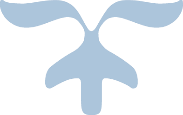


Online Shopping Website

CSCI 313 – Software engineering



Arwa Sameh 211001892

Seif Hussein 221000166

Table of Contents

[Introduction 2](#_bookmark0)

[Scope: 2](#_bookmark1)

[Objective 2](#_bookmark2)

[Technologies used 2](#_bookmark3)

[Overall description 3](#_bookmark4)

[Product perspective 3](#_bookmark5)

[User Characteristics 3](#_bookmark6)

[Constraints: 4](#_bookmark7)

[Functional Requirements 4](#_bookmark8)

[Non-Functional Requirements 5](#_bookmark9)

[Diagrams 6](#_bookmark10)

[Class diagram 6](#_bookmark11)

[Use case Diagram 8](#_bookmark12)

[Sequence diagrams: 11](#_bookmark13)

[State Diagram 12](#_bookmark14)

[Activity Diagram 13](#_bookmark15)

[Interface requirements 14](#_bookmark16)

[System interface 14](#_bookmark17)

[Software Requirements: 19](#_bookmark18)

[Hardware Requirements: 19](#_bookmark19)

[Testing 20](#_bookmark20)

[Functionality testing: 20](#_bookmark21)

[Test Case 1: User Login 20](#_bookmark22)

[Test Case 2: User Signup 21](#_bookmark23)

[Test Case 3: Add to Cart 23](#_bookmark24)

[Test Case 4: View Cart 24](#_bookmark25)

[Test Case 5: Admin Login 25](#_bookmark26)

[Test Case 6: Add Product 26](#_bookmark27)

[Test Case 7: Add User 27](#_bookmark28)

[Test Case 8: View Orders 28](#_bookmark29)

[Conclusion 29](#_bookmark30)

# Introduction

## Scope:

The scope of this project is to develop an online shopping system that provides a comprehensive e-commerce experience for both customers and administrators. The system will allow customers to register, browse products, add items to their cart, place orders, and simulate payments via Cash on Delivery or a dummy card payment method. Administrators will have the ability to manage the product catalog, track orders, and monitor customer interactions. The system will feature secure user authentication, an intuitive interface for browsing and managing products, and a scalable database. While the payment process will be simulated, the project aims to provide a functional, user-friendly platform.

## Objective:

* To create a functional and user-friendly online shopping platform.
* To implement a customer interface that includes registration, login, product browsing, cart management, and order placement.
* To develop an admin interface for managing products, orders, and customers.
* To simulate a payment process using a dummy payment system (Cash on Delivery and Card Payment).

## Technologies used

#### Frontend:

* HTML/CSS: Used for structuring and styling the website.
* Django Templates: Renders dynamic content in HTML.

#### Backend:

* Django: Framework for building and managing the web application.
* Python: Language used for backend logic and server-side functionality.

#### Development Tools:

* VS Code: IDE for coding and debugging.
* GitHub: Platform for version control and collaboration.

# Overall description

## Product perspective:

The Online Shopping System is designed as a small-scale e-commerce platform that allows customers to shop online and administrators to manage the system efficiently. It serves as an independent web application and does not rely on third- party systems for its core functionalities. The system uses Django as the primary backend framework to handle logic, data management, and user authentication, while HTML, CSS are used for the front-end interface.

## User Characteristics:

#### Customers:

* Primary users of the system, engaging in shopping activities.
* Basic internet skills, such as navigating websites and filling out forms.
* Perform tasks like browsing products, adding items to the cart, placing orders, and making payments.
* Expect a simple, intuitive, and error-free interface.

#### Administrators:

* Users with elevated privileges for managing backend operations.
* Perform tasks like adding, updating, and removing products, monitoring orders, and updating order statuses.
* Depends on the dashboard for monitoring system performance and activity.

## Constraints:

* ***Technical Constraints:*** The system will use Django for backend development and HTML, CSS for the frontend. Dummy payment methods (Cash on Delivery and Simulated Card Payment) will be implemented
* ***Scope Constraints:*** The project will exclude real-world payment gateway integration, advanced features like product recommendations, and third- party API integrations. Mobile responsiveness are not a key focus area.

# Functional Requirements

### For Customers:

* **Register and Log In:** Customers can create an account and securely log in to access their profiles.
* **Browse Product Catalog:** Customers can view all available products along with details such as name, price, and description.
* **Add Items to the Cart:** Customers can select products to add to their shopping cart for later checkout.
* **View and Update Cart Items:** Customers can view the cart contents, update quantities, or remove items before placing an order.
* **Place Orders:** Customers can proceed to checkout to confirm their orders.
* **Simulate Payments:** Customers can choose between cash on delivery or a dummy card payment option to complete their purchase.
* **View Order History:** Customers can view a list of their past orders along with details such as order status and payment method.
* **Track Orders:** Customers can track the status of their orders (Pending, Shipped, and Delivered).
* **Update Profile Information:** Customers can update their account details, such as name, email, and address.

### For Admins:

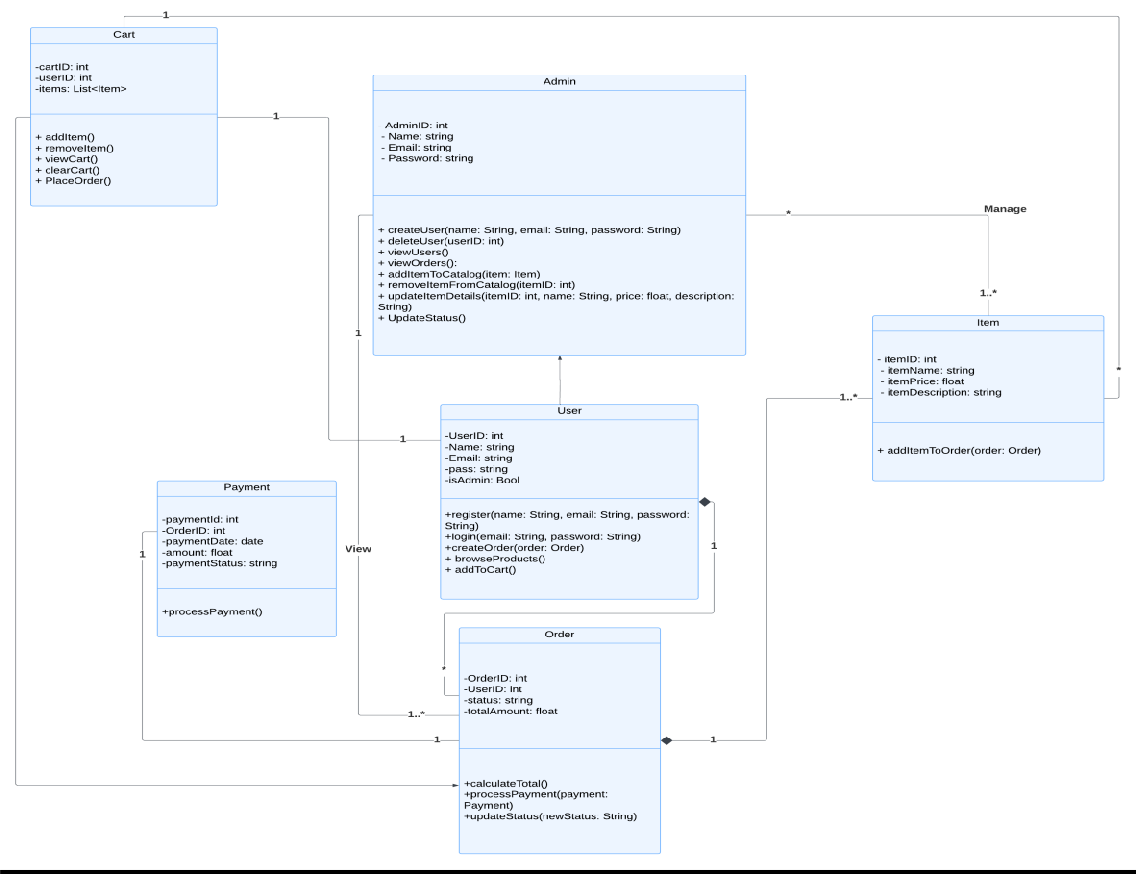
* **Log In as an Admin:** Admins can securely log in to access administrative privileges.
* **Manage the Product Catalog:** Admins can add new products, update existing product details, or delete items no longer available.
* **View and Manage Customer Orders:** Admins can view all customer orders and update their details if necessary.
* **Update Order Statuses:** Admins can change order statuses (Pending, Shipped, and Delivered) to reflect progress.
* **Manage User Accounts:** Admins can view, edit, or delete customer accounts as required.
* **View Analytics:** Admins can access basic reports, such as the number of active users and total sales.
* **Manage Inventory:** Admins can monitor stock levels and update quantities for products in the catalog.

