**Software Requirements**

**Specification**

**for**

**Buyzu**

**Final Version**

|  |  |  |
| --- | --- | --- |
|  |  | **Prepared by**  **Group Name: E7** |
| **GUO Menglong** | **1155191409** | **1155191409@link.cuhk.edu.hk** |
| **MA Heyang** | **1155191433** | **1155191433@link.cuhk.edu.hk** |
| **XUE Guangxuan** | **1155194956** | **1155194956@link.cuhk.edu.hk** |
| **YAN Yitao** | **1155191529** | **1155191529@link.cuhk.edu.hk** |
| **ZHU Keyu** | **1155191834** | **1155191834@link.cuhk.edu.hk** |

**Instructor: *Dr.LAM Tak Kei***

**Course: CSCI3100**

**Date: 09/05/2025**

***Software Requirements Specification for Buyzu***   
  ***Page ii***

# Contents

[CONTENTS II](#_Toc36451)

[REVISIONS II](#_Toc36452)

[1 INTRODUCTION 1](#_Toc36453)

[1.1 DOCUMENT PURPOSE 1](#_Toc36454)

[1.2 PRODUCT SCOPE 1](#_Toc36455)

[1.3 INTENDED AUDIENCE AND DOCUMENT OVERVIEW 1](#_Toc36456)

[1.4 DEFINITIONS, ACRONYMS AND ABBREVIATIONS 2](#_Toc36457)

[1.5 DOCUMENT CONVENTIONS 3](#_Toc36458)

[1.6 REFERENCES AND ACKNOWLEDGMENTS 3](#_Toc36459)

[2 OVERALL DESCRIPTION 5](#_Toc36460)

[2.1 PRODUCT OVERVIEW 5](#_Toc36461)

[2.2 PRODUCT FUNCTIONALITY 5](#_Toc36462)

[2.3 DESIGN AND IMPLEMENTATION CONSTRAINTS 6](#_Toc36463)

[2.4 ASSUMPTIONS AND DEPENDENCIES 7](#_Toc36464)

[3 SPECIFIC REQUIREMENTS 9](#_Toc36465)

[3.1 EXTERNAL INTERFACE REQUIREMENTS 9](#_Toc36466)

[3.2 FUNCTIONAL REQUIREMENTS 19](#_Toc36467)

[3.3 USE CASE MODEL 21](#_Toc36468)

[4 OTHER NON-FUNCTIONAL REQUIREMENTS 23](#_Toc36469)

[4.1 PERFORMANCE REQUIREMENTS 23](#_Toc36470)

[4.2 SAFETY AND SECURITY REQUIREMENTS 24](#_Toc36471)

[4.3 SOFTWARE QUALITY ATTRIBUTES 25](#_Toc36472)

[APPENDIX A – DATA DICTIONARY 25](#_Toc36473)

[APPENDIX B - GROUP LOG 28](#_Toc36474)

***Software Requirements Specification for Buyzu***   ***Page iii***

# Revisions

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Primary Author(s)** | **Description of Version** | **Date Completed** |
| V1.0 | **GUO Menglong**  **MA Heyang**  **Xue Guangxuan**  **Yan Yitao**  **ZHU Keyu** | Initial release of the Software Requirements Specification for the Buyzu project. | 10/02/2025 |
| V2.0 | **GUO Menglong**  **MA Heyang**  **Xue Guangxuan**  **Yan Yitao**  **ZHU Keyu** | Final release of the Software Requirements Specification for the Buyzu project. | 09/05/2025 |

# 1. Introduction

## 1.1 Document Purpose

This document is trying to outline the software requirements specification for the Buyzu, which is an online shopping platform. This platform provides functionality such as user signup and login, browsing products, adding items to the shopping cart, completing purchases and reviewing users’ purchases. This software requirement specification (SRS) focuses on detailing the functional and non-functional requirements of the system. Ensuring product manager, developers, testers and potential follow-on members of the product family to have a clear understanding of system function and limitations.

## 1.2 Product Scope

The Buyzu is designed to provide a clear and intuitive online shopping experience for our users. It allows users to browse products, add items to their shopping carts, checkout, and complete payments securely.

**Key Benefits & Objectives:**

* Provide a user-friendly interface for online shopping.
* Ensure secure transactions with encrypted payment processing.
* Support scalability to develop new features to accommodate users’ preferences.
* Facilitate efficient order management customers.

## 1.3 Intended Audience and Document Overview

This document is intended for:

* **Developers (My groupmates)**: To implement the system based on the defined requirements.
* **Project Managers (Also my groupmates :)**: To oversee development and ensure alignment with project goals.
* **Testers**: To design testing cases and verify system functionality.
* **Clients (maybe) and Professors**: To review the scope, objectives, and expected deliverables of the system.

Suggested reading sequence:

**1: Overall Description** – Provides a high-level overview of the system, including assumptions and dependencies.

**2: Specific Requirements** – Details the core features and expected behaviors of the system.

**3: Non-Functional Requirements** – Specifies performance, security, and usability expectations.

**4: References and Appendices** – Lists related documentation and supplementary materials.

## 1.4 Definitions, Acronyms and Abbreviations

**AES-256**: Advanced Encryption Standard 256-bit

**API**: Application Programming Interface

**AWS**: Amazon Web Services

**AWS S3**: Amazon Web Services Simple Storage Service

**CSS**: Cascading Style Sheets

**CSV**: Comma-Separated Values

**GDPR**: General Data Protection Regulation

**GRS**: Geographically Redundant Storage

**HTTP**: Hypertext Transfer Protocol

**HTTPS**: Hypertext Transfer Protocol Secure

**JSON**: JavaScript Object Notation

**MFA**: Multi-Factor Authentication

**MQTT**: Message Queuing Telemetry Transport

**NGINX**: Engine-X

**OAuth 2.0**: Open Authorization 2.0

**PCI-DSS**: Payment Card Industry Data Security Standard

**POS:** Point of Sale

**RBAC**: Role-Based Access Control

**SLAs:** Service Level Agreements

**SRS**: Software Requirements Specification

**SSO**: Single Sign-On

**SSL/TLS**: Secure Sockets Layer / Transport Layer Security

**TLS**: Transport Layer Security

## 1.5 Document Conventions

This document follows the IEEE formatting standards. The following conventions apply:

**Formatting Conventions:**

* **Font:** Times, size 12 for all text.
* **Spacing:** Fix-spaced text with 1-inch margins, with 0.5 line before and after.
* **Emphasis**: **Bold** are used for comments, notes, and non-standard text that requires special attention.
* **Section Titles:** All **section** and **subsection** titles follow the predefined template to ensure consistency across the document.

**Naming Conventions:**

* **Variable and Function Names**: When referring to system variables or functions within the document, **CamelCase notation** is used, starting with a lowercase letter (e.g., shoppingCart, userProfile).
* **Abbreviations and Acronyms**: Abbreviations and acronyms are spelled out the first time they are used, followed by the acronym in parentheses (e.g., **Software Requirements Specification (SRS)**).

## 1.6 References and Acknowledgments

Institute of Electrical and Electronics Engineers, *IEEE Standard for Software Requirements Specifications (IEEE 830-1998)*, 1998. [Online]. Available: https://ieeexplore.ieee.org/document/720574. [Accessed: 10-Feb-2025].

