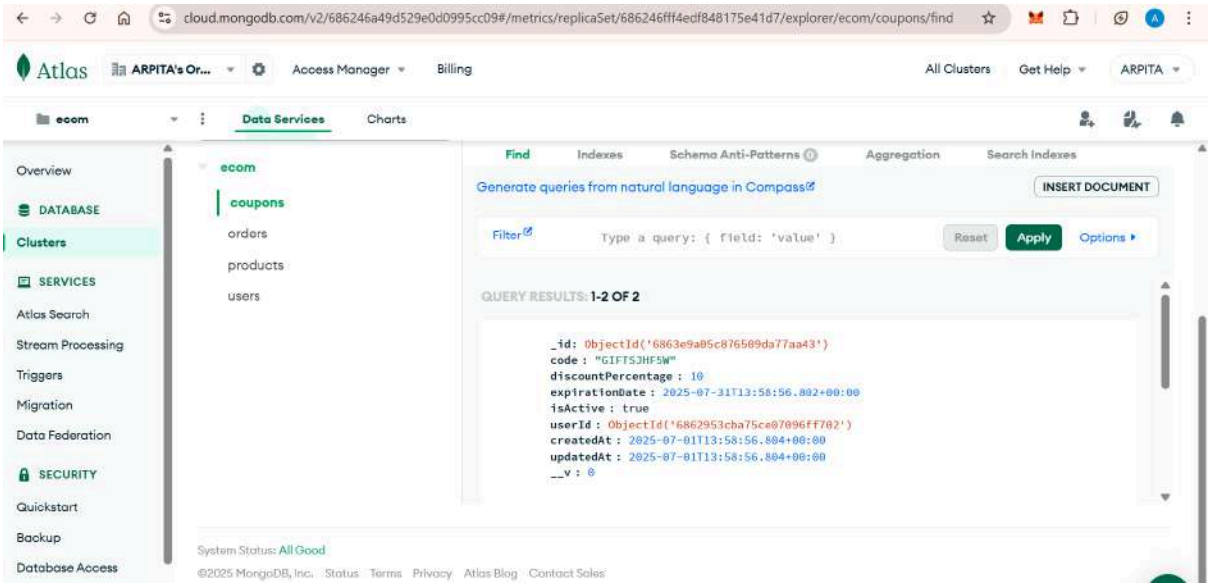
Routes-

payment.route.js

```
import express from "express";
import { protectRoute } from "../middleware/auth.middleware.js";
import { checkoutSuccess, createCheckoutSession } from
"../controllers/payment.controller.js";

const router = express.Router();

router.post("/create-checkout-session", protectRoute, createCheckoutSession);
router.post("/checkout-success", protectRoute, checkoutSuccess);

export default router;
```

Coupon-

coupon.model.js




```
import mongoose from "mongoose";

const couponSchema = new mongoose.Schema(
	{
		code: {
			type: String,
			required: true,
			unique: true,
		},
		discountPercentage: {
			type: Number,
			required: true,
			min: 0,
			max: 100,
		},
		expirationDate: {
			type: Date,
			required: true,
		},
		isActive: {
			type: Boolean,
			default: true,
		},
		userId: {
			type: mongoose.Schema.Types.ObjectId,
			ref: "User",
			required: true,
```

```
                unique: true,
            },
        },
        {
            timestamps: true,
        }
);

const Coupon = mongoose.model("Coupon", couponSchema);

export default Coupon;
```

Controllers-

coupon.controller.js

```
import Coupon from "../models/coupon.model.js";

export const getCoupon = async (req, res) => {
    try {
        const coupon = await Coupon.findOne({ userId: req.user._id, isActive:
true });
        res.json(coupon || null);
    } catch (error) {
        console.log("Error in getCoupon controller", error.message);
        res.status(500).json({ message: "Server error", error: error.message });
    }
};

export const validateCoupon = async (req, res) => {
    try {
        const { code } = req.body;
        const coupon = await Coupon.findOne({ code: code, userId:
req.user._id, isActive: true });

        if (!coupon) {
            return res.status(404).json({ message: "Coupon not found" });
        }

        if (coupon.expirationDate < new Date()) {
            coupon.isActive = false;
            await coupon.save();
            return res.status(404).json({ message: "Coupon expired" });
```

```
		}

		res.json({
			message: "Coupon is valid",
			code: coupon.code,
			discountPercentage: coupon.discountPercentage,
		});
	} catch (error) {
		console.log("Error in validateCoupon controller", error.message);
		res.status(500).json({ message: "Server error", error: error.message });
	}
};
```