# Non-Functional Requirements

* + Usability: The system must have an intuitive and easy-to-navigate interface.
  + Scalability: The system should accommodate a growing number of users and products.
  + Performance: Response time for user actions (browsing products, adding items to the cart) should be minimal.
  + Security: User passwords and sensitive information must be securely stored.
  + Maintainability: The system architecture should allow easy updates and feature additions.

# Diagrams

## Class diagram:

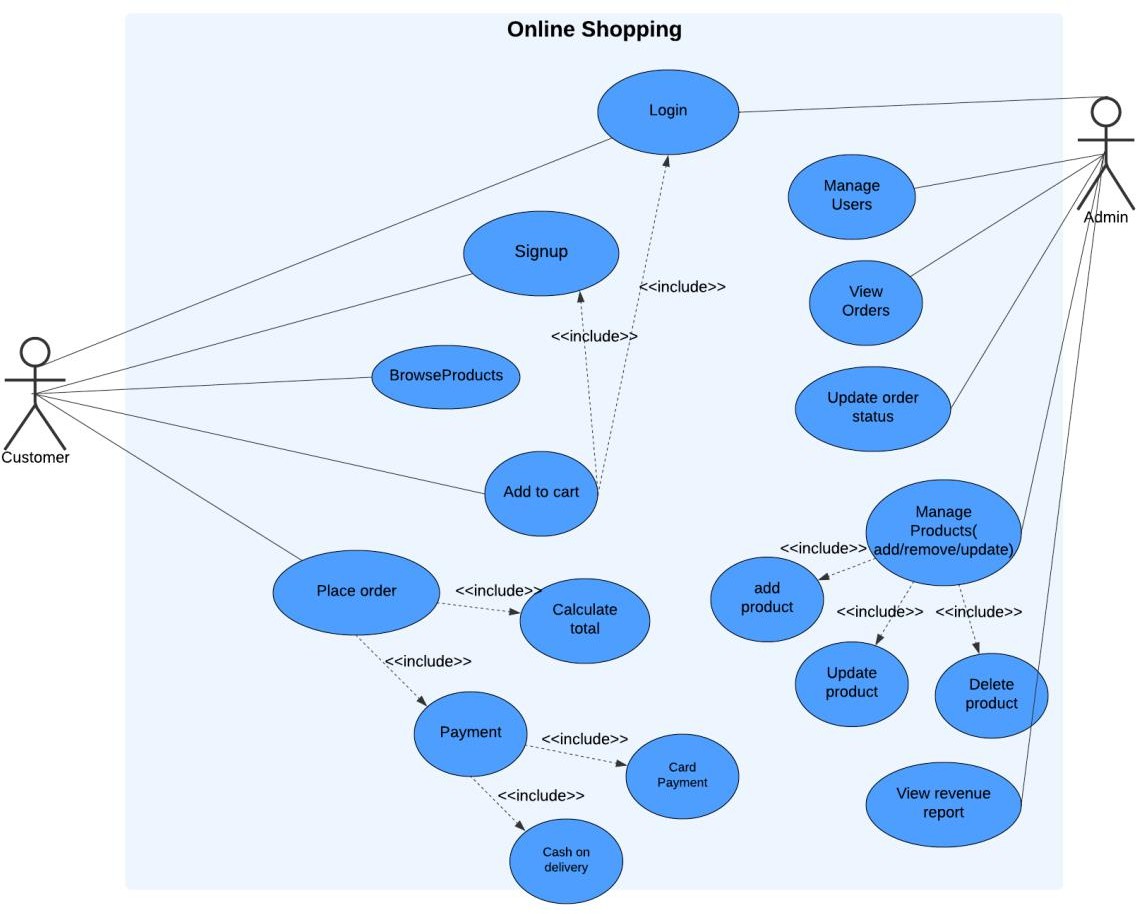
****

The class diagram represents the structure of the online shopping system, detailing the key entities, their attributes, and relationships. Below is a brief explanation of the main components:

1. **User**
   * Represents the customers using the system.
   * Attributes: UserID, Name, Email, Password, isAdmin (to distinguish between customers and admins).
   * Methods: Users can register, log in, browse products, add items to their cart, and place orders.
2. **Admin**
   * A specialized subclass of User with additional privileges.
   * Attributes: AdminID, Name, Email, Password.
   * Methods: Admins can manage users, view and update orders, and add/update/remove products from the catalog.
3. **Item**
   * Represents products available in the catalog.
   * Attributes: itemID, itemName, itemPrice, itemDescription.
   * Methods: Add items to an order, view item details, and update item attributes.
4. **Cart**
   * Temporary storage for items selected by the user before placing an order.
   * Attributes: cartID, userID, items (a list of selected products).
   * Methods: Add, remove, and clear items from the cart; place an order.
5. **Order**
   * Tracks confirmed purchases made by customers.
   * Attributes: orderID, userID, status (e.g., pending, shipped), totalAmount.
   * Methods: Calculate the total order cost, process payment, and update the order status.
6. **Payment**
   * Handles payment details and transactions for orders.
   * Attributes: paymentID, orderID, paymentDate, amount, paymentStatus.
   * Methods: Process the payment for an order.

## Use case Diagram:

The use case diagram represents the primary functionalities of the **Online Shopping System**, highlighting the interactions between the system's actors (**Customer** and **Admin**) and their associated use cases:



***Customer Use Cases:***

### Login and Signup:

* + The customer must either log in or sign up to perform actions like adding items to the cart and placing orders. These use cases ensure secure access to the system.

### Relationships:

* + - <<include>>: "Add to Cart" requires the customer to either log in or sign up first.

### Browse Products:

* + Customers can view the product catalog without logging in. This use case allows unrestricted exploration of available items.

### Add to Cart:

* + To add items to the cart, the customer must first log in or sign up.
  + Includes the ability to view or update cart items, ensuring customers can modify their selections before placing an order.

### Place Order:

* + Customers can proceed to checkout after reviewing their cart. This use case includes calculating the total price of items and proceeding to payment.

### Payment:

* + Payment options include:
    - **Cash on Delivery (COD):** Allows customers to pay upon delivery of goods.
    - **Card Payment:** Simulates a card payment process.

### Relationships:

* + - <<include>>: Both "Cash on Delivery" and "Card Payment" are included in the "Payment" use case.

### Admin Use Cases:

1. **Login as Admin:**
   * The admin must log in to access system management features.