"Non-Functional Requirements in Software Engineering," *GeeksforGeeks*, 03-Jan-2025. [Online]. Available: https://www.geeksforgeeks.org/non-functional-requirements-in-software-engineering/. [Accessed: 10-Feb-2025].

PCI Security Standards Council. "PCI Security Standards," *PCI Security Standards Council*.

[Online]. Available: https://www.pcisecuritystandards.org/standards/. [Accessed: 10-Feb-2025].

Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation).

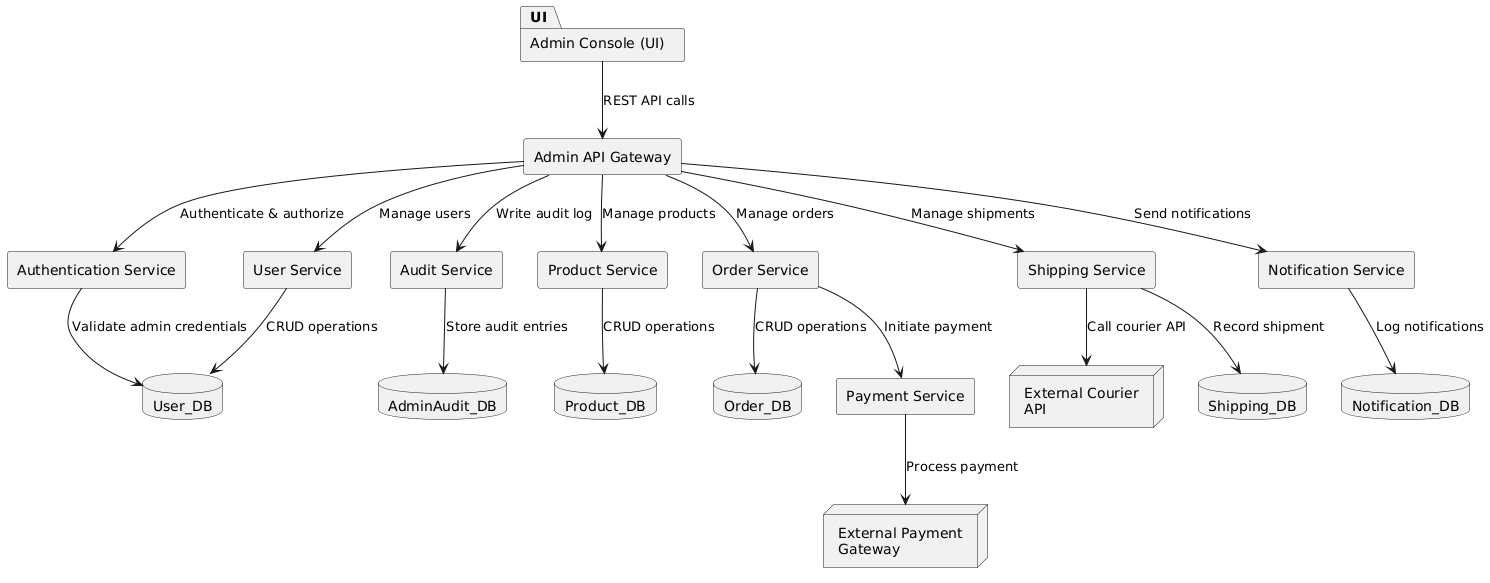
*Official Journal of the European Union*, L 119, 4 May 2016, pp. 1-88. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj/eng. [Accessed: 10-Feb-2025].

S. Cordes, "Non-Functional Requirements Examples and Templates," *The Business Analyst's Toolkit*, 05-Dec-2024. [Online]. Available: https://www.businessanalyststoolkit.com/nonfunctional-requirements-examples-and-templates/. [Accessed: 10-Feb-2025].

# 2. Overall Description

## 2.1 Product Overview [modified]

The Buyzu is a consumer-centric web platform designed to deliver a fast, intuitive, and secure shopping experience tailored to individual users. Built with modern web technologies, the platform prioritizes ease of navigation and efficiency, allowing customers to seamlessly browse a curated catalog of products, add items to a digital cart, complete transactions via trusted payment gateways, and monitor their orders in real time.



## 2.2. Product Functionality

The Buyzu provides the following core functionalities to enable a seamless user experience:

**1.User Signup & Login:**

New users can create accounts via email or social authentication (e.g., Google).

Registered users securely log in to access personalized features.

**2.Product Discovery:**

Users can search for products by keywords, categories, or filters (e.g., price range).

A recommendation system suggests items based on browsing history and preferences.

**3.Cart Management:**

Users can add/remove items to/from a shopping cart, adjust quantities, and view real-time totals.

**4.Checkout & Payment:**

A streamlined checkout process allows users to enter shipping details and select payment methods (e.g., credit card).

**5.Order Tracking:**

Users can view order status (e.g., "Processing," "Shipped") and track deliveries through integrated logistics APIs.

**6.Product Reviews & Ratings:**

Post-purchase, users can submit reviews and ratings to share feedback on products.

These functionalities collectively support the end-to-end shopping journey, from account creation to post-purchase engagement, while prioritizing usability and security.

**7.Admin Service [New]:**

Admin Service provides centralized management of users, products, orders, shipments, and system settings via secure APIs, enforcing role-based access control and comprehensive audit logging.

## 2.3. Design and Implementation Constraints

**Technical Constraints**：

The system must adhere to the COMET (Collaborative Object Modeling and Evolutionary Design) method for software design. This iterative, use-case-driven approach ensures alignment between requirements and object-oriented implementation.All architectural and behavioral diagrams (e.g., class diagrams, sequence diagrams) t be designed using Unified Modeling Language (UML) to standardize system documentation.

**Technology Stack:**

Frontend: React.js (JavaScript/TypeScript) for dynamic UI components.

Backend: Node.js with Express.js framework for RESTful API development.

Database: MySQL for structured data storage, ensuring ACID compliance.

**Performance and Scalability Constraints**：

Critical user actions (e.g., search, checkout) must respond within ≤2 seconds under normal load (1,000 concurrent users).And The backend must support horizontal scaling (e.g., Docker containerization) to handle peak traffic (e.g., holiday sales).

**Operational Constraints:**

Client-side compatibility with modern browsers (Chrome, Firefox, Safari) and mobile devices (iOS/Android).

Concurrency handling (e.g., database locking) to prevent race conditions during high-demand scenarios (e.g., limited stock purchases).

**Development Standards:**

Git with GitHub for collaborative development, following GitFlow branching strategy as for version control.

## 2.4. Assumptions and Dependencies

The following assumptions are critical to the project’s design and implementation. If proven invalid, they may require significant adjustments to the system:

**Third-Party Service Availability:**

The payment gateway and logistics APIs (WeChat Pay/Alipay) will maintain stable uptime and provide consistent response formats.

**User Behavior:**

Users will primarily access the website via modern browsers (Chrome, Firefox, Safari) and mobile devices.

**Network Reliability:**

Users have stable internet connectivity during critical actions (e.g., checkout).

Data Security Compliance:

Third-party services (e.g., Stripe) comply with PCI-DSS standards for payment processing.

***The project relies on the following external components and resources:***

Libraries and Frameworks:

React.js: Frontend UI development.