Route-

coupon.route.js

```
import express from "express";
import { protectRoute } from "../middleware/auth.middleware.js";
import { getCoupon, validateCoupon } from "../controllers/coupon.controller.js";

const router = express.Router();

router.get("/", protectRoute, getCoupon);
router.post("/validate", protectRoute, validateCoupon);

export default router;
```

Project Link

Live Demo: https://youtu.be/TaA0Jzx1nKQ?si=svuInHK9T_mwtFTX
Deployment Link: https://commerce-7je4.onrender.com/

Weekly Report

WEEK 1
Date:6.6.25
Topic name: Node.js Installation and Backend Directory Structure

- Creating a Node.js Server, how to install packages and using nodemon for updating the server according to the changes in the code.

```
> JS server1.js > ...
const server=require("http")
server.createServer((req,res)=>{
    res.writeHead(200,{"Content-Type":"text/html"});
    res.write("<h1 style='color:red'>WELCOME TO SERVER 1</h1>");
    res.write("<h1>ELCOME EVERYONE</h1>");
    res.end();
}).listen(9000)
```

Date:9.6.25
Topic name: File Structure and Github Overview

- A basic Node.js and Express project typically includes a `server.js` or `app.js` file, a `routes/` folder for route handling, and a `package.json` for managing dependencies. It may also have `controllers/`, `views/`, and `public/` folders based on complexity.
- On GitHub, this structure helps organize code clearly, enables version control, and supports team collaboration.

npm init –y

npm i nodemon

```
{
 "name": "class1",
```

```
  "version": "1.0.0",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "app":"nodemon app.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "ejs": "^3.1.10",
    "nodemon": "^3.1.7",
    "pug": "^3.0.3"
  }
}
```

App.js

```
var server=require('http')
server.createServer((request,response)=>{
    response.writeHead(200,{"content-type":"text/html"})
```

```
    response.write("GOOD AFTERNOON")
    response.write("<br>")
    response.write("WELCOME")
    response.write("<br>")
    response.write("<h1 style='color: blue;'>GUNJANA DAS</h1>")
    response.end()
}).listen(8001)
```

File system

```
//console.log(process.argv[2])


// const fs=require("fs")
// const input=process.argv


// fs.writeFileSync(input[2],input[3])


const fs=require('fs')
const path=require('path')
const html=path.join(__dirname,'html')
const js=path.join(__dirname,'js')
```

```
for(i=1;i<=1000;i++){

    fs.writeFileSync(html+"/app"+i+".html","<h1>HII STUDENTS</h1>")

    fs.writeFileSync(js+"/app"+i+".js","<h1>HII STUDENTS</h1>")


}
```

```
const path=require('path')

const fs=require('fs')

const jsFile=path.join(__dirname,'js')

const phpFile=path.join(__dirname,'php')


for(i=1;i<=1000;i++){

    fs.writeFileSync(jsFile+'/app'+i+".js","good morning")
```

```
    fs.writeFileSync(phpFile+'/app'+i+".php","good morning")
}


fs.readFile(`${jsFile}/app1.js`,'utf-8',(err,item)=>{
    if(err) throw err
    console.log(item)
})
fs.appendFile(`${jsFile}/app1.js`," .hellow students",(err)=>{
    if (!err) console.log("ok done")
})


fs.unlink(`${jsFile}/app1.js`,(err)=>{
    if(!err) console.log("deleted")
})


fs.rename(`${jsFile}/app2.js`,"new2.js",(err)=>{
    if(err) throw err
    console.log("renamed")
})
```

```
{
  "name": "class4",
  "version": "1.0.0",
  "main": "app1.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "app1":"nodemon app1.js",
    "app2":"nodemon app2.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "express": "^4.21.1",
    "nodemon": "^3.1.7"
  }
}
```

```
const server=require('express')
const app=server()


app.get("/home",(req,res)=>{
    res.send("<h1>GOOD MORNING</h1>")
})


app.get("/about",(req,res)=>{
    res.send("<h1>this is about page</h1>")
})


app.get("/",(req,res)=>{
    res.send("<h1>this is home or root page</h1>")
})
app.get("/form",(req,res)=>{
    res.send(`
        <h1>USER FORM</h1>
        <form>
            Name:<input type="text" name="a1" placeholder="enter">
        </form>
```

```
        `)
})


app.get("/json",(req,res)=>{
    res.send([
        {"name":"rami","roll":1},``
        {"name":"rani","roll":2}
    ])
})
app.listen(8008)
```