### Manage Products:

* + Admins can add, update, or delete products in the catalog to keep the inventory current.

### Relationships:

* + - <<include>>: Managing products involves specific actions like adding, updating, or deleting products.

### Manage Users:

* + Admins can view and manage customer accounts, ensuring only valid users are present in the system.

### View Orders:

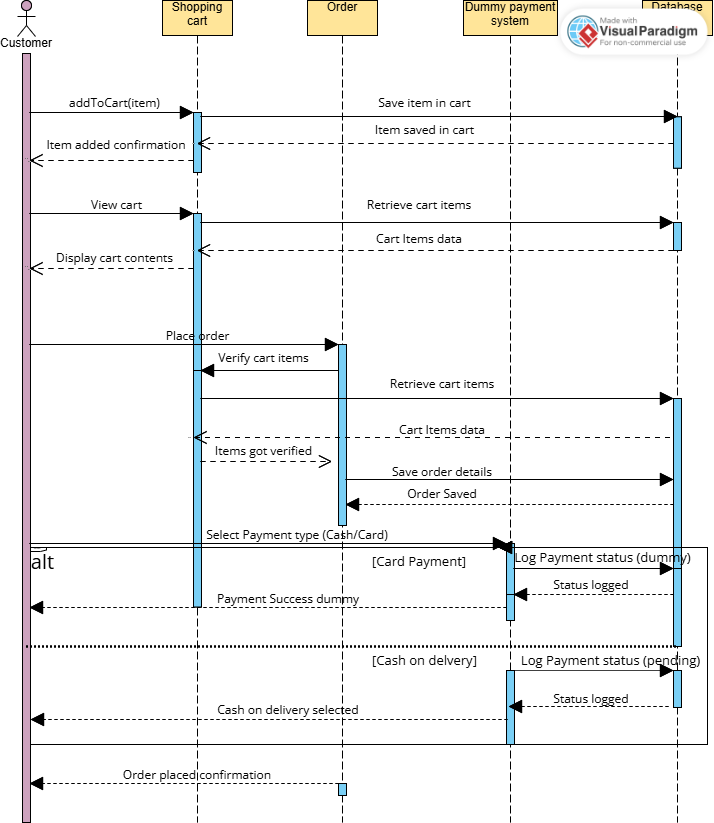
* + Admins can access all customer orders for review and processing.

### Update Order Status:

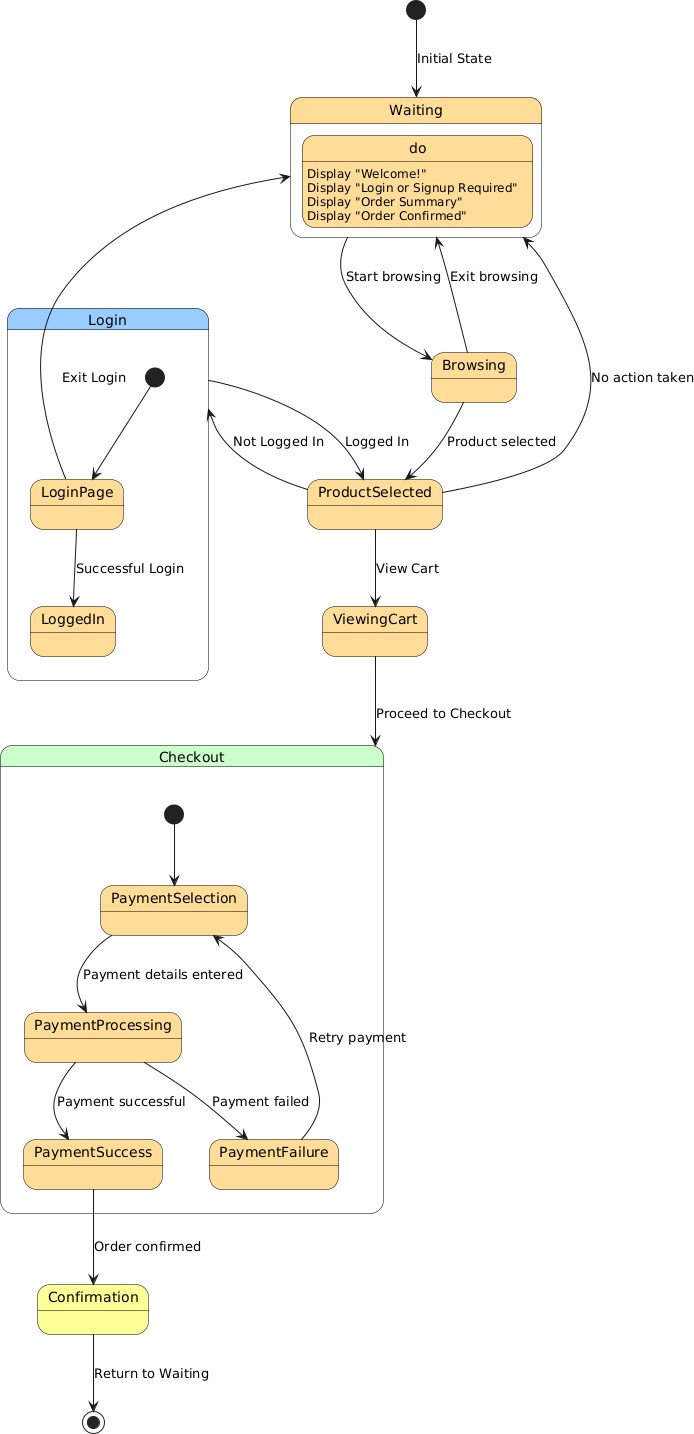
* + Admins can change the status of customer orders (e.g., pending, shipped, delivered) to keep customers informed.

## Sequence diagrams:

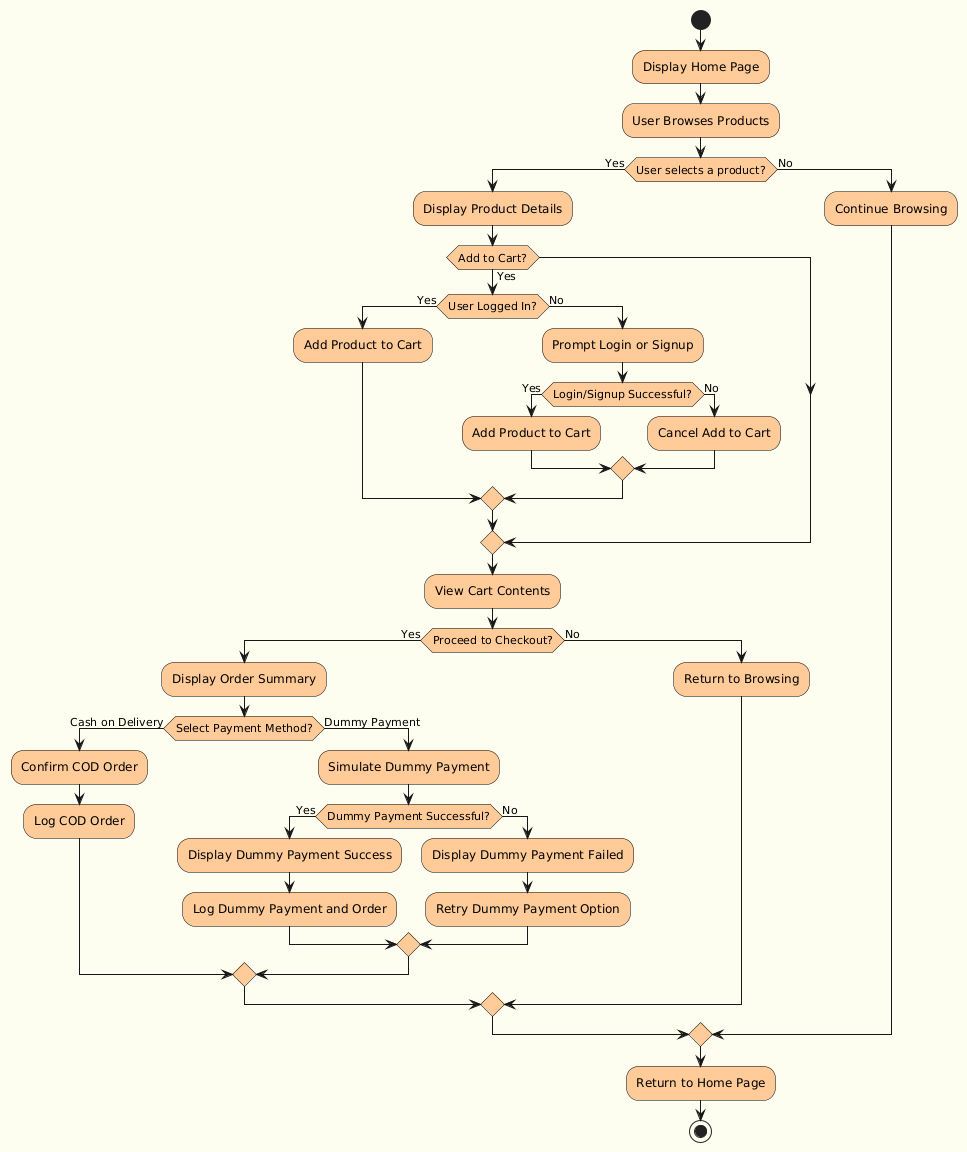
A sequence diagram represents the dynamic behavior of a system, showing how objects interact with each other through messages in a specific sequence.



