CPE-270_Database Systems

Complex Engineering Problem

Multi-Buyer & Vendor E-Commerce Platform

Lab Resource Person

Ma’am Nimra Ikram

Submitted by

Ali Hussain Sindhu (FA23-BCE-013)
Muhammad Ahmad (FA23-BCE-113)

Department of Computer Engineering

Abstract

This project presents the development of a scalable and role-based **E-Commerce Web Application** designed using a full-stack architecture. The system supports multiple user roles, including **Buyer, Vendor, and Admin**, enabling seamless product management, order processing, and analytics functionalities.

The backend is built with **Node.js** and connected to a **SQL Server** database, which handles complex business logic using **stored procedures**, **triggers**, and **role-based access control (RBAC)**. The frontend is developed using **React.js**, offering a user-friendly interface with real-time interaction through RESTful APIs.

Key features of the application include:

- Secure **user authentication and authorization**
- Product catalog with category and subcategory filtering
- Vendor-specific product management and order tracking
- Buyer-centric features such as cart, wishlist, checkout, and order history
- Admin dashboard for user role assignment and system monitoring
- Robust error handling and transaction management

The project follows a **three-tier architecture** comprising the **presentation layer (React frontend)**, **application layer (Node.js server)**, and **data layer (SQL Server)**. This modular approach ensures scalability, maintainability, and efficient separation of concerns.

This documentation outlines the architectural design, database schema, API endpoints, and key implementation details, providing a complete view of the project's functionality and technical scope.

CHAPTER #1
INTRODUCTION

Introduction

Objectives

The central objective of this project is to design and implement a secure, efficient, and scalable relational database system that serves as the backbone of an e-commerce platform. The system is intended to support essential business functionalities, including user account management, product listing, order handling, transaction processing, and customer feedback. The database should integrate smoothly with both the frontend and backend components of the application, enabling fast data access and robust performance under load. Key goals also include ensuring data consistency, enforcing integrity constraints, and supporting security features such as access control and injection prevention. Moreover, the database should be capable of accommodating increasing data volume and user traffic as the application scales, thereby promoting long-term reliability and system stability.

Background

Databases are a vital component in the architecture of modern e-commerce systems. These platforms generate and depend on large quantities of structured and unstructured data, including customer profiles, product inventories, order histories, and payment records. A well-designed database enables efficient storage, retrieval, and manipulation of this data while ensuring that operations are performed reliably and concurrently. It also plays a critical role in maintaining transactional accuracy and supporting real- time analytics.

As e-commerce continues to evolve, users expect quick responses, uninterrupted experiences, and secure transactions. These demands place significant pressure on the underlying database infrastructure. This project emphasizes the necessity of a robust and well-normalized relational database, capable of supporting concurrent users, enforcing business rules, and enabling decision-making through analytical queries. In this context, the database not only supports backend processes but also contributes directly to the frontend user experience and overall system effectiveness.

Overview

This project utilizes **Microsoft SQL Server** as the relational database management system, chosen for its performance, reliability, and advanced feature set. The database schema has been carefully normalized to ensure minimal data redundancy and strong data integrity. It models various entities involved in e-commerce operations, such as users (buyers and vendors), product catalogs, order records, transaction logs, and customer reviews. Relationships between these entities are defined clearly to support efficient joins and complex queries.

To handle business logic within the database layer, the implementation makes use of several SQL Server features. **Triggers** automate responses to data changes, such as adjusting product stock levels after a purchase. **Views** are employed to simplify access to aggregated or filtered data, providing a layer of abstraction for reporting and analysis. **Stored procedures** encapsulate recurring logic such as order creation and product updates, improving performance and security by minimizing raw query exposure.

The application's backend is built using **Node.js** and the **Express.js** framework, which serve as the communication layer between the database and the client interface. This backend exposes secure **RESTful APIs** that handle requests such as user registration, login, product browsing, and order placement. On the client side, a **React**-based frontend provides a modern, dynamic, and responsive user interface, allowing users to interact seamlessly with the platform. Altogether, the system represents a complete full-stack solution, with the database functioning as the core infrastructure supporting real-time operations, security, and analytical capabilities in an e-commerce environment.

CHAPTER #2
PROBLEM STATEMENT

Problem Statement

In today’s digital economy, e-commerce platforms serve millions of users conducting real-time transactions across the globe. These platforms depend heavily on robust and well-structured databases to ensure seamless performance, data consistency, and security. However, many such systems suffer due to poorly designed database architectures. Common issues include unnormalized schemas, data redundancy, lack of referential integrity, inefficient query performance, and inadequate mechanisms for ensuring transactional safety. These flaws often lead to slow system responses, data anomalies, inconsistent reporting, and vulnerability to malicious attacks such as SQL injection.

This project specifically addresses the problem of building a secure, scalable, and efficient **relational database system** for an e-commerce platform. It focuses on creating a schema that can handle high transaction volumes, provide real-time access to data, and maintain accuracy across multiple operations such as user management, order processing, product inventory updates, and customer reviews. The goal is to eliminate common inefficiencies by using **database normalization**, **optimized indexing**, and **advanced SQL constructs** like triggers, views, and stored procedures.

Real-World Significance

This problem is highly relevant in the real world, where businesses increasingly rely on e-commerce systems to reach and serve customers. A weak database layer can result in poor user experience, failed transactions, security breaches, and ultimately, a loss of customer trust and revenue. For example, if inventory data is not updated in real-time, customers may attempt to purchase out-of-stock items, causing frustration and operational overhead.

Moreover, with growing competition and data-driven decision-making, businesses require fast, reliable analytics from their platforms. A poorly designed database hampers this by making it difficult to run accurate and timely queries for sales trends, customer behavior, or operational performance.

Therefore, solving this problem is critical for any organization seeking to build a professional, secure, and scalable e-commerce solution. By addressing these real-world challenges, the project demonstrates the importance of sound database design in the development of reliable web applications.

CHAPTER #3
REQUIREMENT ANALYSIS

System Requirements

1.1 Functional Requirements

The database system for the e-commerce platform is designed to support a wide range of core operations essential for business functionality. The primary functional requirements are as follows:

- **User, Buyer, and Vendor Management**
  The system must store and manage detailed information about users, including buyers and vendors. This includes user registration, login credentials, profile data, and role-based access management.
- **Product Cataloging and Management**
  The database supports the addition, modification, and deletion of products. Each product is associated with categories, pricing, stock levels, and images. Vendors can update their product listings, while buyers can browse categorized inventories.
- **Shopping Cart and Order Processing**
  Buyers can add products to a shopping cart and proceed to checkout. The system handles order creation, linking users, products, and transactions. It updates inventory levels in real time and maintains records of payment status and delivery tracking.
- **Review and Rating System**
  Buyers are allowed to submit product reviews and ratings after completing a purchase. These reviews are stored and associated with both the product and the user, enabling future customers to make informed decisions.
- **Data Analytics and Reporting**
  Database **views** are implemented to generate summarized and analytical insights, such as top-selling products, order trends, and vendor performance. These views serve both business intelligence purposes and administrative monitoring.

1.2 Non-Functional Requirements

Beyond core operations, the system is expected to meet several critical non-functional requirements to ensure performance, security, and long-term maintainability:

- **Performance**
  The system must provide fast response times for both read and write operations. Efficient query execution is ensured through proper indexing and optimized schema design. Stored procedures are used for heavy or repetitive queries to reduce processing overhead.
- **Scalability**
  The database must be capable of handling increased data volume and concurrent user activity as the platform grows. Its structure should support horizontal and vertical scaling without significant performance degradation.
- **Security**
  The system implements strict data access controls based on user roles (e.g., admin, buyer, vendor). Measures such as input validation, parameterized queries, and stored procedures help prevent SQL injection and unauthorized data manipulation.
- **Reliability**
  The database adheres to **ACID (Atomicity, Consistency, Isolation, Durability)** properties to ensure transaction reliability. Backup strategies and recovery plans are incorporated to protect data against unexpected failures or corruption.
- **Usability**
  Although primarily back-end focused, the system is designed to integrate seamlessly with frontend interfaces. Well-structured views and stored procedures simplify data access for developers and allow for easy maintenance and debugging.

