

# **Multi Buyer and Vendor Ecommerce**

**Complex Engineering Problem**

**Final Report**

**Spring-25**



Department of Computer Engineering

COMSATS University Islamabad, Lahore Campus

## Project Detail

Project Group Members			
Sr.#	Reg. #	Student Name	Email ID
(i)	FA23-BCE-013	Ali Hussain Sindhu	<a href="mailto:FA16-BCS-013@cuilahore.edu.pk">FA16-BCS-013@cuilahore.edu.pk</a>
(ii)	FA23-BCS-113	Muhammad Ahmad	<a href="mailto:FA16-BCS-113@cuilahore.edu.pk">FA16-BCS-113@cuilahore.edu.pk</a>

\*The candidates confirm that the work submitted is their own and appropriate credit has been given where reference has been made to work of other

## **Abstract**

Our project, AQI Predictor, is an application to predict the quality of Air aka Air Quality Index (AQI) Level using images. The user can input the images by capturing directly from the smartphone's camera or already captured ones from the gallery. The app uses a trained model that's hosted on local server at the backend. The trained model is a Convolutional Neural Network that uses machine learning techniques to help us in predicting the Air Quality level (AQL) correctly. The model is trained on a relevant dataset in order to achieve the accuracy. Along with the app, a website will be created that will be synchronized with the app and the data will be collected. The data then will be used to generate a map of the areas the Air quality level has been calculated for. This will help people in deciding which areas have what kind of air quality. The quality of air will be predicted by the application providing us Air Quality Level which will further tell us more details such as if the air is harmful or good and how necessary precautions can be taken. These factors and results will help us and the authorities including the Government and relevant governmental and other bodies decide certain measures to make the air quality better.