## State Diagram:

****

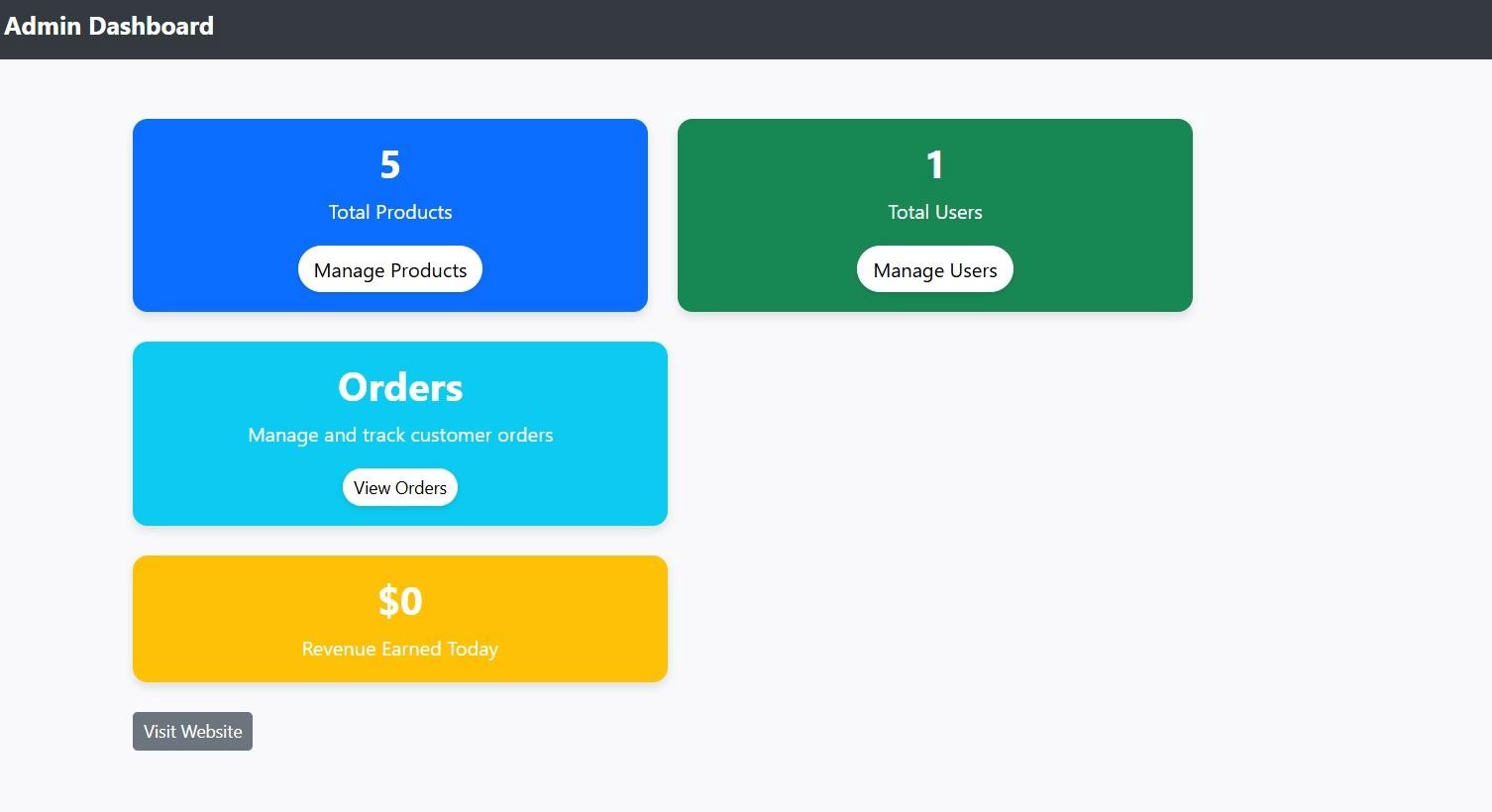
## Activity Diagram:

****

# Interface requirements

## System interface:

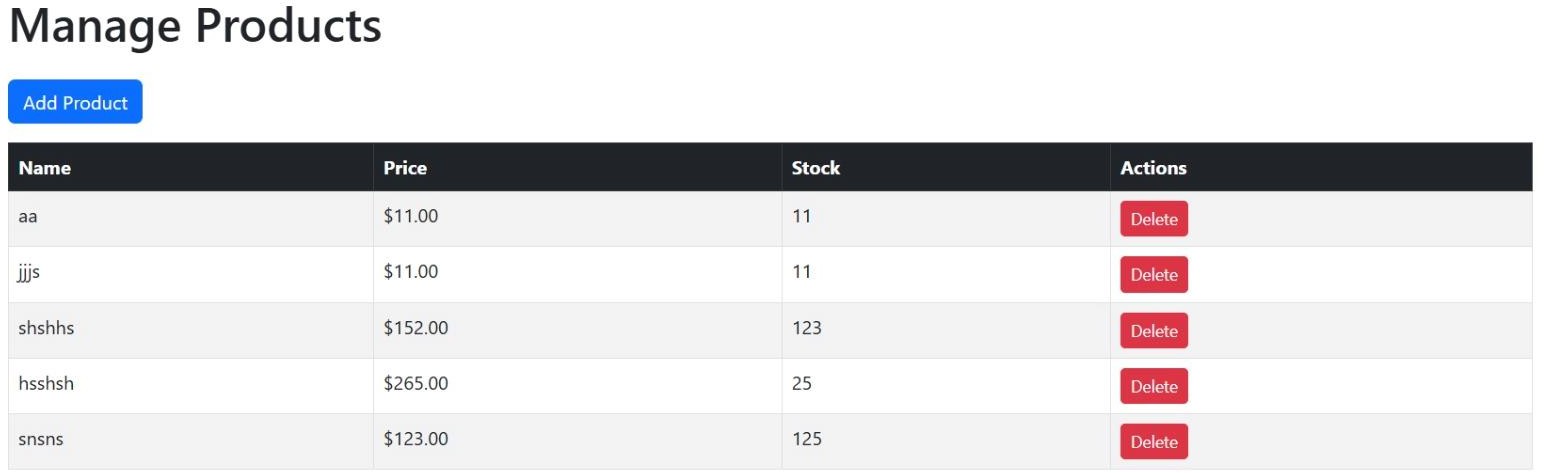
* **Admin Dashboard**



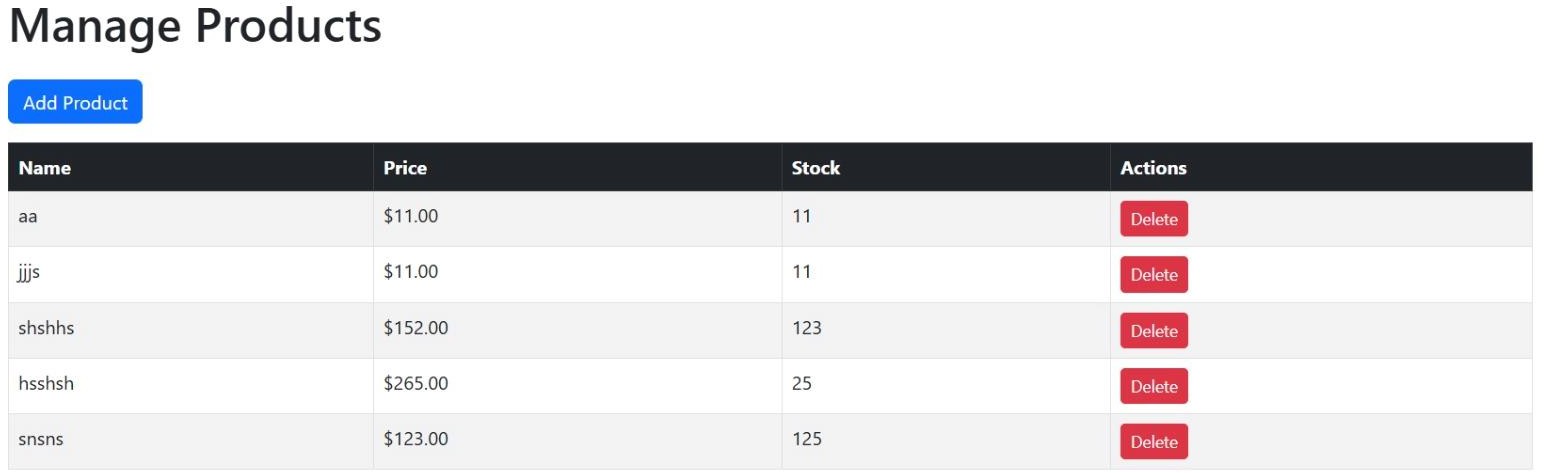
The admin can Manage products, Manage user, view orders and view the revenue earned for the day

### Admin products management

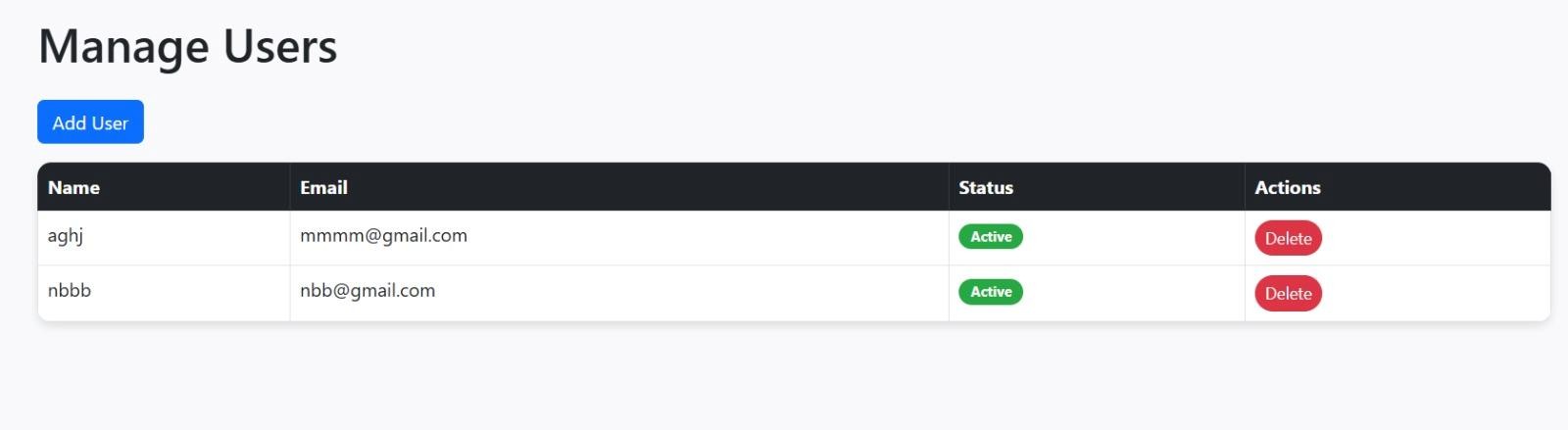
* + An admin can Add and view products.

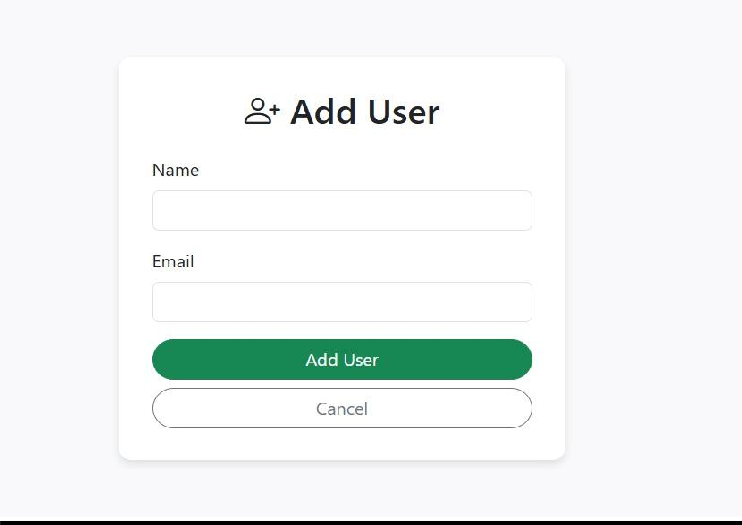


* + An admin can delete products.