CHAPTER #4
SYSTEM DESIGN

System Design

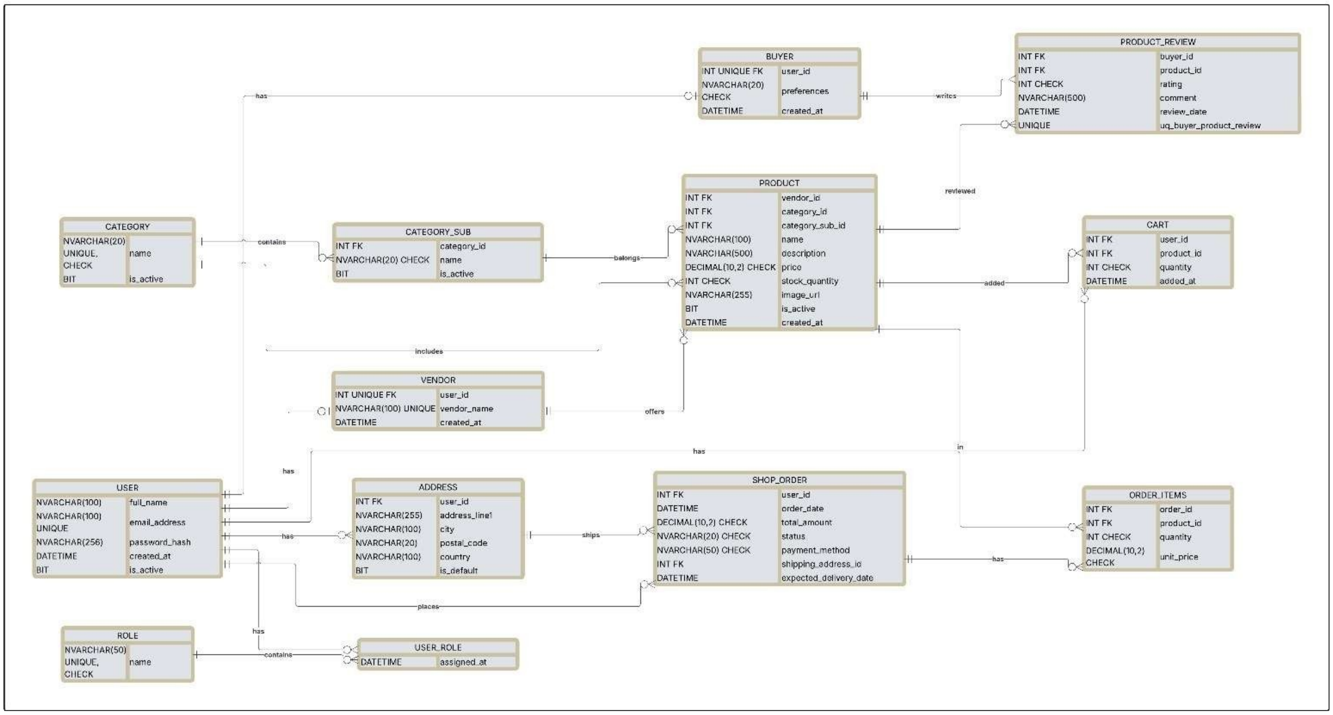
Database Design

The foundation of this project is a well-structured relational database designed to support all core operations of the e-commerce platform. The database is normalized to minimize redundancy, ensure consistency, and support efficient query execution.

The **Entity-Relationship Diagram (ERD)** is illustrated in **Figure 4.1: "ER_Diagram.jpg"**, which shows the key entities and their relationships, including:

- **Users**: Contains general user data including roles (buyer/vendor).
- **Products**: Stores product information along with foreign keys to vendor and category.
- **Orders**: Links buyers to purchased products, with timestamps and status.
- **Order_Details**: Captures the many-to-many relationship between orders and products.
- **Reviews**: Enables buyers to submit feedback for products.
- **Cart**: Temporarily stores selected products before checkout.

This schema is also visually summarized in **Figure 4.1**



BUYER	
INT UNIQUE FK	user_id
NVARCHAR(20) CHECK	preferences
DATETIME	created_at

PRODUCT_REVIEW	
INT FK	buyer_id
INT FK	product_id
INT CHECK	rating
NVARCHAR(500)	comment
DATETIME	review_date
UNIQUE	uq_buyer_product_review

CATEGORY	
NVARCHAR(20) UNIQUE, CHECK	name
BIT	is_active

CATEGORY_SUB	
INT FK	category_id
NVARCHAR(20) CHECK	name
BIT	is_active

PRODUCT	
INT FK	vendor_id
INT FK	category_id
INT FK	category_sub_id
NVARCHAR(100)	name
NVARCHAR(500)	description
DECIMAL(10,2) CHECK	price
INT CHECK	stock_quantity
NVARCHAR(255)	image_url
BIT	is_active
DATETIME	created_at

CART	
INT FK	user_id
INT FK	product_id
INT CHECK	quantity
DATETIME	added_at

VENDOR	
INT UNIQUE FK	user_id
NVARCHAR(100) UNIQUE	vendor_name
DATETIME	created_at

USER	
NVARCHAR(100)	full_name
NVARCHAR(100) UNIQUE	email_address
NVARCHAR(256)	password_hash
DATETIME	created_at
BIT	is_active

ADDRESS	
INT FK	user_id
NVARCHAR(255)	address_line1
NVARCHAR(100)	city
NVARCHAR(20)	postal_code
NVARCHAR(100)	country
BIT	is_default

SHOP_ORDER	
INT FK	user_id
DATETIME	order_date
DECIMAL(10,2) CHECK	total_amount
NVARCHAR(20) CHECK	status
NVARCHAR(50) CHECK	payment_method
INT FK	shipping_address_id
DATETIME	expected_delivery_date

ORDER_ITEMS	
INT FK	order_id
INT FK	product_id
INT CHECK	quantity
DECIMAL(10,2) CHECK	unit_price