# Table of Contents

**Project Details** ..... 2

**Table of Contents** ..... 6

**1. Introduction** ..... 13

**2. Requirement Analysis** ..... 18

    2.1 Functional Requirements ..... 18

    2.2 Non-Functional Requirements ..... 19

**3. System Design** ..... 21

    3.1 Database Design (ER Diagram & Schema) ..... 21

    3.2 Front-End Design ..... 23

    3.3 System Integration (Front-End & Database) ..... 24

**4. Implementation** ..... 25

    4.1 Tools and Technologies Used ..... 25

    4.2 Database Implementation (Queries, Triggers) ..... 26

    4.3 Front-End Implementation ..... 28

        4.3.1 CRUD Operations ..... 28

        4.3.2 Challenges and Solutions ..... 29

**5. Testing** ..... 31

**6. Conclusion** ..... 36

    7.1 Summary of Work ..... 36

    7.2 Lessons Learned ..... 37

    7.3 Future Enhancements ..... 37

**7. References** ..... 38

# **CHAPTER #1**

# **INTRODUCTION**

# 1. Introduction

## 1.1 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.

## 1.2 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.

## 1.3 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**

## 2. 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**

# **REQUIRMENT**

# **ANALYSIS**

### 3. System Requirements

#### 3.1 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.

#### 3.2 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**

## 4 System Design

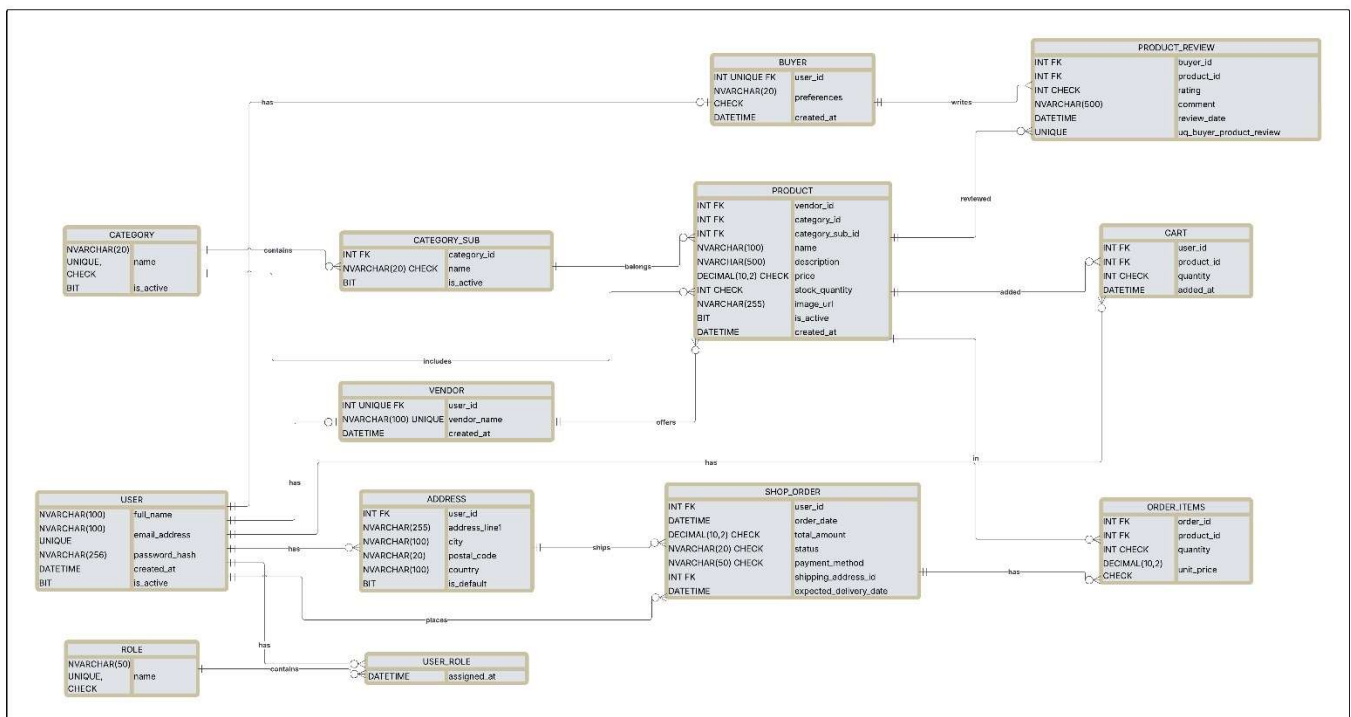
### 4.1 Database Design

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**

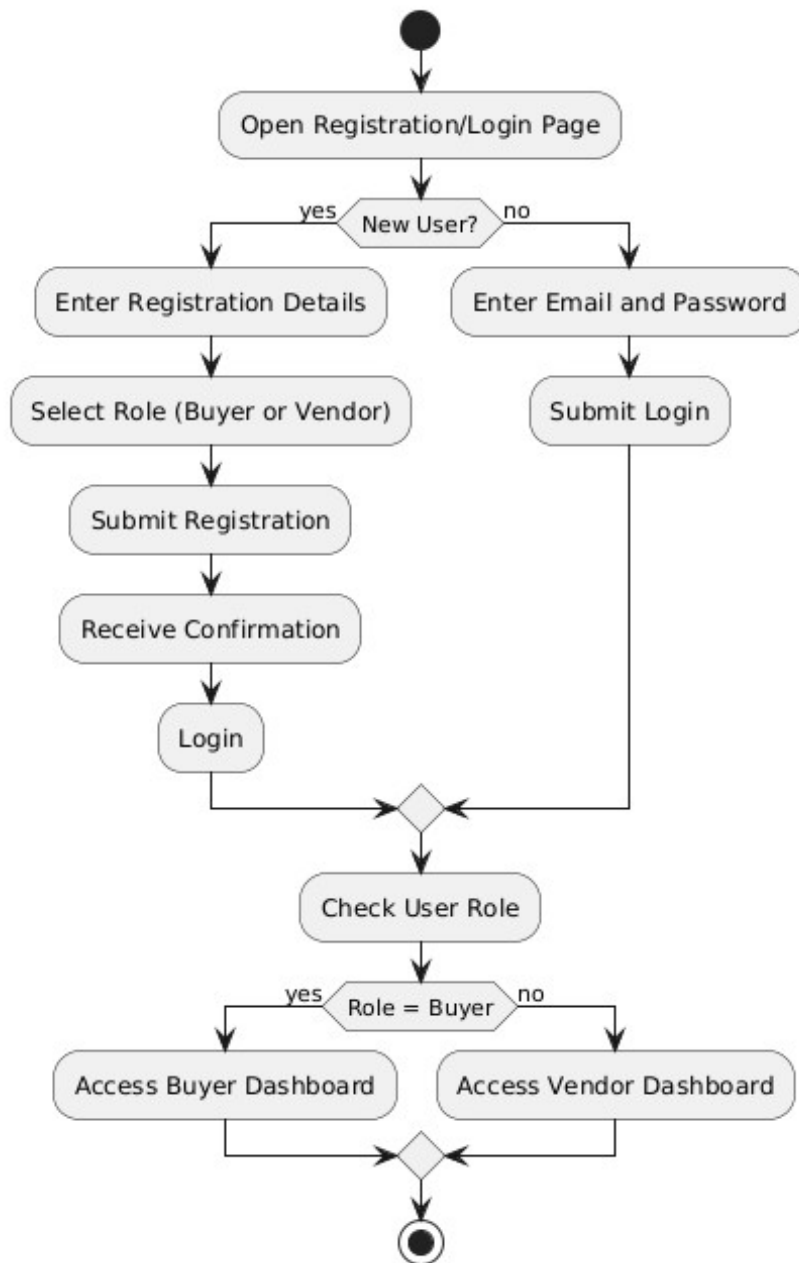


## 4.2 Activity Diagram

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.

