Online Store.

Phase Two: DOCUMENTATION.

TEAM MEMBERS:

1. Hana AbdEl Kader Attia. ID: 320240068.
2. Jana Haitham Mohamed. ID: 320240070.

Supervisors:

Dr. Ahmed Anter

Eng. Nourhan Waleed

Introduction:

Shop Hub is a web-based e-commerce platform developed to provide a streamlined, minimalistic shopping experience. The system allows users to browse diverse categories such as accessories, shoes, and clothing. It focuses on high performance and a clean user interface, bridging the gap between basic structural web design and functional, data-driven web applications.

1. *Project Description & objectives:*

Description:

It is a small web-based online store that allows users to browse, search, and purchase items such as accessories, shoes, or clothes. Each product has details like name, price, and image. Users can add items to their shopping cart, view the total cost, and proceed to check out.

Objective:

The main goal of this project is to create a **user-friendly and responsive e-commerce website** that demonstrates the connection between frontend and backend technologies. The system aims to show how a real online store works, displaying products from a database, managing user actions like adding to cart, and storing orders securely.

2. ***Business Analysis:***

I. ***Stakeholders:***

Stakeholder	Role / Interest
Customer (User)	Browse and buy products online. Needs a simple, clear interface and secure checkout.
Store Admin	Manages products (add, update, delete) and views customer orders.
Developers (Team)	Design and build the system; ensure proper connection between frontend and backend.
Hosting Provider	Provides online space for website and database hosting.
Payment Service (Simulated)	Demonstrates checkout behavior without real payment integration.

II. ***Marketing Analysis:***

The online store will target people who want an easy and at home shopping experience.

We studied other simple pages and stores like:

- DeFacto
- Amazon
- SHEIN

It focuses on a **lightweight, fast, and minimalistic shopping experience**, using fewer animations and cleaner pages than large platforms.
We'll use:

- Attractive homepage banners to catch user attention.
- A mobile-friendly responsive design to reach more users.

III. Layout:

Here's the **structural layout** (the flow and components of the project):

Main Pages:

1) Home Page:

- Header with logo and navigation bar.
- Featured products carousel.
- Product categories.

2) *Products Page:*

○ List of all available items with images, prices, and "Add to Cart" button.

3) *Login / Sign-Up Page:*

○ Two main sections:

1. **Login Form:**
   • Fields: Email and Password.
   • Button: Login.

2. **Sign-Up Form:**
   • Fields: Full Name, Email, Password.
   • Button: Create Account.

4) Cart Page:

○ Displays all added items, quantity, and total price.
○ "Remove" or "Checkout" options.

5) Admin Page (Backend):

○ Login for admin.
○ Add, edit, or delete products.
○ View orders.

6) *Database Tables:*

○ **users** (id, username, password).
○ **products** (id, name, price, description, image).
○ **orders** (id, user_id, total, date).
○ **cart_items** (id, product_id, quantity, user_id).

7) *Checkout Page:*

○ Shows a summary of all items added to the cart:
   • Product name.
   • Price.
   • Total amount.
○ Includes a simple Shipping Form:
   • Address.
   • City.
   • ZIP Code.
   • "Place Order" button.
○ When submitted, it confirms the order and thanks the customer.

8) Footer:

- ○ Copyright information.
- ○ Contact email or social media links.
- ○ Quick links like About Us or Privacy Policy.

3. Requirements Specification:

I. Functional Requirements:

These requirements define what the system allows users to do:

A. Product Browsing: Users must be able to view a list of products with images, names, and prices.

B. Filtering: Users must be able to filter products by category (e.g., Accessories, Shoes, Clothes).

C. Cart Management:

* Users can add items to the cart.

* Users can update quantities (increment/decrement).

* Users can remove items from the cart.

Constraint: Cart data is managed via JSON objects and processed via PHP sessions and stored in the MySQL cart_items table.

D. Authentication:

* Users must be able to Log in and Sign Up.

* System must distinguish between "User" and "Admin" roles.

E. Checkout Process:

* Access to the checkout page is restricted to logged-in users only.

* Users must provide shipping details to complete the order.

F. Admin Management:

* Admins must be able to Add new products.

* Admins must be able to Edit existing product details.

* Admins must be able to Delete products.

II. Non-Functional Requirements:

These requirements define how the system performs:

a. Unobtrusive JavaScript: All event handling (clicks, form submissions) is managed via addEventListener in external script files. No inline onclick attributes are used in HTML.

b. Adaptability: The layout adapts to mobile devices (screens smaller than 768px) using CSS Media Queries.

c. Data Persistence: User accounts, product inventory, and order history are stored permanently in a MySQL database, ensuring data is accessible from any device upon login.

d. Modularity: Code is organized into a centralized script.js file that routes logic based on the active page.

4. System Architecture & Data Model:

1. Presentation Layer: XHTML, CSS, and jQuery (Frontend).
2. Application Layer: Python with Flask (Server-side logic handling requests).
3. Data Layer: SQLite Database (store.db) for lightweight, server-less data storage.