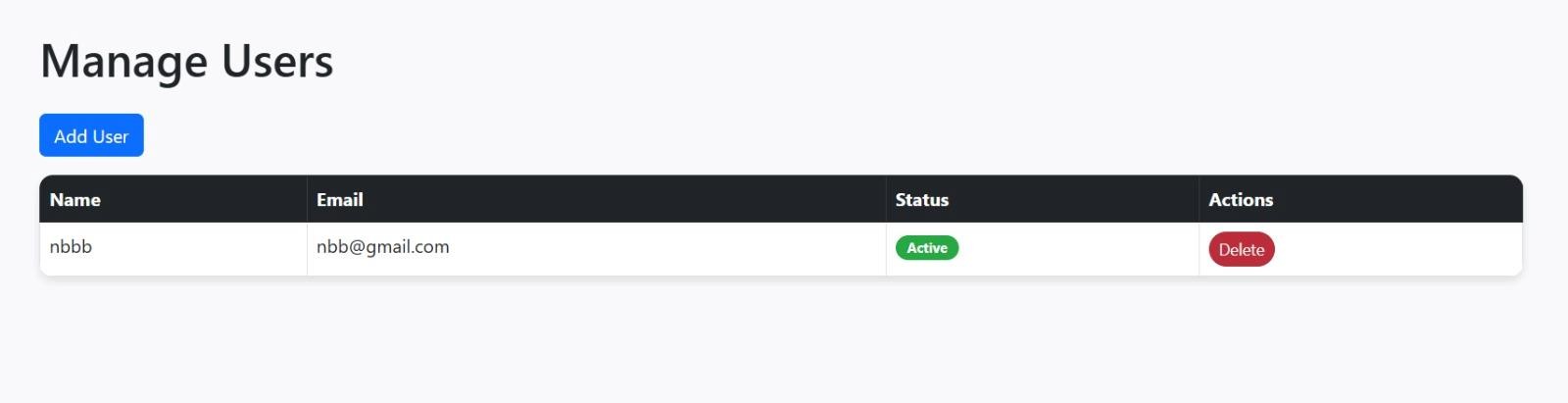


### Admin Users Management

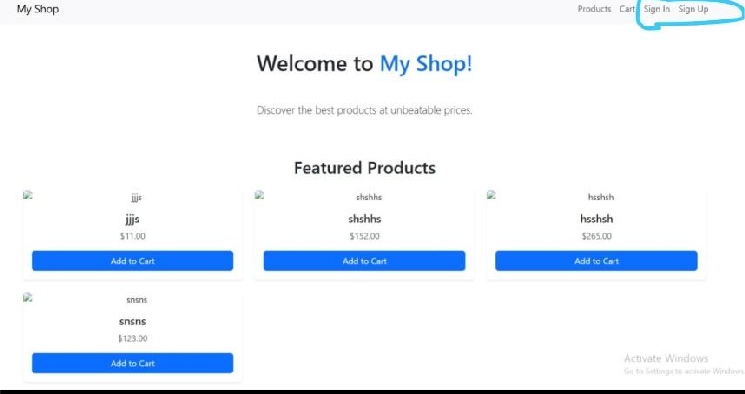
* + An admin can View Users.
* Add users



