Software Project Report Format

SEMESTER: SPRING 2025

COURSE CODE: CSE412

SECTION: 02

GROUP: 03

DATE: 19/05/2025

1. Tausif Ahmed	(2022-1-60-315)
2. Md. Mehedi Hasan	(2021-2-60-112)
3. Khandaker Hasan Mahmud	(2021-2-60-144)
4. Tarek Hasan	(2020-3-60-025)

Submitted to:

Yasin Sazid

Computer Science and Engineering (CSE)

Chapter 1. Introduction

The objective of the Software Design Document (SDD) is to provide a thorough and detailed design blueprint for the development of an E-Commerce Web Application. It serves as the bridge between system requirements and system implementation. It serves to establish a common vision of the system structure, components, behavior, and interactions between all of the stakeholders such as developers, designers, testers, and clients.

Project Overview (Brief summary of the system)

The E-Commerce Web Application is a dynamic online platform that allows users to browse, purchase, and manage products effortlessly. With the increasing trend of online shopping, this project aims to create a user-friendly, secure, and scalable marketplace where customers can find a wide range of products, compare prices, and make purchases with no bother. This web application will cater to both customers and sellers, ensuring smooth order processing, payment transactions, and product management. Unlike traditional shopping methods, this platform provides 24/7 accessibility, secure payment gateways, and real-time order tracking. This is an ecommerce web application developed using PHP and MySQL, with Bootstrap for the user interface. The application allows visitors to browse products, view individual product descriptions, leave feedback in the comment section, and view seller information. Sellers need to register and await admin confirmation before they can start posting their products for sale.

Objectives and Scope

Objectives:

• Create a responsive eCommerce platform

• Develop an efficient and user-friendly eCommerce platform.

• Implement secure payment gateways for online transactions.

• Integrate real-time order tracking and notifications.

• Ensure data security and privacy for users.

Scope:

• User Registration & Authentication: Secure sign-up/login functionality.

• Product Management: Admin panel for adding, editing, and removing products.

• Shopping Cart & Checkout: Allow users to add products to the cart and place orders.

• Order Management: Users can track their orders, and admins can manage shipping.

• Review & Rating System: Users can review and rate products.

Stakeholders

Every software project involves different stakeholders who have a vested interest in its success. For this E-Commerce Web Application, the key stakeholders include:

Customers: People who browse, purchase, and interact with the platform. Their satisfaction is crucial for business growth.

Sellers (Vendors): Businesses or individuals who list their products on the platform. They need a simple and effective way to manage inventory and sales.

Administrators: The platform owners and managers who ensure everything runs smoothly, including user management, product approvals, and fraud prevention.

Payment Providers: Banks, payment gateways (COD, Bikash, etc.), and financial institutions that handle transactions securely.

Delivery Partners: Logistics companies responsible for shipping and handling orders, ensuring timely delivery.

Developers & System Administrators: The technical team that maintains, updates, and secures the platform.

Technology Stack (Programming languages, frameworks, databases)

This part presents the frontend and backend technologies for the application that were carefully selected, as well as tools utilized for handling the database, user login, deployment, and design. The selected stack considers aspects such as growth, maintenance ease, the speed at which developers can develop, and user experience.

Frontend: React.js / HTML / CSS / JavaScript

Backend: Node.js (Express.js) / PHP

Database: MySQL / XAMPP

Hosting & Deployment: Github / Netlify / Vercel

Authentication: JWT / OAuth 2.0

Design Tools: Figma (for UI/UX wireframes)

Chapter 2. Software Requirements Specification & Analysis

It is the process of collecting, learning, and documenting what a software system should do. It produces a document known as an SRS that is a list of all the functional and non-functional requirements in clear terms.

2.1 Stakeholder Needs & Analysis

Identifying Primary & Secondary Stakeholders:

Primary Stakeholders: Customers, Sellers, Administrators.

Secondary Stakeholders: Payment Providers, Delivery Partners.

Methods Used for Requirement Elicitation:

Surveys & Questionnaires: To gather user preferences regarding shopping behavior.

Interviews: Conducted with business owners and potential customers to understand requirements.

Observations: Reviewing competitor websites to identify industry best practices.

Focus Groups: Engaging users for feedback on UI/UX design.

2.2 List of Requirements

- Functional Requirements (FRs):

User Registration & Authentication: Users can sign up, log in, with passwords.

Product Catalog Management: Sellers can add, update, and delete product listings.

Shopping Cart & Wishlist: Users can add items to cart or save them for later.

Order Placement & Tracking: Users can place orders and track their delivery status.

Payment Processing: Secure checkout process with multiple payment options.

Review & Ratings System: Customers can rate and review products.

- Non-Functional Requirements (NFRs):

Security: Secure login, and safe payment processing.

Performance: The website should load within 3 seconds for optimal user experience.

Scalability: Should support up to minimum 1,000 concurrent users.

Availability: The system should have 95% uptime.

Usability: The interface should be easy to navigate for all age groups.

- Extra-Ordinary Requirements (Wow Factors):

Augmented Reality (AR) Product Preview: Allows customers to visualize how a product (e.g., furniture, clothing, eyewear) will look in real life before purchasing.

Smart Order Tracking with Live Updates: Real-time shipment tracking with push notifications on order status changes.

2.3 Quality Function Deployment

Customer Requirements (CRs) List:

• Easy navigation and product search

• Secure and fast payment processing

• High-quality images and product descriptions

• Reliable customer support

Engineering Requirements (TRs) List:

• Implementation of responsive UI for all devices

• Integration of SSL security certificates

• Development of scalable database architecture

• Optimization for fast page loading

QFD Matrix (House of Quality):

• Rows: Customer Requirements (CRs)

• Columns: Technical Requirements (TRs)

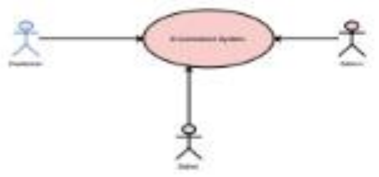
• Relationship Mapping: Strong, Medium, Weak connections between CRs and TRs

Customer Requirements (CRs)	**Technical Requirements (TRs)**	**Relationship (Strong/Medium/Weak)**
CR1: Users need a fast login process	TR1: Implement OAuth-based login	Strong
CR2: System should handle high traffic	TR2: Use a scalable cloud database	Strong
CR3: Users want a mobile-friendly UI	TR3: Implement responsive web design	Medium
CR4: Ensure data security	TR4: Encrypt user data	Strong
CR5: Easy and quick checkout process	TR5: Implement one-click checkout	Strong
CR6: Website should load fast	TR6: Optimize images and cache system	Strong

