E-Commerce Web Application

A PROJECT REPORT

Submitted by

Gerald Alan Raj (RA2211056010147)

Mamtha Sri (RA2211056010161)

Under the Guidance of

Dr. N Vasudevan

Assistant Professor, Department of Data Science and Business Systems

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEMS

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR– 603 203

NOV 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR–603 203

BONAFIDE CERTIFICATE

Register No: RA2211056010147, RA2211056010161 Certified to be the bonafide work done by **Gerald Alan Raj, Mamtha Sri** of III year/V sem B.Tech Degree Course in the subject – **21IPE336T – Advanced Java Programming** in **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**, Kattankulathur for the academic year 2024-2025.

Date:

Faculty in Charge
Dr. N.Vasudevan
Assistant Professor
Data Science and Business Systems
SRMSIT -KTR

HEAD OF THE DEPARTMENT
Dr. V.Kavitha
Professor
Data Science and Business Systems
SRMIST - KTR

ABSTRACT

The E-commerce Web Application is a streamlined and efficient solution designed to enable online shopping experiences for users. It provides an intuitive and responsive interface where users can browse a wide range of products, add them to a shopping cart, and proceed through the checkout process seamlessly. The system is built using Angular 17 for the frontend, Spring Boot for the backend, and SQL for database management, ensuring a robust and scalable platform. At the core of the system is a dynamic product catalog, where users can view detailed product information such as name, price, and description. The application allows users to easily manage their shopping cart, view ordered items, and make purchases with a smooth checkout process. Additionally, users have the ability to delete items from their order if needed. The backend is responsible for retrieving product data from the SQL database and processing user orders through Spring Boot APIs. The application's user-friendly interface ensures that customers can efficiently browse products and make informed purchasing decisions. Built with modern web technologies, the system offers a high-performance, cross-platform compatible experience, providing users with a seamless and secure shopping journey. This E-commerce Web Application serves as an essential tool for businesses to offer an online shopping platform, optimize product listings, and streamline order management. Its ease of use, flexibility, and comprehensive features make it an ideal solution for delivering a functional and interactive e-commerce experience.

TABLE OF CONTENTS

CHAPTER 1

1. Introduction

The E-commerce Web Application is designed to provide a user-friendly online shopping experience by integrating modern web technologies to facilitate product browsing, cart management, and order placement. The application leverages Angular 17 for the frontend, Spring Boot for the backend, and SQL for database management, ensuring seamless data flow between the user interface and the backend services. Users can easily explore a catalog of products, add items to their cart, and proceed with the checkout process, all while enjoying a responsive and intuitive interface. The backend is responsible for managing product data, processing transactions, and ensuring smooth communication with the database. This system aims to streamline the online shopping process by offering essential features such as viewing product details, managing orders, and deleting items from the cart. The project demonstrates the integration of frontend and backend technologies to build a fully functional e-commerce platform that can be expanded and customized for real-world applications. Through this application, businesses can provide a reliable, efficient, and engaging platform for customers to shop online.

CHAPTER 2

2. Project Objectives

The main objective of this project is to develop a simple yet functional e-commerce web application that provides a seamless shopping experience for users. The system aims to enable users to easily browse and search for products, add them to a shopping cart, and proceed with a secure checkout process. Additionally, the project focuses on creating a robust backend that effectively manages product data, processes user orders, and ensures smooth communication between the frontend and database. Key objectives include implementing features such as product listing, cart management, order placement, and the ability to view and delete ordered items. The project also seeks to demonstrate the integration of Angular 17, Spring Boot, and SQL, providing a full-stack solution for modern e-commerce platforms. Ultimately, the objective is to deliver an intuitive, scalable, and efficient solution that could be adapted to meet the needs of small to medium-sized businesses seeking to establish or enhance their online presence.

