

ONLINE STATIONARY SHOPPING SYSTEM

CS23333 – Object Oriented Programming using Java

Submitted by

AARTHI A P - 231001003

ABINAYA M - 231001005

Of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY



DEPARTMENT OF INFORMATION TECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE

(An Autonomous Institution)

NOVEMBER 2024

RAJALAKSHMI ENGINEERING COLLEGE
BONAFIDE CERTIFICATE

Certified that this project titled “**ONLINE STATIONARY SHOPPING SYSTEM**” is the bonafide work of “**AARTHI AP (231001003), ABINAYA M (231001005)**” who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr.P.Valarmathie

HEAD OF THE DEPARTMENT

Information Technology

Rajalakshmi Engineering College

SIGNATURE

Mrs.Usha S

COURSE INCHARGE

Information Technology

Rajalakshmi Engineering College

This mini project is submitted for CS23333 – Object Oriented Programming using Java held on _____

INTERNAL EXAMINAR

EXTERNAL EXAMINAR

ABSTRACT

The Stationery Haven website serves as an efficient and user-friendly e-commerce platform catering to stationery needs. With a focus on simplicity and functionality, it provides essential shopping features such as easy product ordering, real-time availability status, and transparent pricing to ensure informed purchasing decisions. The platform includes detailed product descriptions, enabling customers to understand item specifications thoroughly. A robust cart system simplifies the purchasing process, allowing users to manage their orders with ease. The website supports secure payment options, including UPI and net banking, ensuring smooth and trustworthy transactions. Upon completion of a purchase, an instant receipt is generated, enhancing customer confidence and convenience. Designed with a seamless interface and efficient backend processes, Stationery Haven prioritizes user satisfaction by offering a reliable, hassle-free shopping experience tailored to modern needs.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	i
	LIST OF FIGURES	iii
	LIST OF TABLES	iv
1	INTRODUCTION	1
	1.1 PROBLEM STATEMENT	1
	1.2 OBJECTIVE OF THE PROJECT	2
	1.3 ORGANIZATION OF THE REPORT	2
2	SYSTEM DESIGN	3
	2.1 OVERVIEW	3
	2.2 LIST OF MODULES	3
	2.3 UML MODELING	4
	2.3.1 USE CASE DIAGRAM OVERVIEW	4
	2.3.2 ENTITY-RELATIONSHIP DIAGRAM	6
	2.3.3 DATA FLOW DIAGRAM	7
	2.4 SYSTEM SPECIFICATION	9
	2.4.1 FUNCTIONAL REQUIREMENTS	10
3	DESIGN	11
	3.1 DESIGN	11
	3.2 DATABASE DESIGN	16
	3.3 IMPLEMENTATION (CODE)	18
4	CONCLUSION	23
	REFERENCES	24

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NUMBER
3.1	USE CASE DIAGRAM OF ONLINE STATIONARY SHOPPING SYSTEM	5
3.2	ENTITY RELATIONSHIP DIAGRAM OF ONLINE SHOPPING SYSTEM	6
3.3	DATA FLOW DIAGRAM OF ONLINE STATIONARIES SHOPPING SYSTEM	8
3.4	LOGIN/SIGN UP PAGE	11
3.5	SIGN UP PAGE	11
3.6	LOGIN PAGE	12
3.7	MAIN PAGE	12
3.8	ACKNOWLEDGEMENT PAGE	13
3.9	RECEIPT PAGE	13
3.1	ADDRESS PAGE	14
3.11	PAYMENT GATEWAY PAGE	14
3.12	PAYMENT SUCCESSFUL PAGE	15
3.13	PAY ON DELIVERY PAGE	15

LIST OF TABLES

TABLE NO	TABLE NAME	PAGE NUMBER
3.1	PRODUCTS TABLE	16
3.2	USERS TABLE	17

CHAPTER 1

INTRODUCTION

1.1.PROBLEM STATEMENT

Purchasing stationery items can sometimes be a long and difficult process. Customers may face issues like visiting stores only to find items unavailable or using online platforms that do not provide clear information about products, prices, or stock levels. These challenges make it hard for users to find and buy the products they need efficiently.

There is a need for an online platform that simplifies the stationery shopping process. This platform should:

1. Provide **real-time updates** on product availability so customers can plan their purchases.
2. Show **clear pricing information** to help users make cost-effective decisions.
3. Offer **detailed product descriptions** so customers can understand the items before buying.
4. Include an **easy-to-use cart system** to manage selections before checkout.
5. Support **secure payment options**, such as UPI and net banking, for a safe shopping experience.
6. Generate **instant receipts** and offer reliable order tracking to build trust with users.

The **Stationery Haven** project aims to address these challenges by creating a simple, efficient, and secure e-commerce platform. This website will focus on improving the shopping experience for customers by offering modern and user-friendly features.

1.2.OBJECTIVE OF THIS PROJECT

The objective of the Stationery Haven project is to develop an intuitive, secure, and efficient e-commerce platform that simplifies the online shopping experience for customers purchasing stationery items. The platform aims to provide real-time product availability updates, ensuring that customers can easily find the items they need without any inconvenience. It will offer clear and transparent pricing along with detailed product descriptions to assist customers in making informed purchasing decisions. The project will also include a seamless shopping experience through an easy-to-use cart system, allowing customers to manage their selections effortlessly. In addition, the platform will integrate secure payment options, such as UPI and net banking, to cater to a wide range of customers. To enhance trust and customer satisfaction, the system will generate instant order receipts and provide reliable order tracking. Ultimately, the goal is to create a user-friendly platform that is scalable and adaptable, capable of handling increased traffic and expanding product listings, thus meeting the needs of customers while supporting the long-term growth of the business.

1.3.ORGANIZATION OF THE REPORT

CHAPTER 1 – INTRODUCTION

CHAPTER 2 – SYSTEM DESIGN

CHAPTER 3 – IMPLEMENTATION

CHAPTER 4 – CONCLUSION

CHAPTER 2

SYSTEM DESIGN

2.1. OVERVIEW

The Online Shopping System is designed using a client-server architecture, seamlessly integrating the frontend, backend, and database components to deliver an efficient and user-friendly experience. The frontend serves as the interactive interface, built using technologies like HTML, CSS, and JavaScript, allowing users to browse products, search for items, manage their shopping cart, and complete purchases. It sends user requests, such as login or adding items to the cart, to the backend for processing.