navbar using express

```
const server =require('express')
const app=server()


app.get("/",(req,res)=>{
    res.send(`\
        <a href="/about">ABOUT</a>


        `)
})
```

```
app.get("/about",(req,res)=>{
    res.send("about page")
})

app.listen(8009)
```

https://github.com/arpi2003ta/vt_2025

Date: 11.6.25

Topic name: Express with Pug and EJS Templates

- Understand how to configure and use **EJS** and **Pug** with Express.
- Be able to pass data to templates and render dynamic content.
- Know how to **create forms**, accept user input, and respond with personalized views.

- How to set up EJS in an Express app,Syntax basics: <%= %>, <% %>, and how to embed JS directly in HTML,Passing data from the Express route to the EJS view.

```
doctype html
html(lang="en")
    head
        meta(charset="UTF-8")
        title= pageTitle
        link(rel="stylesheet", href="/css/style.css")
    body
        h1=pageTitle
        h2 HEllow, #{info.name}
        h3 STUDENT KNOWN LANGUAGES
        if info.languages.length > 0
            ul
                each lang in info.languages
                    li=lang
        else
            p No languages found.
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>STUDENT INFO</title>
</head>
<body>
    <h1><%= pageTitle %></h1>
    <h2><%= info.name %></h2>
        <h1>KNOWN LANGUAGES</h1>
        <% if(info.languages.length > 0){%>
        <ul>
            <% info.languages.forEach(function(lang) { %>
                <li><%= lang %></li>
            <% }) %>
        </ul>
        <% } else {%>
        <p>UNKNOWN LANGUAGES</p>
        <% }%>
</body>
</html>
```

```
const server=require("express");
const app=server();
const path=require("path")
const HTML=path.join(__dirname,"HTML")
 const file1=`${HTML}/Signup.html`
//app.set("view engine","ejs");//for using when as ejs
app.set("view engine","pug");//for using as pug
app.use(server.static(path.join(__dirname,"public")));
app.get("/pug",(req,res)=>{
    const user={
        name:"Raj kumar jha",
        languages: ["python", "javascript", "java"]
    }
    res.render('info', {
        pageTitle: "STUDENT INFO",
        info:user
    })
})
app.get("/ejs2",(req,res)=>{
    const user={
        name:"Raj kumar jha",
        languages: ["python", "javascript", "java"]
    }
    res.render('info', {
        pageTitle: "STUDENT INFO",
        info:user
    })
})
```

Date:13.6.25

Topic name: Node.js | Express | Nodemailer | Fake Data | Path | FS

- Creation of fake data of users to send mails to multiple users at a glance,description of different packages like cors,expresss,body-parser
- Usage of nodemailer, and path function to join.Creating basic **GET** and **POST** routes.Using `express.json()` middleware to parse JSON bodies.How to respond with plain text or JSON.Logging request data and server events.

```
const nodemailer=require("nodemailer")
const transporter=nodemailer.createTransport({
    service:'gmail',
    auth:{
        user:'shonababuu30@gmail.com',
        pass:'zrzt ivzm wczg khzy'
    }
})
const recipients=[
    'suvabiswas246@gmail.com',
    'suvajit.biswas.iotcs26@heritageit.edu.in'
]
async function sendBulkMails(){
    for(const recv of recipients){
        const mailOptions={
            from:'shonababuu30@gmail.com',
            to: recv,
            subject:'greeding',
            text:'Hello ,\n\n this is a test mail from suva'
        }
        try{
            const info=await transporter.sendMail(mailOptions)
           console.log("mail is sent to everyone")
    }catch(err){

        console.error( err);
    }
}
}
```