CHAPTER 3

3. Technologies Used

3.1 Angular 17

Overview: Angular 17 is a powerful, open-source front-end web application framework developed by Google. Angular uses TypeScript, which allows for strong typing and better maintainability. It provides a modular structure, efficient data binding, routing, and dependency injection, making it ideal for building scalable and interactive web applications. Angular's component-based architecture ensures reusability, and its extensive set of tools and libraries allows for seamless integration with APIs and backend services.

3.2 Spring Boot

Overview: Spring Boot is an open-source Java-based framework used to create stand-alone, production-grade Spring-based applications. It simplifies the process of configuring and deploying Java applications by offering embedded servers, auto-configuration, and a wide range of pre-built templates. In this project, Spring Boot serves as the backend framework to handle RESTful API endpoints, user authentication, order processing, and database interactions. It enables quick development and deployment of secure, scalable, and maintainable web services while integrating smoothly with the front end.

3.3 SQL (Structured Query Language)

Overview: SQL is a standard programming language used to manage and manipulate relational databases. In this project, SQL is employed to manage the storage and retrieval of product data, user orders, and other related information in a relational database. It enables efficient querying, updating, and deleting of data. SQL's ability to handle complex queries, join multiple tables, and ensure data integrity through constraints makes it an essential technology for managing structured data in a scalable and consistent manner.

3.4 HTML/CSS

Overview: HTML (HyperText Markup Language) and CSS (Cascading Style Sheets) are fundamental technologies for web development. HTML structures the content of the web pages, such as headings, paragraphs, and images, while CSS is used for styling the content, controlling layout, colors, fonts, and overall design. In this project, HTML and CSS are used to create the layout and design of the frontend application, ensuring it is visually appealing, user-friendly, and responsive across different devices and screen sizes.

3.5 JavaScript/TypeScript

Overview: JavaScript is a dynamic programming language commonly used for creating interactive and responsive web applications. TypeScript, a superset of JavaScript, adds static typing and enhanced features, which help in large-scale applications and improve code maintainability. Angular uses TypeScript to write more robust and type-safe code, making development easier and reducing potential errors in the application. TypeScript's integration with Angular allows developers to create clean, maintainable code for complex frontend functionality.

3.6 RESTful APIs

Overview: REST (Representational State Transfer) is an architectural style used to design networked applications. RESTful APIs are based on HTTP methods (GET, POST, PUT, DELETE) and allow communication between the frontend and backend. In this project, RESTful APIs are used to handle requests between the Angular frontend and the Spring Boot backend, facilitating operations such as retrieving product data, adding items to the cart, and processing orders. RESTful APIs ensure that the application is scalable, easy to maintain, and can interact with various systems and platforms.

CHAPTER 4

4. System Architecture

4.1 Frontend (Client-Side)

The frontend of the E-commerce Web Application is built using Angular 17, which provides a modern, component-based framework for developing dynamic and responsive user interfaces. It handles the presentation layer and user interactions, including displaying product listings, managing the shopping cart, and processing orders. The user interface (UI) is created with HTML and CSS to ensure a visually appealing and responsive design that adapts to different screen sizes and devices. TypeScript is used to implement business logic, ensuring type safety and better code maintainability. Angular's services manage HTTP requests to interact with the backend, retrieving product data and sending order-related information. The Angular Router enables seamless navigation between pages, such as from product listings to the shopping cart or checkout process.

4.2 Backend (Server-Side)

The backend of the application is powered by Spring Boot, a robust Java framework that simplifies the development of stand-alone, production-ready applications. It exposes RESTful APIs to allow communication between the frontend and backend, handling requests such as retrieving product data, adding items to the cart, processing orders, and managing order history. The backend processes business logic and ensures that data is consistently managed using Spring Data JPA, which abstracts the interaction with the SQL database. The service layer in Spring Boot validates and processes requests before storing or retrieving information from the database. The backend ensures secure and efficient transaction processing while maintaining the integrity of user data.