ROLE	
NVARCHAR(50) UNIQUE, CHECK	name

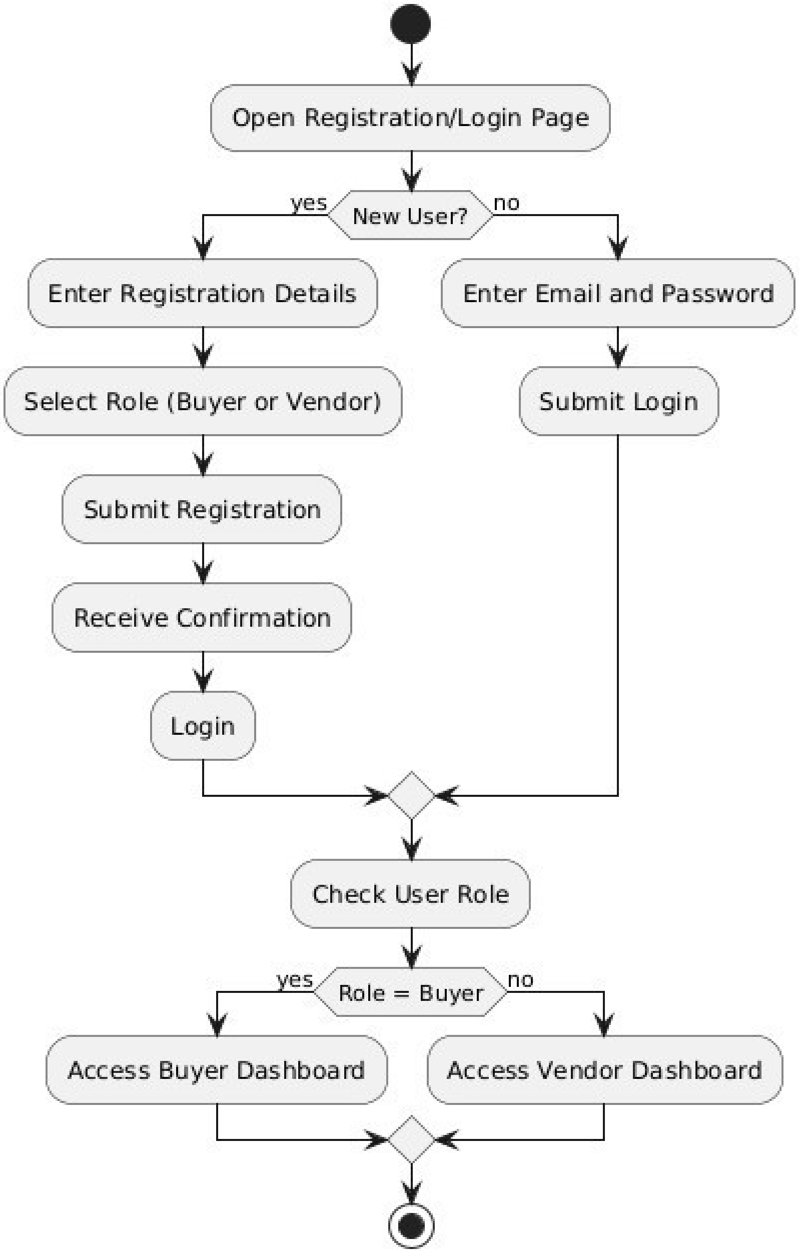
USER_ROLE	
DATETIME	assigned_at

Activity Diagram

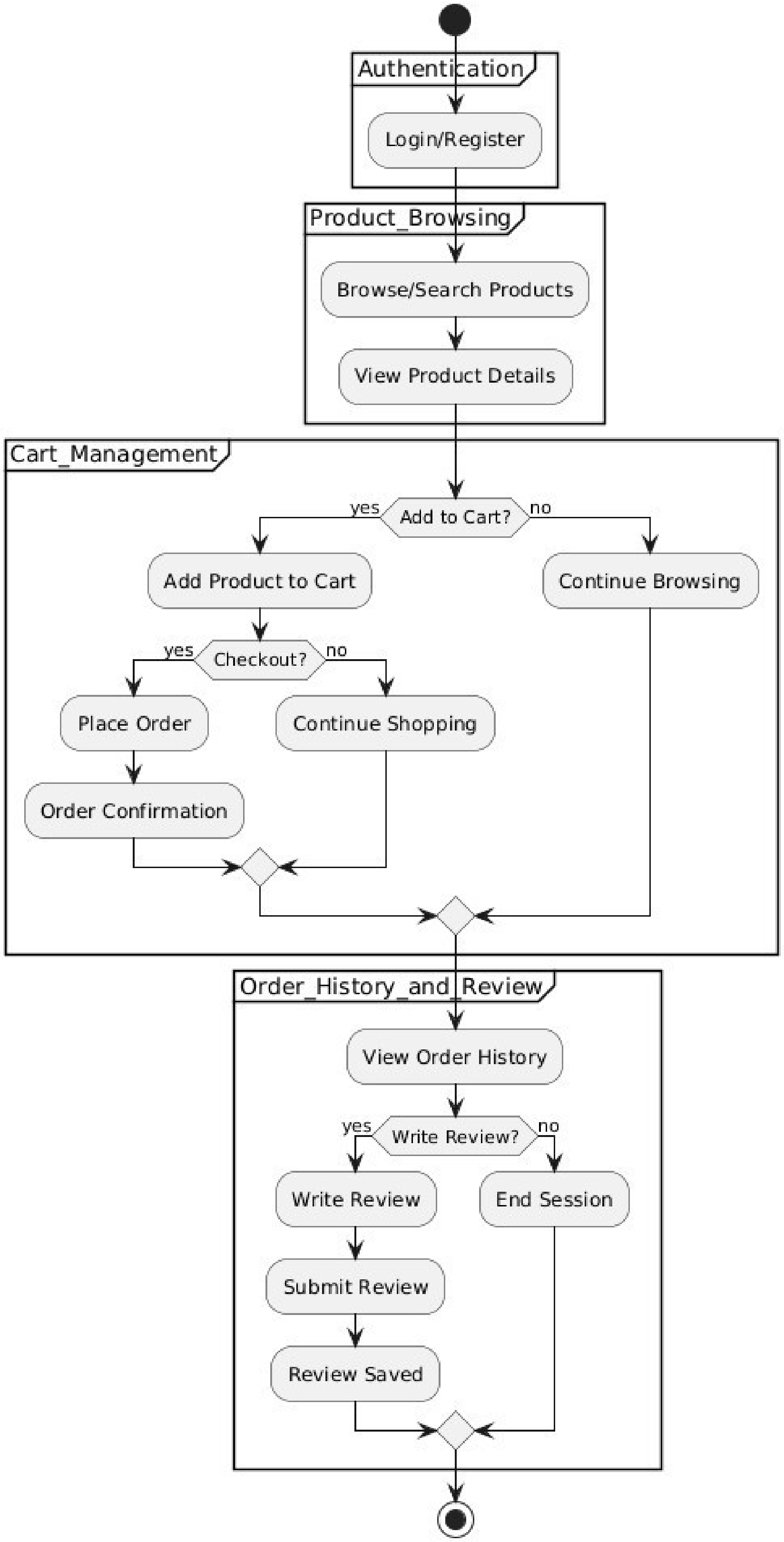
Activity diagrams provide a high-level view of the system workflows and logic behind the main processes such as user registration, product purchase, and order fulfillment.

Key activity diagrams included:

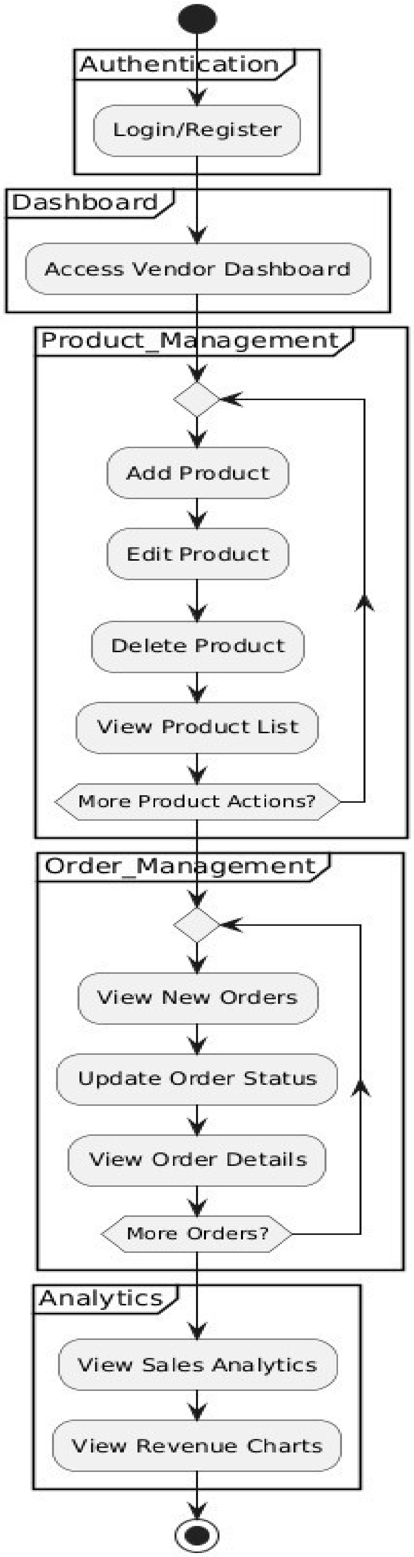
User Registration Activity.