Node.js: Backend API development.

MySQL: Database management.

Cloud Infrastructure:

AWS EC2: Initial deployment of the backend server.

Open-Source Tools:

Jest/Mocha: Unit testing frameworks.

Git/GitHub: Version control and collaboration.

# 3. Specific Requirements

## 3.1 External Interface Requirements

#### 3.2.1. User Interfaces

The user interface for the e-commerce platform will be designed to provide an intuitive and seamless shopping experience. The following are the six main interfaces that Buyzu provides to users (Due to technical and time issues, the pictures are simple web pages, and the content displayed may be slightly different from the description):

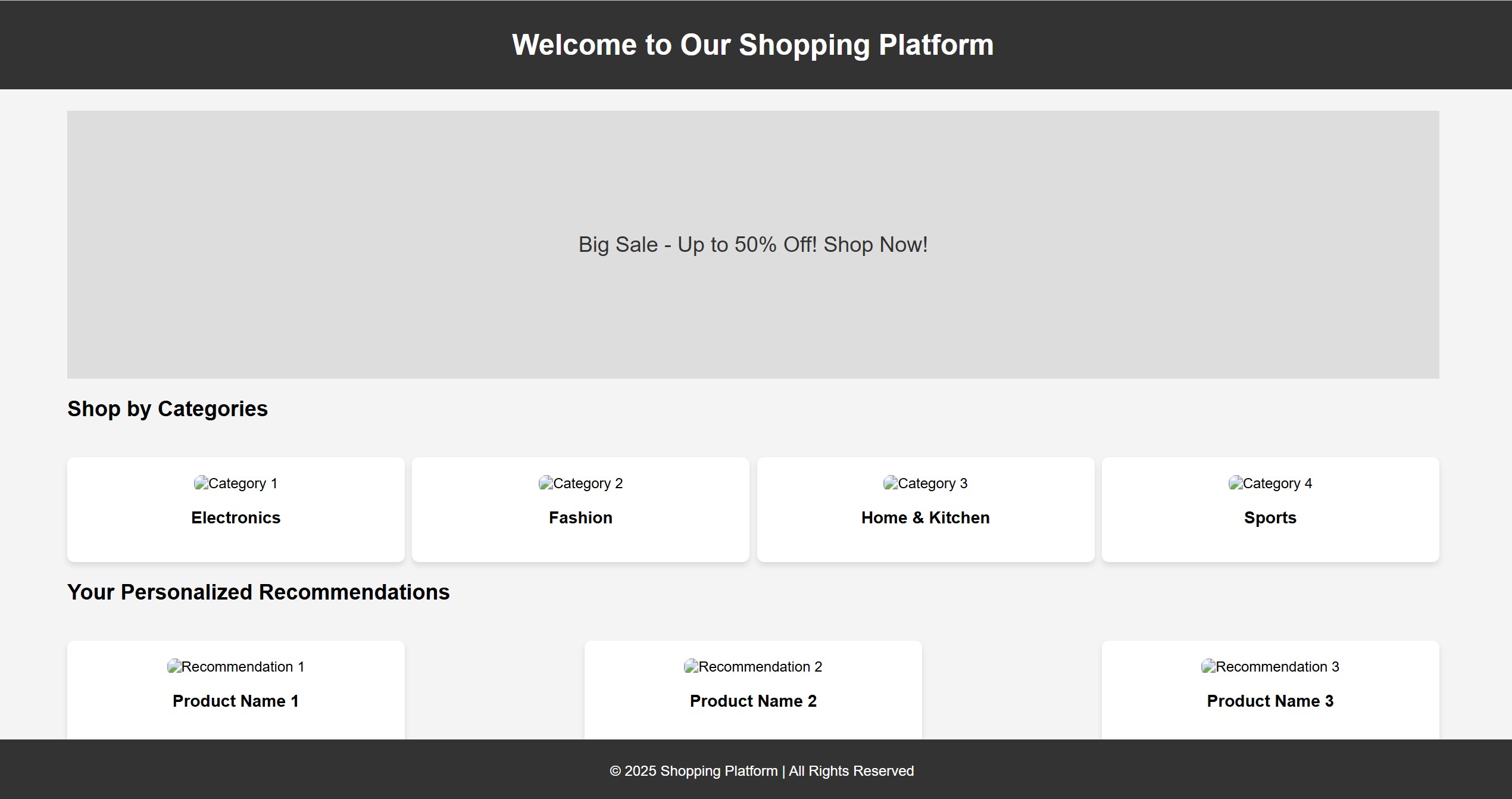
1. **Homepage** 
   * **Purpose**: Primary entry point for users to discover products and promotions.
   * **Components**:

o **Promotional Banner**:

* + - * + Displays dynamic sales campaigns (e.g., "Big Sale - Up to 50% Off! Shop Now!").
        + Rotating banners managed via the admin dashboard.

o **Product Categories**:

* + - * + Grid layout showing categories (e.g., *Electronics*, *Fashion*, *Home & Kitchen*, *Sports*).
        + Clickable icons/images redirecting to category-specific product listings. o **Personalized Recommendations**:
        + Algorithm-driven product suggestions based on user browsing history (e.g., *Recommendation 1*, *Recommendation 2*).
        + Displayed as a carousel with product images, names, and prices.



1. **Product Search & Filter Page** 
   * **Purpose**: Enables users to search and refine product selections.
   * **Components**:

o **Search Bar**:

* + - * + Free-text input field with autocomplete suggestions.
        + Example query: "wireless headphones".

o **Filters**:

* + - * + **Price Range**: Slider or dropdown (e.g., 0−50, 50−100).
        + **Brand**: Checkbox list of available brands.
        + **Rating**: Star-based filtering (e.g., "4 Stars and above").

o **Sorting Options**:

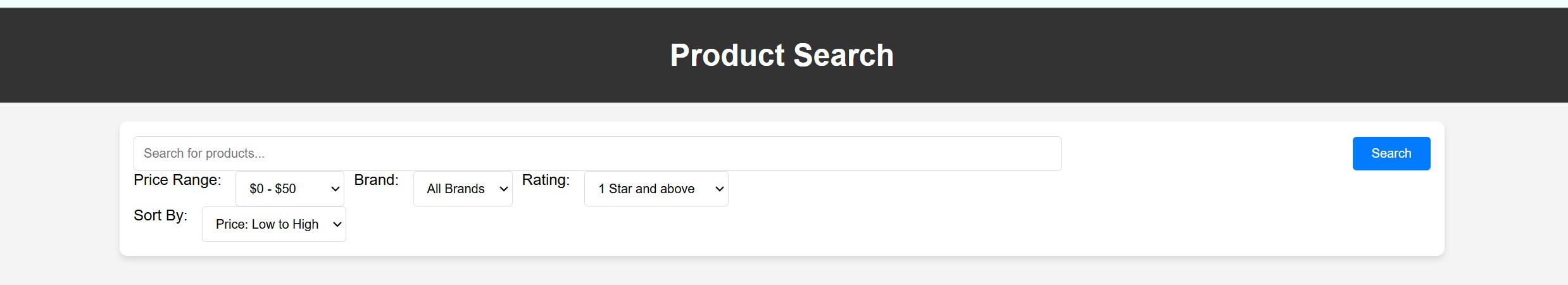
▪ Dropdown menu for sorting by price (Low to High), relevance, or popularity.