* + Delete Users

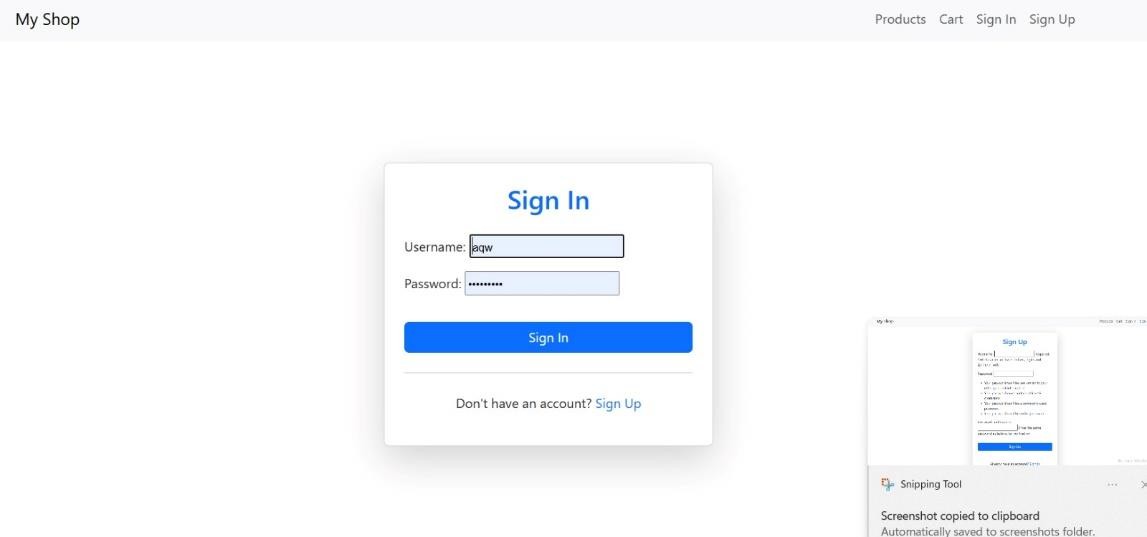


* **Our homepage**

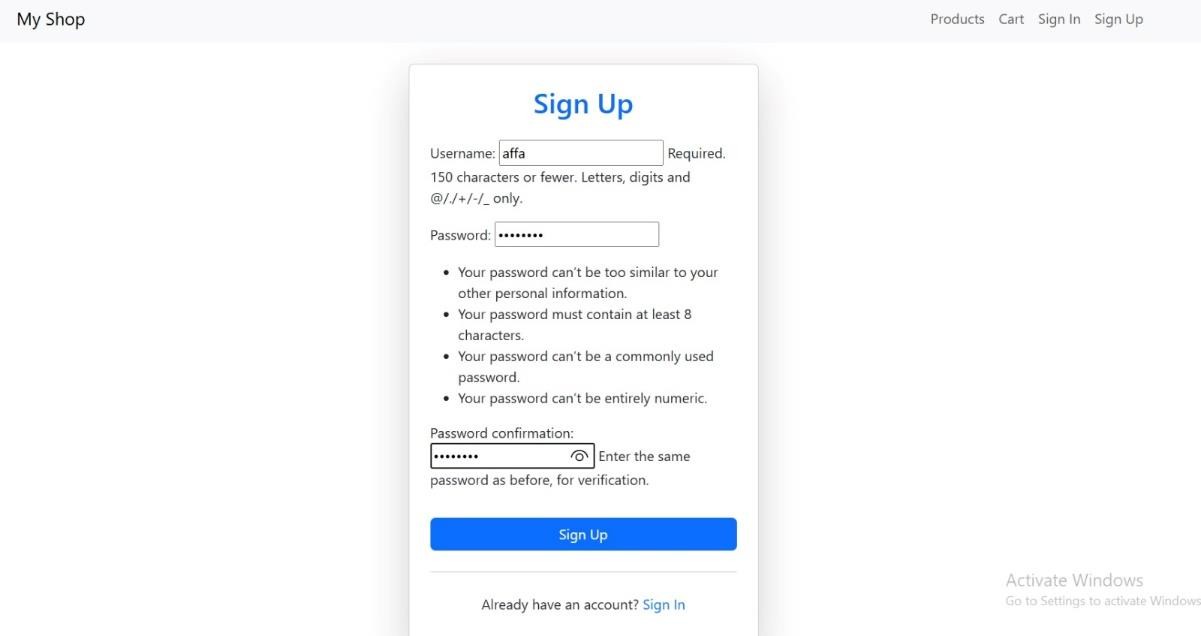


* **Sign in/Sing up**

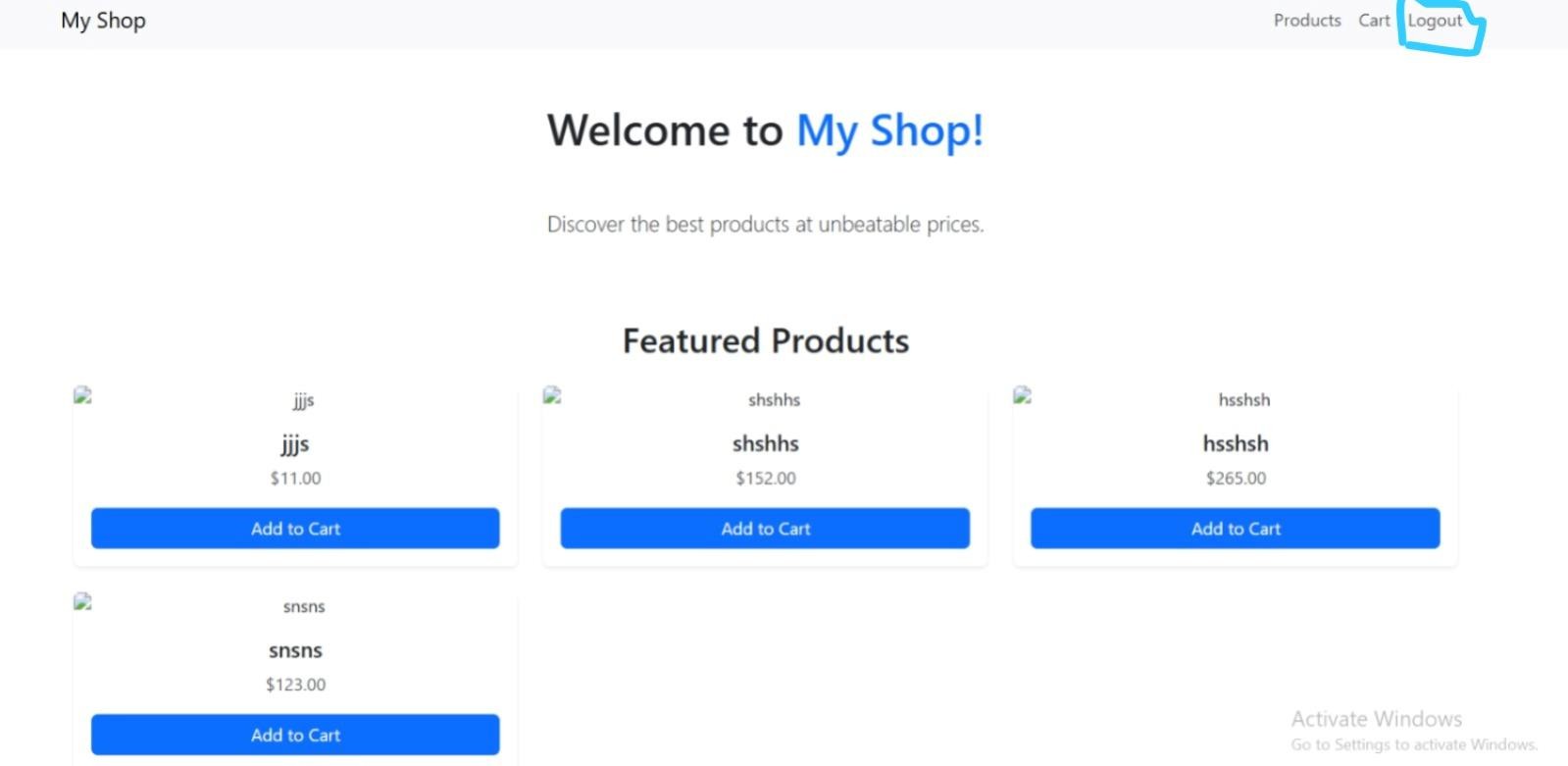
A Customer Must be registered to be able to log into our website



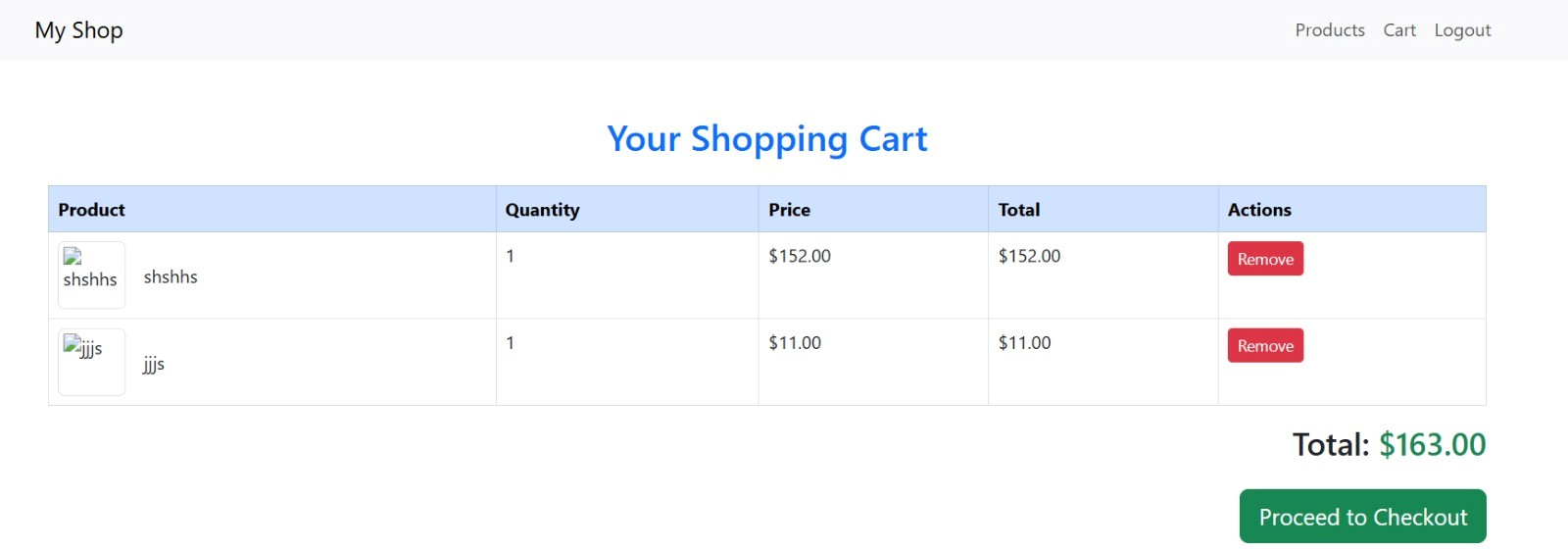
A customer should meet a specific criteria for the required info to be able to register successfully into our website



* **View products**

After signing in the user should be able to add products to the cart successfully

* **View Cart**

Upon finishing shopping a user can navigate to Cart to view or delete the chosen Items and view the total amount then proceed to checkout successfully to make the payment.

## Software Requirements:

Since it is a web application that requires a backend scripting language like Python, along with a relational database to store transactions and user data. A modern web browser is needed for user interaction.

## Hardware Requirements:

As an internet-based application, it requires devices with active internet connectivity (PCs, laptops, smartphones) via WAN, LAN, or Ethernet. The server must have sufficient processing power, memory, and storage to ensure smooth operation.

# Testing

## Functionality testing:

Test Case 1: User Login

**Project Name:** My shop **Test Case ID:** 001 **Module Name:** Login

**Test Title:** Verify user can log in with valid credentials

**Description:** Test login functionality with valid and invalid credentials

**Pre-conditions:**

* User must have a registered account.

**Dependencies:**

* Database must contain valid user credentials.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| 1 | Navigate to the login page |  | Login page is displayed | Login page displayed successfully | Pass |  |
| 2 | Enter valid username and password | username: aaa, Password: Arwa123 | User is logged in and redirected to the homepage | User redirected to homepage | Pass |  |
| 3 | Enter invalid username | Email: invalid | "Invalid email or password" error message displayed | Error message displayed | Pass |  |
| 4 | Leave email or password blank | Email: blank, Password: blank | "Please enter your email and password" error message displayed | Error message displayed | Pass |  |

**Post-conditions:**

* User is logged in, and session details are stored in the database.

Test Case 2: User Signup

**Project Name:** My Shop **Test Case ID:** 002 **Module Name:** Signup

**Test Title:** Verify user can sign up with valid and invalid details

**Description:** Test the signup functionality with valid and invalid inputs.

**Pre-conditions:**

* User navigates to the signup page.