The backend, developed in Java using JDBC, acts as the intermediary between the frontend and the database. It handles the core business logic, validates inputs, processes user actions, and ensures secure data exchange. Features like user registration, login authentication, cart management, product catalogue, updates, and order placement are implemented in the backend. The backend efficiently interacts with the database, powered by Oracle SQL, which securely stores and manages critical information such as user details, product data, cart contents, and order histories. The database ensures data integrity and supports robust querying for operations like product searches and inventory updates.

This integration of the frontend, backend, and database ensures a cohesive system where user actions are processed seamlessly, data is managed securely, and the shopping experience remains smooth and scalable.

2.2. LIST OF MODULES

1. User Management Module
2. Product Management Module
3. Transaction Management Module
4. Cross-Origin Resource Sharing(CORS) Handling Module
5. Error Handling and Response Module
6. Database Connectivity Module

2.3. UML MODELING

2.3.1. USE CASE DIAGRAM OVERVIEW:

The use case diagram for the Stationery Haven e-commerce platform visualizes the interactions between the users and the system, representing the functional requirements and core activities of the platform. The primary actors in this diagram include:

1. **Customer:** The end-user who interacts with the platform to browse products, add them to the cart, make payments, and track orders.
2. **Admin:** The system administrator who manages product listings, inventory, and user activity on the platform.

Key use cases for Customer include:

- **Browse Products:** The customer can view the list of available stationery items and filter them based on categories or search queries.
- **View Product Details:** The customer can click on any product to see detailed descriptions, pricing, and availability status.
- **Add to Cart:** The customer can add selected products to their shopping cart.
- **Manage Cart:** The customer can modify the cart, including adding, removing, or adjusting the quantity of items.
- **Checkout and Payment:** The customer can proceed to checkout, select a payment option (UPI, net banking), and complete the transaction securely.
- **View Order Receipt:** After successful payment, the customer receives an instant receipt and order confirmation.
- **Track Orders:** The customer can track the status of their orders through the platform.

Key use cases for Admin include:

- **Manage Products:** The admin can add, update, or remove products from the system based on inventory and sales requirements.
- **Manage Orders:** The admin can view and update the status of customer orders.
- **View Analytics:** The admin can analyse sales data, user behaviour, and product performance.

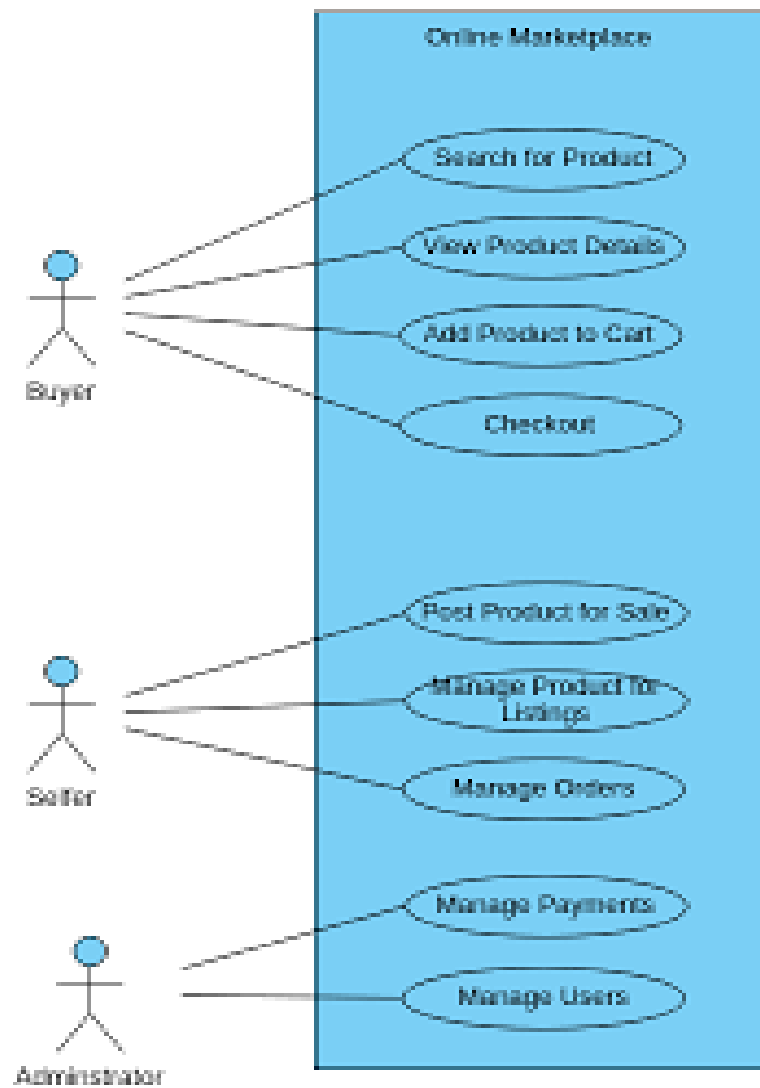


Figure 2.1. Use Case Diagram of Online Stationary Shopping System

2.3.2. ENTITY RELATIONSHIP DIAGRAM:

The ER diagram represents the structure and relationships in the Online Shopping System. It begins with the User entity, which stores details such as name, email, password, and address. Users can place Orders, each containing multiple OrderDetails that specify the products, quantities, and subtotals. The Product entity holds information about items, including their name, description, price, and stock quantity, and is categorized under the Category entity for better organization. Users can write Reviews.