1. **Product Details Page** 
   * **Purpose**: Displays comprehensive product information and facilitates purchase decisions.
   * **Components**:

o **Product Images**:

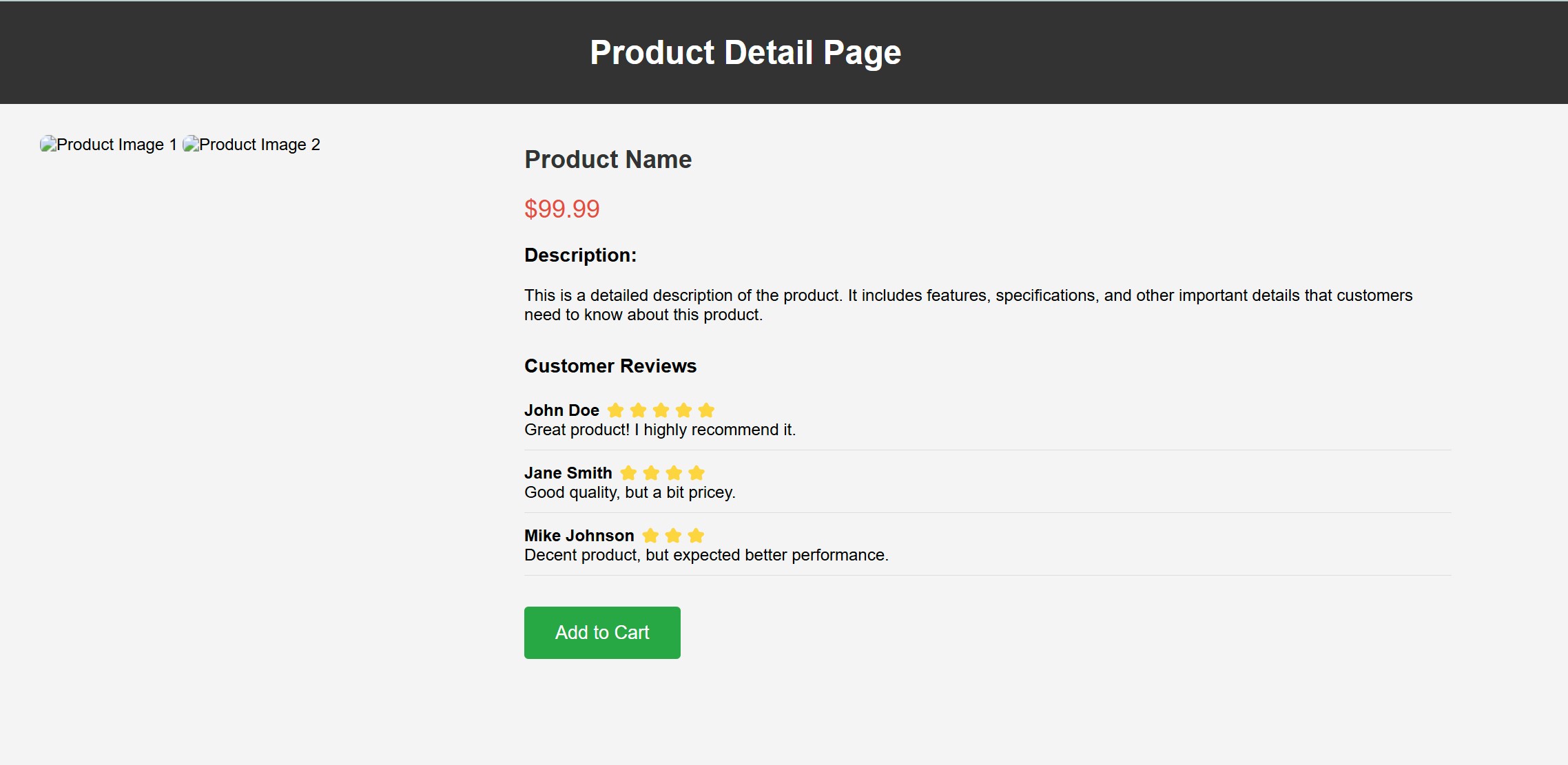
▪ High-resolution gallery with zoom functionality (e.g., *Product Image 1*, *Product Image 2*).

o **Description Section**:

▪ Detailed specifications (e.g., dimensions, materials) and feature highlights.

o **Customer Reviews**:

* + - * + Star ratings and user comments (e.g., "Great product! I highly recommend it.").
        + Option to sort reviews by date or helpfulness. o **Action Buttons**:
        + **Add to Cart**: Adds item to the shopping cart.
        + **Buy Now**: Direct checkout option.



1. **Shopping Cart Page** 
   * **Purpose**: Manages selected items before checkout.
   * **Components**:

o **Cart Items Table**:

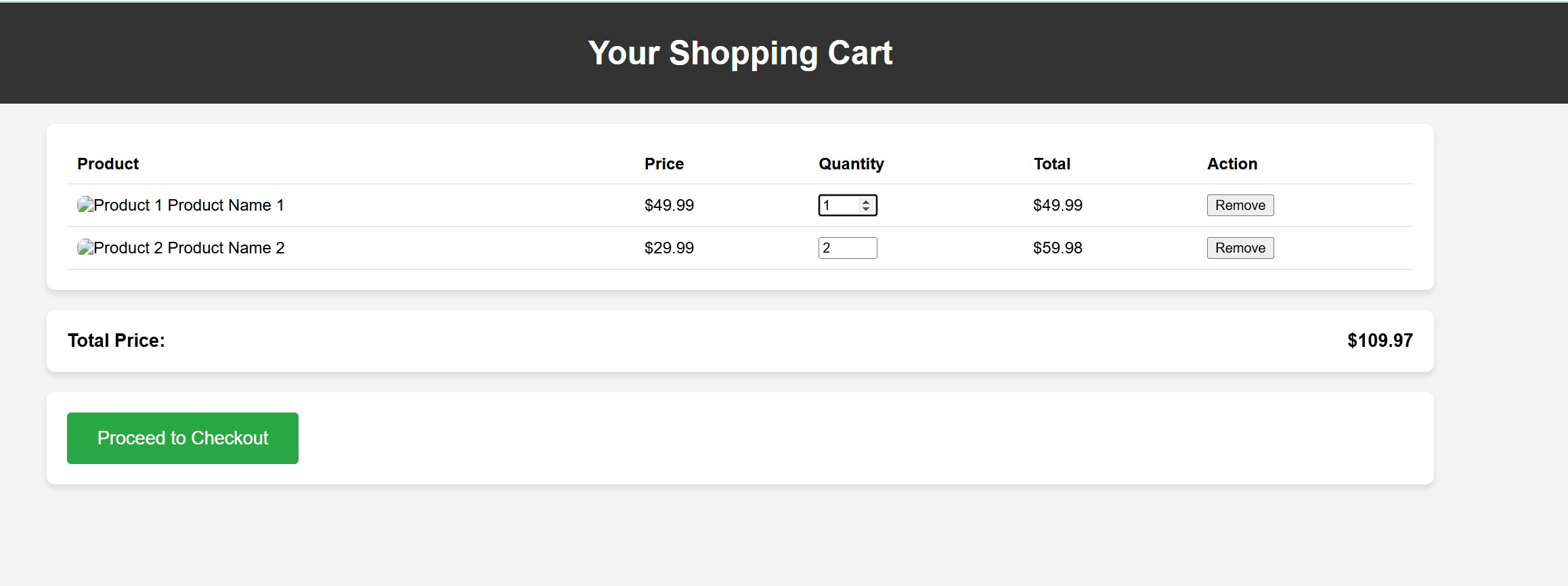
* + - * + Lists product names, prices, quantities, and total per item.
        + Example row: *Product Name 1 | 49.99**1*49.991*49.99 | Remove*.

o **Total Price Summary**:

▪ Displays subtotal, taxes (if applicable), and grand total.

o **Checkout Actions**:

* + - * + **Proceed to Checkout**: Redirects to payment and shipping page.
        + **Continue Shopping**: Returns to previous browsing page.