5. Problem Statement:

Many commercial e-commerce platforms are burdened with "feature bloat," resulting in slow load times and confusing navigation. Furthermore, many educational projects fail to strictly separate structure (XHTML) from behavior (JavaScript). Shop Hub addresses these challenges by utilizing modular architecture, database management, and a "Gatekeeper" security system to ensure a fast, secure, and organized shopping environment.

6. Market and Related Companies:

Our team analyzed industry leaders including **Amazon, SHEIN, and DeFacto**. While major platforms offer vast inventory, they often overwhelm the user. Shop Hub focus on a lightweight mobile-first design, fewer distracting animations, and a faster path from product discovery to checkout.

7. Methodology & Technical Implementation:

The project was built adhering to professional web standards and the "Separation of Concerns" principle:

- **XHTML & CSS:** The structure follows well-formed XHTML standards (properly nested, lowercase tags). Presentation is handled entirely by external CSS files, utilizing **Floats** for layout and **Media Queries** to ensure the site is fully responsive on devices smaller than 768px.
- **DOM Manipulation (jQuery):** We integrated jQuery to manage the Document Object Model efficiently. This allows the system to update the product grid and cart totals dynamically without requiring page refreshments.
- **Data Interchange (JSON):** The application uses **AJAX** for data interchange. The frontend sends JSON data to the Python backend, which responds with JSON status messages (e.g., {"status": "success"}).
- **Validation (Regex):** To ensure data integrity, all forms (Login, Sign-Up, and Shipping) use **Regular Expressions**.
    - *Email:* /^[^\s@]+@[^\s@]+\.[^\s@]+$/ ensures valid contact info.
    - *ZIP Code:* Numeric-specific patterns ensure shipping accuracy.

- **Unobtrusive JavaScript:** Following best practices, we moved all logic to an external script.js. There are zero inline event handlers (no onclick); all interactions are managed via addEventListener.
- **Routing Logic:** A Python Flask application (app.py) intercepts requests to the server (mimicking legacy server.php endpoints) and routes them to the appropriate Python functions for processing.
- **Asynchronous Data Handling:** We implemented AJAX to bridge the gap between the frontend and the database. For example, when an admin saves a product, a jQuery AJAX POST request is sent to the server. The Python backend receives this, executes an SQL **INSERT** command, and returns a success signal.Session
- **Database Management:** Instead of a complex MySQL setup, we utilized **SQLite**. This allows the entire database to exist as a single file (store.db) within the project folder, making the project portable and easy to deploy without external server configuration.

Component	Technology
Server Environment	**Python (Flask Web Framework)**
Backend Language	**Python**
Database	**SQLite**
Data Transfer	**AJAX (Asynchronous JavaScript and XML)**
Frontend Scripting	**jQuery & JavaScript**
Markup & Styling	**XHTML & CSS**
Validation	**Regex (Regular Expressions)**

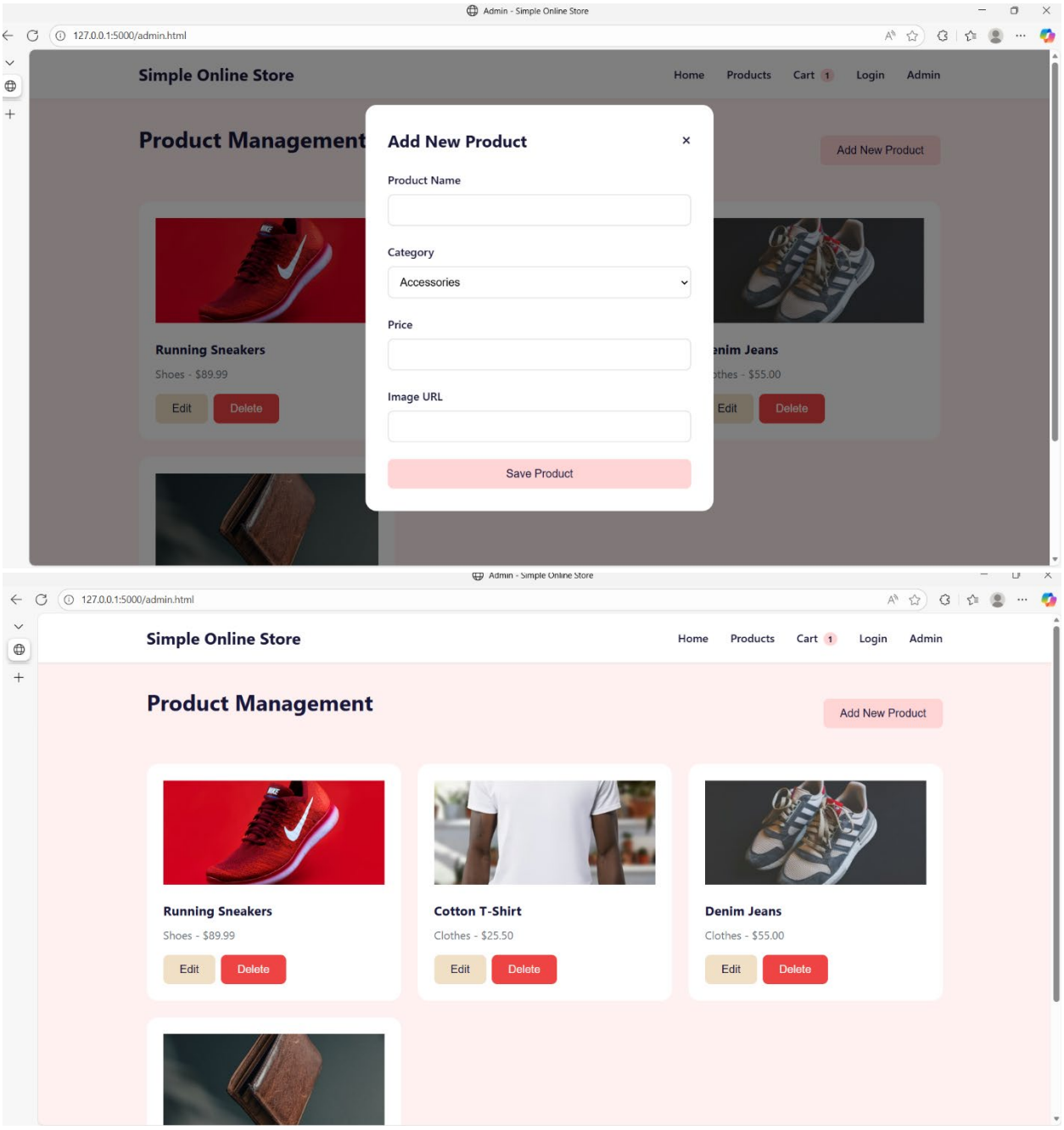
8. Proposed Architecture:

The system follows a **3-Tier Client-Server Architecture**:

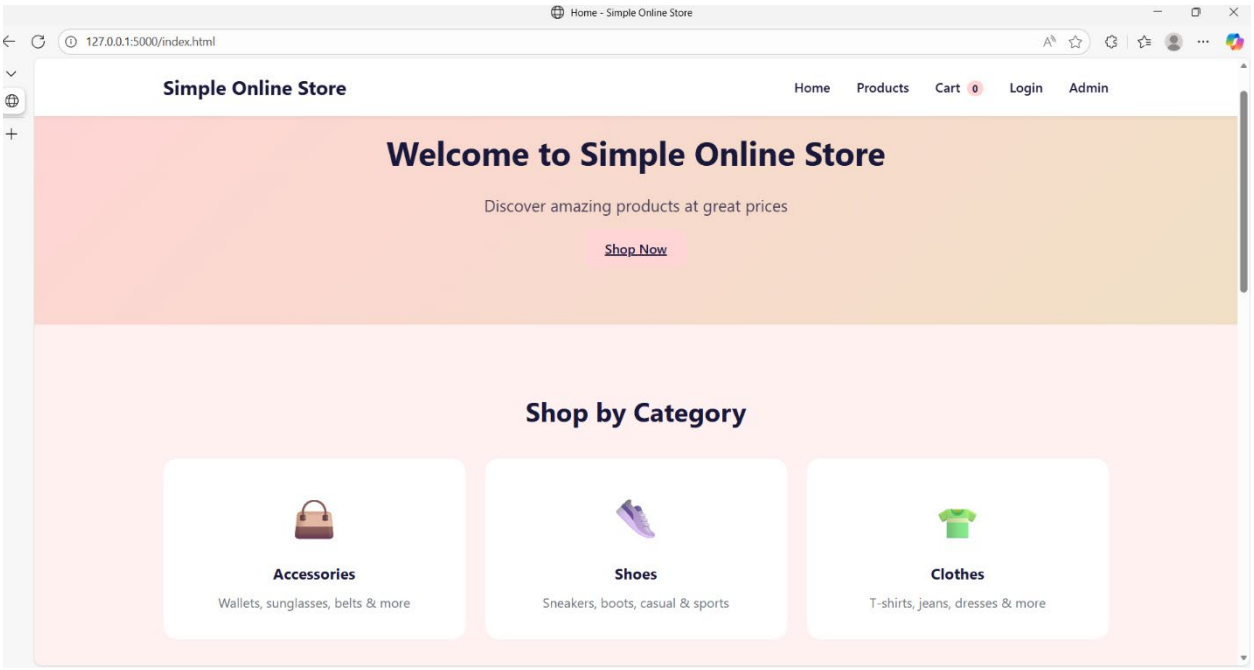
- **Presentation Layer:** Developed using XHTML, CSS, and jQuery to provide a responsive user interface.
- **Logic Layer:** Python scripts running via the Flask framework manage the business logic, API endpoints (routing), and data processing.
- **Data Layer:** A relational **SQLite** database provides permanent storage for users, products, and orders, ensuring data persists even after the server is restarted.
- **Communication:** jQuery AJAX acts as the bridge, allowing the frontend to communicate with the PHP backend asynchronously.