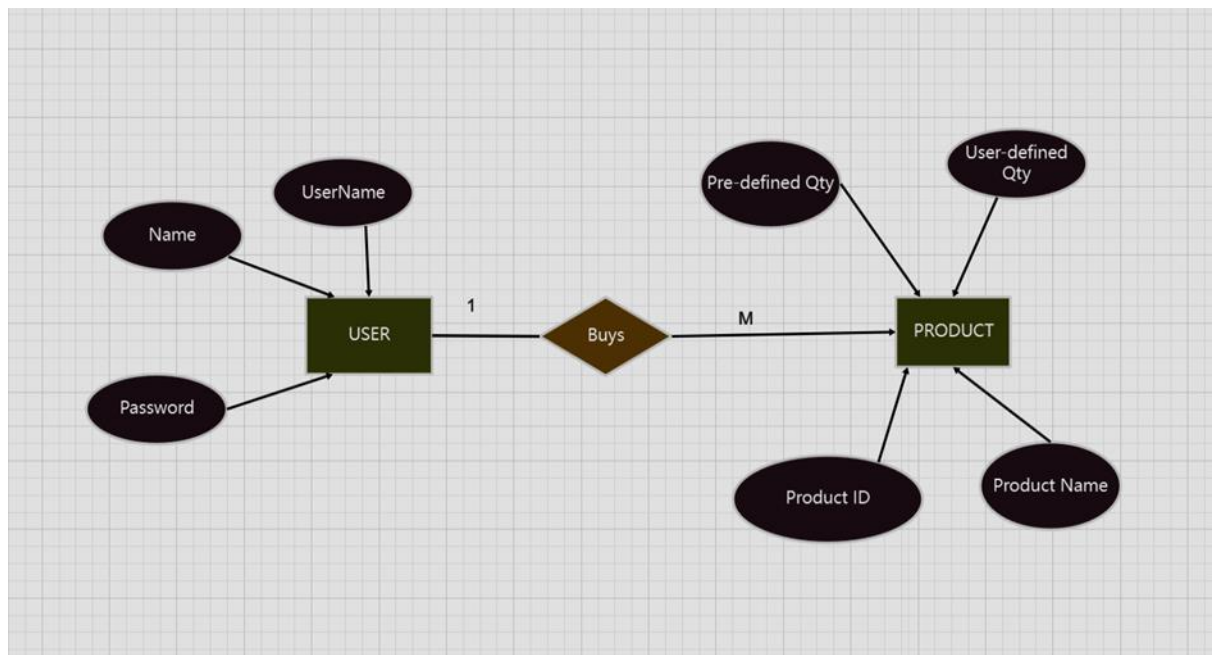


Figure 2.2. Entity Relationship diagram of Online Shopping System

2.3.3. DATA FLOW DIAGRAM:

The Data Flow Diagram (DFD) for the Stationery Haven project illustrates how data moves through the system, showcasing the interactions between users, system components, and external entities. It helps visualize the system's data flow and provides a clear understanding of its operations.

Level 0: Context Diagram

At the highest level, the system is represented as a single entity that interacts with two main actors: Customer and Admin. The Customer interacts with the platform to browse products, make purchases, and track orders, while the Admin manages product listings, inventory, and order processing.

Level 1: Detailed Data Flow

In this level, the system is broken down into key processes and their corresponding data flows:

- **Browse and Search Products:**
Input: Customer's search query
Output: Product details retrieved from the Product Database
- **Add to Cart:**
Input: Selected product by customer
Output: Updated items in the Shopping Cart
- **Checkout and Payment:**
Input: Customer's cart details and payment information
Output: Payment transaction processed by the Payment Gateway, and order confirmed in the Order System
- **Track Orders:**
Input: Customer order ID
Output: Order status retrieved from the Order Database
- **Admin Product Management:**
Input: Product data updates from Admin
Output: Updated product information stored in the Product Database

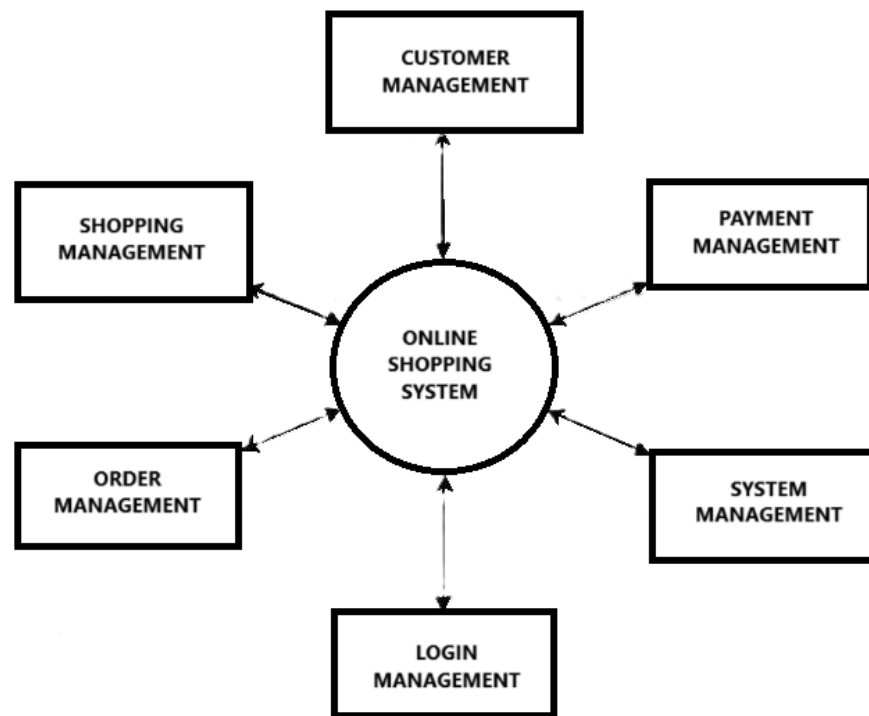


Figure 2.3. Data Flow Diagram of Online Stationaries Shopping System

2.4. SYSTEM SPECIFICATION

Frontend:

1. HTML: For structuring web pages.
2. CSS: For styling and creating responsive, visually appealing designs.
3. JavaScript: For adding interactivity and handling dynamic user actions.

Backend:

1. Programming Language: Java
2. Database Connectivity: JDBC (Java Database Connectivity)
3. Server: Built-in Java HTTP Server for handling client requests and responses.

Database:

1. Database Management System (DBMS): Oracle SQL
2. Database Features:
 - Tables for users, products, cart, and orders.
 - Constraints for data integrity (e.g., primary keys, foreign keys, unique constraints).

Development Tools:

1. Integrated Development Environment (IDE): Visual Studio Code with Java support.
2. Database Tools: Oracle SQL Developer Compatible with Windows
3. Browser Compatibility: Works on modern browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge, for frontend interaction.

2.4.1. FUNCTIONAL REQUIREMENTS

1. User Management:

- Allow users to register by providing personal details such as name, email, password, and address.
- Enable users to log in securely using their credentials.
- Provide functionality for users to update their profile information and reset passwords if needed.

2. Product Management:

- Display a catalog of products with details such as name, description, price, and availability.
- Enable users to search for products using filters like category, price range, and keywords.
- Allow admin users to add, update, or remove products from the catalogue.

3. Shopping Cart:

- Allow users to add products to their cart with specified quantities.
- Provide functionality to view, update, or remove items from the cart.
- Display the total price of items in the cart, including applicable taxes.

4. Order Management:

- Enable users to place orders directly from their cart.
- Provide order confirmation with details such as order ID, product summary, and total amount.

5. Payment Processing:

- Support multiple payment methods, such as credit/debit cards, net banking, or wallets.
- Validate payment details and provide status updates (e.g., successful, pending, or failed).