1. **Checkout/Payment Page** 
   * **Purpose**: Collects shipping details and processes payments.
   * **Components**:

o **Shipping Details Form**:

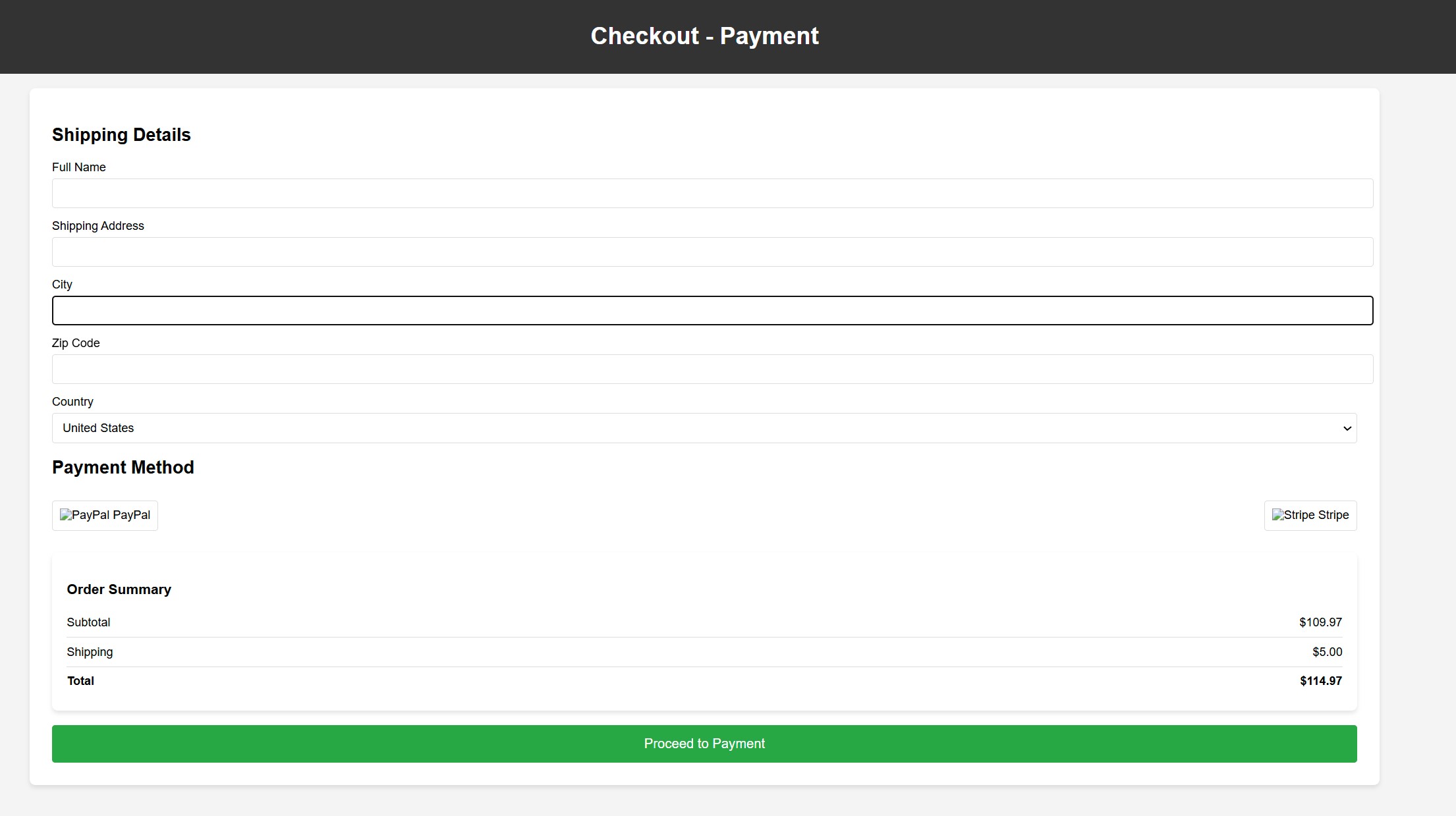
* + - * + Fields for *Full Name*, *Shipping Address*, *City*, *Zip Code*, and *Country*.
        + Example: *123 Main St, Springfield, IL, 62701, United States*. o **Payment Method Selection**:
        + Options for *PayPal*, *Credit Card*, and other third-party gateways.

o **Order Summary**:

* + - * + Displays subtotal, shipping fees, and total amount.
        + Example: *Subtotal: 100.97**Shipping:*100.97*Shipping*:*5.00 | Total: $114.97*.

o **Proceed to Payment Button**:

▪ Redirects to the selected payment gateway for transaction completion.



1. **User Account Dashboard** 
   * **Purpose**: Central hub for managing user-related data.
   * **Components**:

o **Orders Section**:

* + - * + Displays order history with *Order ID*, *Date*, *Status*, and *Total*.
        + Example: *Order #12345 | 2025-02-01 | Shipped | $99.59*.

o **Addresses Section**:

* + - * + Lists saved shipping addresses with options to edit or delete.
        + Example: *123 Main St, Springfield, IL*.