Buyer’s Activity



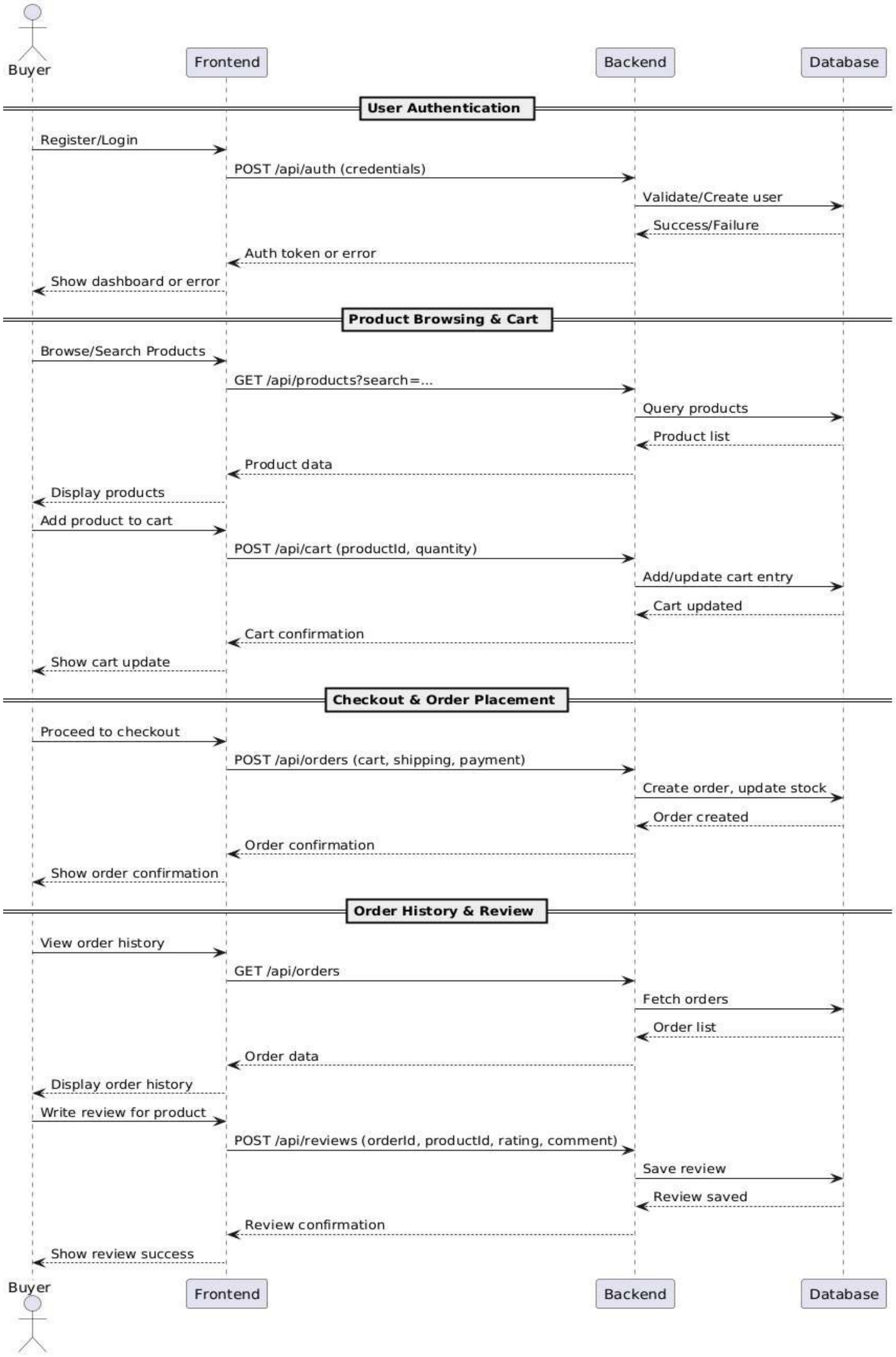
Vendor’s Activity



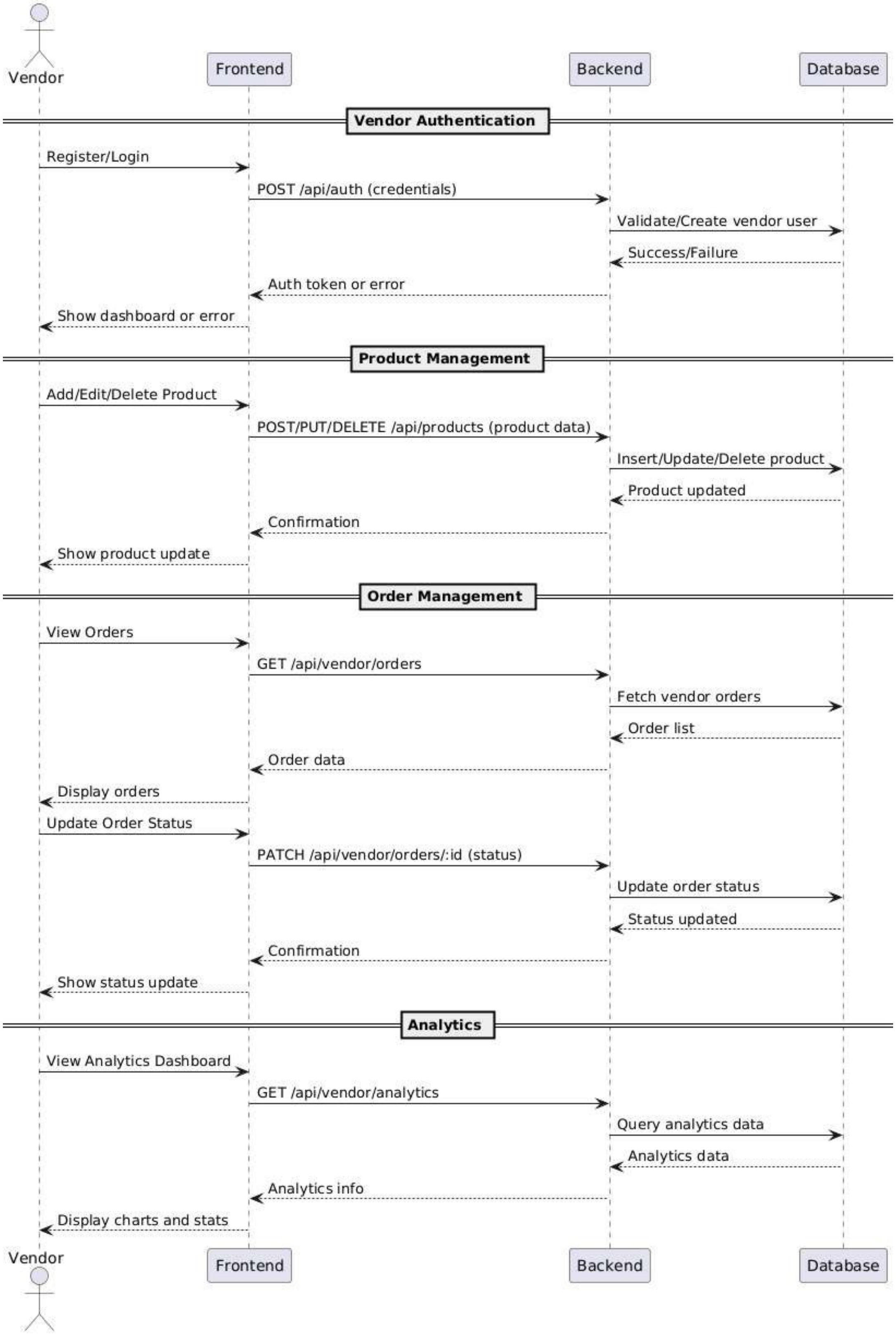
Sequence Diagram

Sequence diagrams explains the whole working of the system.

