**Project by:** Assaad AKEL & Rida NASSER

**Major:** Data Science

**Course:** Web Application

**Course Instructor:** Dr. Nobar KASSABIAN

**Project Title: “Flask E-commerce Web App: Luxury Watches Store”**

**PLAN:**

1. **Introduction**
2. Target Audience
3. Functionalities
4. **Project Overview**
5. Directory Structure
6. **User Functionality Breakdown**
7. Login
8. Register
9. User Page
10. Dashboard page
11. View Cart page
12. **Admin Functionality Breakdown**
13. Admin Login
14. Admin Panel
15. Add Products
16. Edit & Delete Products
17. View & Delete Users
18. **UI Details**
19. Login Page
20. Registration Page
21. Product Management Pages
22. Shopping Cart
23. Contact Form
24. **Business Logic Overview**
25. User Authentication
26. Product Management
27. Shopping Cart Operations
28. Email Communication
29. **Conclusion**
30. Challenges Faced
31. Future Enhancements
32. **Appendix**

# **Introduction:**

This report details the development of a web application using Flask, a lightweight Python web framework. The objective was to create an e-commerce platform for a Luxury Watches store, catering to a discerning clientele seeking high-end timepieces.

## **Target Audience:**

Dr. Nobar KASSABIAN, Instructor of the course.

## **Functionalities:**

The application offers a user-friendly experience for both customers and the store owner (admin).

Customers can:

* Browse available watches and add them to their shopping carts.
* Checkout and complete a purchase (future implementation).
* Send inquiries to the store owner through a contact form.

The administrator has access to a dedicated dashboard for managing the store's inventory. Key functionalities include:

* Adding new watch products to the database with detailed descriptions, pricing, and images.
* Editing existing watch listings for accurate information and updates.
* Deleting watches from the store's offerings.
* Viewing product statistics and trends with interactive visualizations.

# **Project Overview**

The application utilizes Python 3.11.5 with Flask version 3.0.2 and Werkzeug version 3.0.1 for server- side functionality. Flask-SQLAlchemy is employed as an Object Relational Mapper (ORM) to seamlessly interact with a SQLite3 database for data persistence.

## **Directory Structure:**

The project is organized into a clear and maintainable directory structure:

**app:** This folder houses the core application logic in Python files:

* **\_\_init\_\_.py:** Initializes the application and configures the database connection.
* **models.py:** Defines the data models representing products, users, and other entities within the database.
* **routes.py:** Handles all routing logic, mapping URLs to corresponding functions that control the application's behavior.