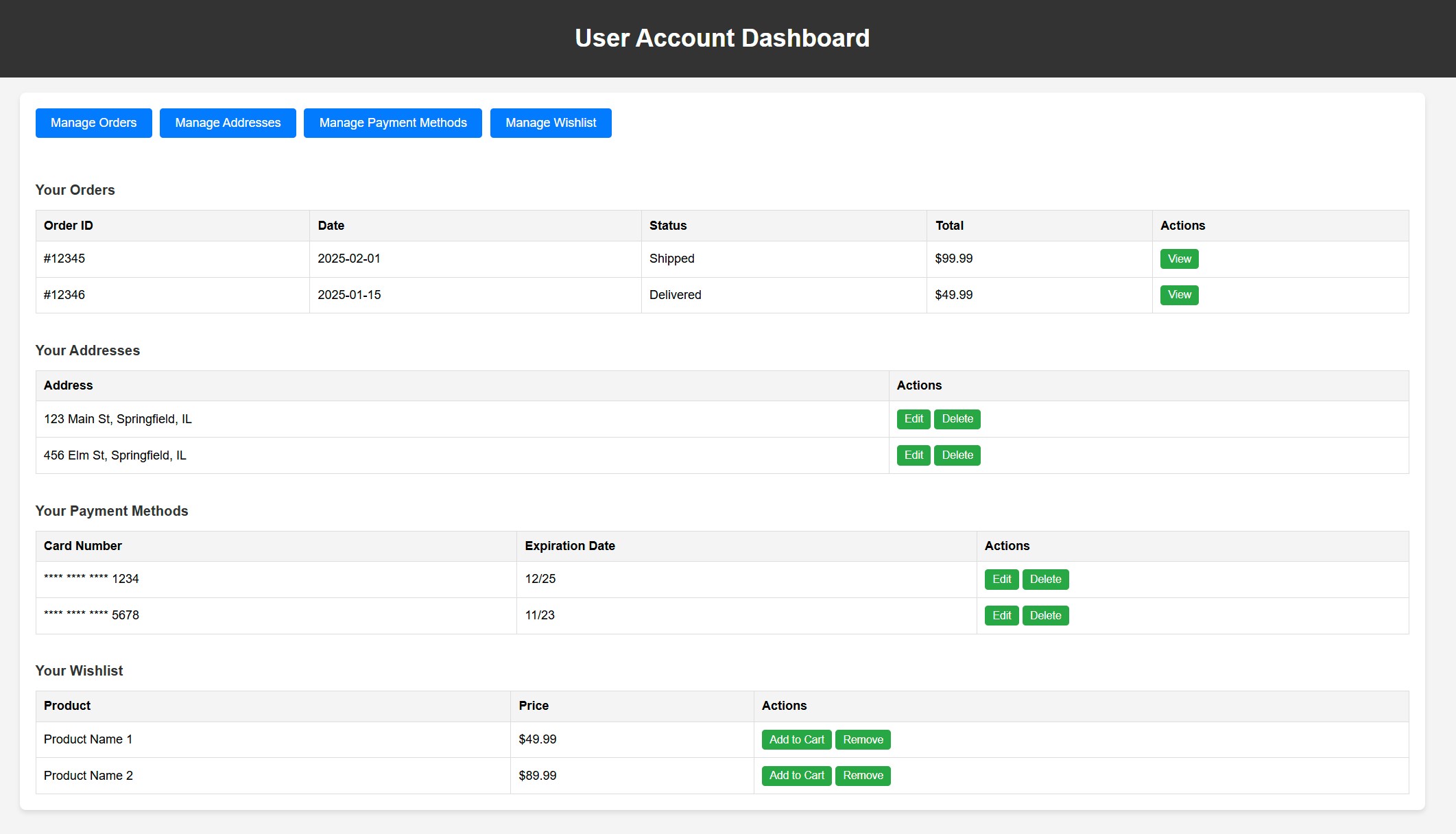
o **Payment Methods Section**:

* + - * + Displays saved payment cards with *Card Number* and *Expiration Date*.
        + Example: *Card ending in 1234 | Expires 12/25*.

o **Wishlist Section**:

* + - * + Lists saved products with *Product Name*, *Price*, and actions (*Add to Cart* or *Remove*).
        + Example: *Product Name 1 | $49.99 | Add to Cart | Remove*.

**Screenshot**:



1. **Admin Dashboard [New]**
   * **Purpose**: Central hub for managing users, products, orders, reviews, and system metrics.
   * **Components**:

o **User Management Section**:

* + - * + Overview and controls for user accounts and roles.

o **Product Management Section**:

* + - * + Create, update, delete, and view product catalog.

o **Order Management Section**:

▪ Track and modify order statuses and details.

o **Review Management Section**:

▪Approve, reject, or edit customer product reviews.

**Screenshot**:

一張含有 文字, 螢幕擷取畫面 的圖片

AI 產生的內容可能不正確。

**Interaction Flow**

* 1. Users browse promotions and categories on the Homepage.
  2. Search for specific products using keywords and filters.
  3. View product details and add items to the cart. 4. Review cart contents and proceed to checkout.
  4. Enter shipping details and select a payment method.
  5. Manage orders, addresses, and wish lists via the User Account Dashboard.

**Technical Notes**

* + **Responsive Design**:
    - * All interfaces adapt to screen sizes (desktop, tablet, mobile).
      * Mobile version uses collapsible menus and touch-friendly buttons.
  + **Accessibility**:
    - * Alt text for images, ARIA labels for screen readers.
      * Keyboard navigation support.

**3.2.2. Hardware Interfaces**

The e-commerce platform will interact with the following hardware components to ensure seamless operations and data synchronization. Detailed requirements are listed below: **1. Servers**

* **Web/Application Servers**:
  + **Protocol**: HTTP/HTTPS (RESTful APIs) for frontend-backend communication.
  + **Data Format**: JSON payloads with UTF-8 encoding.
  + **Load Balancing**: Deploy redundant servers with NGINX load balancers to handle high traffic (e.g., during promotions).
  + **Fault Tolerance**: Automatic failover to backup servers if primary servers exceed 90% CPU/RAM usage.
* **Database Servers**:
  + **Storage**: Use MySQL clusters for transactional data (orders, payments) and MongoDB for unstructured data (user logs, product reviews).
  + **Replication**: Real-time master-slave replication to ensure data consistency.
  + **Backup**: Daily encrypted backups stored on AWS S3. **2. Payment Terminals**
* **POS Systems**:
  + **Integration**: Support QR code scanning via USB/Bluetooth-enabled devices (e.g., Honeywell scanners).
  + **Communication**: JSON-based API endpoints for transaction confirmation and refunds.
  + **Security**: End-to-end TLS 1.3 encryption for payment data transmission.
* **Mobile Payment Devices**:
  + **Compatibility**: Ensure compatibility with Android/iOS-based POS systems (e.g., Alipay Box, WeChat Pay terminals).
  + **Offline Mode**: Cache transactions locally if network connectivity is lost, with auto-sync upon reconnection.

**3. Logistics Devices**

• **Barcode Scanners**:

* **Protocol**: Support USB, Bluetooth, and Wi-Fi connectivity.
* **Data Format**: Scanned barcode data (EAN-13 or UPC-A format) transmitted as plain text via MQTT protocol.
* **Error Handling**: Retry failed scans up to 3 times, then trigger an alert to warehouse staff.

**4. User-End Devices**

• **Mobile/Desktop Clients**:

o **Resolution Support**: Responsive UI for screen resolutions ranging from 320px (mobile) to 1920px (desktop). o **Hardware Sensors**:

▪ Utilize device cameras for QR code scanning (via WebRTC API).

**3.2.3. Software Interfaces**

The platform integrates with the following external systems:

1. **Payment Gateways**:

**Alipay/WeChat Pay**: HTTPS-based API integration for secure transaction processing.

Requirements:

SSL/TLS encryption for data in transit.

Support for refunds and order status synchronization.

1. **Third-Party Login**:

OAuth 2.0 authentication for social media logins (WeChat, QQ).

1. **Logistics APIs**:

Integration with SF Express and ZTO APIs to fetch shipping statuses and generate tracking numbers.

**Security Constraints**:

* All external APIs must comply with PCI-DSS standards for payment processing.
* Rate limiting and token-based authentication for API access.