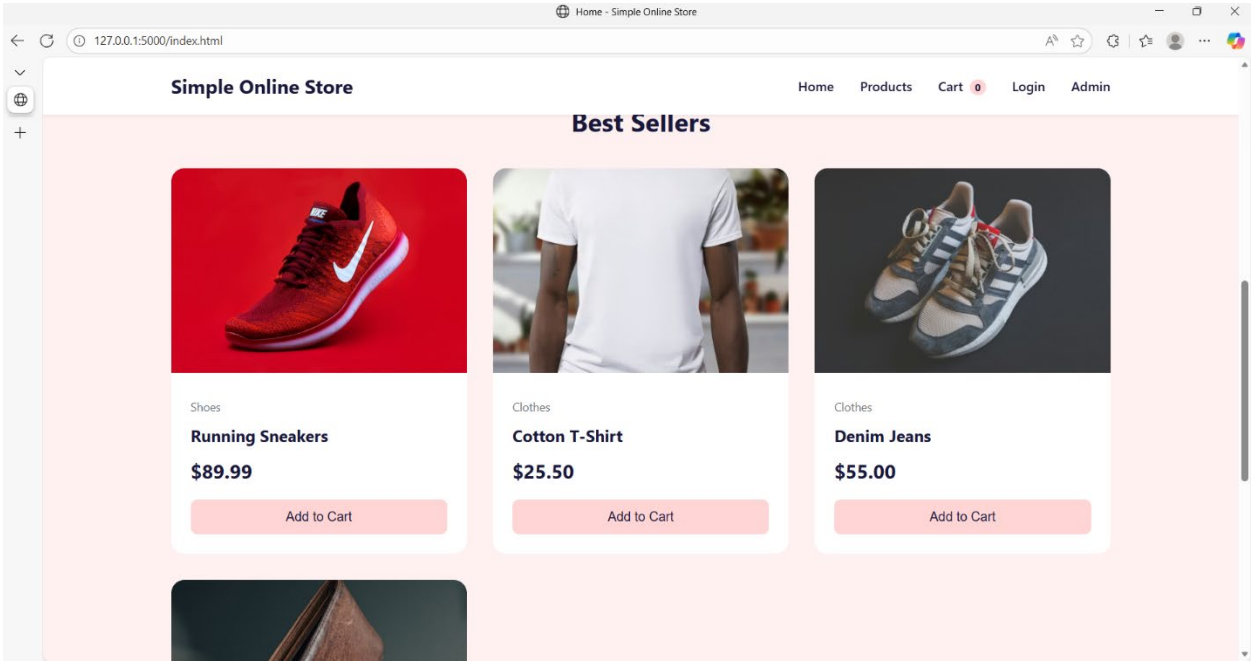
9. Results:

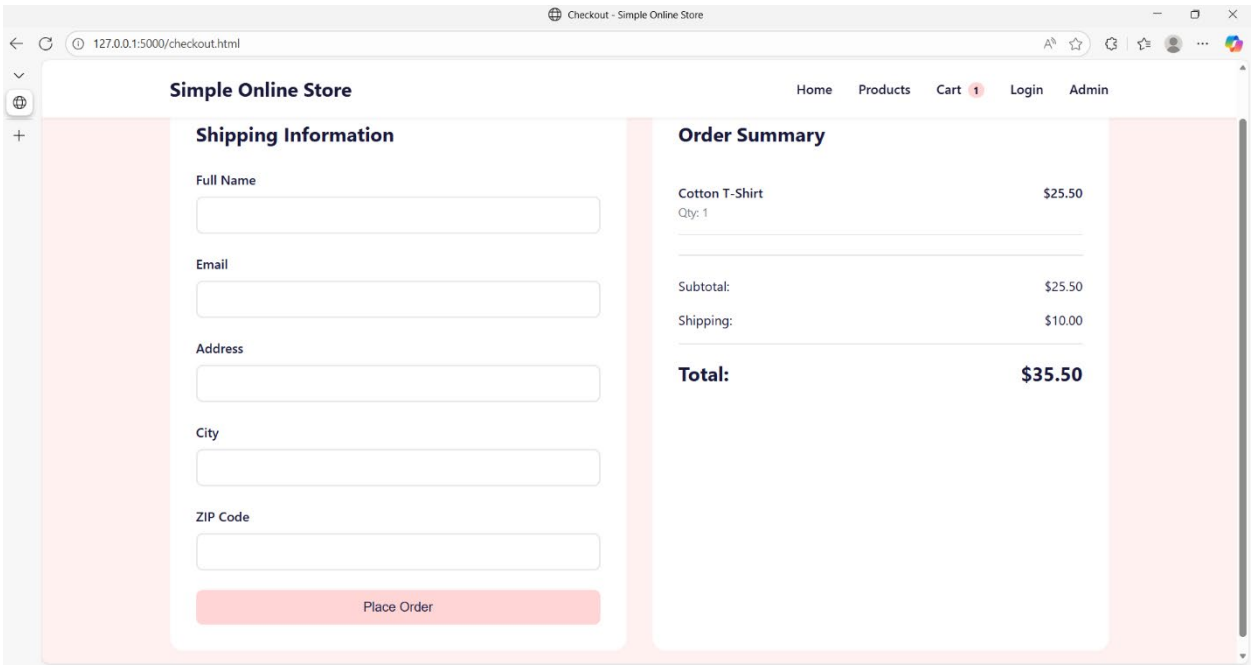
The final system provides a seamless transition between the user-facing storefront and the administrative backend. Users can filter by category and manage their cart effectively, while administrators have full CRUD (Create, Read, Update, Delete) control over the product inventory.