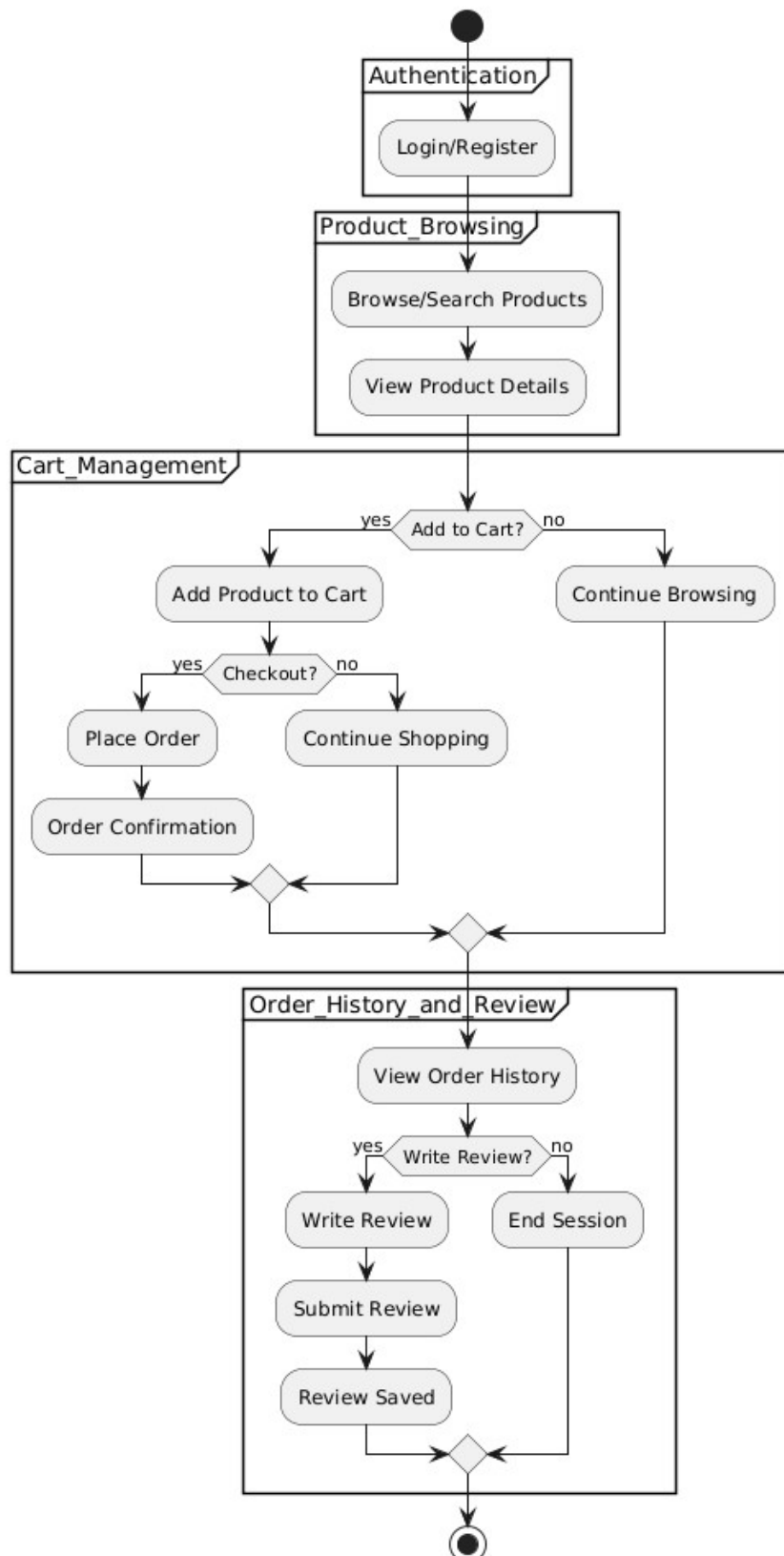
Key activity diagrams included:

### 4.2.1 User Registration Activity.

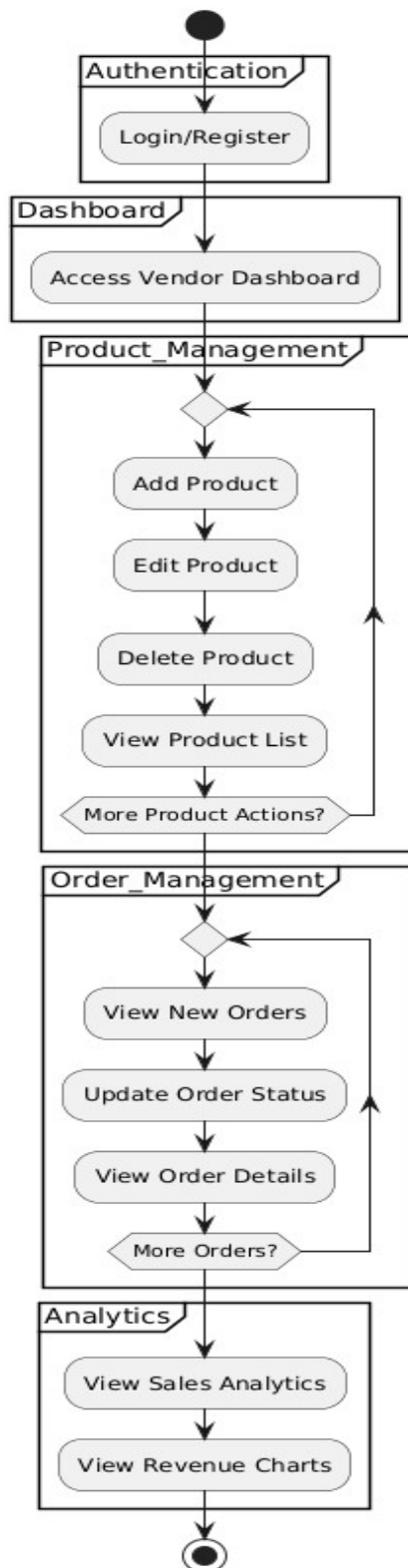




#### 4.2.2 Buyer's Activity



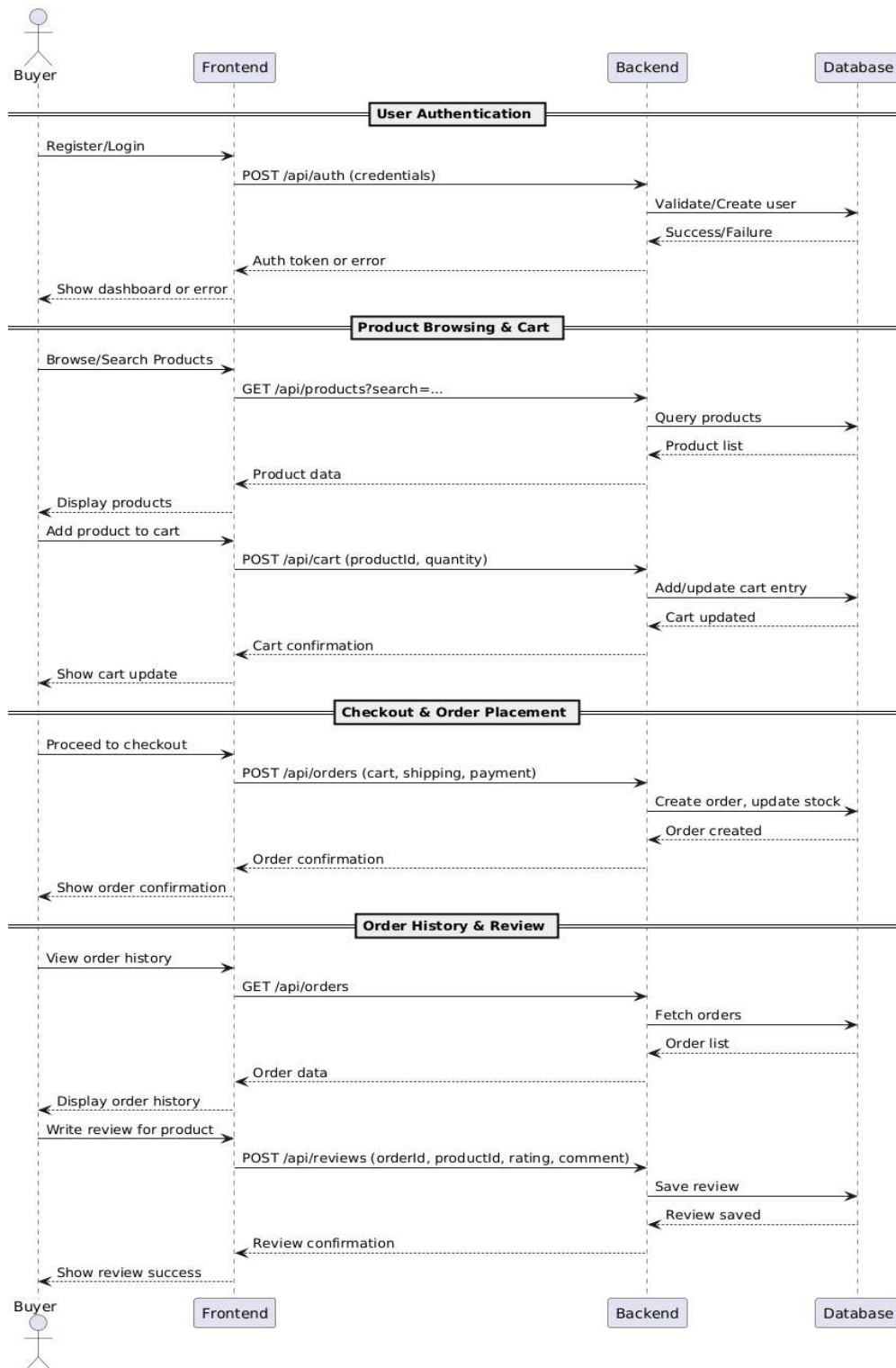