## 3.2. Functional Requirements

**3.2.1. User Signup:**

The system shall allow users to register for an account using either (1) valid email address (2) SSO. When using the first method of registration, the system will check if the email domain name exists initially. If it is already in use, a reminder will be pushed to direct the user to the login page. If it is not in use, a verification email will be sent to the email address. The user will need to check the verification code and enter it into the verification box provided by the system to complete the email address verification. The system will then ask the user to enter a unique username and strong password to complete the registration process. For the second method of registration, the system should support service providers that offer authentication options (e.g. Google, Facebook). Once the user has selected one of the specific service providers, the system shall establish a secure session (e.g. using OAuth 2.0 as mentioned in 3.1.3 above) and redirect the user to its interface for authentication. After authentication is complete, the system will create a pass token and account information based on the returned external information.

**3.2.2. User Login:**

The system shall provide two login methods corresponding to the registration, namely username login and SSO login. When using username login, the system needs to verify the existence of the corresponding record in the encrypted database based on the unique username and high strength password provided by the user. If the record does not exist, the user is prompted to re-enter or reset the account. When resetting an account, the user is asked to enter a tied email address. If the email address exists in the system's encrypted database, a reset email is automatically sent to the email address. The reset email contains a unique username and temporary login password. After obtaining the above information and re-entering the temporary login information, the system detects and automatically jumps to the password reset screen, asking the user to re-enter the high strength password and confirm. If the email address does not exist, the system displays an error message and asks the user to re-enter the email address. When using SSO login, the system shall redirect the user to the authentication page provided by the identity provider. The data exchange during this process shall be within a secure session. If SSO authentication fails, the system should require the user to reauthenticate or log in using another method.

**3.2.3. Product Discovery:**

The system should have the function of recommending the corresponding products according to the user's browsing records and purchasing habits. This function shall be realised by deep learning algorithms and neural networks included in the system. When the user has a clear need, the system should allow the user to conduct fuzzy and associative searches using product keywords through the search box embedded at the top of the page. Once the user has identified a product subcategory, the system shall jump to a search page that displays the product in question. The system should provide the user with multiple filters for additional criteria (e.g., price range, positive reviews first, sales first, newest on shelf, etc.). When the user clicks on a link to a specific product, the system shall redirect the user to the corresponding product detail page, where information about the merchant offering the product, delivery time information, real user reviews, and more product descriptions are displayed.

**3.2.4. Cart Management:**

The system should allow users to add products to the shopping cart from the product details page. On the shopping cart page, the system shall support users to modify product options and delete products. The system shall support the user to partially or fully select the products in the shopping cart for checkout and display the dynamic total price of the selected products.

**3.2.5. Checkout & Payment:**

The system shall allow the user to check out the products selected from the shopping cart interface on a dedicated checkout page. The system shall allow the user to select a shipping address and display the available logistics methods and shipping rates on the checkout page. The system shall provide the user with different payment methods and submit payment information to a third-party payment portal via a secure channel and obtain payment results.

**3.2.6. Order Tracking:**

The system should be integrated with a logistics API to capture real-time tracking information of shipped orders. The system should categorise shipped orders into different stages (e.g. processing, shipping, completed) and update them in real time via the logistics API.

**3.2.7. Product Reviews & Ratings:**

When the user confirms receipt of the goods, the system should provide the user with an entry point for feedback and complaints. The system should allow users to rate the quality of the product, the seller's service, and the speed of logistics, and allow users to choose whether to publicly display this review.

**3.2.8. Admin Service [New]:**

When submitting user feedback or complaints after receiving an order, the system should provide administrators with a centralized interface for making changes and resolving issues. Administrators can view and filter entries by product, customer, rating category (product quality, seller service, logistics speed), or complaint type; approve, reject, or edit review content; toggle the public visibility of each review; respond to customers directly from within the dashboard; perform mass modifications; and export summary reports of rating and complaint trends to drive continuous service improvement.

## 3.3. Use Case Model

*TO DO: Provide a use case diagram that will encapsulate the entire system and all actors.*

#### 3.3.1. Use Case #1 (use case name and unique identifier – e.g. U1)

*TO DO: Provide a specification for each use case diagram*

**Author –** MA, Heyang 1155191433

**Purpose** – User Login

**Requirements Traceability –** Traces to Requirement 3.4.2 (User Login Requirements) **Priority** – High

**Preconditions** – (1) The user has previously registered via email or through SSO. (2) The encrypted database contains the necessary account records and tied email information. (3) The user has set up a secure connection to the Internet.

**Post conditions** – (1) A valid user session is established upon successful login. (2) An authentication token is generated and associated with the session. (3) The user is redirected to the appropriate dashboard or landing page.

**Actors** – **User:** Initiates the login process. **System:** Verifies credentials and handles session management. **SSO Providers (e.g., Google, Facebook):** Authenticate the user during SSO login.

**Extends –** Password Reset Use Case (handles scenarios where credentials are invalid or password reset is invoked).

**Flow of Events**

1. Basic Flow

**Username Login:**

1. The user selects the username login method.
2. The user enters the unique username and high-strength password.
3. The system checks the encrypted database for the corresponding record.
4. If the record is found and credentials match, the user is successfully logged in, and a secure session is established.

**SSO Login:**

1. The user selects SSO as the login method.
2. The system redirects the user to the identity provider’s authentication page, establishing a secure session (e.g., using OAuth 2.0).
3. The identity provider authenticates the user and returns the required external account information.
4. The system generates a pass token, creates the corresponding account information (if necessary), and establishes a user session.
5. Alternative Flow - a secondary flow of events due to infrequent conditions

**Username Login – Invalid Credentials:**

* 1. If the system does not find a matching record, it prompts the user to re-enter the username and password or choose account reset.
  2. If the user selects account reset, the system requests the tied email address.
  3. The system verifies the email address in the encrypted database.
  4. If found, a reset email with the unique username and temporary login password is sent.
  5. The user enters the temporary credentials, and the system redirects them to the password reset screen where a new high-strength password is set.

**SSO Login – Authentication Failure:**

* 1. If SSO authentication fails (e.g., due to incorrect credentials or network issues), the system displays an error message.
  2. The user is prompted to re-authenticate or choose an alternative login method.

1. Exceptions - Exceptions that may happen during the execution of the use case **Username Login:**

If repeated invalid credential attempts occur, the system may trigger security measures (such as temporary account lockout).

If the tied email is not found during the reset process, an error message is displayed, prompting the user to re-enter the email correctly.

**SSO Login:**

Network failures or issues with the identity provider may disrupt SSO authentication, requiring the user to retry.

**Includes** (other use case IDs)

Password Reset Use Case

• Session Management and Token Generation Use Case

**Notes/Issues** - Any relevant notes or issues that need to be resolved

一張含有 文字, 圖表, 行, 字型 的圖片

AI 產生的內容可能不正確。

# 4. Other Non-functional Requirements

## 4.1. Performance Requirements

**4.1.1. Response Time:**

(1). Under standard network conditions, the system should be able to process user authentication requests in 2 seconds or less under normal load (around 1,000 concurrent users), and in 5 seconds or less under peak load (around 10,000 concurrent users), within 99% of the time.

(2). Under normal load and network conditions, the response time for critical functions should not exceed 2 seconds in 99% of the time.

*Rationale*: Ensure a seamless user experience in different circumstances, even during a high internet traffic period.