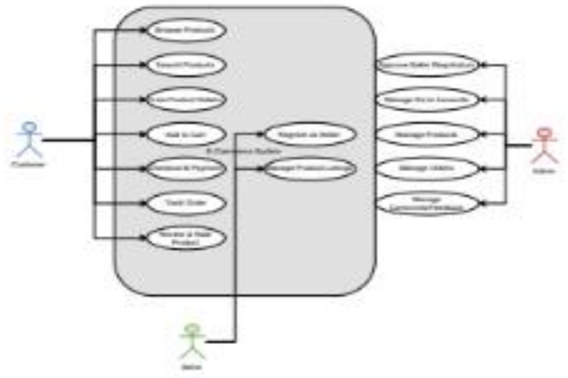
2.4 Requirements Modeling

Use Case Diagrams

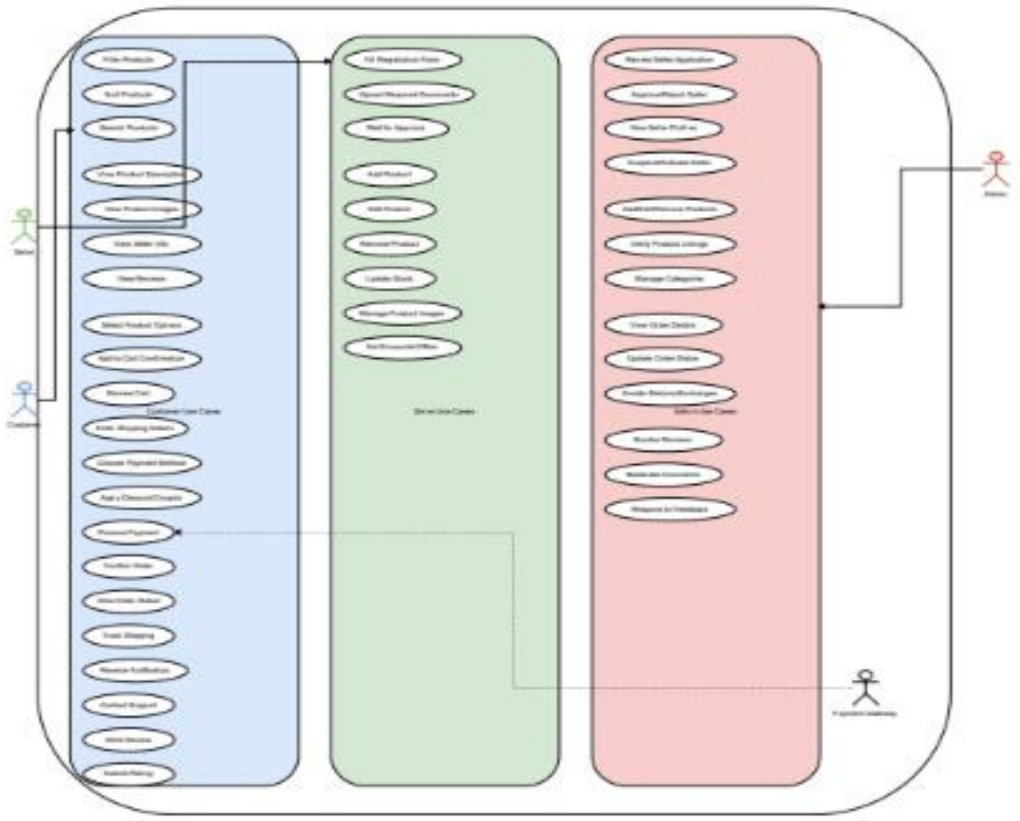
Use Case Level 0



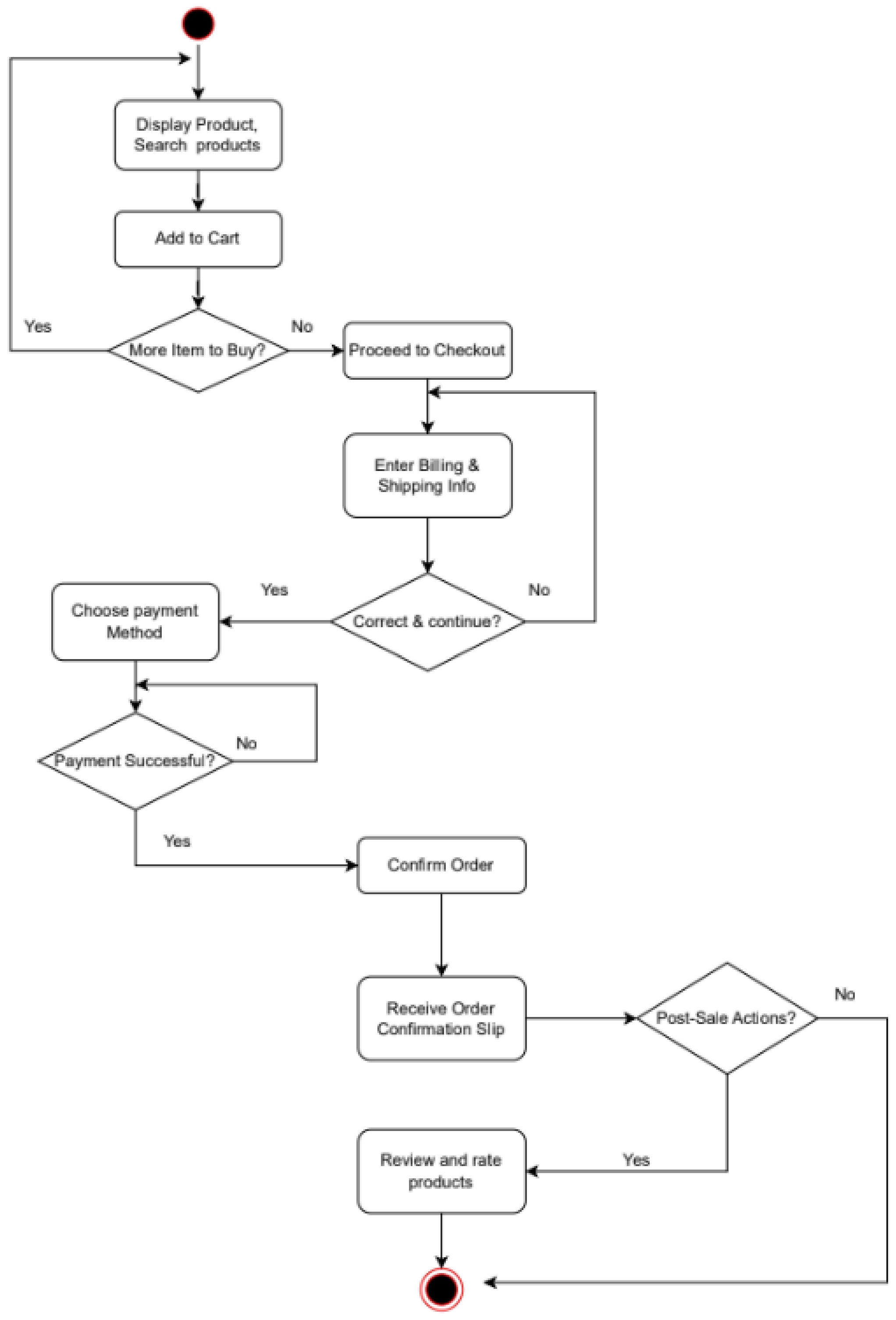
Use Case Level 1



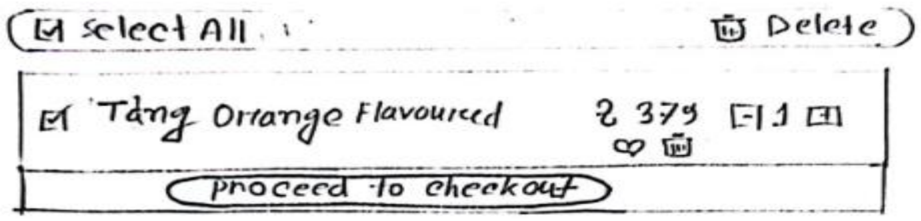
Use Case Level 2



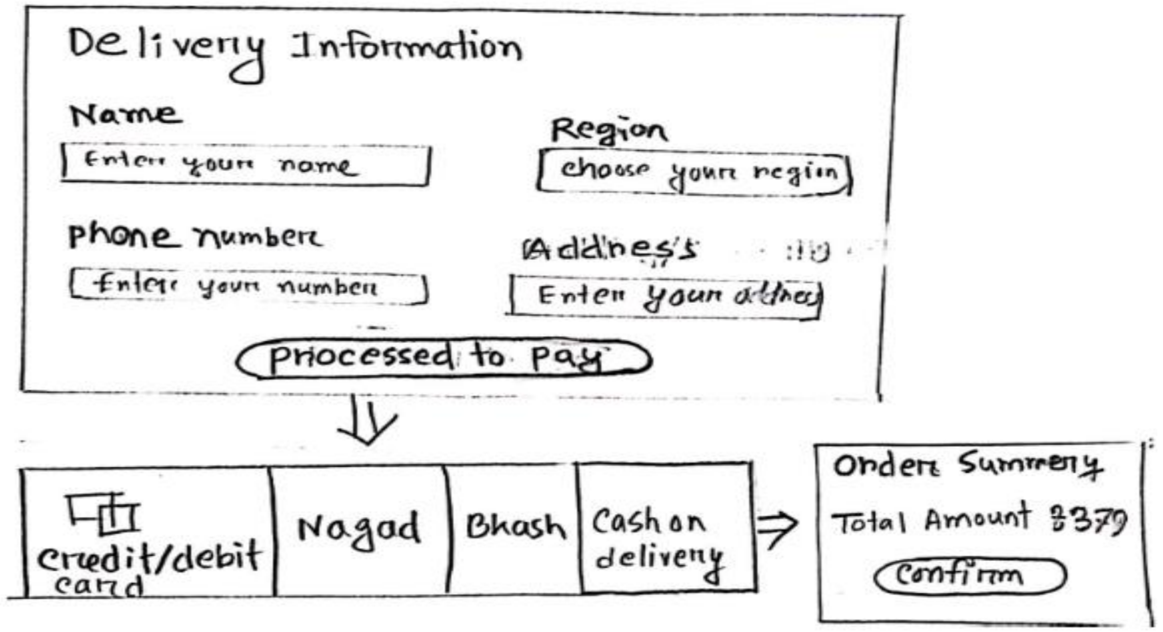
Activity Diagrams



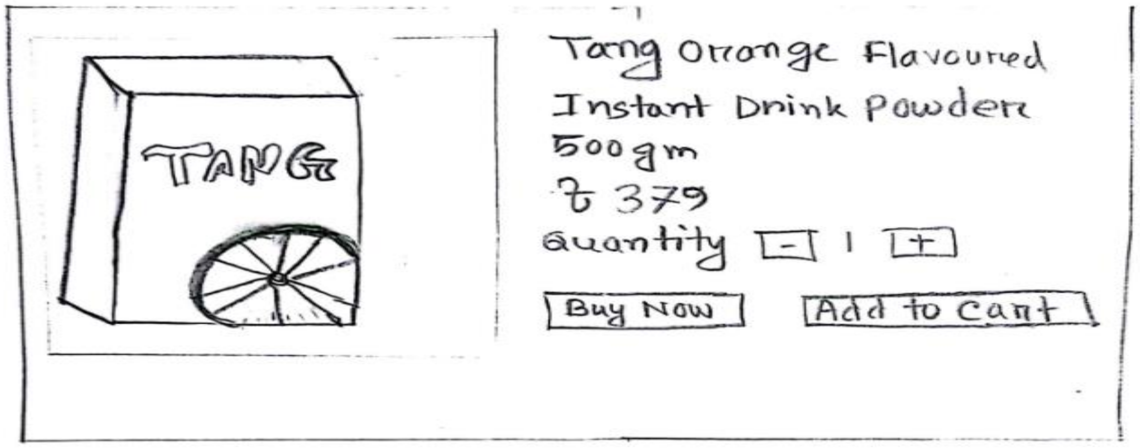
UI Sketches



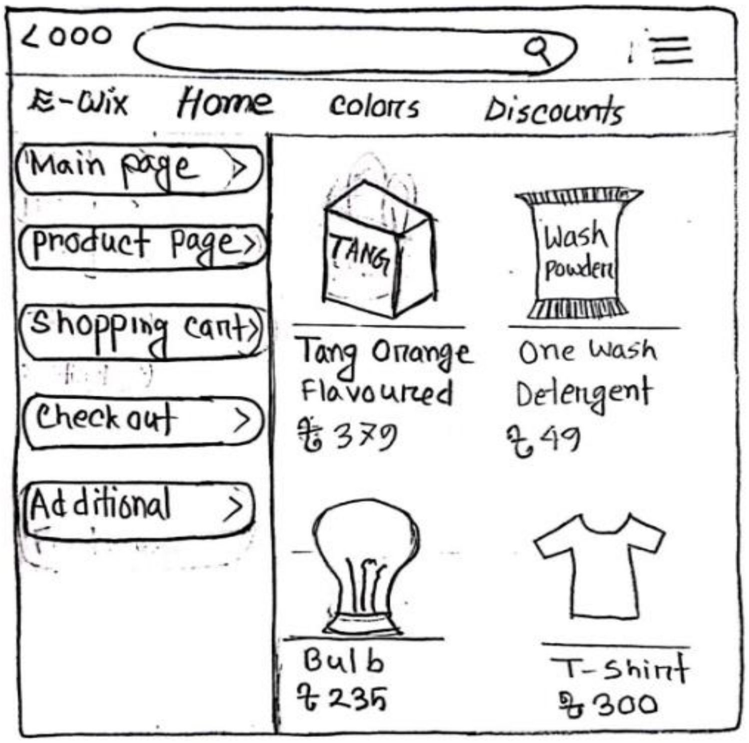