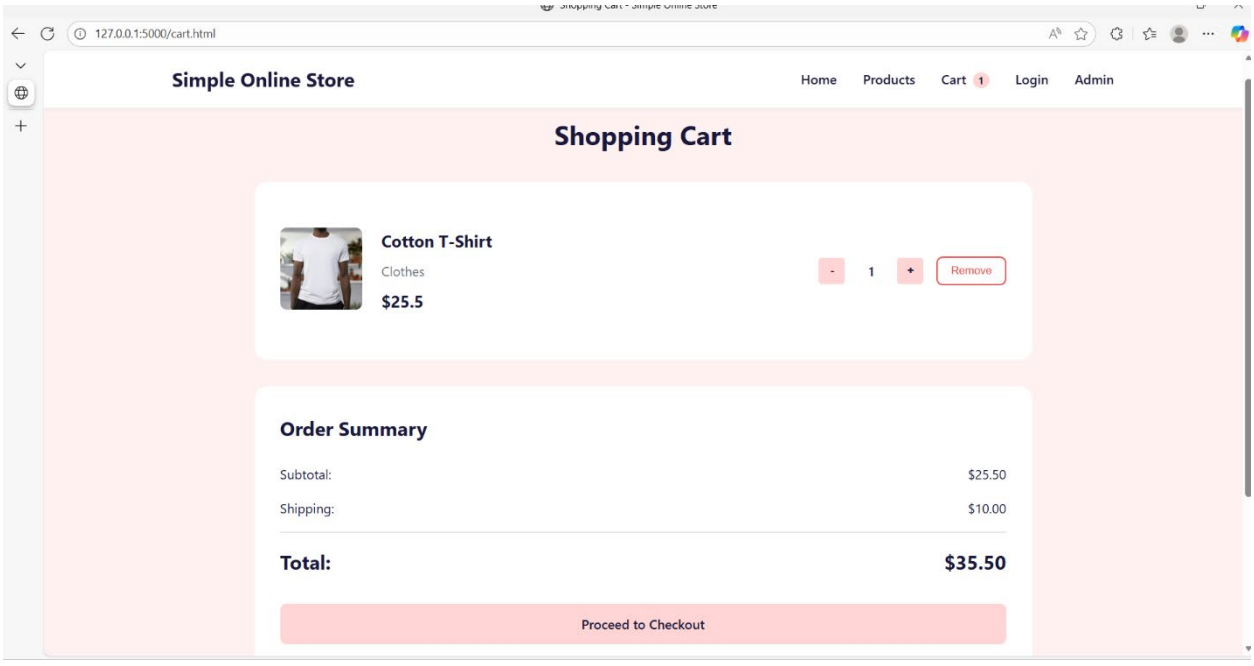

Admin - Simple Online Store
127.0.0.1:5000/admin.html
Simple Online Store
Home
Products
Cart 1
Login
Admin
Product Management
Add New Product
Add New Product
Product Name
Category
Accessories
Price
Image URL
Save Product
Running Sneakers
Shoes - $89.99
Edit
Delete
nim Jeans
Edit
Delete