4.3 Interaction Between Frontend and Backend

The frontend and backend communicate through HTTP requests (GET, POST, DELETE) using Angular's HttpClient. When the user interacts with the interface, such as viewing products or adding items to the cart, the frontend sends requests to the backend's RESTful APIs. These requests allow the frontend to retrieve product data, update the cart, or place an order. The backend processes these requests, performs necessary validations, and interacts with the database to store or retrieve data. The response from the backend, typically in the form of JSON, is sent back to the frontend, which updates the UI accordingly. This interaction ensures a dynamic, responsive shopping experience for the user, with real-time updates to the cart and order information.

4.4 Data Flow Within the System

The data flow within the system begins when the user interacts with the frontend to browse products or add items to the cart. The frontend sends a GET request to the backend to retrieve the product list, which is then displayed to the user. When the user adds items to the cart, a POST request is sent to the backend, which updates the cart data in the database. Once the user is ready to checkout, another POST request is sent to create an order, storing the relevant information in the Checkout Table. If any changes are needed, such as deleting an item from the cart, the frontend sends a DELETE request to the backend, which updates the cart or order data in the database. Throughout this process, data is exchanged in JSON format, ensuring that the frontend and backend stay synchronized and up-to-date.

CHAPTER 5

5. Features

Add a New Product: Authorized users can add new products to the catalog, providing details like name, price, description, and other attributes.

Add a Product to Cart: Users can add products to their shopping cart, with the option to specify quantity and review the cart before checkout.

View Ordered Items: Users can view the list of items they have ordered, including product details and order status.

Delete an Item: Users have the ability to remove a product from the cart or delete an ordered item from their order history.

Update Quantity in Cart: Users can modify the quantity of products in their cart, adjusting the number of items they wish to purchase before proceeding to checkout.

CHAPTER 6

6. Implementation Details

6.1 Database Schema Design

The database schema for this E-commerce Web Application consists of two main tables: products and checkout.

1. **Products Table**:

   This table stores the details of the products available in the system.

   - **Columns**:
     - id: An auto-incremented integer that uniquely identifies each product.
     - name: A varchar field that stores the product name (up to 255 characters).
     - description: A text field that provides a detailed description of the product.
     - price: A decimal field that stores the price of the product, formatted to support values with two decimal places (e.g., 99999999.99).
     - image: A varchar field that stores the image URL or filename of the product's image.

   The **products** table serves as the primary source for product information that will be displayed to users. It provides flexibility to add new products and update existing ones.

2. **Checkout Table**:

   This table stores information about the orders placed by users, linking the ordered products with their respective details.

- **Columns**:
  - order_id: An auto-incremented integer that uniquely identifies each order.
  - product_id: An integer referencing the ID of the product from the **products** table.
  - product_name: The name of the product being ordered.
  - quantity: The number of units of the product ordered.
  - price: The price of the product at the time of ordering.
  - order_date: A timestamp of when the order was placed, with the default set to the current timestamp.

The **checkout** table is crucial for tracking user orders, including product details, quantities, and order timestamps. This data is stored once the user completes their purchase and is used for order history and tracking.

6.2 API Endpoints and Routes

The backend of this project is powered by **Spring Boot** and offers several API endpoints to interact with the database and provide necessary functionality to the frontend. Below is an overview of the key API endpoints:

1. GET /api/products: Fetch All Products

- **Route**: @GetMapping
- **Description**: Retrieves a list of all products from the **products** table. This endpoint calls a service method to fetch all products and returns them as a JSON array to the frontend.

2. POST /api/products/checkout: Place an Order (Checkout)

- **Route**: @PostMapping("/checkout")
- **Description**: This endpoint is called when a user proceeds to checkout. It accepts a list of products (cart items) and saves them in the **checkout** table. The cart items are saved along with their quantities, prices, and other order details.