Checkout



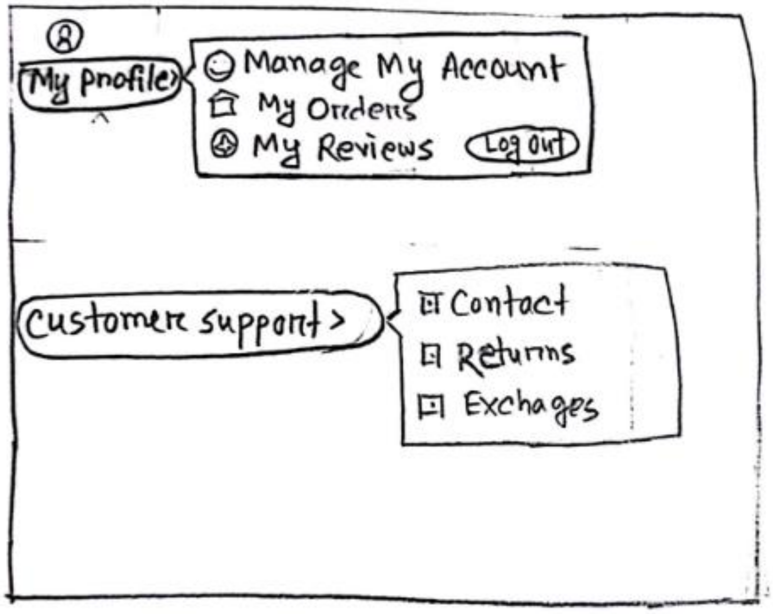
Product page



Main page



Additional Features



Chapter 3. Software Design

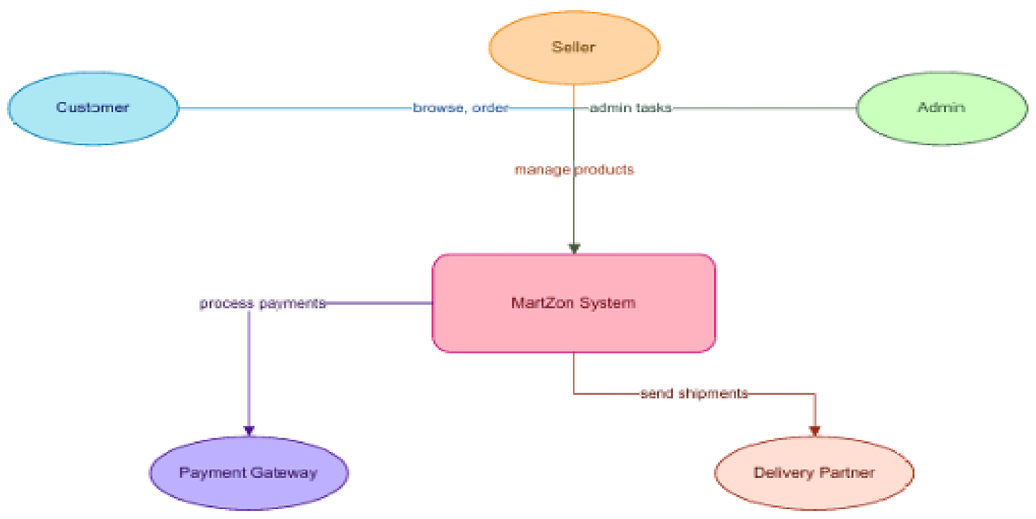
Software design is the practice of creating a plan for the creation of a software system. It describes how the system is constructed in order to fulfill the requirements, such as layout, components, movement of the data and the usage by the users.