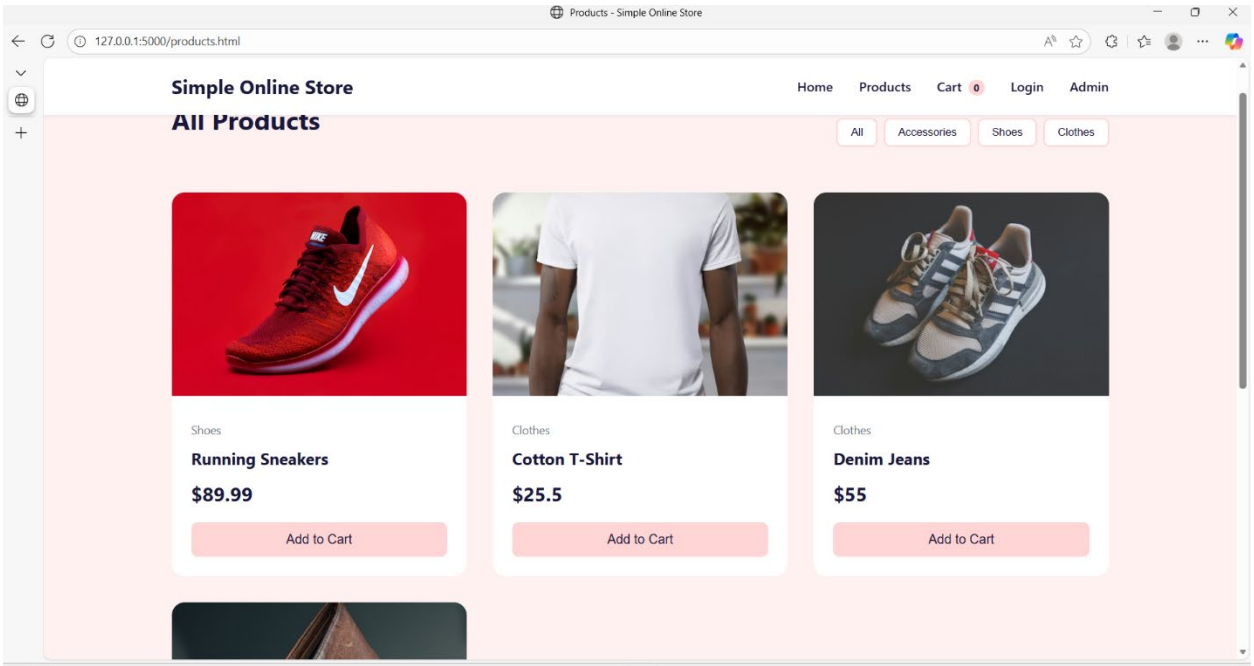
Admin - Simple Online Store
127.0.0.1:5000/admin.html
Simple Online Store
Home
Products
Cart 1
Login
Admin
Product Management
Add New Product
Running Sneakers
Shoes - $89.99
Edit
Delete
Cotton T-Shirt
Clothes - $25.50
Edit
Delete
Denim Jeans
Clothes - $55.00
Edit
Delete

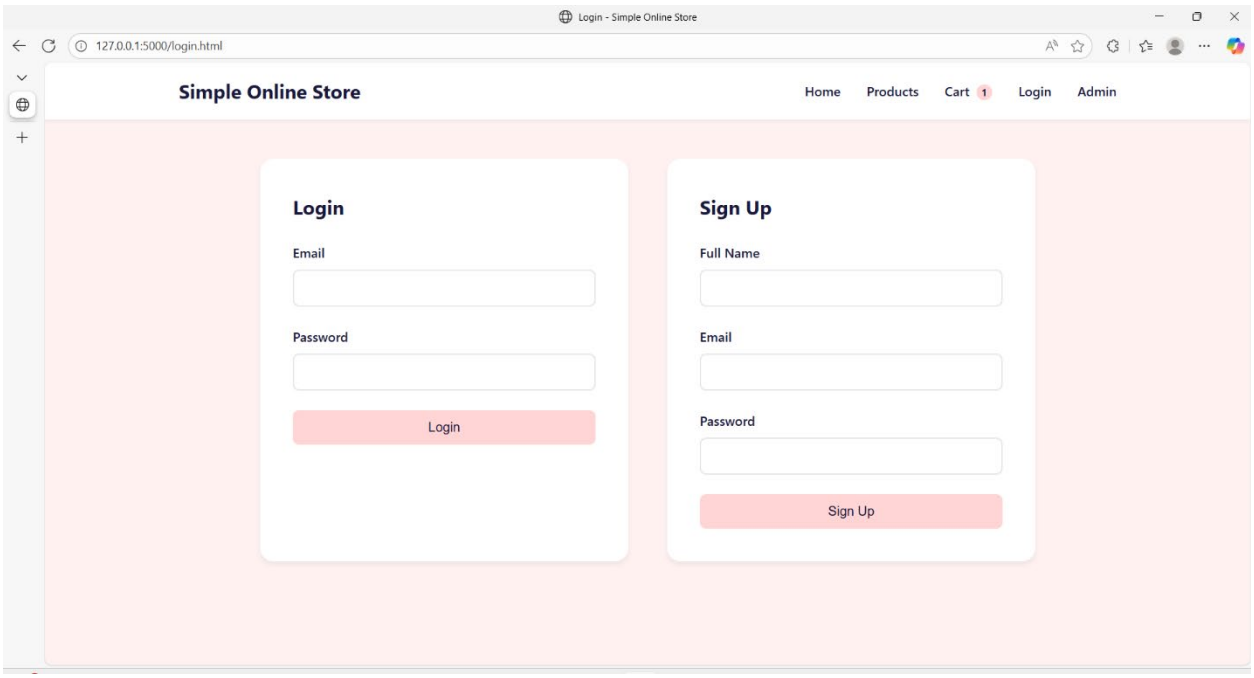

Home - Simple Online Store
127.0.0.1:5000/index.html
Simple Online Store
Home
Products
Cart 0
Login
Admin
Welcome to Simple Online Store
Discover amazing products at great prices
Shop Now
Shop by Category
Accessories
Wallets, sunglasses, belts & more
Shoes
Sneakers, boots, casual & sports
Clothes
T-shirts, jeans, dresses & more