### 4.2.3 Vendor's Activity



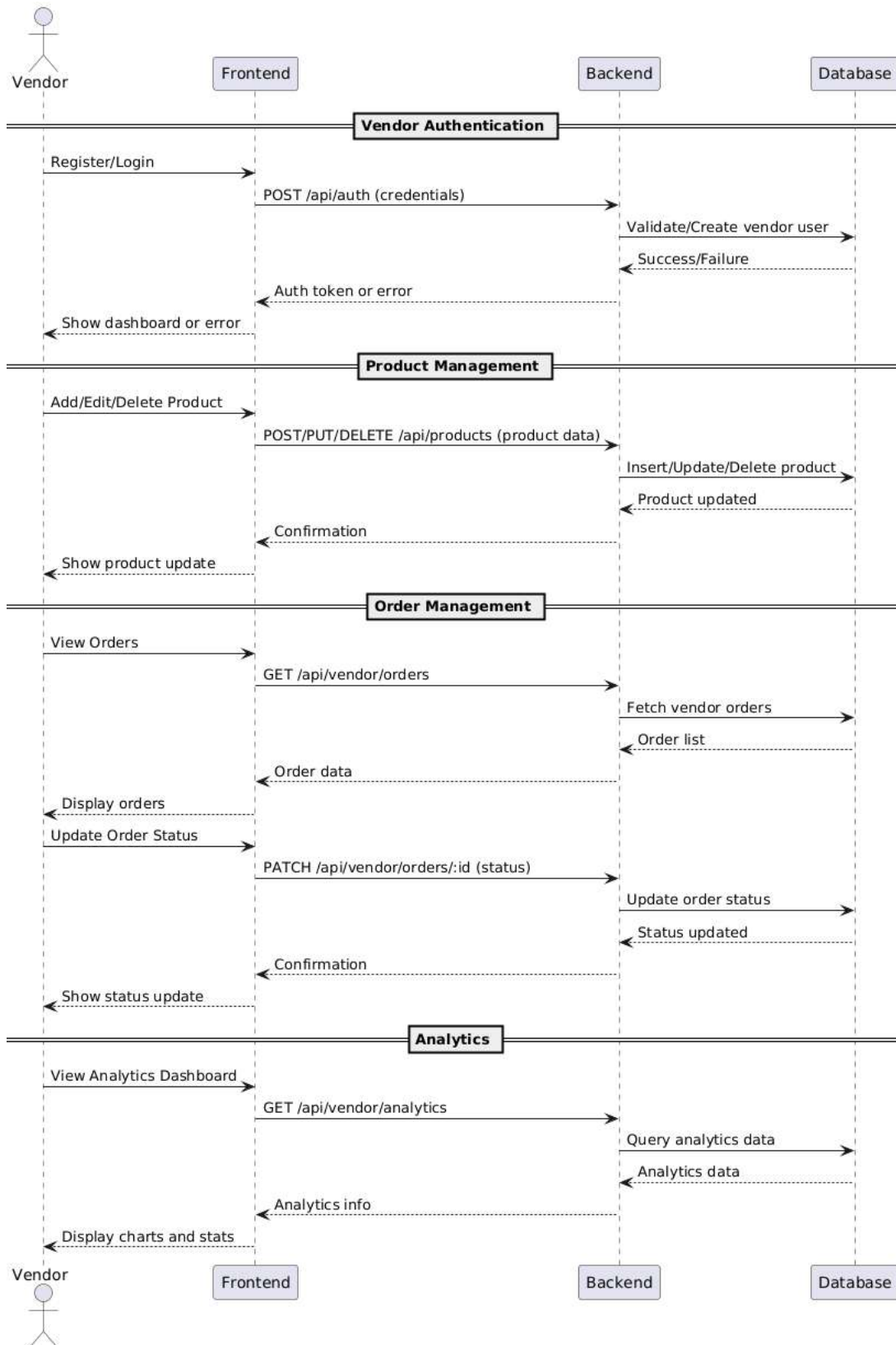
## 4.3 Sequence Diagram

Sequence diagrams explain the whole working of the system.