**instance:** Contains the initialized database file (SQLite3) used by the application.

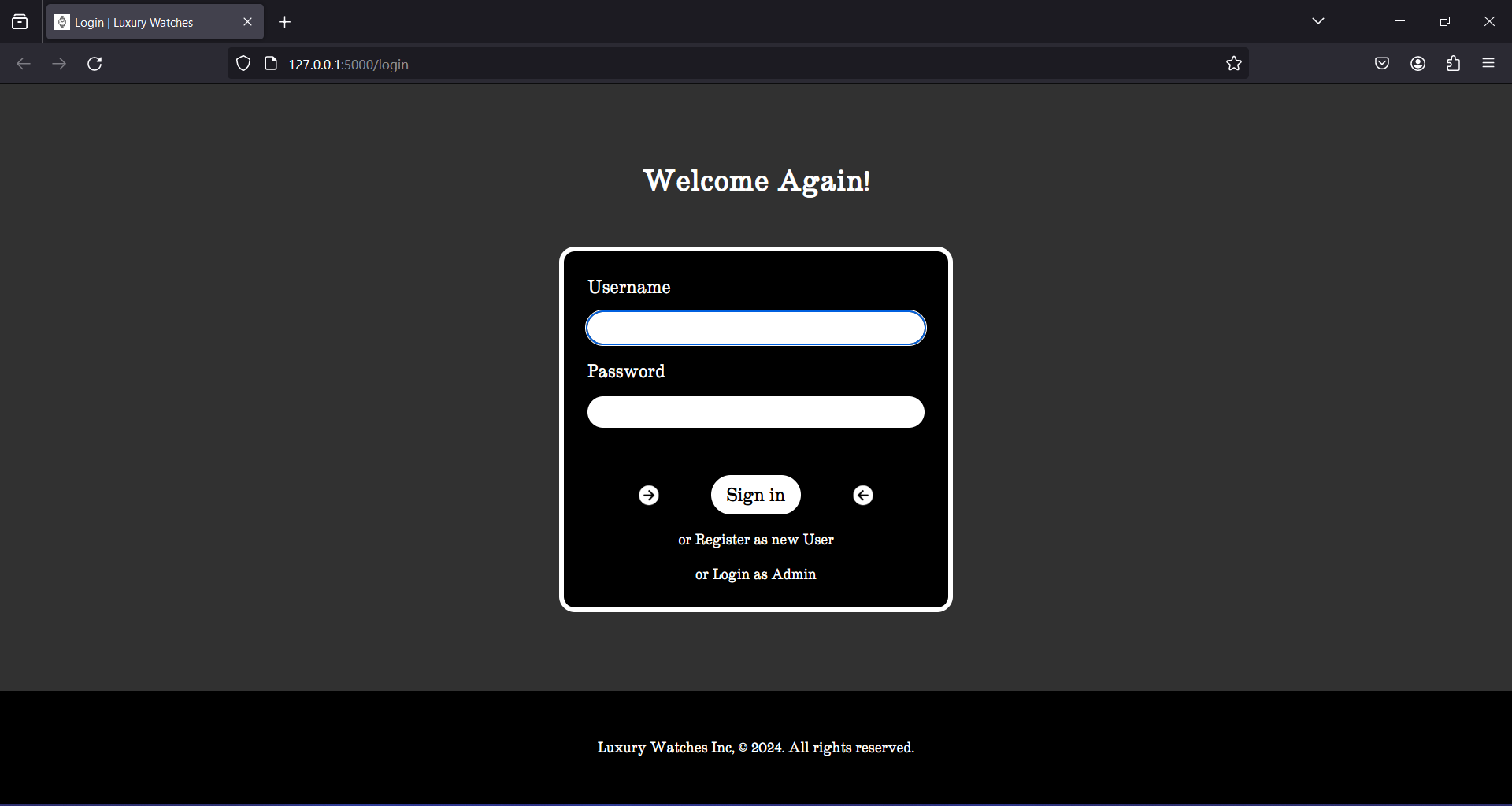
**static:** Stores all static files such as Cascading Style Sheets (CSS), JavaScript (JS) files, web fonts, and an "assets" subfolder for images.