CHAPTER 3

DESIGN

3.1. DESIGN

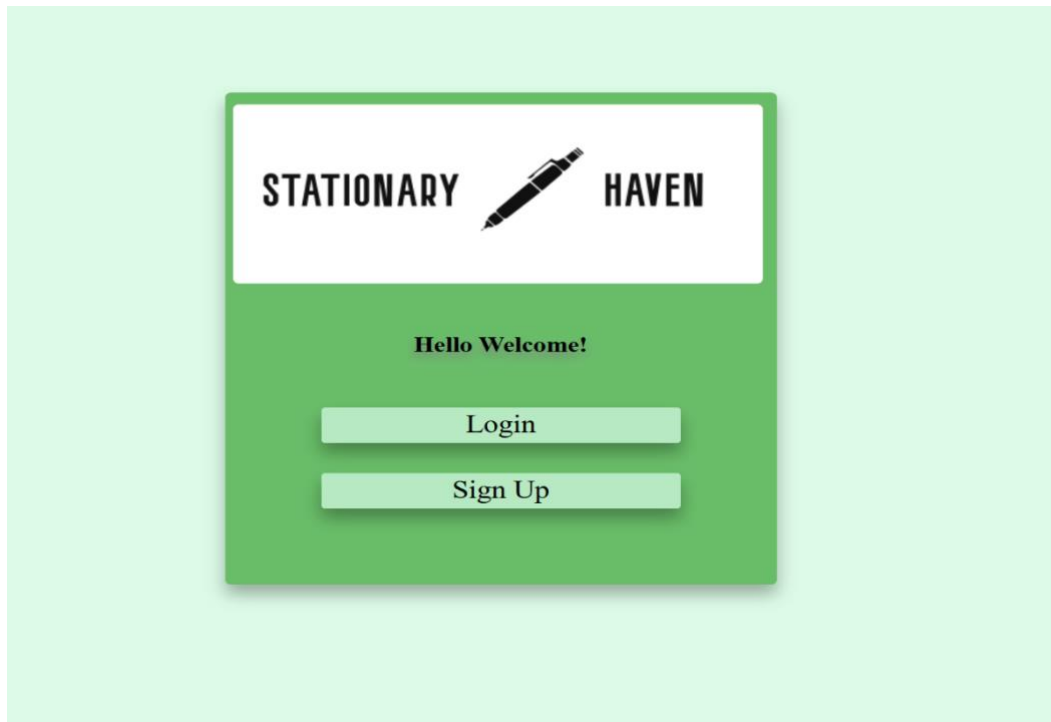


Figure 3.1. Login/Sign Up Page

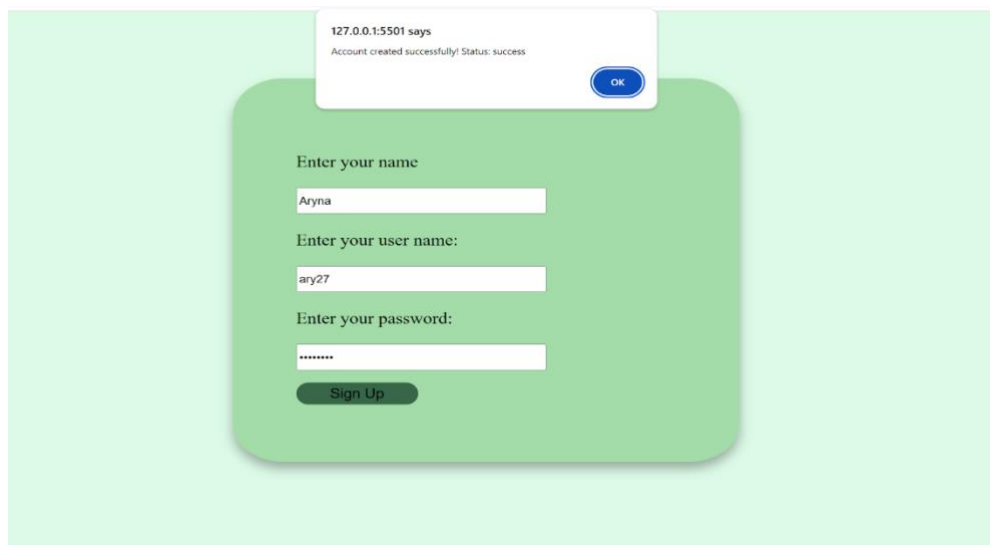


Figure 3.2. Sign Up Page

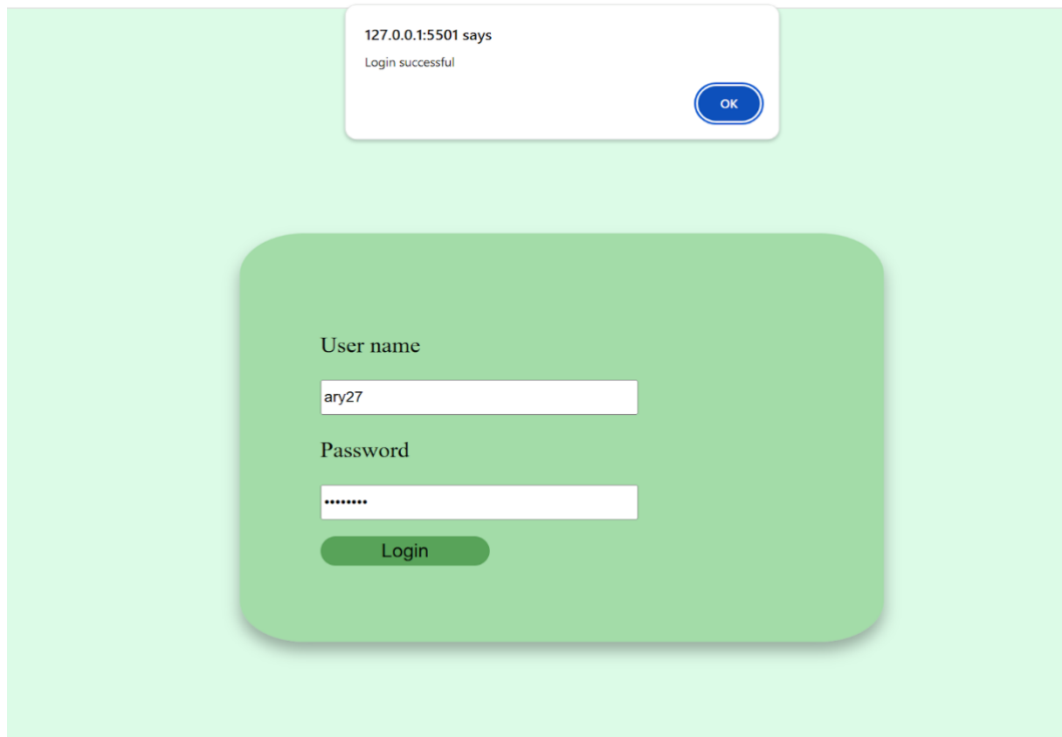


Figure 3.3. Login Page

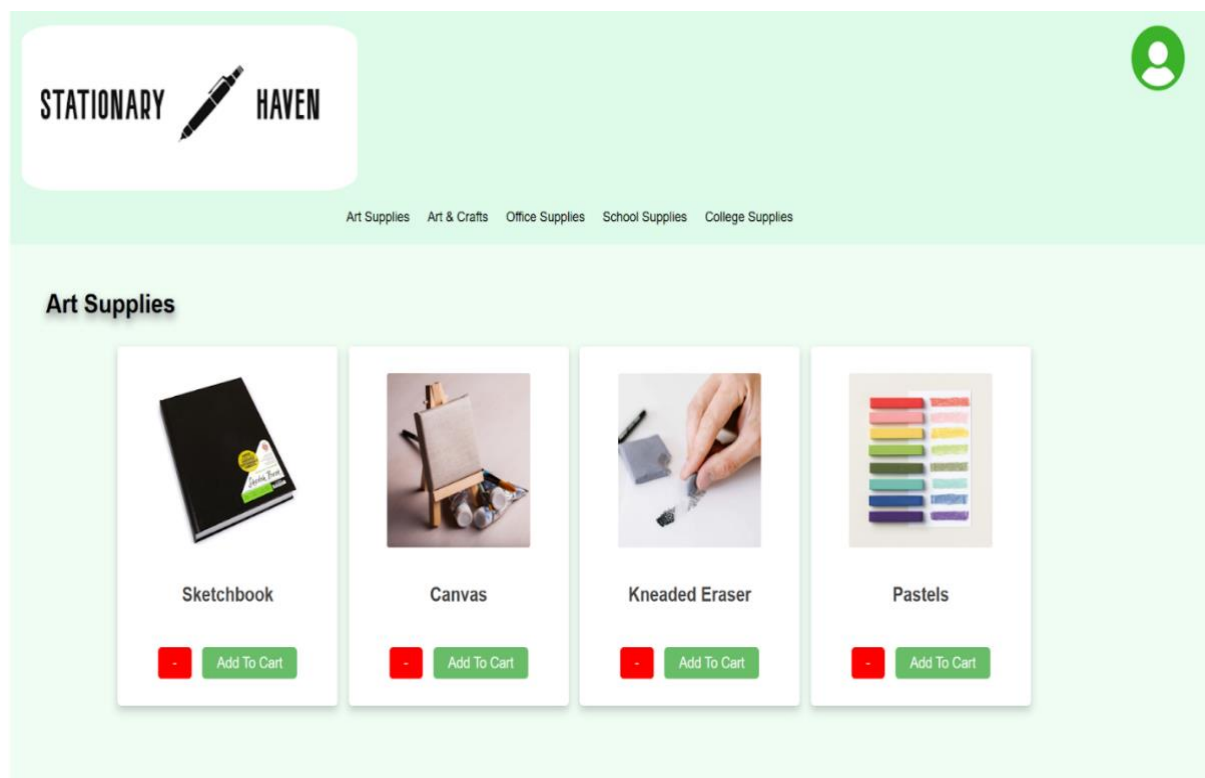


Figure 3.4. Main Page

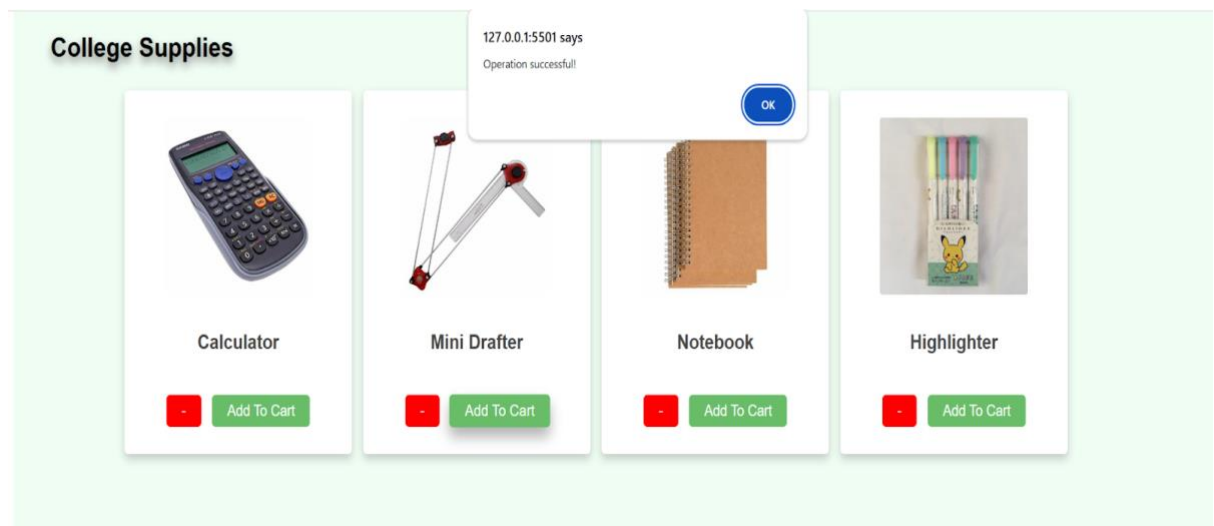


Figure 3.5. Acknowledgement Page