Buyer Sequence Diagram

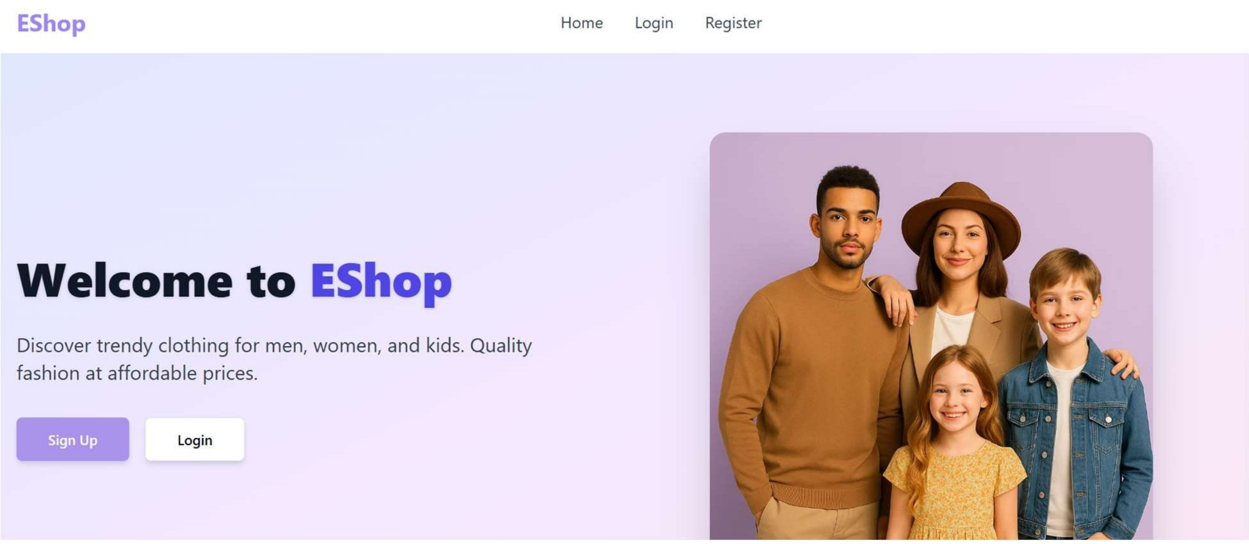


Vendor Sequence Diagram



Front-End Design

Home Page

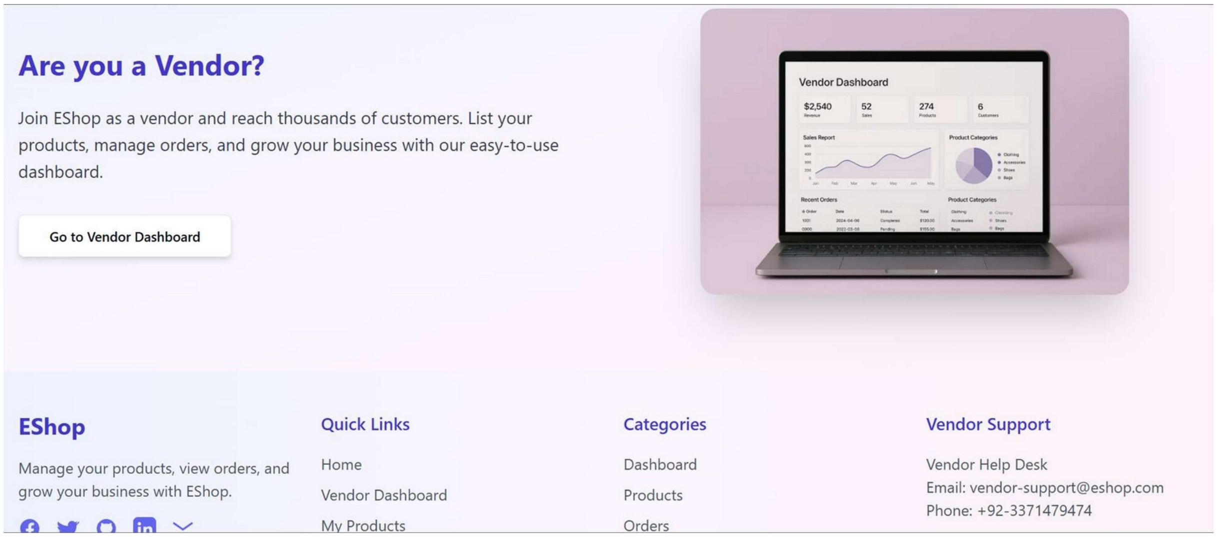


EShop

Home Login Register

Welcome to EShop

Discover trendy clothing for men, women, and kids. Quality fashion at affordable prices.



Are you a Vendor?

Join EShop as a vendor and reach thousands of customers. List your products, manage orders, and grow your business with our easy-to-use dashboard.

EShop

Manage your products, view orders, and grow your business with EShop.

Quick Links

Home
Vendor Dashboard
My Products

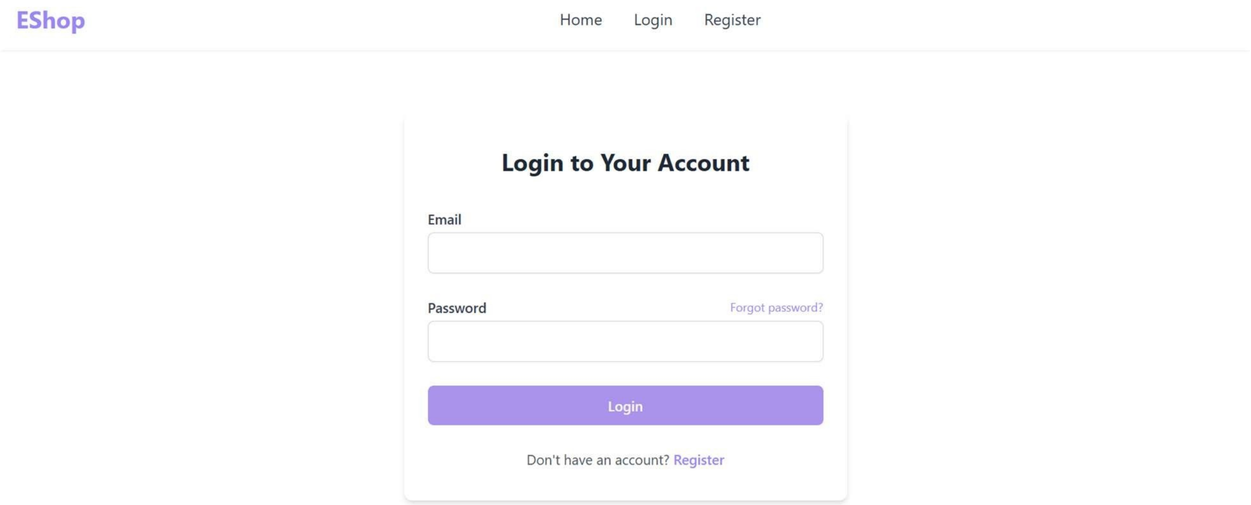
Categories

Dashboard
Products
Orders

Vendor Support

Vendor Help Desk
Email: vendor-support@eshop.com
Phone: +92-3371479474

Login Page



Login to Your Account

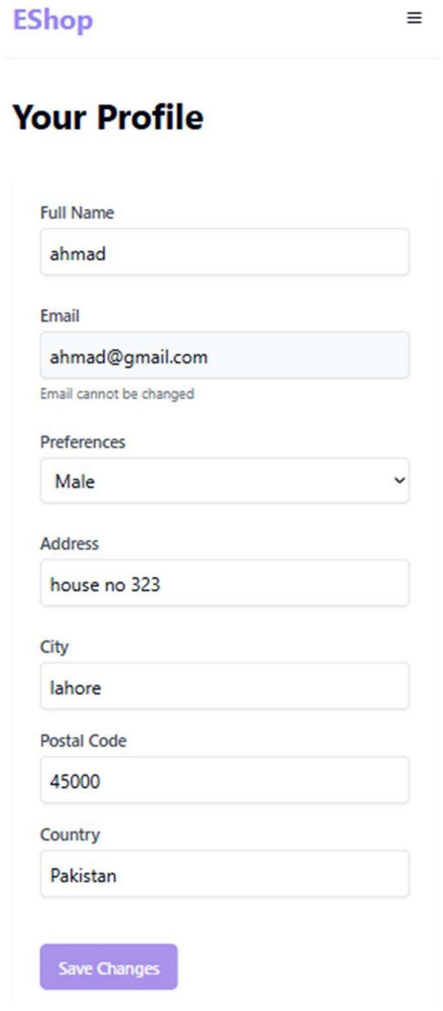
Email

Password

Forgot password?