3. GET /api/products/orders: View Order History

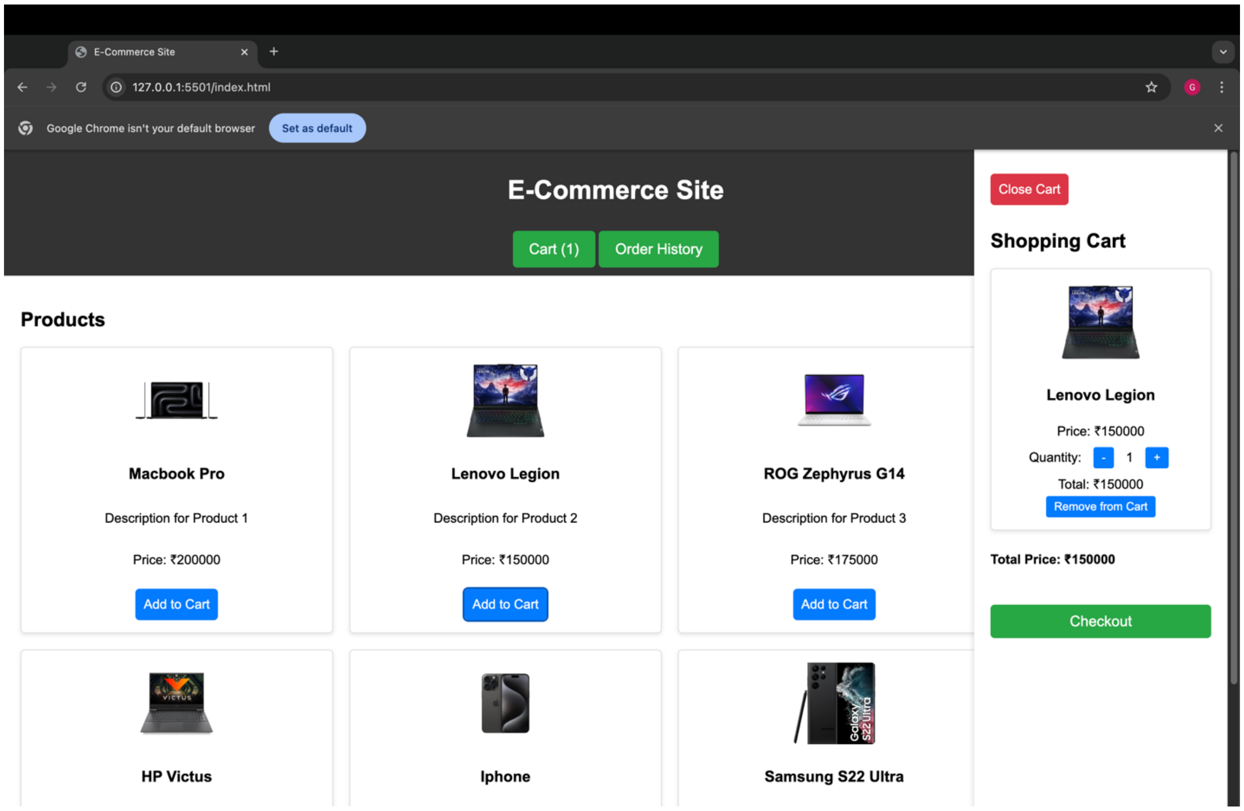
- **Route**: @GetMapping("/products/orders")
- **Description**: This endpoint retrieves the order history of a user. It fetches orders from the **checkout** table, including product names, quantities, prices, and order dates.