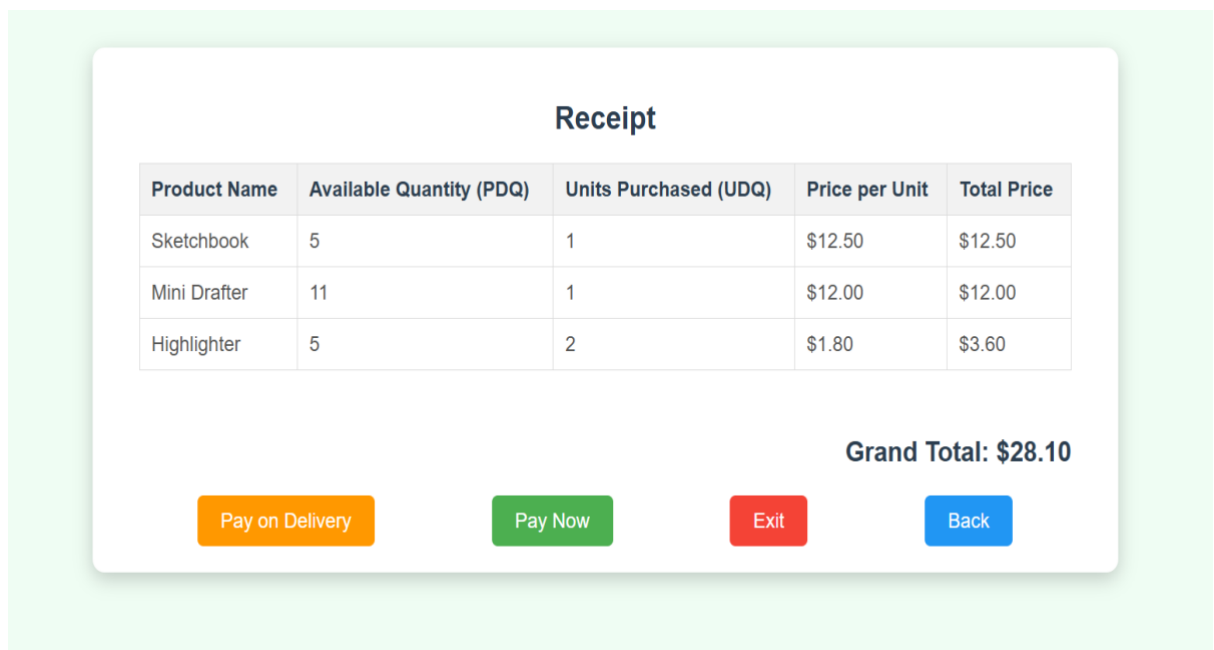
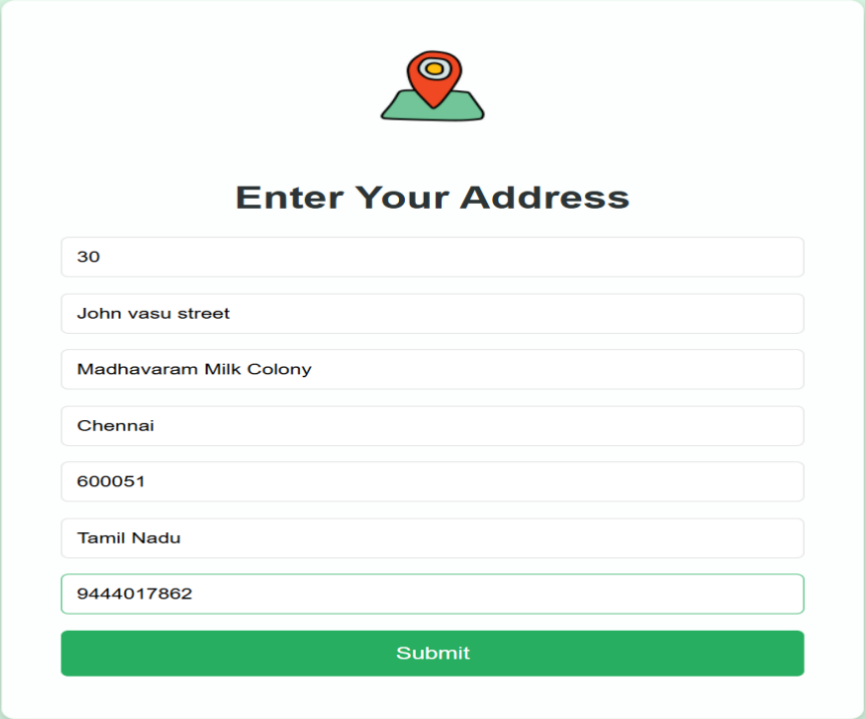


Figure 3.6. Receipt Page



The image shows a web form titled "Enter Your Address" with a location pin icon at the top. The form consists of eight input fields stacked vertically, each containing a specific address component. At the bottom of the form is a green "Submit" button.