Don't have an account? Register

Profile



Your Profile

Full Name

ahmad

Email

ahmad@gmail.com

Email cannot be changed

Preferences

Male

Address

house no 323

City

lahore

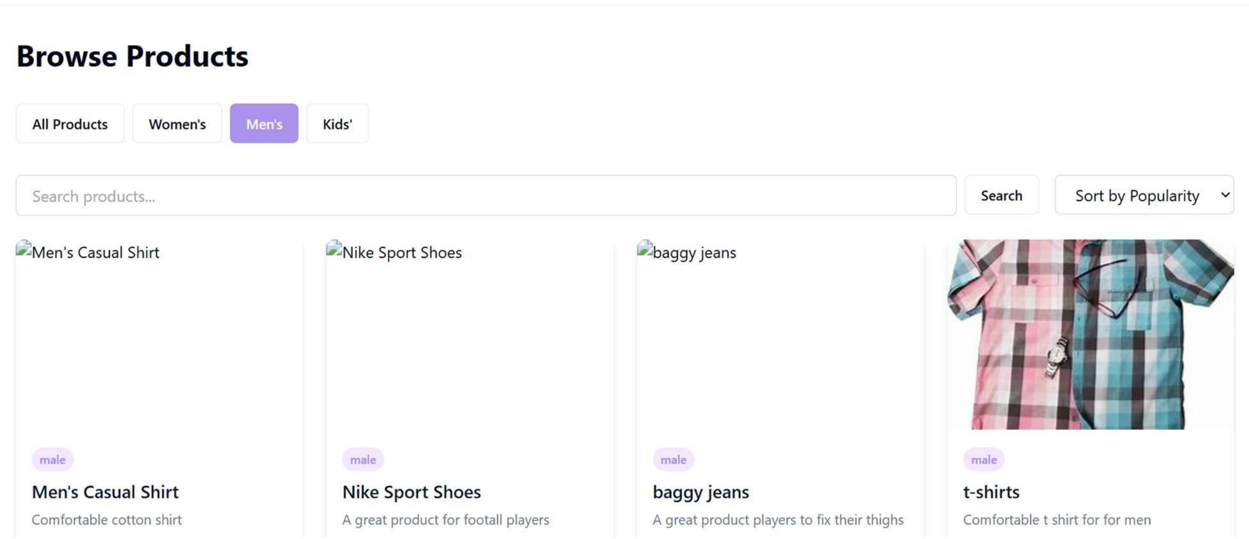
Postal Code

45000

Country

Pakistan

Buyer Dashboard



Browse Products

Men's Casual Shirt

Nike Sport Shoes

baggy jeans

male

Men's Casual Shirt

Comfortable cotton shirt

male

Nike Sport Shoes

A great product for footall players

male

baggy jeans

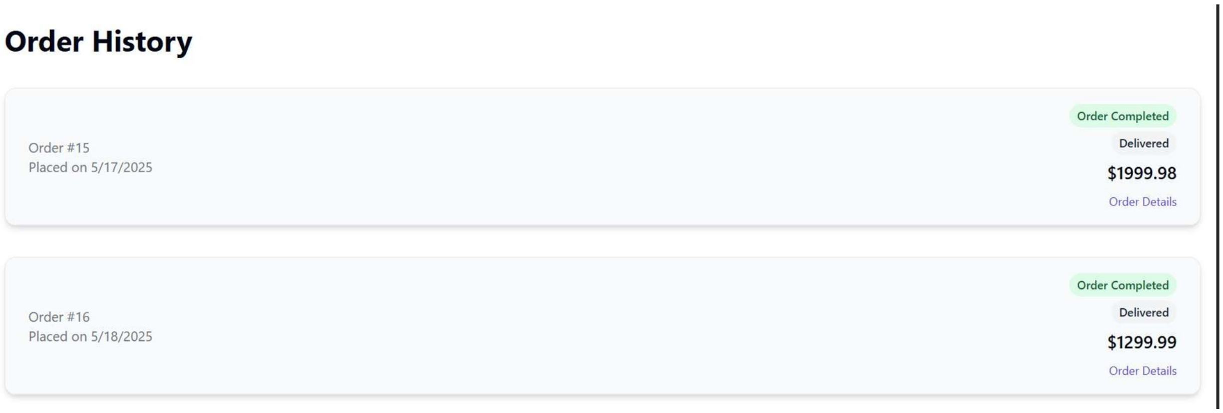
A great product players to fix their thighs

male

t-shirts

Comfortable t shirt for for men

Order History



Order History

Order #15

Placed on 5/17/2025

Order Completed

Delivered

$1999.98

Order #16

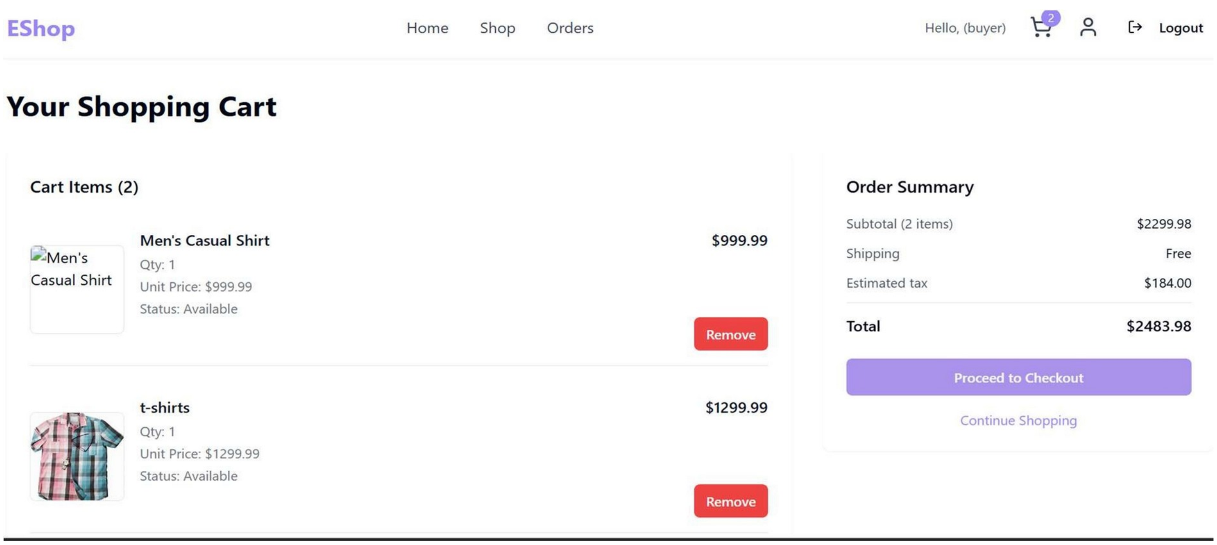
Placed on 5/18/2025

Order Completed

Delivered

$1299.99

Cart View



Your Shopping Cart

Cart Items (2)