**templates:** Holds the HTML templates that define the application's user interface (UI) layout and structure.

**app.py:** The main entry point. When the application runs, this file starts the Flask app.

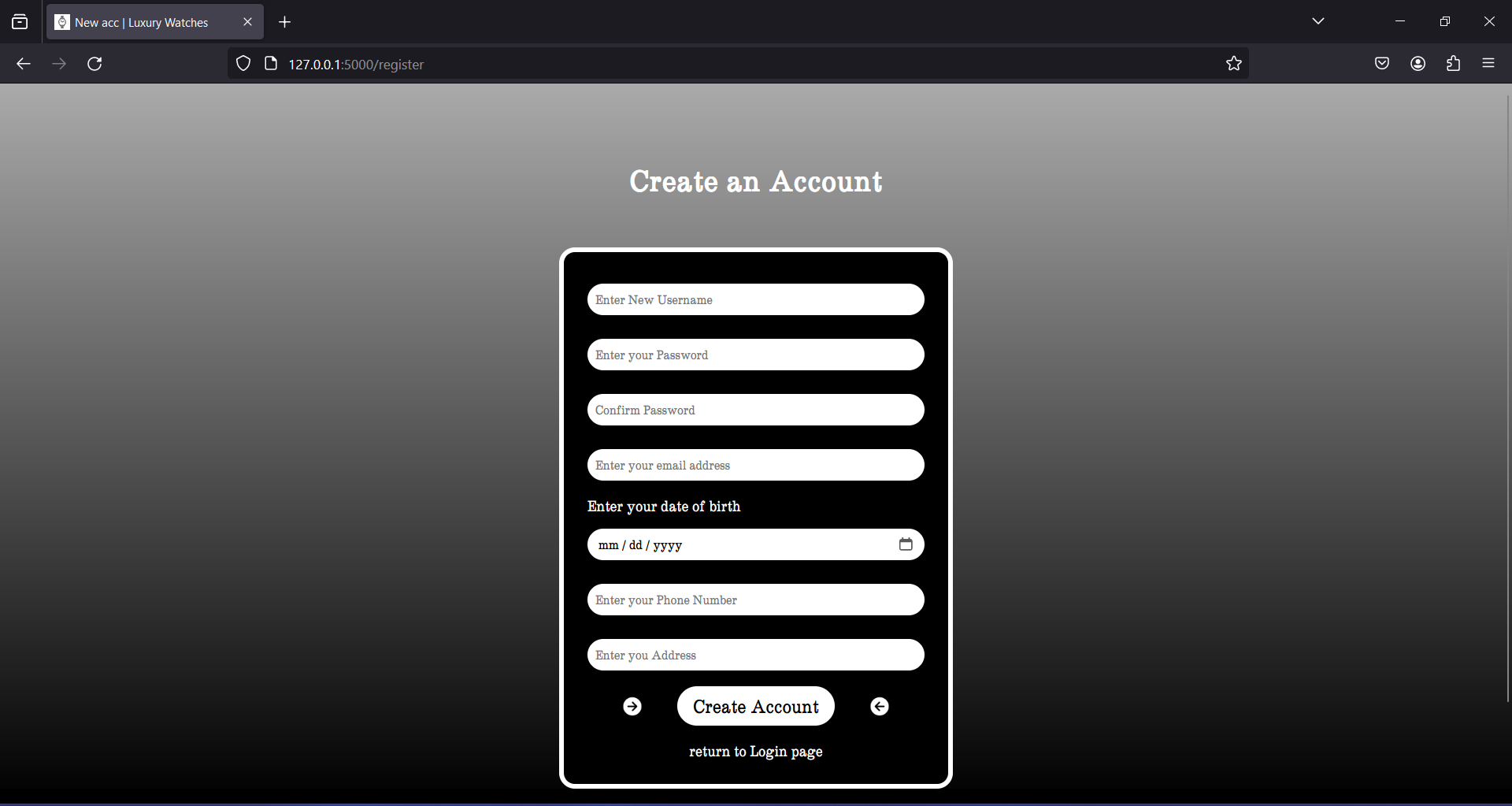
# **User Functionality Breakdown**

## **Login:**

****

As a registered user, enter your username and your password and click on the **‘Sign in’** button to login. If you’re not registered, click on the **‘Register as new User’** button, this will redirect you to the Register page. If you’re an admin, click on the **‘Login as Admin’** button, this will redirect you to the Admin Login page – mentioned later in the report.