Enter Your Address

30

John vasu street

Madhavaram Milk Colony

Chennai

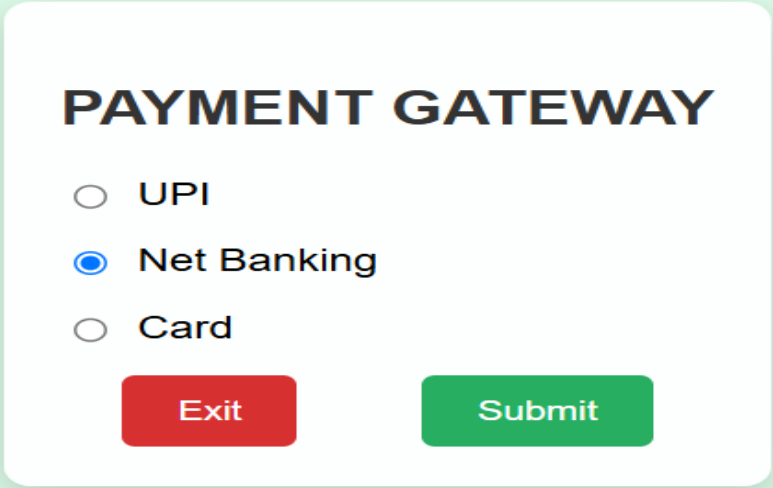
600051

Tamil Nadu

9444017862

Submit

Figure 3.7. Address Page



The image shows a web form titled "PAYMENT GATEWAY" with three radio button options. The "Net Banking" option is selected. At the bottom are two buttons: a red "Exit" button and a green "Submit" button.