3.1 Architectural Design

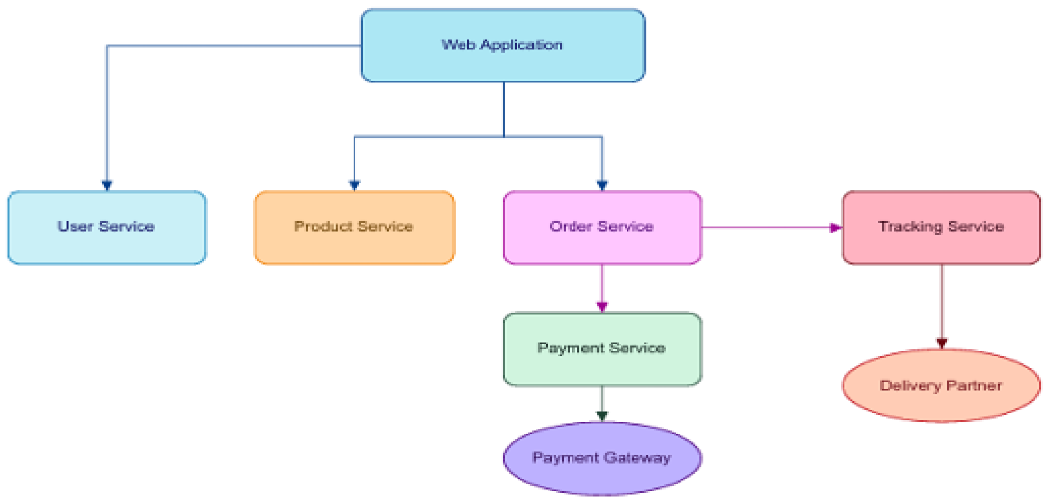
Architectural Context Diagram

Architectural Context Diagram

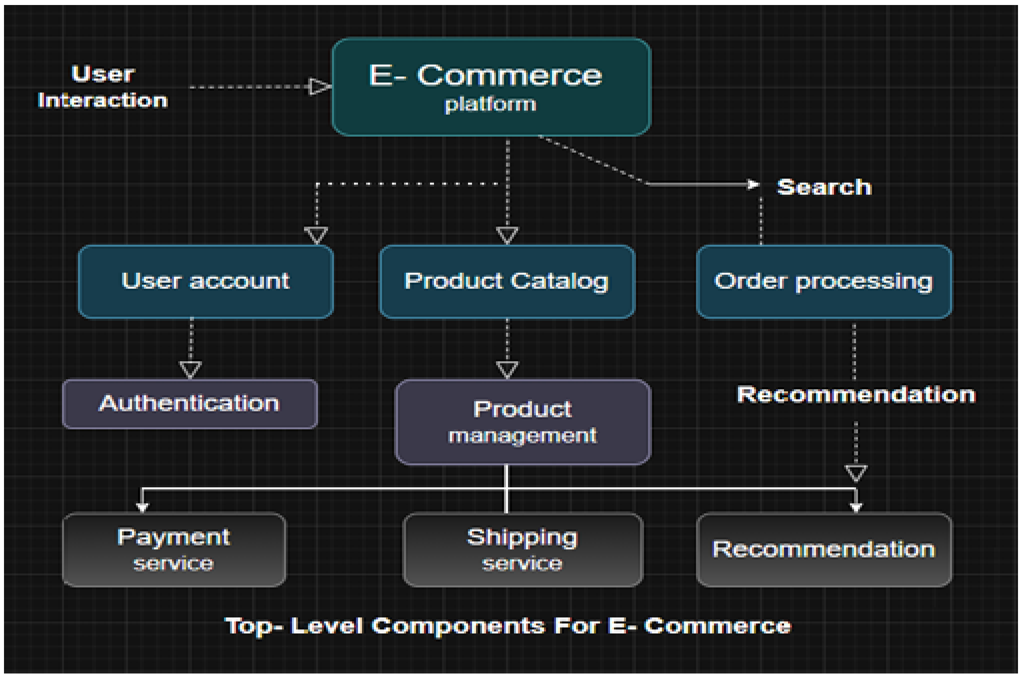
<u>**Architectural Context Diagram (Level 0)**</u>



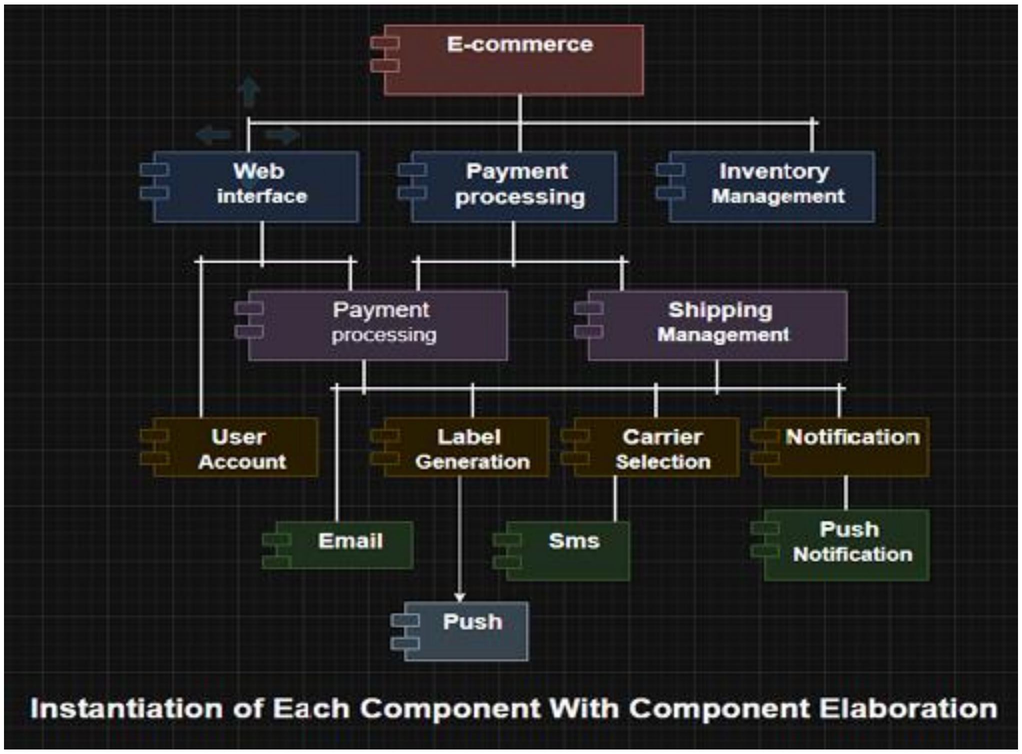
<u>**Architectural Context Diagram (Level 1)**</u>