## **Register:**

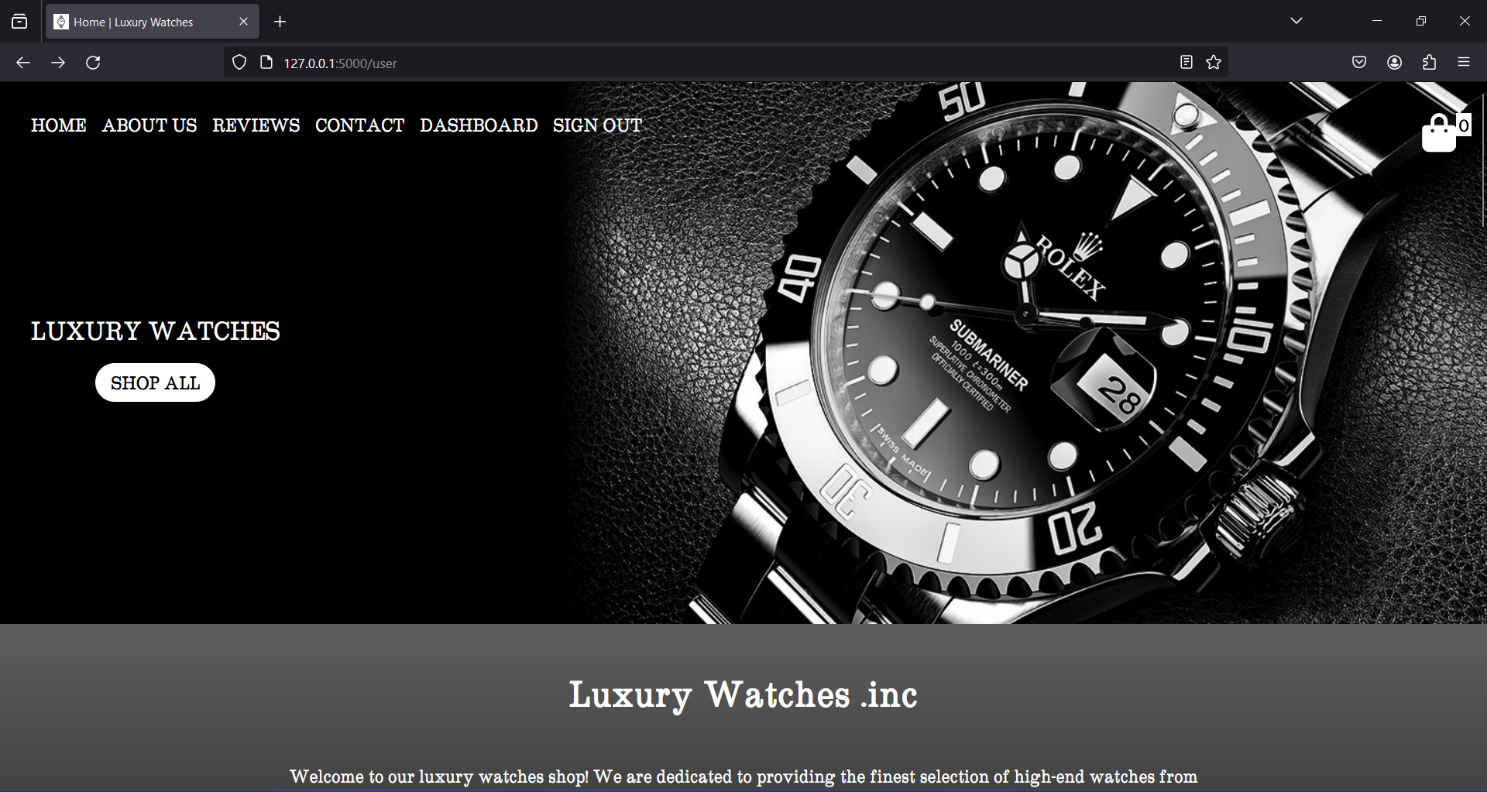


Begin by entering a username, a password, and confirm your password. If your username is not unique or your password is less than 8 characters, you will need to register again.

Enter your email address, your birth date, your phone number, and your address. If your phone number is not unique, you will need to register again.

Once you click on the **‘Create Account’** button, you should be redirected to the **Login** page, which means you’re registered successfully. If you don’t get redirected and you stayed in the **Register** page, you will need to register again.