Men's Casual Shirt

Men's Casual Shirt $999.99

Qty: 1

Unit Price: $999.99

Status: Available

t-shirts $1299.99

Qty: 1

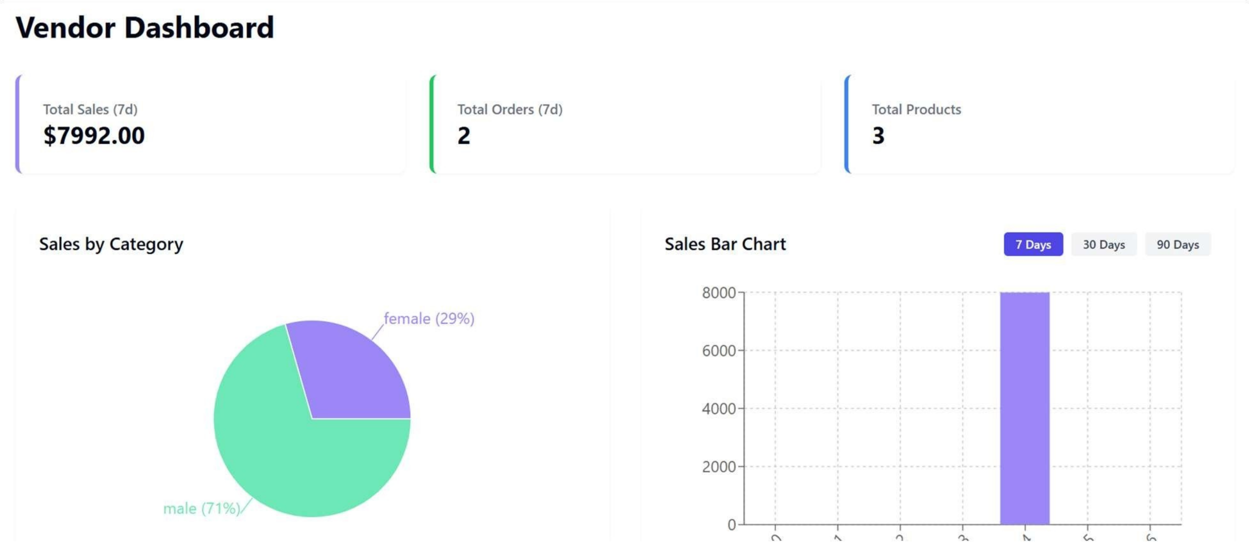
Unit Price: $1299.99

Status: Available

Order Summary

Subtotal (2 items)	$2299.98
Shipping	Free
Estimated tax	$184.00
Total	$2483.98

Vendor Dashboard



Vendor Dashboard

Total Sales (7d)
$7992.00

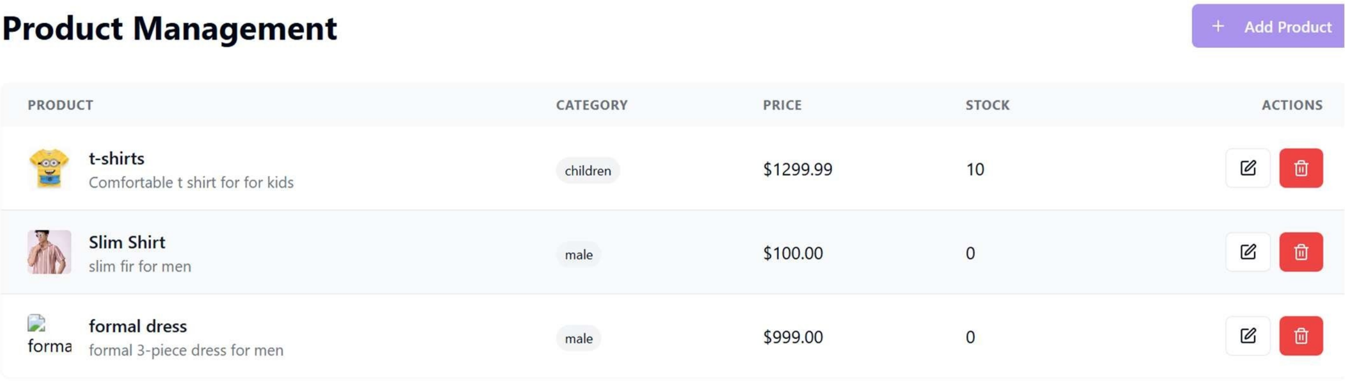
Total Orders (7d)
2

Total Products
3

Sales by Category

Sales Bar Chart

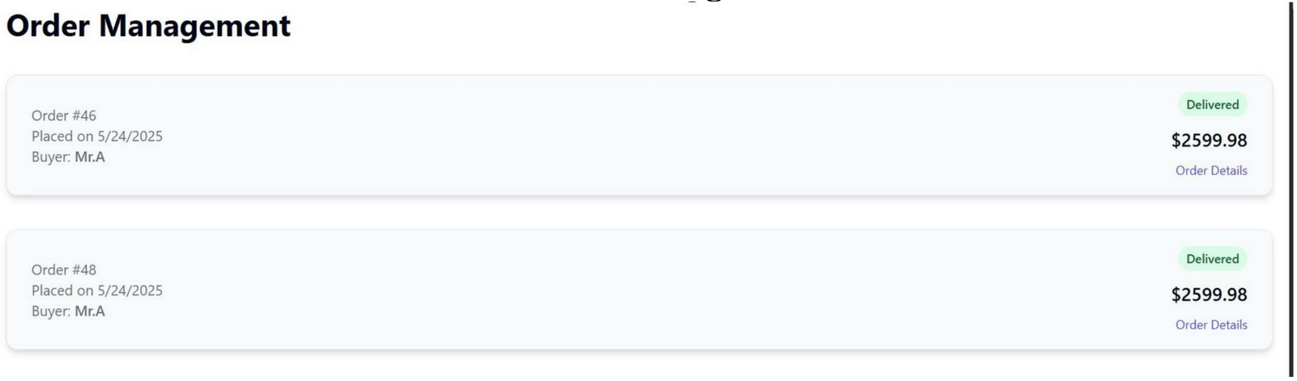
Product Management



Product Management

PRODUCT	CATEGORY	PRICE	STOCK	ACTIONS
t-shirts Comfortable t shirt for for kids	children	$1299.99	10	✎ 🗑
Slim Shirt slim fir for men	male	$100.00	0	✎ 🗑
forma formal dress formal 3-piece dress for men	male	$999.00	0	✎ 🗑

Order Management



Order Management

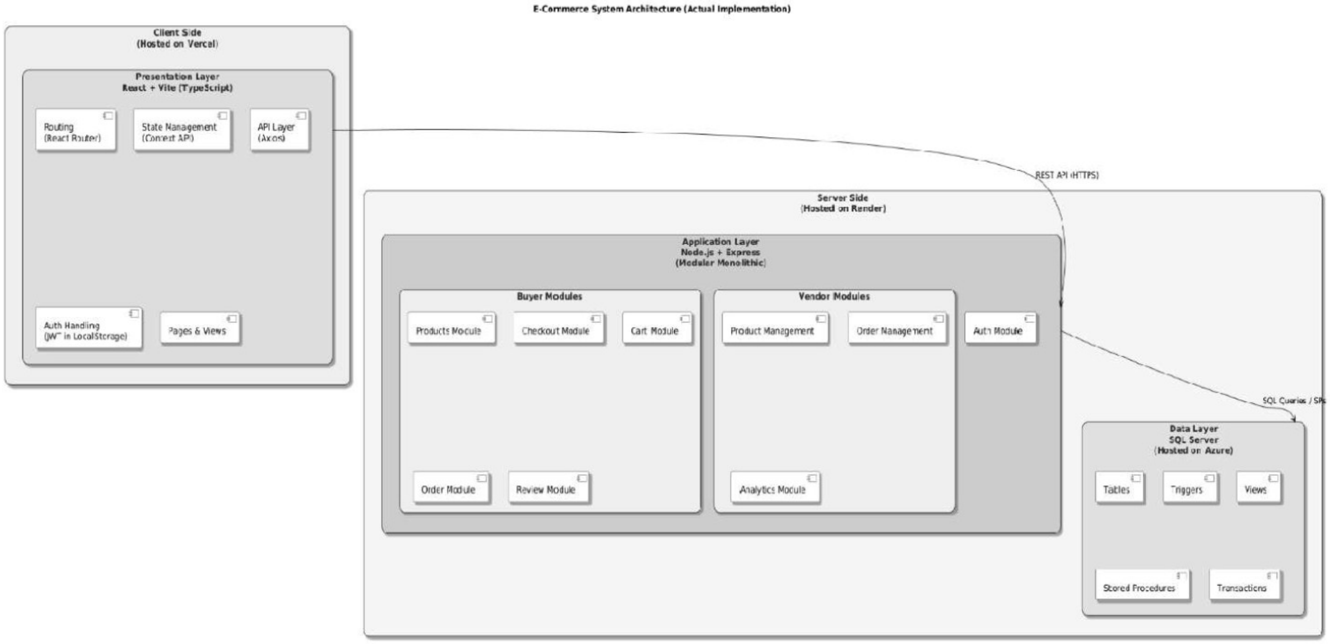
Order #46
Placed on 5/24/2025
Buyer: **Mr.A**

Delivered
$2599.98
Order Details

Order #48
Placed on 5/24/2025
Buyer: **Mr.A**

Delivered
$2599.98
Order Details

Integration



CHAPTER #5
TESTING

Testing

Testing was carried out to ensure that all system components function as expected and meet the project's functional and non-functional requirements. The testing process involved validating both backend and frontend functionalities using industry-standard tools like **Postman** and **Chrome Developer Tools**.