**4.1.2. Throughput:**

The system should be able to handle 10,000 transactions per minute with a success rate of 99%, including retries after a failed attempt.

*Rationale*: Support high-volume data use of platform (for sales events or promotion) without the degradation of performance.

**4.1.3. Scalability:**

(1). The system should support an extension of a certain function or modules easily without requiring full reinstallation.

(2). The system should automatically adjust compute resources (e.g., AWS instances) to handle 200% traffic surges within 20 minutes, without the degradation of performance or manual intervention.

*Rationale*: Ensure elasticity for any feature expansion, rapid business growth or unpredicted demand spikes without redesign or reinstallation.

## 4.2. Safety and Security Requirements

**4.2.1. Access Control:**

(1). The system should use MFA for all user logins and RBAC to secure user permission.

(2). Five consecutive failed login attempts to the system should trigger account lockouts.

(3). Every unauthorized request to a resource, login failures and administrative actions in the system should be recorded and stored for audit over the next two years.

(4). The system should require double authorization, such as email verification or dual approval (e.g., OAuth 2.0 workflow), when a user is performing a sensitive operation (e.g., making a payment).

*Rationale*: Prevent potential threats and unauthorized access to user or admin account.

**4.2.2. Data Encryption:**

(1). All Personally Identifiable Information and user data should be encrypted with AES-256 (TLS

1.3+ in transit, encrypted databases at rest), and data backups must be stored in GRS with 7-day retention.

(2). Daily data backups should be encrypted and stored in AWS S3 with 30-day retention

(3). In the payment section, a dedicated firewalls should be used to safeguard the data transmit between user and a third-party paying software.

*Rationale*: Ensure the data security as well as compile with the GDPR and PCI DSS regulation.

## 4.3. Software Quality Attributes

**4.3.1. Reliability:**

(1). The system should achieve a minimum of 99.9% monthly uptime (excluding scheduled maintenance) according to SLAs regulations, with automated failovermechanisms to switch system to standby servers within 15 seconds once an error occurs.

(2). The scheduled maintenance windows of the system should be restricted to 2 hours per month during off-peak hours, which can be achieved by adopting a modular design for features and ensuring the continuous output of system logs to make maintenance more efficient.

**4.3.2. Adaptability:**

(1). The system should support the demanded hardware or cloud-hosted servers without code modification, with a microservices architecture with containerization tools like Docker and orchestration system like Kubernetes to ensure consistent deployment across platforms.

(2). The system should provide responsive user interface (from mobile phone to 4K screen) withflexible layouts (e.g., CSS Grid or Flexbox) or media queries.

**4.3.3. Usability:**

(1). The design of user interface of system should be concise and intuitive for non-technical users

(2). The system should include some accessibility features for users with disabilities (e.g., screen reader for dyslexia or visually impaired users).

(3). A help guide to the system should be easily accessible in the main user interface.

**4.3.4. Portability**

The system should support high efficiency data migration by storing and exporting data in open formats (e.g., CSV, JSON)

# Appendix A – Data Dictionary (modified)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Data Type** | **Description** | **Format/Constraint**  **s** | **Related**  **Operations** |
| UserID | INTEGER | Unique  identifier for the user  (Primary Key) | Auto-increment, cannot be modified | User Signup,  Login, Order  Tracking |
| Username | VARCHAR(50) | Unique username chosen by the user during registration | 4-50 characters, only letters, numbers, and underscores allowed | User Signup, Login |
| PasswordHash | BINARY(60) | Encrypted storage of user password | Encrypted using bcrypt, minimum length of 8 characters, must include uppercase, lowercase, and numbers | User Signup,  Login,  Password  Reset |
| Email | VARCHAR(255  ) | User’s verified email address | Must follow RFC 5322 standard and pass email verification process | User Signup, Notifications |
| AuthToken | VARCHAR(255  ) | Authenticatio n token for user sessions | JWT format, valid for 24 hours | User Login, API Access |
| ProductID | CHAR(12) | Unique  identifier for a product | 12-character alphanumeric string, follows EAN-13 standard | Product  Search, Cart  Management |
| ProductPrice | DECIMAL(10,2  ) | Current price of the product | ≥0.01, two decimal places | Product  Display,  Checkout |
| CartID | INTEGER | Unique  identifier for the shopping  cart | Linked to UserID, each user has only one active cart | Cart  Management |
| OrderStatus | ENUM | Current status of an order | Possible values:  Processing /  Shipped / Delivered  / Cancelled | Order  Tracking |
| PaymentTransactionI D | VARCHAR(36) | Transaction ID returned by the payment gateway | UUID format | Checkout & Payment |
| ShippingAddress | TEXT | Full shipping address provided by the user | Must include country, postal code, and street  address | Checkout,  Order  Tracking |
| ProductRating | TINYINT | User rating for a product | Integer between 1 and 5 | Product  Reviews &  Ratings |
| SessionID | CHAR(32) | User session ID | MD5 hash, 32character hexadecimal value | All  Authenticate d Operations |
| Roles | VARCHAR(100) | Comma-separated list of admin roles (e.g. “superadmin,product\_mgr”); at least one required | VARCHAR(100), NOT NULL | On admin create/update: validate presence and correctness of each role token |
| CreatedAt | DATETIME | Default CURRENT\_TIMESTAMP | DATETIME, NOT NULL | INSERT: database auto-populates this field  SELECT: read for audit or reporting |
| UpdatedAt | DATETIME | Default CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP | DATETIME, NOT NULL | UPDATE: database automatically updates this field  SELECT: read to determine last modification time |
| LastLogin | DATETIME | Nullable; timestamp of last successful login | DATETIME, NULLABLE | On successful authentication: UPDATE to NOW()  SELECT: display or audit user’s most recent login |
| FailedLoginCount | INTEGER | Default 0; increments on each failed login attempt | INTEGER, NOT NULL, Default 0 | On authentication failure: increment by 1  On successful login: reset to 0  During login checks: read to enforce lockout or throttling policies |

# Appendix B - Group Log

**Project Title:** Buyzu

**Date Range:** 2025-1-18 to 2025-2-10

**Meeting 1**

* **Date:** 2025-1-18
* **Time:** 5:30 pm to 7:30 pm
* **Location:** Online
* **Attendees:** Guo Menglong, MA Heyang, Xue Guangxuan, YAN Yitao, ZHU Keyu
* **Agenda:**

1. Project scope discussion
2. Task assignment
3. Timeline planning

* **Key Decisions:**
* Defined project objectives
* Assigned roles to members
* Set deadlines for initial tasks

**Meeting 2**

* **Date:** 2025-2-8
* **Time:** 7:30 pm to 9:30 pm
* **Location:** Online
* **Attendees:** Guo Menglong, MA Heyang, Xue Guangxuan, YAN Yitao, ZHU Keyu
* **Agenda:**

1. assigned workload of SC2
2. Defined project name and content
3. Revising project timeline

**Meeting 3**

* **Date:** 2025-2-10
* **Time:** 5:30 pm to 7:30 pm
* **Location:** University Library
* **Attendees:** Guo Menglong, MA Heyang, Xue Guangxuan, YAN Yitao, ZHU Keyu
* **Agenda:**

1. Worked together to finish SC2
2. Fixed problems and adjust scope