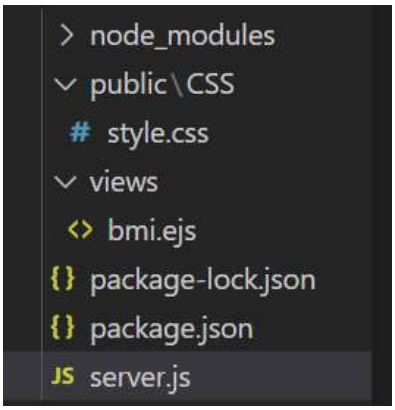
```
const {faker}=require('@faker-js/faker');
const fs=require("fs")
const generateFakeuserData=(num)=>{
    const users=[]
    for(let i=0;i<num;i++){
        const user={
            id:i+1,
            name:faker.name.firstName(),
            email:faker.internet.email(),
            phone:faker.phone.number(),
            address:{
                street:faker.address.streetAddress(),
                city:faker.address.city(),
                country:faker.address.country(),
            },
            crate_at:faker.date.past().toISOString()

        }
        users.push(user)
    }
    return users
}
const user_fakeData=generateFakeuserData(50)
//storing in a file
fs.writeFileSync("Store_fake_user_data.json",JSON.stringify(user_fakeData,null,2))
console.log("Fake user data generated and stored in Store_fake_user_data.json");
```

WEEK 2

Date:16.6.25

Topic name:BMI calculator

- How to use cors and its installation for data sharing across different platforms.
- Using body-parsers to calculate the bmi of the user from frontend to the backend.
- File structure



```
const express=require("express")
const app=express()
const path=require("path")
const cors=require("cors")
const bodyParser=require("body-parser")
//const folder_name=path.join(__dirname,"views")
app.set("view engine","ejs")
app.use(express.static(path.join("public")))
app.use(cors())
app.use(bodyParser.urlencoded({extended:true}))
app.get("/",(req,res)=>{
    res.render('bmi')
})
app.post("/bmi",(req,res)=>{
    const w=parseFloat(req.body.weight)
    const h=parseFloat(req.body.height)
    if(!w || w<=0 || !h || h<=0){
        res.send("Please enter valid weight and height")
    }else{
        const bmi=w/(h*h)
        let message=""
        if(bmi<1.5){
            message="normal"
        }else if(bmi<3){
            message="You are overweight"
        }else{
            message="You are obese"
        }
        res.render("bmi",{
```

Date:18.6.25
Topic name: Mongo Db and its handling.

- Installation of mongodb , explanation of how mongodb works,collection,creation of collection in a particular cluster of a database.
- Explanation of different methods used in mongo db,query injection and creation of document for a collection
- Explanation about bson and why its used over json and installation of mongosh package in the local device and its usage through the cmd.

Date:20.6.25
Topic name: Further explanation of DB.

- To connect with backend and database downloading of packages into the system mongodb and mongosh.
- Function of .env file and how to integrate the mondb using the mongo backend url provided by the mongodb application when a db is created.
- As mongodb is database which doesnot have a schema, so how it can be used by making a schema

WEEK 3

Date: 24.6.25

Topic name: Creating database in MongoDB

· Using a 4 file system creating a database in atlas using node. First we made an API then uploaded “product.json” file to backend database using mongoose.

· Node and express is used to make the api work and the backend is connected with frontend using it while nodemon run the server automatically.

Date: 25.6.25

Topic name: Creating an email, name json database in ATLAS

· Creating a file structure like,

· Connection creator

· Schema of database

· Main file

· Using nodemon, cors, body-parser, express made a js program that connected frontend with backend and updated the atlas.

Date: 27.06.25

Topic name: React installation and initialisation

· Making ORM(mapping backend items) also use view (using frontend via backend), using the command “npx create app app1” where “app1” is app name and “npx” is app extractor initiate it with “npm start” , the file structure looks like

· Using components named file inside “src” creating different functional components and use it

· Learned to use functional component and class component

Date:28.6.25

Topic name: React use

· Learned about jsx files which is js embedded html thus created forms also we can embed html in js file, these help us to perform logical portions with lightweight functionality.

WEEK 4

Date: 02.07.25

Topic name: React Event trigger /State change

· Using event triggering in react to initiate functions.

· Initiating constant variables with their initial states using function from react named useState and triggering them based on any function.