Home - Simple Online Store
127.0.0.1:5000/index.html
Simple Online Store
Home
Products
Cart 0
Login
Admin
Best Sellers
Shoes
Running Sneakers
$89.99
Add to Cart
Clothes
Cotton T-Shirt
$25.50
Add to Cart
Clothes
Denim Jeans
$55.00
Add to Cart


Checkout - Simple Online Store
127.0.0.1:5000/checkout.html
Simple Online Store
Home
Products
Cart 1
Login
Admin
Shipping Information
Full Name
Email
Address
City
ZIP Code
Place Order
Order Summary
Cotton T-Shirt
Qty: 1
$25.50
Subtotal:
$25.50
Shipping:
$10.00
Total:
$35.50


127.0.0.1:5000/cart.html
Simple Online Store
Home
Products
Cart 1
Login
Admin
Shopping Cart
Cotton T-Shirt
Clothes
$25.5
-
1
+
Remove
Order Summary
Subtotal:
$25.50
Shipping:
$10.00
Total:
$35.50
Proceed to Checkout





10. Future Work:

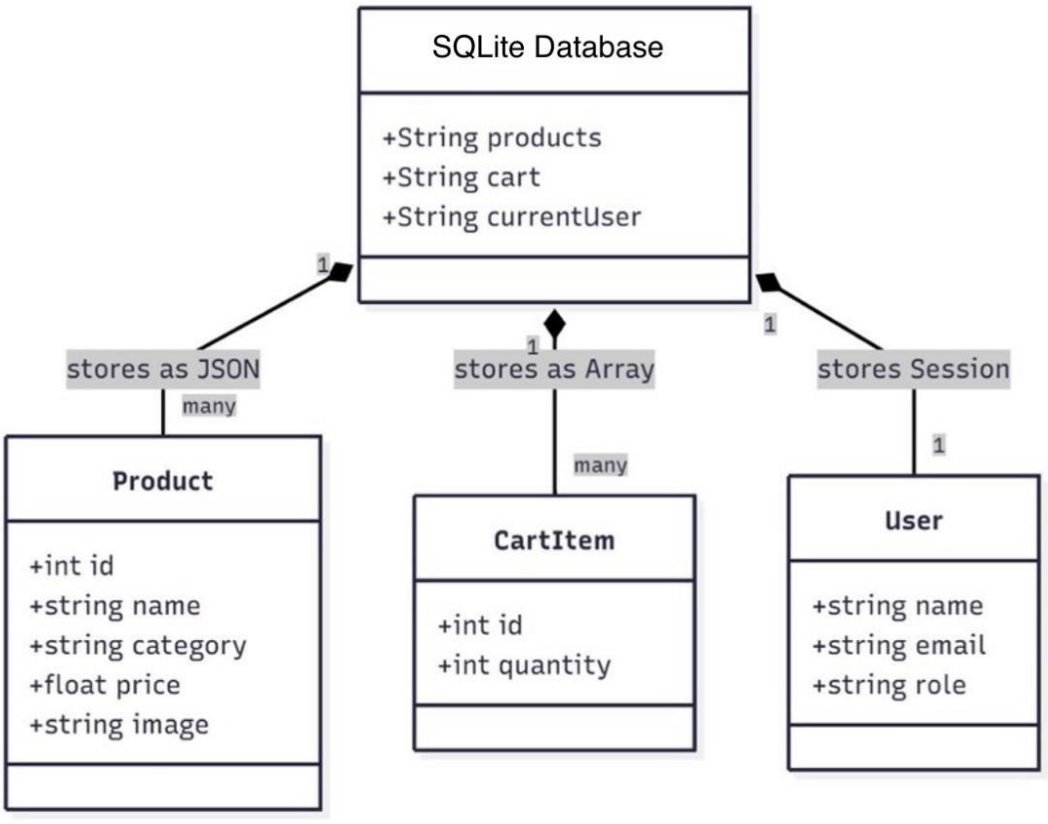
To evolve Shop Hub beyond Phase Two, the following steps are proposed:

- **Security Enhancement:** Implementing Crypt password hashing for database security.
- **Image Upload System:** Creating an admin interface to upload product images directly to the server.
- **Live API Integration:** Connecting real-world payment gateways like PayPal or Stripe.