**Dependencies:**

* Database must allow new user creation.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| 1 | Navigate to the signup page |  | Signup page is displayed | Signup page displayed | Pass |  |
| 2 | Enter valid username, password, and confirmation | Username: affa, Password: Test@1234, Password Confirmation: Test@1234 | Account is created successfully, and user is redirected to the homepage | Account created successfully | Pass |  |
| 3 | Enter mismatched password confirmation | Password: Test@1234, Password Confirmation: Test123 | "Password confirmation does not match" error message displayed | Error message displayed | Pass |  |
| 4 | Enter a password less than 8 characters | Password: Test12, Password Confirmation: Test12 | "Your password must contain at least 8 characters" error message displayed | Error message displayed | Pass |  |
| 5 | Enter a common password | Password: password123, Password | "Your password can't be a commonly used password" error message displayed | Error message displayed | Pass |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
|  |  | Confirmation: password123 |  |  |  |  |
| 6 | Leave any required field blank | Username: blank, Password: Test@1234 | "This field is required" error message displayed | Error message displayed | Pass |  |
| 7 | Enter username with invalid characters | Username: user@\*123 | "Only letters, digits, and @/./+/-/\_ characters are allowed" error message displayed | Error message displayed | Pass |  |
| 8 | Use an already registered username | Username: affa, Password: Test@1234 | "This username is already taken" error message displayed | Error message displayed | Pass |  |
| 9 | Enter a password that is entirely numeric | Password: 12345678, Password Confirmation: 12345678 | "Your password can't be entirely numeric" error message displayed | Error message displayed | Pass |  |

**Post-conditions:**

* A new user is created and stored in the database if all validations pass.
* User can log in with the newly created account.

Test Case 3: Add to Cart

**Project Name:** My shop

**Test Case ID:** 003

**Module Name:** Add to Cart

**Test Title:** Verify user can add products to the cart

**Description:** Test the functionality of adding products to the cart

**Pre-conditions:**

* User must be logged in.
* Product catalog must be available.

**Dependencies:**

* Product database must contain at least one product.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| 1 | Navigate to the product catalog |  | Product catalog is displayed | Product catalog displayed | Pass |  |
| 2 | Select a product | Product ID: 12345 | Product details page is displayed | Product details displayed | Pass |  |
| 3 | Click "Add to Cart" |  | Product is added to the cart, and cart count is updated | Product added to the cart | Pass |  |
| 4 | Add the same product again |  | Quantity of the product in the cart increases | Quantity updated in the cart | Pass |  |

**Post-conditions:**

* Product is added to the cart, and cart details are updated in the database.

Test Case 4: View Cart

**Project Name:** My shop **Test Case ID:** 004 **Module Name:** Cart

**Test Title:** Verify user can view items in the cart

**Description:** Test the ability to view and update cart contents

**Pre-conditions:**

* User must be logged in.
* User must have items in the cart.

**Dependencies:**

* Cart data must be stored and retrievable from the database.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| 1 | Navigate to the cart page |  | Cart page is displayed | Cart page displayed successfully | Pass |  |
| 2 | Verify cart contents |  | All added items with correct quantities are displayed | Items displayed correctly | Pass |  |
| 3 | Update item quantity | Quantity: 2 | Cart updates successfully, and total price is recalculated | Cart updated successfully | Pass |  |
| 4 | Remove an item |  | Item is removed from the cart | Item removed successfully | Pass |  |

**Post-conditions:**

* Cart data reflects the latest updates.

## Test Cases for the management part:

Test Case 5: Admin Login

**Project Name:** My Shop **Test Case ID:** Admin\_001 **Module Name:** Admin Login

**Test Title:** Verify admin login functionality

**Description:** Test the admin login functionality with valid and invalid credentials.

**Pre-conditions:**

* Admin account must exist in the database.

**Dependencies:**

* Admin credentials must be stored in the database.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| 1 | Navigate to the admin login page |  | Admin login page is displayed | Admin login page displayed | Pass |  |
| 2 | Enter valid admin credentials | Username: admin, Password: Admin@1234 | Admin is logged in and redirected to the admin dashboard | Admin logged in successfully | Pass |  |
| 3 | Enter invalid credentials | Username: admin, Password: wrongpassword | "Invalid username or password" error message displayed | Error message displayed | Pass |  |
| 4 | Leave username or password blank | Username: blank, Password: Admin@1234 | "Please enter both username and password" error message displayed | Error message displayed | Pass |  |

**Post-conditions:**

* Admin is logged in, and session details are stored in the database.

Test Case 6: Add Product

**Project Name:** My Shop

**Test Case ID:** Admin\_002

**Module Name:** Product Management

**Test Title:** Verify admin can add new products

**Description:** Test the ability to add new products to the catalog.

**Pre-conditions:**

* Admin must be logged in.

**Dependencies:**

* Product data must be saved in the database.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| 1 | Navigate to the "Add Product" page |  | "Add Product" page is displayed | Page displayed successfully | Pass |  |
| 2 | Enter valid product details | Name: Laptop, Price: 1200, Description: Gaming Laptop | Product is added successfully, and confirmation message is displayed | Product added successfully | Pass |  |
| 3 | Leave mandatory fields blank | Name: blank, Price: 1200 | "All fields are required" error message displayed | Error message displayed | Pass |  |
| 4 | Enter invalid price | Name: Laptop, Price: -10 | "Price must be a positive number" error message displayed | Error message displayed | Pass |  |

**Post-conditions:**

* New product is saved in the database and visible in the catalog.

Test Case 7: Add User

**Project Name:** My Shop

**Test Case ID:** Admin\_003

**Module Name:** User Management

**Test Title:** Verify admin can add new users

**Description:** Test the ability to add new users through the admin panel.

**Pre-conditions:**

* Admin must be logged in.

**Dependencies:**

* Database must support adding new user entries.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| 1 | Navigate to the "Add User" page |  | "Add User" page is displayed | Page displayed successfully | Pass |  |
| 2 | Enter valid user details | Name: aqw  Password: Pass1234 | User is added successfully, and a confirmation message is displayed | User added successfully | Pass |  |
| 3 | Leave mandatory fields blank | Name: blank | "All fields are required" error message displayed | Error message displayed | Pass |  |
| 4 | Use an already registered user | user: arwa | "username already exists" error message displayed | Error message displayed | Pass |  |

**Post-conditions:**

* The new user is added to the database and can log in with the provided credentials.

Test Case 8: View Orders

**Project Name:** My Shop

**Test Case ID:** Admin\_004

**Module Name:** Order Management

**Test Title:** Verify admin can view and manage customer orders

**Description:** Test the ability to view and update customer orders.

**Pre-conditions:**

* Admin must be logged in.
* Orders must exist in the database.

**Dependencies:**

* Database must store orders.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| 1 | Navigate to the "Orders" page |  | List of orders is displayed | Orders displayed successfully | Pass |  |
| 2 | Click on an order to view details |  | Order details page is displayed | Order details displayed | Pass |  |
| 3 | Update order status | Status: Shipped | Order status is updated, and a confirmation message is displayed | Status updated successfully | Pass |  |

**Post-conditions:**

* Order details reflect the updated status in the database.

# Conclusion

In conclusion, this project presents a functional and user-friendly online shopping system that allows customers to easily browse products, manage their cart, place orders, and complete payments through a dummy payment system. The system also features an administrative interface for managing users, products, and orders, providing a comprehensive platform for both end-users and administrators.

**Github link:**

https://github.com/arwasameh123/shopping\_website.git