## **User page:**

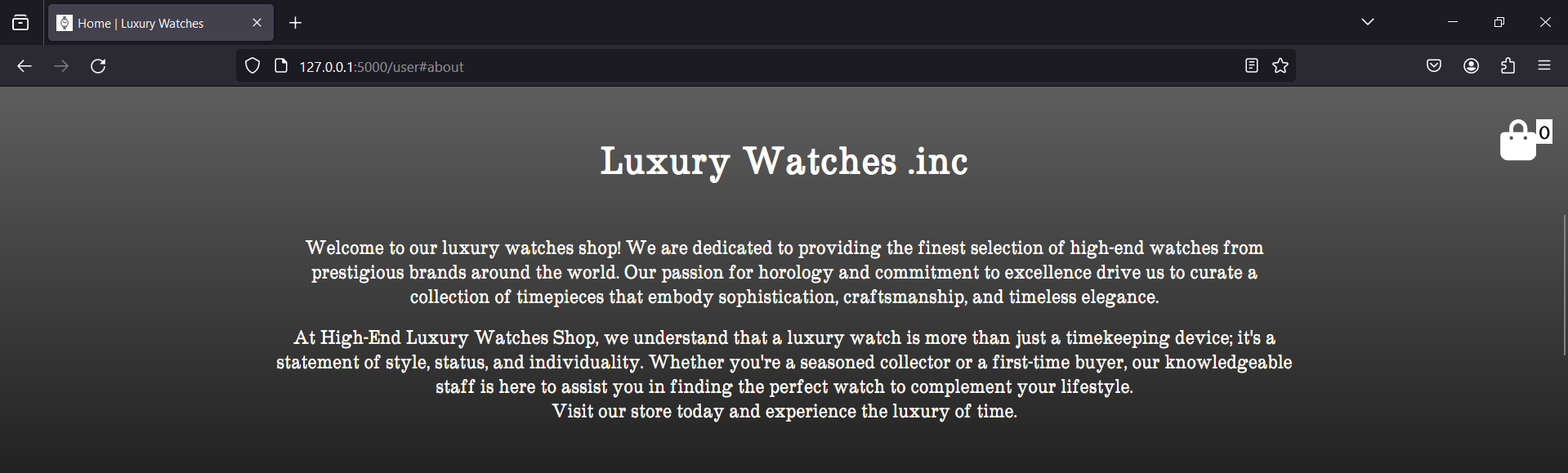


Once logged in, you will be redirected to the **User** page. Use the top menu to navigate through this page.

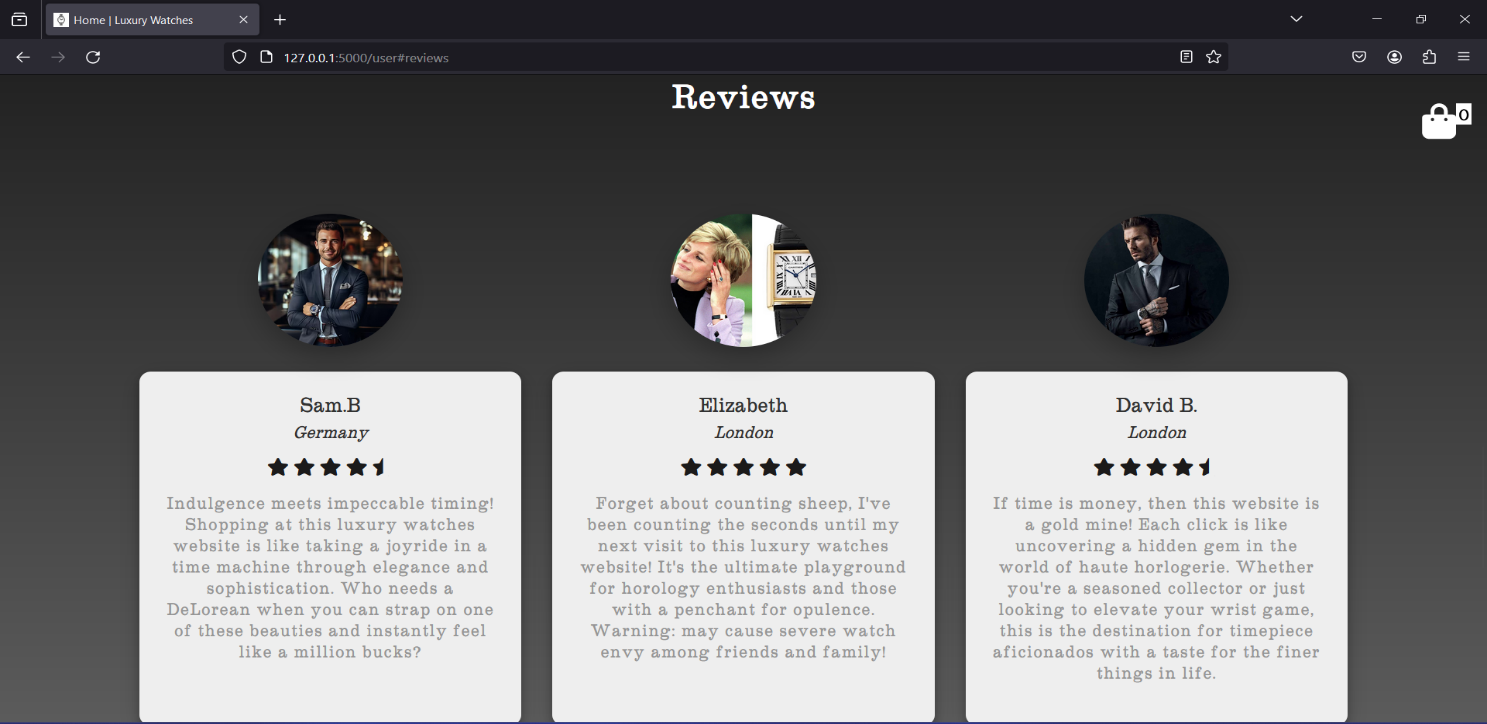
It contains the ‘HOME’ section – appearing in the photo, the ‘ABOUT US’ section which tells you more about the store, the ‘REVIEWS’ section which shows you our customer reviews, the ‘CONTACT’ section through which you can send inquiries or feedbacks to the store owner by email, the ‘DASHBOARD’ section which will redirect you to the **Dashboard** page – mentioned later in the report, and the ‘SIGN OUT’ button which will sign you out of your account.