### 4.3.1 Buyer Sequence Diagram

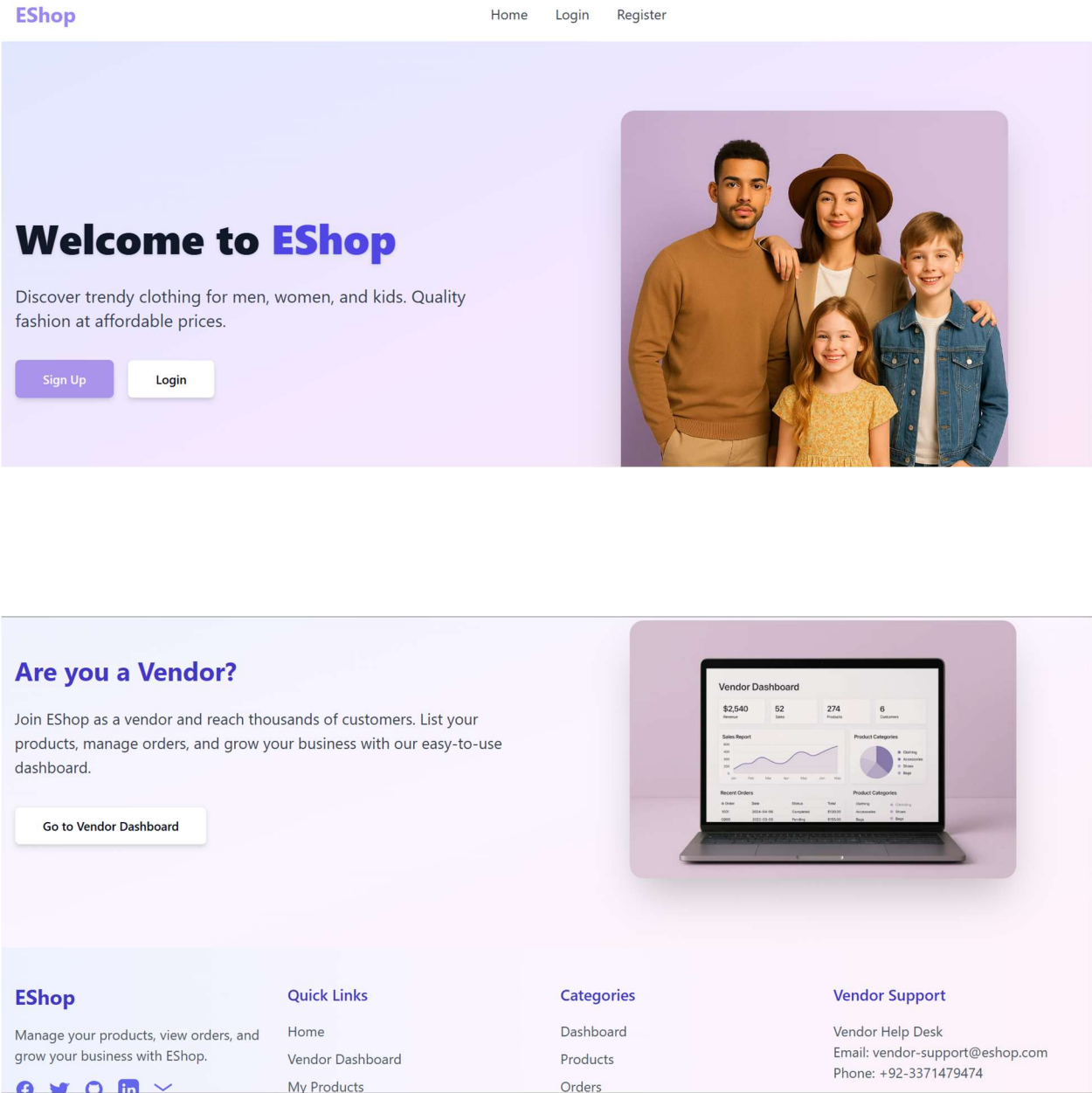


### 4.3.2 Vendor Sequence Diagram



## 4.4 Front-End Design

### Home Page



# Login Page

EShop

[Home](#) [Login](#) [Register](#)

## Login to Your Account

Email

Password

[Forgot password?](#)

Login

Don't have an account? [Register](#)