Top-Level Component Diagram

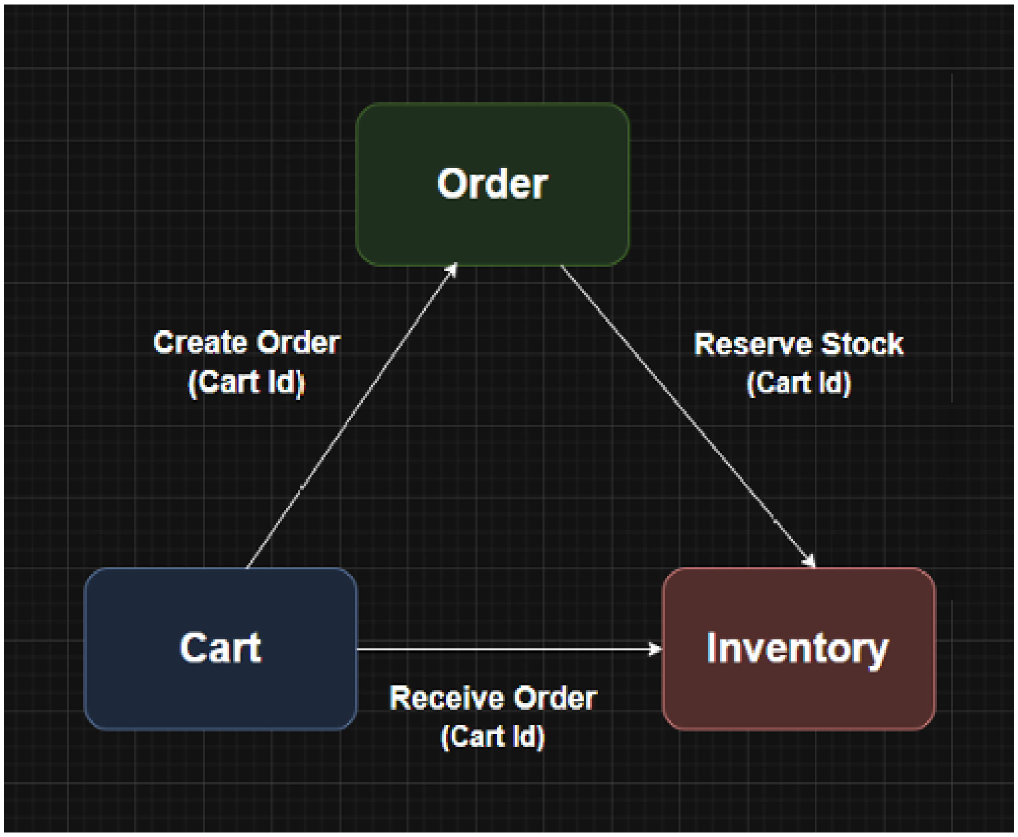


Instantiation of Each Component with Component Elaboration

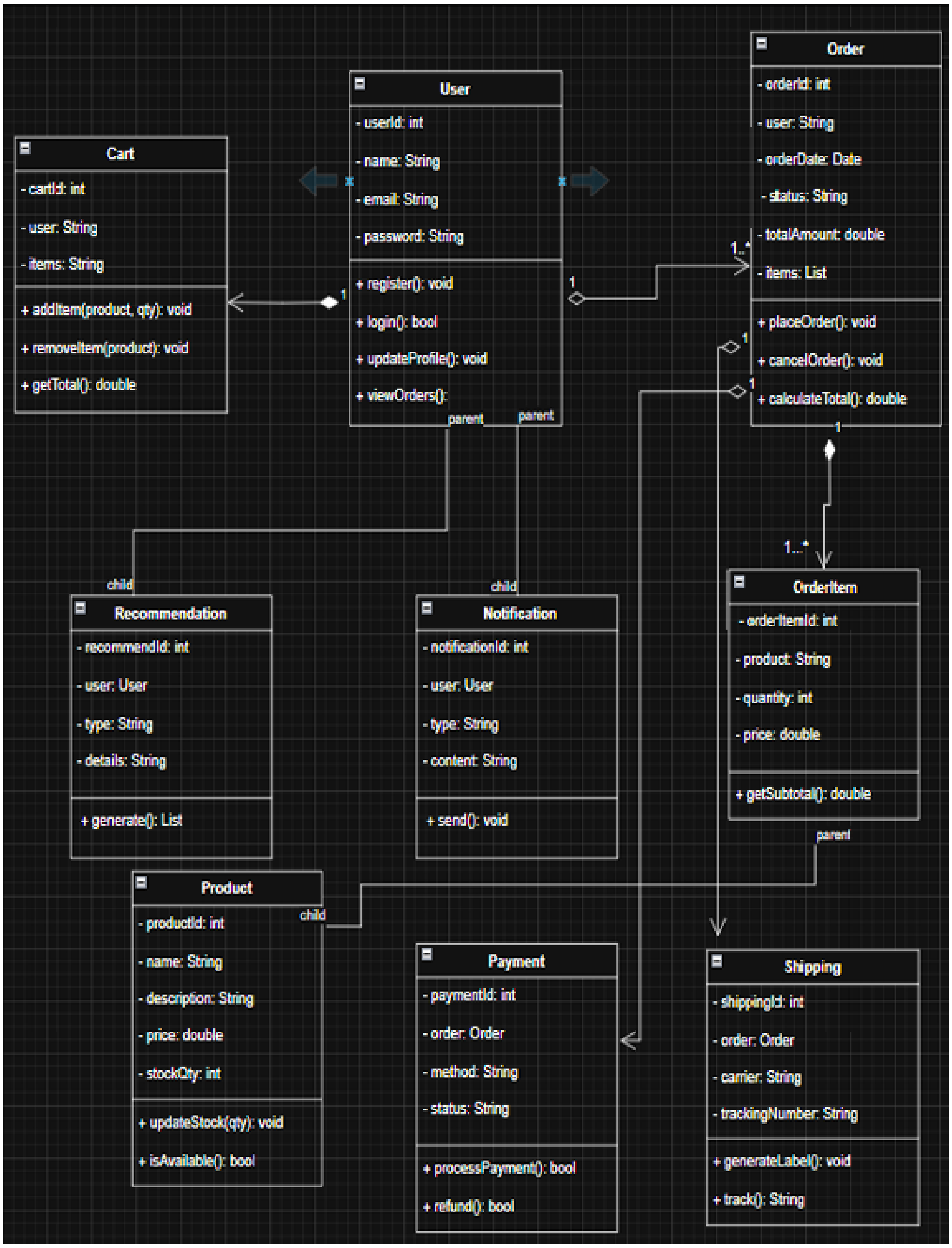


3.2 Component-Level Design

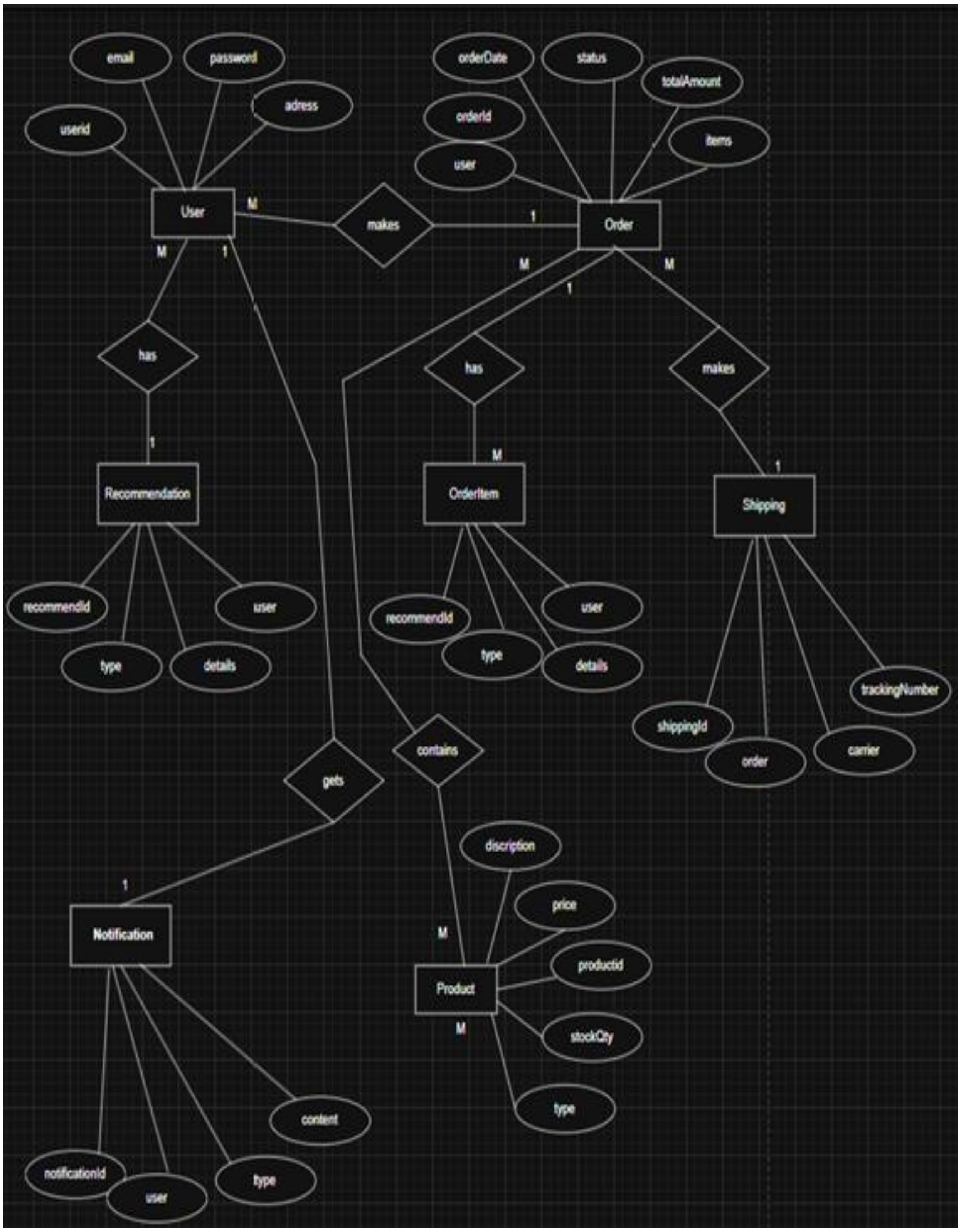
Elaboration of Design Components



Class Diagram



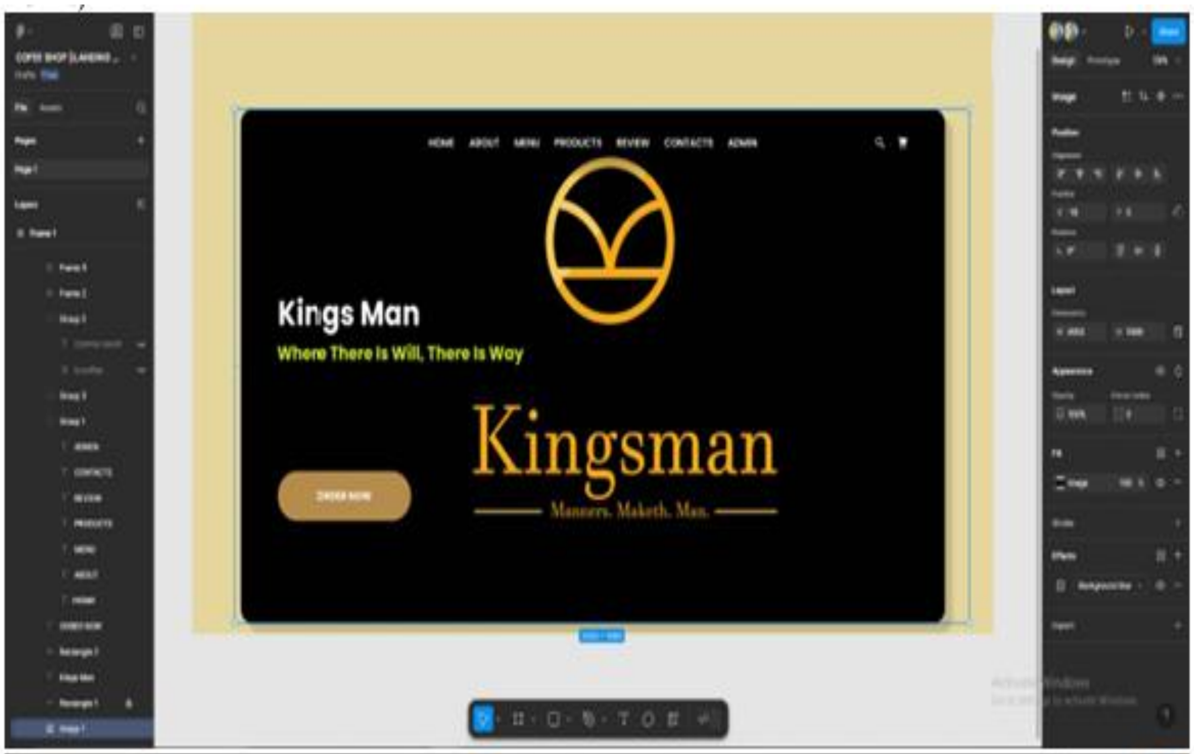
Database Design

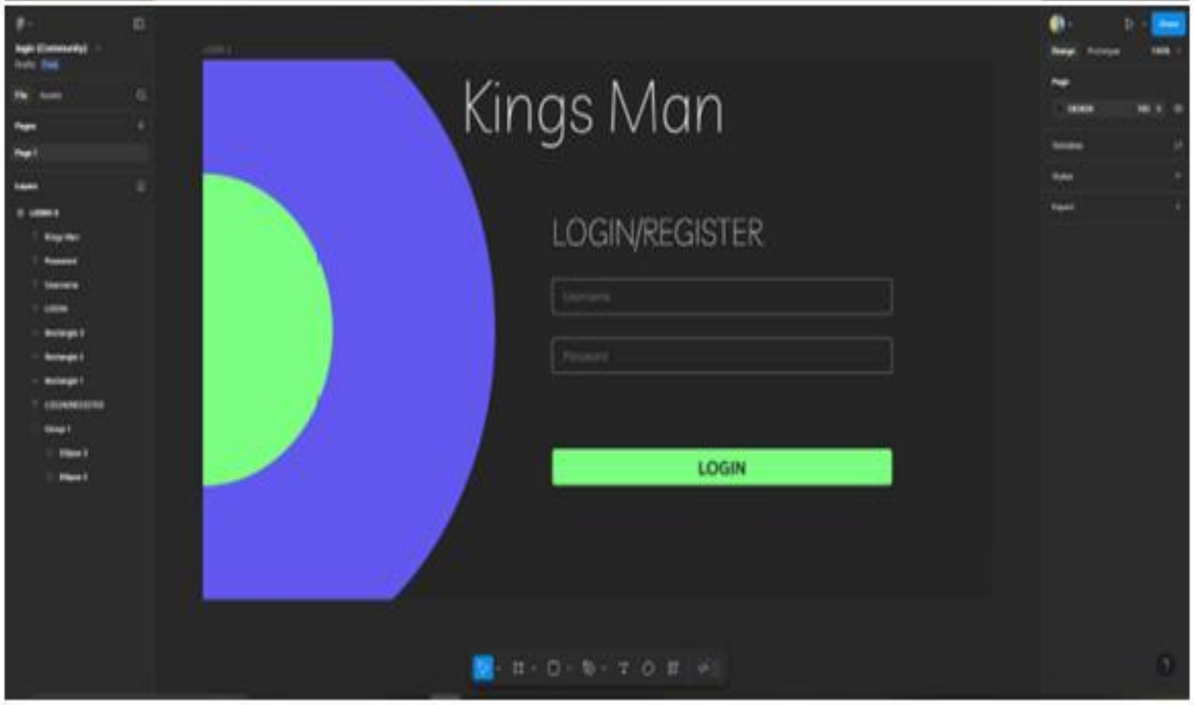


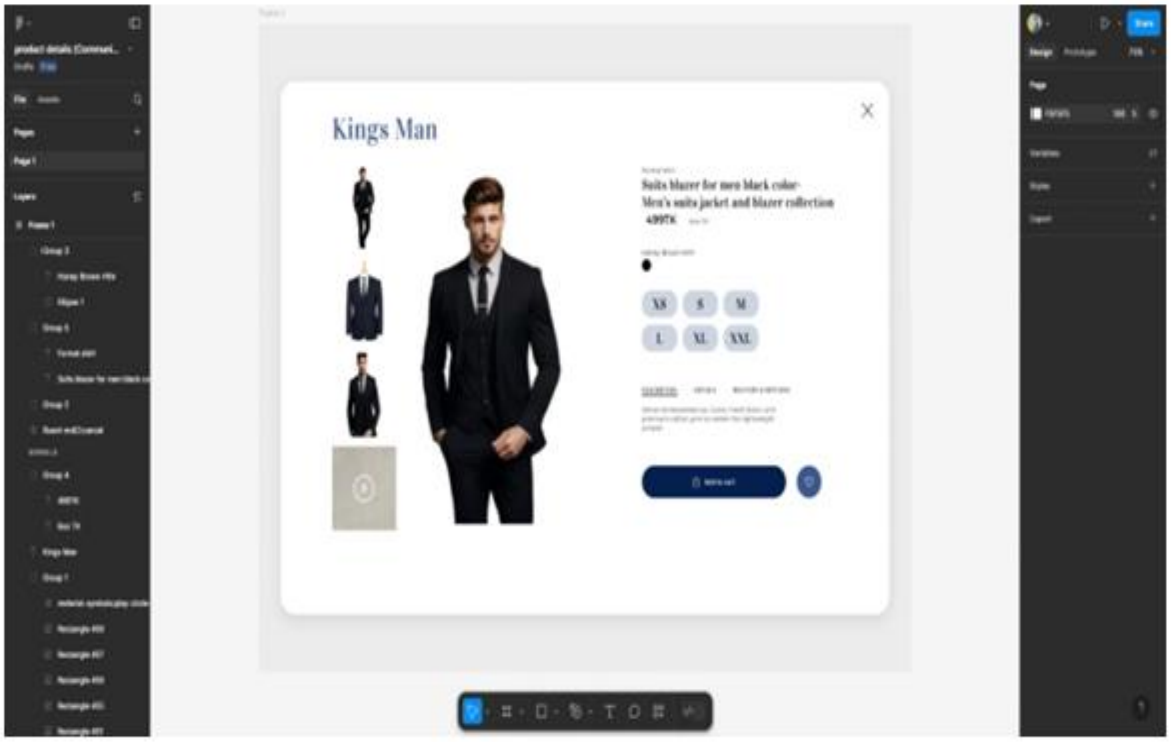
3.3 User Interface Design

UI Wireframes/Mockups/Prototypes

(Home Page, Product Detail Page, Shopping Cart, Checkout Page, Login Forms)

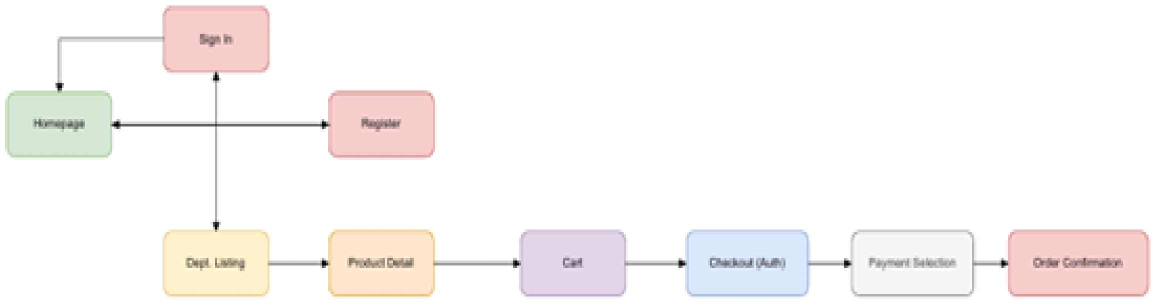

HOME ABOUT MENU PRODUCTS REVIEW CONTACTS ADMIN
Kings Man
Where There Is Will, There Is Way
ORDER NOW
Kingsman
Manners. Maketh. Man.



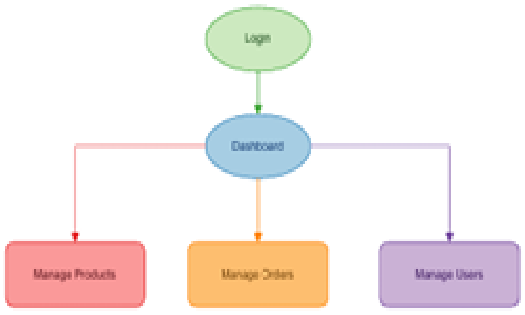

Kings Man
Suits blazer for men black color-
Men's suits jacket and blazer collection
499TK
XS
S
M
L
XL
XXL

Navigation Flow

Navigational Flow User



Navigational Flow Admin



Chapter 4. Implementation

Implementation involves taking the software design and converting it into real code using a programming language. It entails coding, compiling, and assembling the code so that the software can execute.