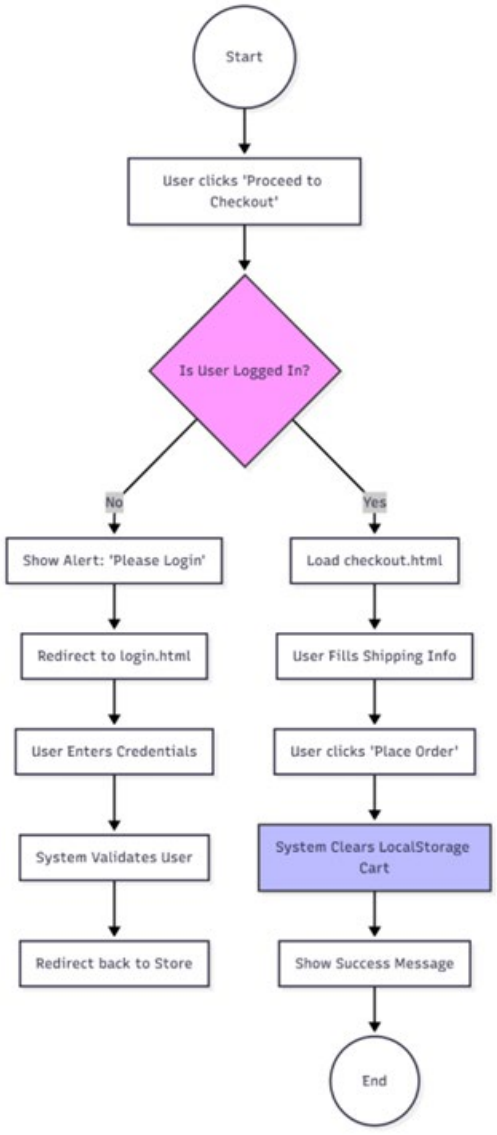
- **Advanced Security:** Implementing server-side session management and password hashing (crypt).
- **Live Payments:** Integrating a functional payment gateway API (e.g., Stripe) to replace the simulated checkout.
- **Dynamic Search:** Implementing AJAX-based live search for products.

11. UML Diagrams:

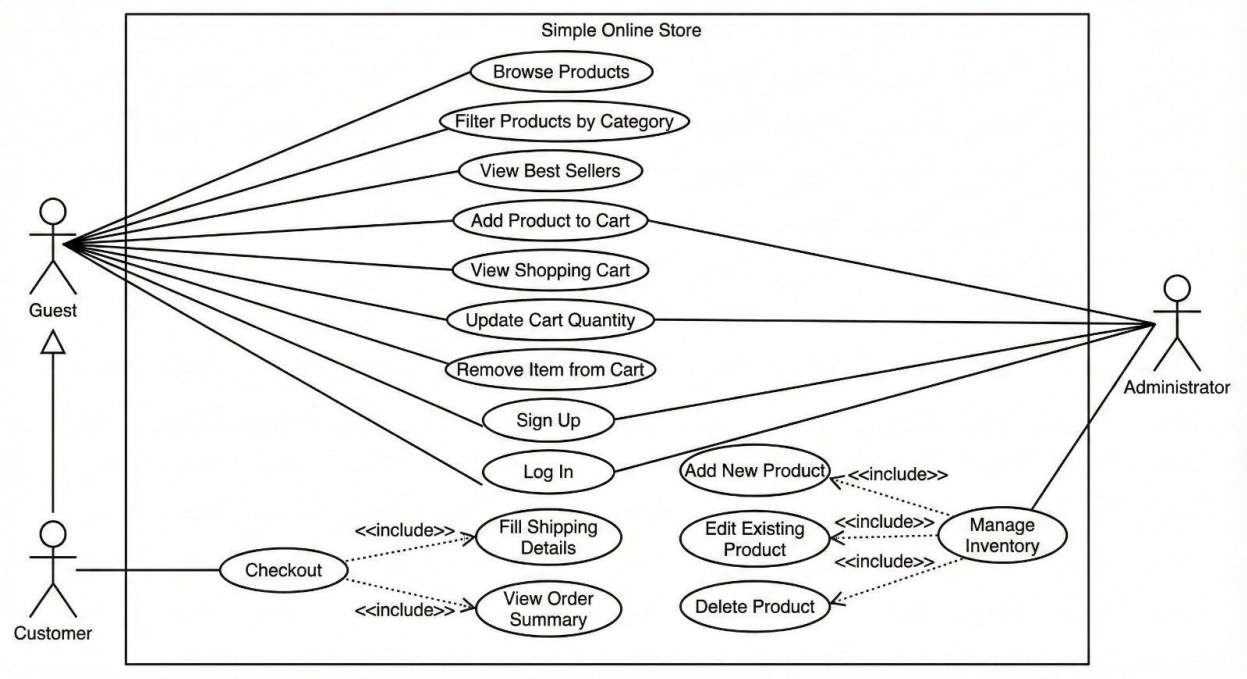
1. This diagram illustrates the interactions between the Users, whether it's User or Admin, and the System.



2. This diagram details the flow of the Checkout process, specifically the security check that ensures a user is logged in before accessing the checkout form



3. This diagram shows the main system classes and the relationships between them, illustrating how users, products, carts, and orders interact within the online store.



GitHub link:

https://github.com/hanaabdelkader583-creator/Simple-Online-Store/tree/main