# Profile

EShop



## 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

Save Changes

# Buyer Dashboard

## Browse Products

[All Products](#) [Women's](#) [Men's](#) [Kids'](#)




Men's Casual Shirt

male

**Men's Casual Shirt**

Comfortable cotton shirt




Nike Sport Shoes

male

**Nike Sport Shoes**

A great product for footall players




baggy jeans

male

**baggy jeans**

A great product players to fix their thighs



t-shirts

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 Details](#)

Order #16  
Placed on 5/18/2025

Order Completed  
Delivered  
**\$1299.99**  
[Order Details](#)

## Cart View


EShop

[Home](#) [Shop](#) [Orders](#)

Hello, (buyer)   [Logout](#)

### Your Shopping Cart

**Cart Items (2)**




**Men's Casual Shirt**

Qty: 1  
Unit Price: \$999.99  
Status: Available

**\$999.99**

[Remove](#)



**t-shirts**

Qty: 1  
Unit Price: \$1299.99  
Status: Available

**\$1299.99**

[Remove](#)

**Order Summary**

Subtotal (2 items)  
Shipping  
Estimated tax

\$2299.98  
Free  
\$184.00

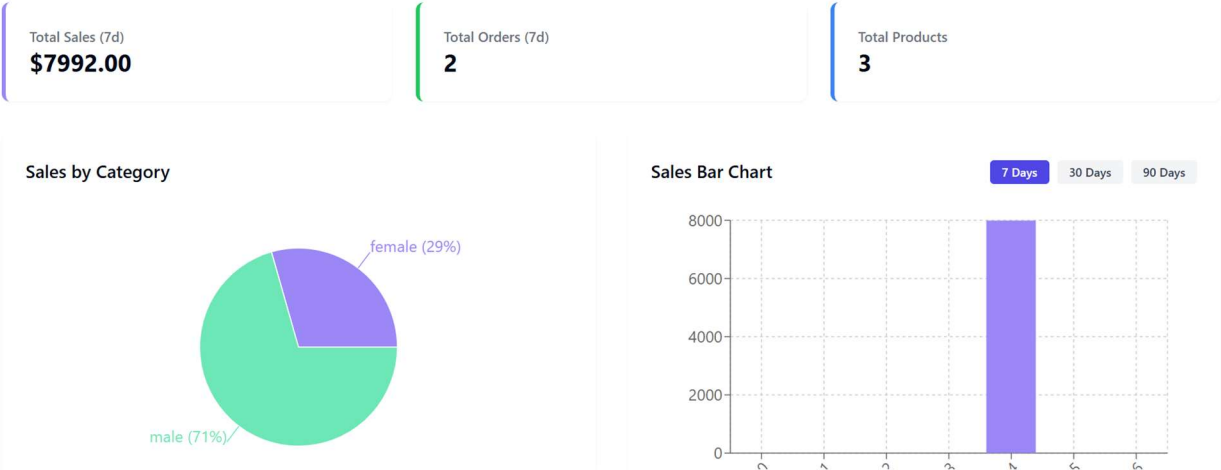
**Total**

**\$2483.98**

[Proceed to Checkout](#)  
[Continue Shopping](#)

# Vendor Dashboard










## Vendor Dashboard



# Product Management

## Product Management

+ Add Product

PRODUCT	CATEGORY	PRICE	STOCK	ACTIONS
 <b>t-shirts</b> Comfortable t shirt for for kids	children	\$1299.99	10	 
 <b>Slim Shirt</b> slim fir for men	male	\$100.00	0	 
 <b>formal dress</b> formal 3-piece dress for men	male	\$999.00	0	 