Backend API Testing (Postman)

To ensure the reliability and robustness of the backend system, thorough API testing was conducted using Postman. This helped validate business logic, security, and data flow across all endpoints.

Key Areas Tested Using Postman

1. **Endpoint Validation**
   All API routes (e.g., /register, /login, /products, /orders) were tested for correct request methods and routes. This ensured that APIs respond accurately to both valid and invalid paths or HTTP methods.
2. **Input Validation & Error Handling**
   Each API was tested with valid, missing, and incorrect input data to check whether the server returns proper error messages (e.g., 400 Bad Request, 401 Unauthorized). This confirms that server-side validation is implemented correctly.
3. **Authentication & Authorization**
   Login APIs were tested for generating tokens or sessions. Protected routes were tested using headers (e.g., Bearer tokens) to ensure unauthorized users cannot access restricted data.
4. **Data Consistency Checks**
   After successful operations (e.g., order placement or user registration), follow-up GET requests were made to confirm that the database reflects accurate and consistent changes.
5. **SQL Injection Prevention**
   Malicious inputs were submitted through parameters and form fields to simulate SQL injection attacks. The system's defense mechanisms—like parameterized queries or ORM protections— were verified to be working.

6. **Response Time & Status Codes**
   Each API was evaluated for acceptable response time under normal conditions and confirmed to return the correct HTTP status codes (200 OK, 201 Created, 404 Not Found, 500 Internal Server Error, etc.).
7. **Business Logic Enforcement**
   Triggers and stored procedures were indirectly tested—for example, placing an order and then verifying if the inventory stock was automatically reduced, or if an audit table was updated.

4.1Frontend Testing (Chrome Developer Tools)

Frontend testing focused on verifying the responsiveness, usability, and real-time behavior of the application. The **Chrome Developer Tools** suite was used to inspect network requests, check console logs, and analyze layout rendering.

Key Areas Tested:

- **Form Validations**: Ensured input fields on registration, login, and checkout forms enforce correct patterns and required fields.
- **Network Requests**: Confirmed that API calls from the frontend return correct responses and update the UI accordingly.
- **Responsiveness**: Tested UI on different screen sizes (mobile, tablet, desktop).
- **Error Handling**: Verified that failed API calls or invalid user actions display meaningful error messages.
- **Component Rendering**: Ensured that components render dynamically based on user actions (e.g., login state, cart updates).

CHAPTER #6

IMPLEMENTATION

Implementation

Tools and Technologies

The system was developed using a combination of frontend, backend, and database technologies. Development tools were selected to ensure compatibility, performance, scalability, and ease of integration. Version control and testing tools were also utilized to maintain code quality and system reliability throughout the development lifecycle.

Database Implementation

The database was designed using a relational model and implemented using a robust database management system. The schema was normalized to ensure data consistency and to eliminate redundancy. Relationships between entities were established using appropriate constraints, and indexing strategies were applied to optimize query performance.

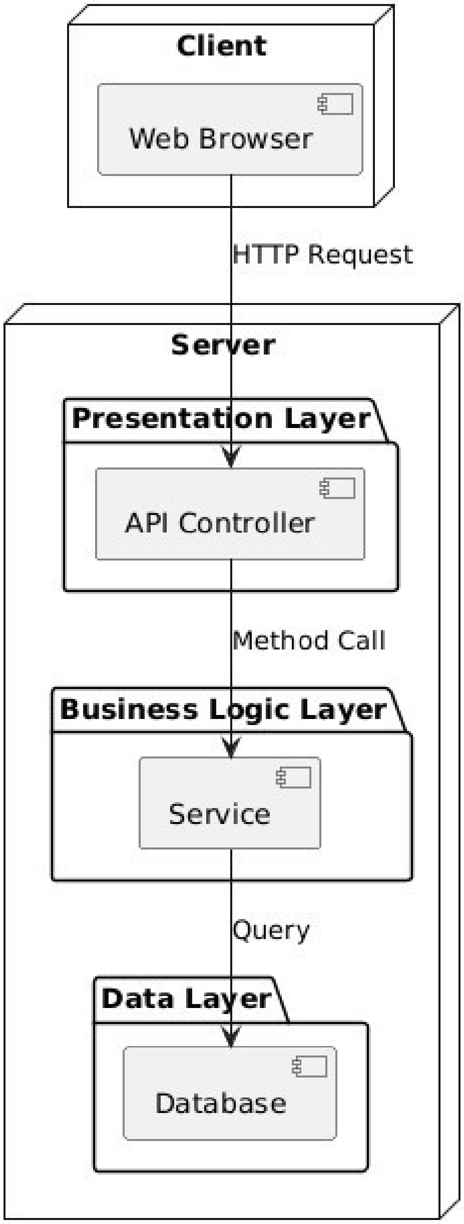
Advanced features such as triggers, views, and stored procedures were integrated to enforce business rules, automate processes, and simplify complex operations. The implementation also included data validation mechanisms, backup planning, and measures to ensure the integrity and security of transactional data.

Front-End Implementation

The user interface was designed to provide a smooth and intuitive experience across various devices. Core features such as user registration, product browsing, shopping cart, and order placement were implemented to interact dynamically with the backend services.

The frontend communicates with the backend via secure API calls to fetch and update data. Client-side validation, state management, and responsive design principles were applied to enhance usability and performance.

Inter Module Conectivity



CHAPTER #7
CONCLUSION

Conclusion

Summary

The primary goal of this project was to design and implement a scalable, secure, and efficient database-driven system tailored for an e-commerce platform. Through careful planning and execution, the project successfully achieved key objectives such as structured data management, secure user operations, smooth integration between the front-end and back-end, and reliable transaction processing. By incorporating best practices in database design and full-stack development, the final system demonstrates robust functionality and practical applicability in real-world e-commerce scenarios.

Lessons Learned

Working on this project provided valuable insights into the end-to-end development lifecycle of database-centric applications. Important technical lessons included applying normalization principles, ensuring data consistency, managing secure API interactions, and implementing business logic at the database level. Additionally, the project helped reinforce non-technical skills such as project planning, version control, collaborative debugging, and documentation. It also emphasized the importance of testing, security considerations, and performance optimization in full-stack development.

Future Enhancements

While the current version of the system fulfills essential requirements, several enhancements can be considered to improve and expand its functionality:

- **Admin Panel:** Introducing a comprehensive dashboard for managing users, products, and transactions more effectively.
- **Recommendation System:** Implementing AI-based product recommendations based on user behavior and past purchases.
- **Payment Gateway Integration:** Adding secure and real-time payment processing for a complete purchase cycle.
- **Mobile Application Support:** Extending the platform to mobile devices for broader accessibility.
- **Automated Reporting:** Generating business intelligence reports for sales, inventory trends, and customer insights.

These future improvements can make the system more intelligent, user-friendly, and adaptable to evolving business needs.