Code Structure Overview: The E-Commerce Web Application is structured into modular PHP scripts that collectively manage user interaction, product selection, cart handling, and order processing. Below is an overview of the major components and their roles:

1. User Module: Handles user-related functionalities such as registration, login, and session control.

Files:

- register.php: User registration with validation and password hashing.
- login.php: Authenticates users using stored credentials.
- logout.php: Ends user session securely.

2. Product Module: Manages the listing and display of products on the storefront.

Files:

- products.php: Displays available products (from database or static list).
- product_detail.php: Shows individual product info.

3. Cart Module: Allows users to add, remove, and update products in their cart.

Files:

- cart.php: Stores cart items in sessions, supports quantity control.
- Uses session management

4. Checkout & Order Module: Handles the checkout process and simulates order placement or payment.

Files:

- checkout.php – Validates cart, processes order confirmation.
- order_success.php (optional) – Displays after successful checkout.

5. Backend/Database Module: Assumed if database is used, though not shown in your uploaded files.

Files:

- db_connect.php: Connects to the database (for users/products/orders).
- Database tables: users, products, orders, etc.

6. Validation and Security: Includes input validation and secure practices.

- Password validation in register.php.
- Quantity validation in cart.php.
- Cart empty check in checkout.php.

7. UI/UX (Optional Static Files): May include styles and scripts for better user experience.

Files (if included):

- style.css – Visual styling of forms and pages.
- script.js – Client-side interactivity (optional).

8. Session and Navigation Control:

- Session is started and used across files to store login state and cart data.
- Navigation may be managed through a shared header/footer or menu links.

GitHub Repository URL

https://github.com/tausifahmed1234/CSE412

Chapter 5. Software Testing

Software Testing is the process of checking whether the software works correctly and meets the user's requirements. It helps find bugs (errors) in the program before it is released. E-Commerce Web Application is a site that allows individuals to search and purchase products from various categories. The system is designed to facilitate shopping with essential functionalities such as signing up and logging in, searching for products, filter options, adding to cart, ordering products and secure payment.

5.1 White Box Testing

White box testing is a software testing technique in which the internal structure, design, and code of the software are tested. The tester has full knowledge of the code and writes test cases to verify logical paths, loops, conditions, and data flow within the program.

Unit Testing on Two Classes: Perform unit testing on at least two classes using control flow graphs.

1. Selected Classes: Shopping Cart and Order

CFG 1: cart.php: Add to Cart Logic

• Start

• Check if the product exists in the database.

• If quantity is greater than 0:

• Add item to the cart.

• Else:

• Show an error message.

• End

CFG 2: register.php: User Registration Logic

• Start

• Validate the email format.

• If valid, check if the password is strong.

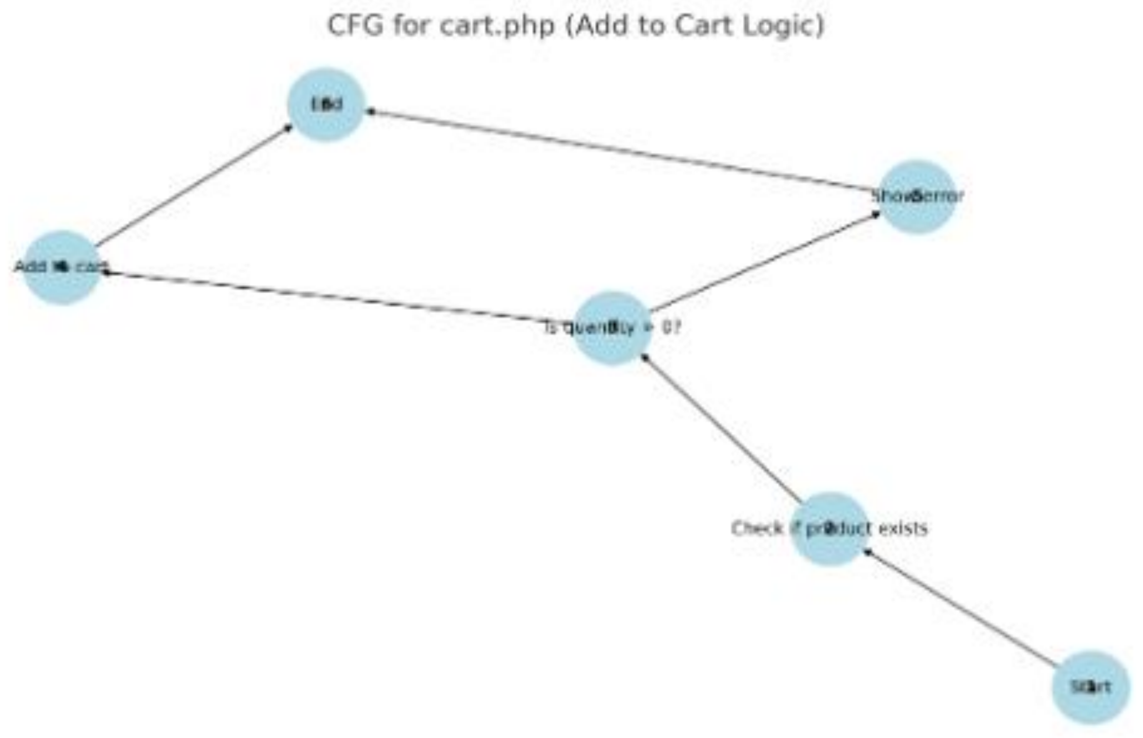
• If strong, save user and finish.

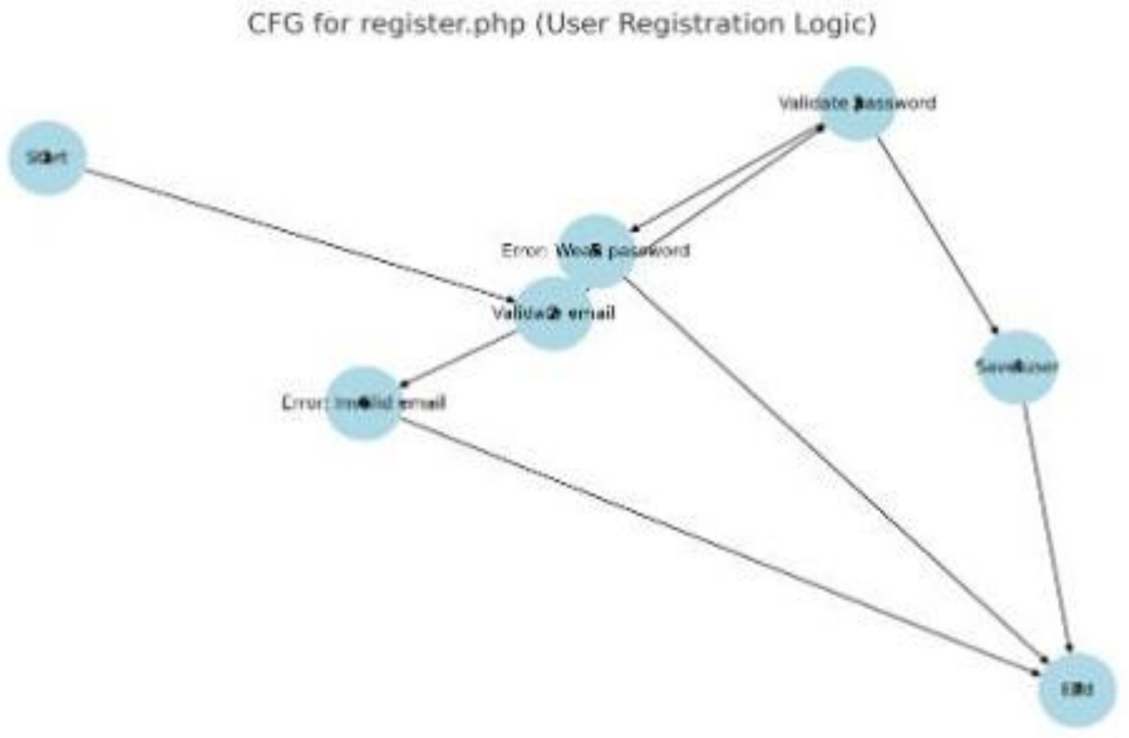
• Else, show password error.

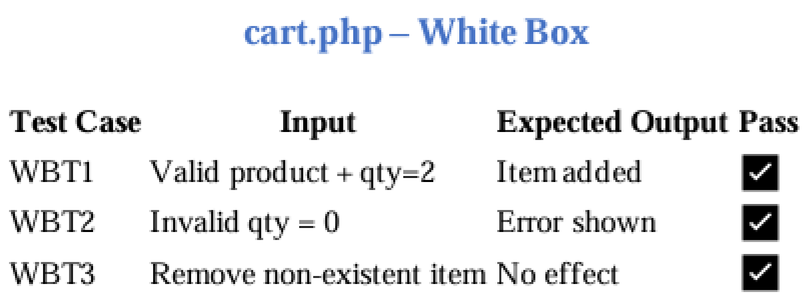
• If email is invalid, show email error.