PAYMENT GATEWAY

☐ UPI

☒ Net Banking

☐ Card

Exit Submit

Figure 3.8. Payment Gateway Page

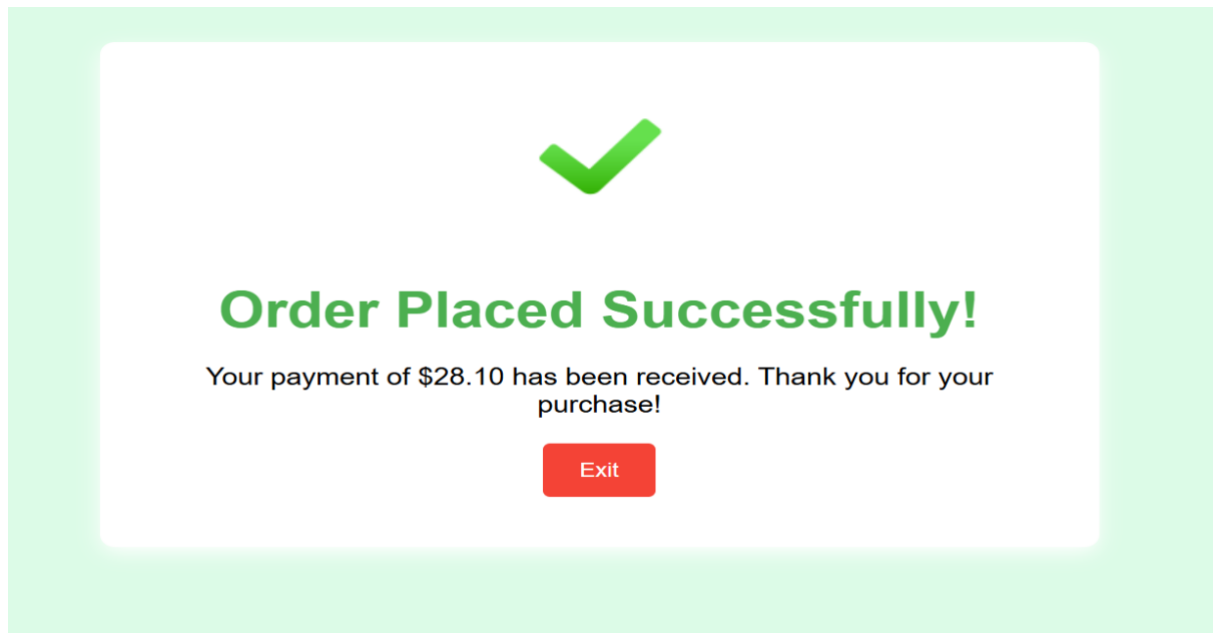


Figure 3.9. Payment Successful Page

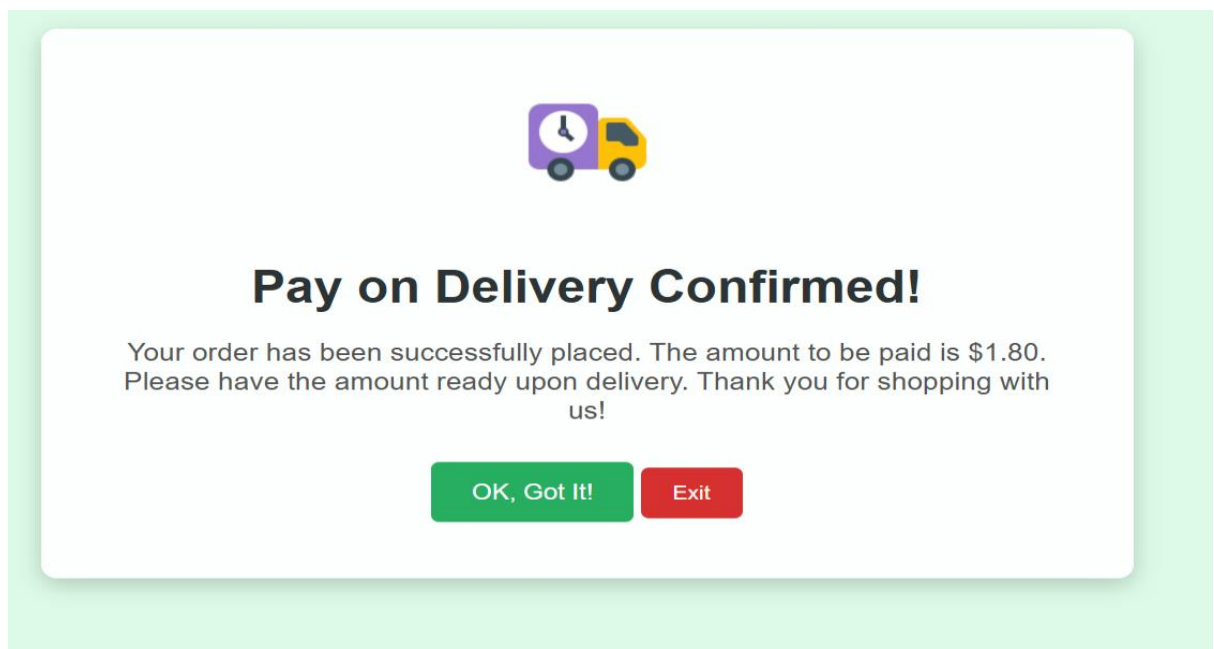


Figure 3.10. Pay on Delivery Page

3.2. DATABASE DESIGN

Products Table

Purpose: Maintains information about the available products in the inventory.

Attributes:

- Product ID (PID): Unique identifier for each product (Primary Key).
- Product Name (PNAME): Descriptive name of the product.
- Price: Cost of the product.
- Physical Display Quantity (PDQ): Total available stock.
- User Demand Quantity (UDQ): Quantity added to the user's cart.

Key Highlights:

- Tracks inventory levels accurately.
- Supports operations for adding or removing items to/from the cart.

PID	PNAME	PRICE	PDQ	UDQ
-----	-----	-----	-----	-----
P001	Sketchbook	12.5	5	0
P002	Canvas	25	13	0
P003	Kneaded Eraser	1.5	9	0
P004	Pastels	15	10	0
P005	Brushes	8	17	0
P006	Adhesive Vinyl	5	20	0
P007	Poster Colors	7.5	12	0
P008	Markers	6	18	0
P009	Paper Clips	1	10	0
P010	Register	4	10	0
P011	Folders	3.5	11	0

Table 3.1. Products Table

PID	PNAME	PRICE	PDQ	UDQ
-----	-----	-----	-----	-----
P012	Desk Organizer	15	20	0
P013	Pencil Case	2.5	18	0
P014	Pens	1.2	13	0
P015	Pencils	.8	10	0
P016	Geometry Box	5.5	5	0
P017	Calculator	18	14	0
P018	Mini Drafter	12	11	0
P019	Notebooks	2	16	0
P020	Highlighter	1.8	4	0

Table 3.1. Products Table

Users Table

Purpose: Manages user authentication details.

Attributes:

- Username: Unique identifier for each user (Primary Key).
- Password: Stores the password securely for authentication purposes.

Key Highlights:

- Ensures unique usernames to avoid duplication.
- Supports secure login operations.