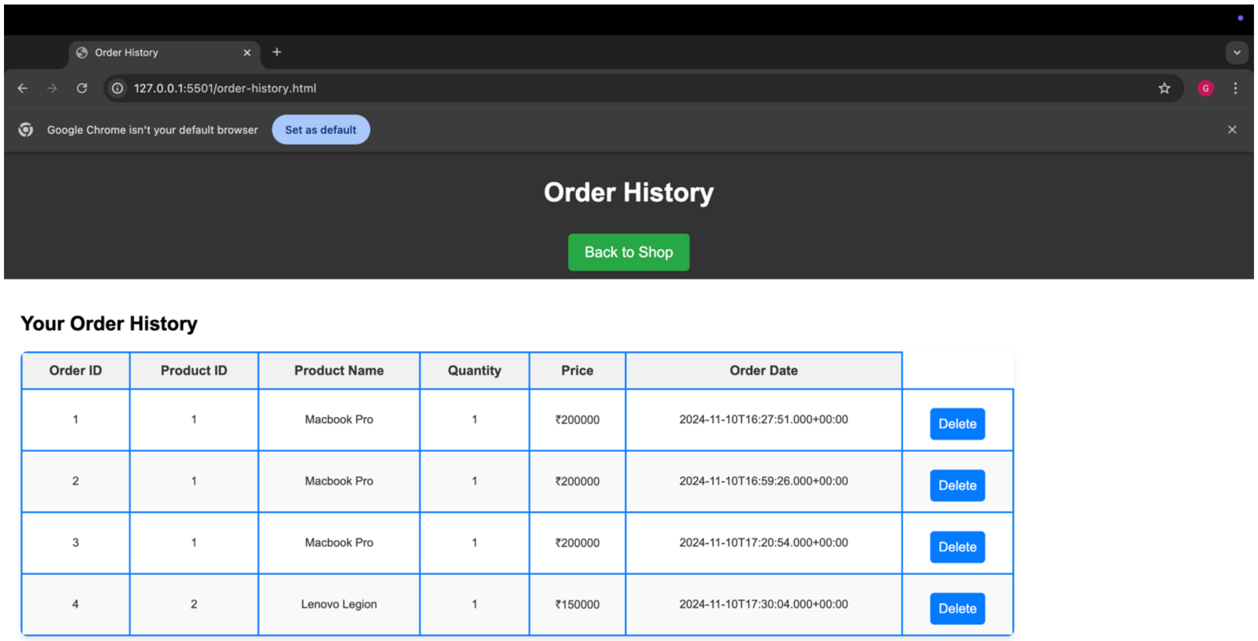
4. DELETE /api/products/orders/{orderId}: Delete an Order

- **Route**: @DeleteMapping("/products/orders/{orderId}")
- **Description**: This endpoint allows the deletion of an order by orderId from the order history. It deletes the corresponding record from the **checkout** table and returns a success message upon successful deletion.

CHAPTER 7

7. Result and Discussion




Order History
127.0.0.1:5501/order-history.html
Google Chrome isn't your default browser
Set as default
Order History
Back to Shop

Your Order History

Order ID	Product ID	Product Name	Quantity	Price	Order Date	
1	1	Macbook Pro	1	₹200000	2024-11-10T16:27:51.000+00:00	Delete
2	1	Macbook Pro	1	₹200000	2024-11-10T16:59:26.000+00:00	Delete
3	1	Macbook Pro	1	₹200000	2024-11-10T17:20:54.000+00:00	Delete
4	2	Lenovo Legion	1	₹150000	2024-11-10T17:30:04.000+00:00	Delete

CHAPTER 8

8. Conclusion

In conclusion, this E-commerce Web Application effectively integrates key technologies such as Angular, Spring Boot, and SQL to build a robust and efficient platform. The project successfully implements essential functionalities, including product listing, cart management, order placement, and order history management, providing users with a seamless shopping experience. The backend, built with Spring Boot, handles business logic and database interactions through well-structured API endpoints, ensuring smooth communication with the frontend and database. The SQL database schema is designed to support efficient data retrieval and management of products and orders. The Angular frontend delivers a dynamic and responsive user interface, interacting with the backend to reflect real-time updates on the user's actions. This project lays a strong foundation for an e-commerce platform, with potential for future enhancements such as user authentication, payment integration, and inventory management. The modular design and use of modern technologies make the system scalable and adaptable to future requirements, paving the way for continued development and growth.