• End

2. Control flow graphs:

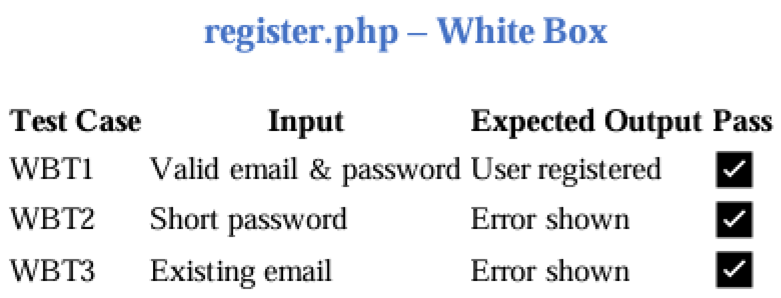






cart.php – White Box

Test Case	Input	Expected Output	Pass
WBT1	Valid product + qty=2	Item added	✓
WBT2	Invalid qty = 0	Error shown	✓
WBT3	Remove non-existent item	No effect	✓



register.php – White Box

Test Case	Input	Expected Output	Pass
WBT1	Valid email & password	User registered	✓
WBT2	Short password	Error shown	✓
WBT3	Existing email	Error shown	✓

5.2 Black Box Testing

Black box testing is a software testing method that examines the functionality of an application without knowing its internal code structure. The tester focuses on input and output and checks whether the system behaves as expected.

Functional Testing on Two Core Features

Use boundary value analysis for functional testing of at least two core features

a. Selected Features

• User Registration

• Order Placement

b. Test Cases (Boundary Value Examples)

• User Registration:

- Test with min/max lengths of username and password.
- Empty fields or special characters.

• Order Placement:

- Test with 0 items, 1 item, and maximum allowed items.
- Payment value edge cases (e.g., 0, 1, 99999).

Feature A: User Registration – Black Box

TC ID	Test Scenario	Input	Expected Output	Actual Output	Pass/Fail
BBT_REG_01	Minimum username length (3 chars)	abc	Accepted	Accepted	✓
BBT_REG_02	Maximum username length (20 chars)	usernameusername1234	Accepted	Accepted	✓
BBT_REG_03	Empty username field	``	Error: Required field	Error	✓
BBT_REG_04	Password at minimum length (6 chars)	123456	Accepted	Accepted	✓
BBT_REG_05	Password below min (5 chars)	12345	Error: Too short	Error	✓
BBT_REG_06	Special characters in username	user@name	Error: Invalid characters	Error	✓

Feature A: Order Placement– Black Box

TC ID	Test Scenario	Input	Expected Output	Actual Output	Pass/Fail
BBT_ORDER_01	Place order with 0 items	0 items	Error: Empty cart	Error	✓
BBT_ORDER_02	Place order with 1 item (min)	1 item	Order success	Success	✓
BBT_ORDER_03	Place order with max allowed items (e.g., 100)	100 items	Order success	Success	✓
BBT_ORDER_04	Payment amount = 0	₹0	Error: Invalid amount	Error	✓
BBT_ORDER_05	Payment amount = 1	₹1	Order success	Success	✓
BBT_ORDER_06	Payment amount = 99999	₹99999	Order success	Success	✓

5.3 Bug Detection and Solution

Bug detection and solution in software development involve finding and correcting errors in code to ensure a software product functions as expected and is free from errors.

Identification, Solution and Retesting of Bugs:

Bug 1: cart.php (Allows Quantity = 0)

Incorrect Code:

```
Bug 1 – cart.php (Incorrect Code)
if (isset($_POST['add_to_cart'])) {
    $quantity = $_POST['quantity'];
    // No validation for quantity
    $_SESSION['cart'][] = $product;
}
```

Fixed Code:

```
Bug 1 – cart.php (Fixed Code)
if (isset($_POST['add_to_cart'])) {
    $quantity = $_POST['quantity'];
    if ($quantity <= 0) {
        echo 'Invalid quantity';
        exit;
    }
    $_SESSION['cart'][] = $product;
}
```

Bug 2: checkout.php – Checkout Without Cart Items

Incorrect Code:

```
Bug 2 – checkout.php (Incorrect Code)
if (isset($_POST['checkout'])) {
    // No check for empty cart
    processPayment();
}
```

Fixed Code:

```
Bug 2 – checkout.php (Fixed Code)
if (isset($_POST['checkout'])) {
    if (empty($_SESSION['cart'])) {
        echo 'Cart is empty';
        exit;
    }
    processPayment();
}
```

Bug 3: register.php – Accepts Weak Passwords

Incorrect Code:

```
Bug 3 – register.php (Incorrect Code)
$password = $_POST['password'];
// No check for password length
$hashed = password_hash($password, PASSWORD_DEFAULT);
```

Fixed Code:

```
Bug 3 – register.php (Fixed Code)
$password = $_POST['password'];
if (strlen($password) < 6) {
    echo 'Password must be at least 6 characters.';
    exit;
}
$hashed = password_hash($password, PASSWORD_DEFAULT);
```

After applying each fix, all failed test cases were retested and passed successfully.

Chapter 6. Deployment

Deployment Platform

The application is hosted on a local development server using XAMPP (Apache 2.4, PHP 8.x, MySQL 8.x). XAMPP was selected because it provides a complete, easy-to-install environment for running PHP and MySQL on Windows, macOS, or Linux without additional configuration. This setup closely mirrors a production LAMP stack, ensuring compatibility and streamlined troubleshooting during development and testing.

Deployment Process

1. XAMPP Installation

- Download and install the latest XAMPP package from the Apache Friends website.
- Verify that Apache and MySQL services start without errors via the XAMPP Control Panel.

2. Project Directory Configuration

- Copy the project folder (ecommerce-app) into XAMPP's htdocs directory (e.g., C:\xampp\htdocs\ecommerce-app).
- Ensure file and directory permissions are set to allow Apache to read and execute PHP scripts.

3. Database Setup

- Launch phpMyAdmin (http://localhost/phpmyadmin) and create a new database named ecommerce_db.
- Import the provided SQL dump (ecommerce_db.sql) via phpMyAdmin's Import tab to populate necessary tables (products, users, orders, etc.).
- Update database connection credentials in config.php to match the XAMPP MySQL default user (root) and password (blank by default).

4. Server Configuration

- Open httpd.conf (found in xampp/apache/conf/) and confirm that the mod_rewrite module is enabled for clean URL routing.
- Restart Apache to apply any configuration changes.
- Testing & Verification
- In a web browser, navigate to the application URL (see 4.3 below).
- Test critical workflows (user registration, product browsing, cart operations, checkout) to ensure end-to-end functionality.
- Monitor the XAMPP Control Panel logs for any PHP or SQL errors and resolve as necessary.

Website URL

Development URL:

http://localhost/ecommerce-app/index.php

Chapter 7: Conclusion

Learnings

This project presented an excellent opportunity to apply software engineering concepts and principles to deliver an operational E-Commerce site. In all phases of the Software Development Life Cycle (SDLC), we experienced firsthand:

- ✓ **Requirements Specification:** We considered what individuals desire from a user-friendly online shopping site. These range from signing up to selecting products, cart management, and finally ordering.

- ✓ **Design Principles:** The system is separated into distinct parts (cart, user, checkout), using simple design and keeping various functions separated for maintainability

- ✓ **Implementation:** We developed the core backend system for user interaction, storing products in a cart and easy input checking using PHP and session handling.

- ✓ **Testing Techniques:** Conduct White Box Testing based on control flow graphs and test cases. Utilized Black Box Testing with Boundary Value Analysis to validate user input and order limits. Found and resolved issues in the registration, cart and checkout sections by thoroughly testing them.

Limitations

Despite achieving core functionality, the system has some limitations:

- No real-time payment gateway integration.
- Lacks advanced user roles (e.g., admin panel).
- No product database or dynamic inventory.
- UI/UX is basic and not responsive for mobile users.
- No Scalability, filtering, or product search functionality.

Future Plan

To enhance the system and align it more closely with modern e-commerce platforms, we plan the following upgrades:

Technical Enhancements:

- Database Integration (MySQL): Store users, products, and orders with persistent data.
- User Authentication: Add secure login, logout, password reset, and session expiry.
- Admin Dashboard: Allow admins to manage products, view orders, and track customers.
- Payment Gateway: Integrate real systems like PayPal or Stripe for real transaction support.

UI/UX Improvements:

- Redesign using Bootstrap or Tailwind CSS for a clean and mobile-responsive layout.
- Add product filters, categories, and search functionality for better navigation.

Security Upgrades:

- Strengthen password policies and add captcha during registration.