USERNA	PASSWORD
-----	-----
radhi	rad@123
art1	art@1
abi	abi@16
sat	sat@01
aka73	aka@73
jan	jan@10
arch	arch@16
anu1	anu@1

Table 3.2. Users Table

3.3. IMPLEMENTATION (CODE)

1. Server Initialization

This snippet shows how the HTTP server is initialized and endpoints are set up

```
HttpServer server = HttpServer.create(new InetSocketAddress(8081), 0);
server.createContext("/login", new LoginHandler());
server.createContext("/validate-login", new ValidateLoginHandler());
server.createContext("/fetch-items", new FetchItemsHandler());
server.createContext("/update-quantity", new UpdateQuantityHandler());
server.createContext("/exit-before-payment", new ExitBeforePaymentHandler());
server.createContext("/exit-after-payment", new ExitAfterPaymentHandler());
server.setExecutor(null); // Creates a default executor
server.start();
System.out.println("Server started on port 8081");
```

2. Login Endpoint

This snippet handles user registration by receiving data from the frontend and inserting it into the database:

```
if ("POST".equalsIgnoreCase(requestMethod)) {
    String jsonData = new
Scanner(exchange.getRequestBody()).useDelimiter("\\A").next();
    String username = extractJsonValue(jsonData, "username");
    String password = extractJsonValue(jsonData, "password");

    try (Connection con = DriverManager.getConnection(jdbcUrl, dbUsername,
dbPassword)) {
        String sql = "INSERT INTO loginpage (username, password) VALUES (?, ?)";
        PreparedStatement stmt = con.prepareStatement(sql);
        stmt.setString(1, username);
        stmt.setString(2, password);
        stmt.executeUpdate();
    }
```



```

        sendResponse(exchange, 200, "{\"status\":\"success\"}");
    } catch (SQLException e) {
        if (e.getErrorCode() == 1) { // Duplicate username
            sendResponse(exchange, 409, "{\"status\":\"error\", \"message\":\"Username
exists\"}");
        }
    }
}

```

3. Login Validation

This snippet validates user credentials by querying the database:

```

String sql = "SELECT COUNT(*) FROM loginpage WHERE username = ? AND
password = ?";
PreparedStatement stmt = con.prepareStatement(sql);
stmt.setString(1, username);
stmt.setString(2, password);
ResultSet rs = stmt.executeQuery();
if (rs.next() && rs.getInt(1) > 0) {
    sendResponse(exchange, 200, "{\"status\":\"success\", \"message\":\"Login
successful\"}");
} else {
    sendResponse(exchange, 401, "{\"status\":\"error\", \"message\":\"Invalid
credentials\"}");
}

```

4. Fetch Products

This snippet retrieves product data and calculates the total price for the user's cart:

```

String sql = "SELECT pname, pdq, udq, price FROM product WHERE udq >= 1";
PreparedStatement stmt = con.prepareStatement(sql);
ResultSet rs = stmt.executeQuery();

StringBuilder jsonBuilder = new StringBuilder("");

```

```

double grandTotal = 0.0;

while (rs.next()) {
    if (jsonBuilder.length() > 1) jsonBuilder.append(",");
    String pname = rs.getString("pname");
    int udq = rs.getInt("udq");
    double price = rs.getDouble("price");
    double totalPrice = price * udq;
    grandTotal += totalPrice;

    jsonBuilder.append(String.format(
        "{\"pname\":\"%s\", \"udq\":%d, \"price\":%.2f, \"totalPrice\":%.2f}",
        pname, udq, price, totalPrice
    ));
}
jsonBuilder.append("]");
String jsonResponse = String.format("{\"products\": %s, \"grandTotal\": %.2f}",
    jsonBuilder, grandTotal);
sendResponse(exchange, 200, jsonResponse);

```

5. Update Product Quantities

This snippet adjusts product quantities based on user actions (add or decrease):

```

String action = extractJsonValue(jsonData, "action");
if ("add".equals(action) && udq < pdq) {
    String updateSql = "UPDATE product SET udq = udq + 1, pdq = pdq - 1 WHERE
pid = ?";
    PreparedStatement updateStmt = con.prepareStatement(updateSql);
    updateStmt.setString(1, pid);
    updateStmt.executeUpdate();
    sendResponse(exchange, "{\"success\": true}");
} else if ("decrease".equals(action) && udq > 0) {
    String updateSql = "UPDATE product SET udq = udq - 1, pdq = pdq + 1 WHERE
pid = ?";

```

```

        PreparedStatement updateStmt = con.prepareStatement(updateSql);
        updateStmt.setString(1, pid);
        updateStmt.executeUpdate();
        sendResponse(exchange, "{\"success\": true}");
    } else {
        sendResponse(exchange, "{\"success\": false, \"message\": \"Invalid
operation\"}");
    }
}

```

6. Reset Quantities on Exit

This snippet resets product quantities if the user exits before or after payment:

```

String updateSql = "UPDATE product SET pdq = pdq + udq, udq = 0";
PreparedStatement updateStmt = con.prepareStatement(updateSql);
updateStmt.executeUpdate();
sendResponse(exchange, "{\"success\": true, \"message\": \"All items returned to
stock\"}");

```

7. CORS Handling

This snippet adds headers to handle cross-origin requests:

```

private static void addCORSHeaders(HttpExchange exchange) {
    exchange.getResponseHeaders().add("Access-Control-Allow-Origin", "*");
    exchange.getResponseHeaders().add("Access-Control-Allow-Methods", "GET,
POST, OPTIONS");
    exchange.getResponseHeaders().add("Access-Control-Allow-Headers", "Content-
Type");
}

```

CHAPTER 4

CONCLUSION

The Stationary Haven project efficiently showcases the core functionalities of a stationary management system in a simplified and user-friendly way. By incorporating features such as user registration, login, product catalog management, inventory updates, and cart functionality, the project effectively handles essential operations for managing a stationary store. The implementation also includes practical functionalities like calculating totals for selected items and real-time inventory updates, which reflect a strong understanding of user-centric features. Additionally, the modular structure of the code ensures scalability and maintainability, making it easier to extend the system with additional features in the future. Overall, this project lays a strong foundation for understanding the design and implementation of stationary management systems. It highlights the interaction between database management, server-side processing, and frontend design, achieving its goal of demonstrating key principles while offering scope for future enhancements.

REFERENCES

- [1] <https://www.geeksforgeeks.org/establishing-jdbc-connection-in-java/>
- [2] <https://www.geeksforgeeks.org/courses/Java-backend-live>
- [3] <https://www.gfg.com>
- [4] <https://www.w3schools.com/tutorials/>
- [5] <https://www.oracle.com/technical-resources/articles/java/webapps.html>
- [6] JAVA TECHNOLOGY IN THE DESIGN AND IMPLEMENTATION OF WEB APPLICATIONS January 2012 Technics Technologies Education Management
- [7] Research Java Web framework based on OSGi December 2011Procedia Engineering