Date: 06.07.25

Topic name: Connecting Frontend, Backend, and MongoDB

· To connect the frontend, backend, and MongoDB in a MERN stack app, the **frontend (React)** communicates with the **backend (Express/Node.js)** using REST APIs. The backend processes requests and interacts with **MongoDB** using **Mongoose** for data storage and retrieval. This full-stack setup ensures seamless data flow between the UI and the database.

```
{
 "name": "serverside",
 "version": "1.0.0",
 "description": "",
 "main": "index.js",
 "scripts": {
   "test": "echo \"Error: no test specified\" && exit 1",
   "server": "nodemon server.js"
 },
 "keywords": [],
 "author": "",
 "license": "ISC",
 "type": "commonjs",
 "dependencies": {
   "bcryptjs": "^3.0.2",
```

```
    "body-parser": "^2.2.0",
    "cors": "^2.8.5",
    "dotenv": "^17.0.1",
    "express": "^5.1.0",
    "mongoose": "^8.16.1",
    "nodemon": "^3.1.10"
  }
}
```

```
MONGO_URI=mongodb://admin:admin123@localhost:27017/admin
PORT=5000
```

Server.js

```
// backend/server.js
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const dotenv = require('dotenv');
const userRoutes = require("./routes/userRoute");
```

```
dotenv.config();


const app = express();


app.use(cors());
app.use(express.json());


app.use('/api/users', userRoutes);


mongoose.connect(process.env.MONGO_URI)
.then(() => console.log('MongoDB connected'))
.catch((err) => console.log('DB Error:', err));


const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

userRouter.js

```
// backend/routes/userRoutes.js
const express = require('express');
const router = express.Router();
```

```
const User = require('../models/User');


// POST: Create User
router.post('/', async (req, res) => {
  try {
    const { name, email } = req.body;
    const newUser = new User({ name, email });
    await newUser.save();
    res.status(201).json(newUser);
  } catch (err) {
    if (err.code === 11000) {
      return res.status(400).json({ message: 'Email already exists' });
    }
    res.status(400).json({ message: err.message });
  }
});


// GET: All Users
router.get('/', async (req, res) => {
  try {
    const users = await User.find();
```

```
    res.json(users);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});


module.exports = router;
```

User.js

```
// backend/models/User.js
const mongoose = require('mongoose');


const userSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
  },
  email: {
    type: String,
    required: true,
```

```
    unique: true,
  },
});


module.exports = mongoose.model('User', userSchema);
```

FRONTEND

APP.js

```
// frontend/src/App.js
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import UserForm from './components/UserForm';


function App() {
  const [users, setUsers] = useState([]);


  const fetchUsers = () => {
    axios.get('http://localhost:5000/api/users')
      .then((res) => setUsers(res.data))
      .catch((err) => console.error(err));
```

```
  };

  useEffect(() => {
    fetchUsers();
  }, []);

  return (
    <div className="App">
      <h1>User Management</h1>
      <UserForm onUserCreated={fetchUsers} />
      <h2>User List</h2>
      <ul>
        {users.map(user => (
          <li key={user._id}>{user.name} ({user.email})</li>
        ))}
      </ul>
    </div>
  );
}

export default App;
```

UserForm.js

```
// frontend/src/components/UserForm.js
import React, { useState } from 'react';
import axios from 'axios';

const UserForm = ({ onUserCreated }) => {
  const [name, setName] = useState('');
  const [email, setEmail] = useState('');
  const [message, setMessage] = useState('');

  const handleSubmit = (e) => {
    e.preventDefault();

    axios.post('http://localhost:5000/api/users', { name, email })
      .then((res) => {
        setMessage('User created successfully');
        setName('');
        setEmail('');
        onUserCreated();
```

```
    })
    .catch((err) => {
      setMessage(err.response?.data?.message || 'Error creating user');
    });
};


return (
  <form onSubmit={handleSubmit}>
    <input
      type="text"
      placeholder="Name"
      value={name}
      onChange={(e) => setName(e.target.value)}
      required
    />
    <input
      type="email"
      placeholder="Email"
      value={email}
      onChange={(e) => setEmail(e.target.value)}
      required
```

```
      />
      <button type="submit">Create User</button>
      {message && <p>{message}</p>}
    </form>
  );
};


export default UserForm;
```