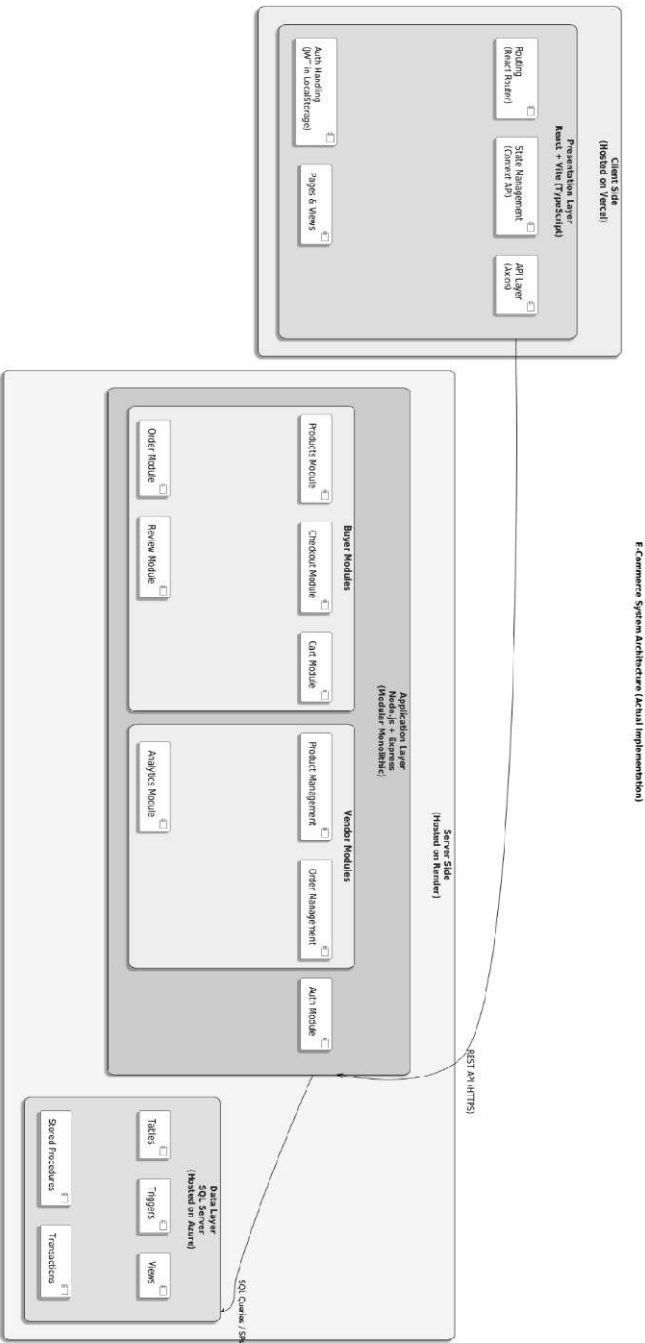
# Order Management

## Order Management

Order #46 Placed on 5/24/2025 Buyer: Mr.A	Delivered <b>\$2599.98</b> <a href="#">Order Details</a>
Order #48 Placed on 5/24/2025 Buyer: Mr.A	Delivered <b>\$2599.98</b> <a href="#">Order Details</a>



4.5 Integration



# **CHAPTER #5**

## **TESTING**

## 5 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**.

### 5.1 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.

### 5.2 Frontend 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**

# **IMPEMENTATION**

## **5.1 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.

## **5.2 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.

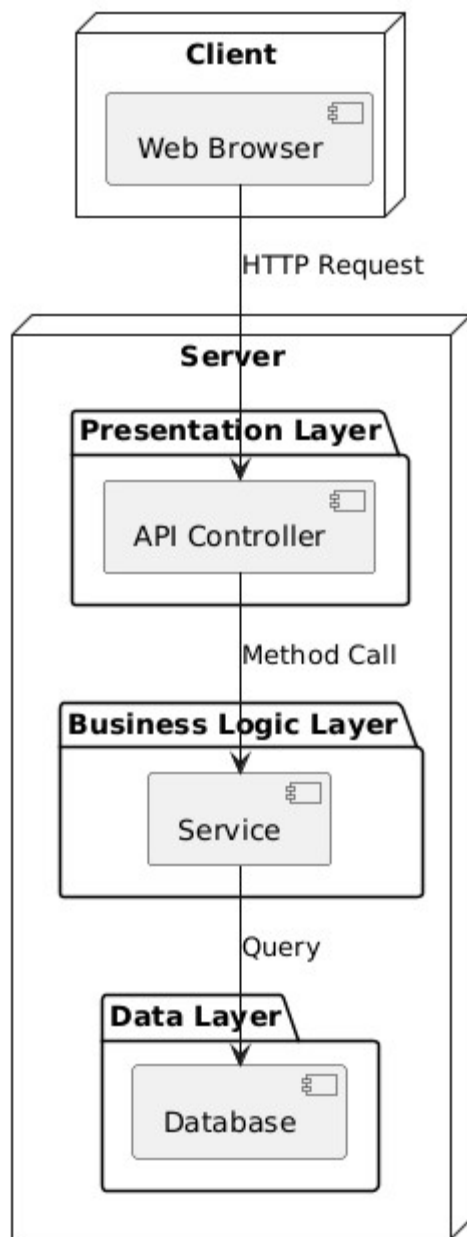
Advanced features such as triggers, views, and stored procedures were integrated to enforce business rules, automate processes, and simplify complex operations. The implementation also included data validation mechanisms, backup planning, and measures to ensure the integrity and security of transactional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

## **5.3 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.

## 5.4 Inter Module Connectivity



# **CHAPTER #7**

# **CONCLUSION**



## 6 Conclusion

### 8.1 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.

### 8.2 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.

### 8.3 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.