The ‘HOME’ section contains a ‘SHOP ALL’ button that navigates you to the ‘PRODUCTS’ section – mentioned later in the report.

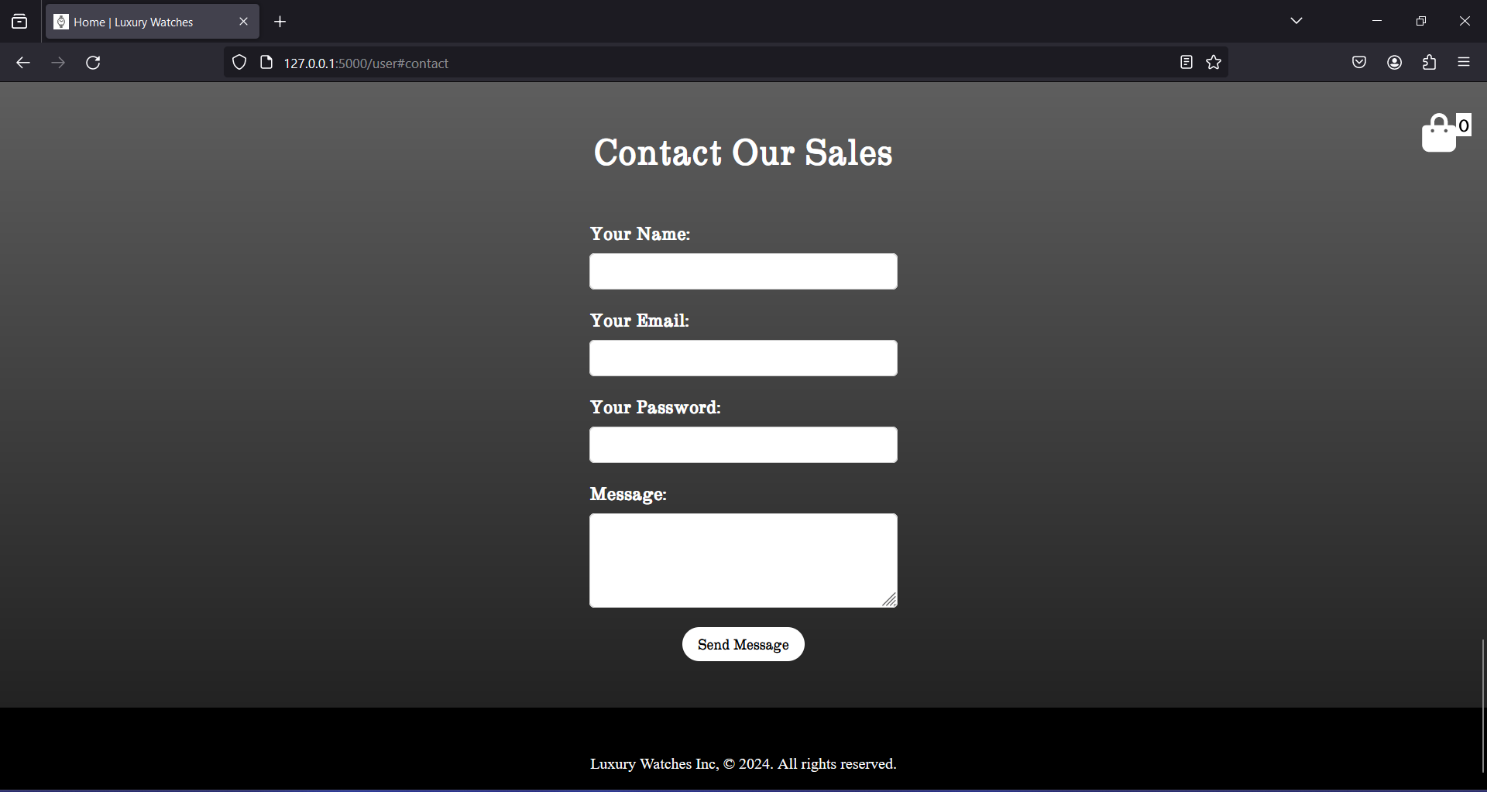
‘ABOUT US’:



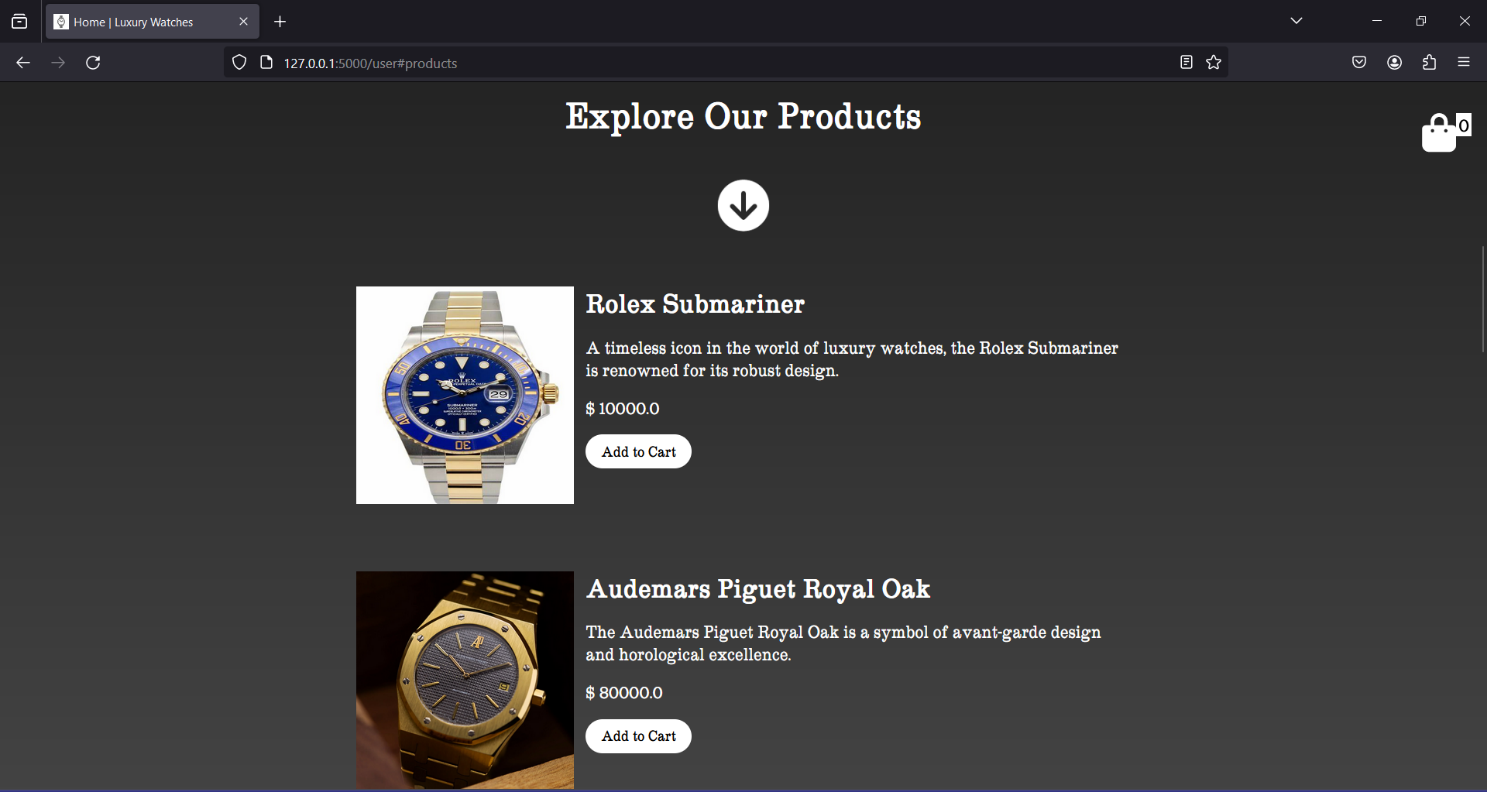
‘REVIEWS’:



‘CONTACT’:



‘PRODUCTS’:



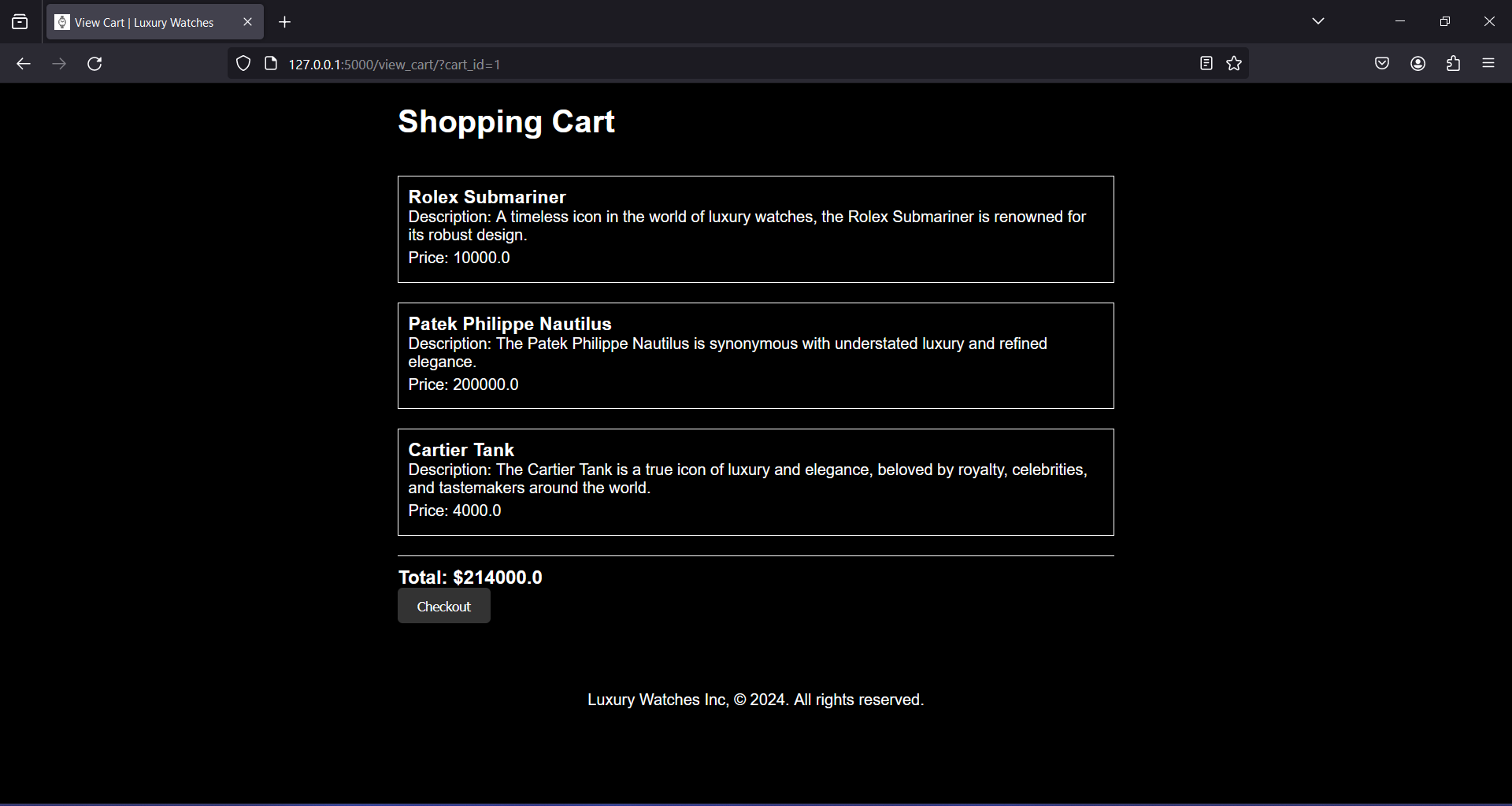
The user can buy the products using the ‘Add to Cart’ button. He can review his cart and checkout his orders in the ‘View Cart’ page by clicking on the top-right button in the **User** page – mentioned later in the report.

## **Dashboard page:**



In this page, the user can see graph insights about the price trends of our products, which can help him decide when and what product to buy.

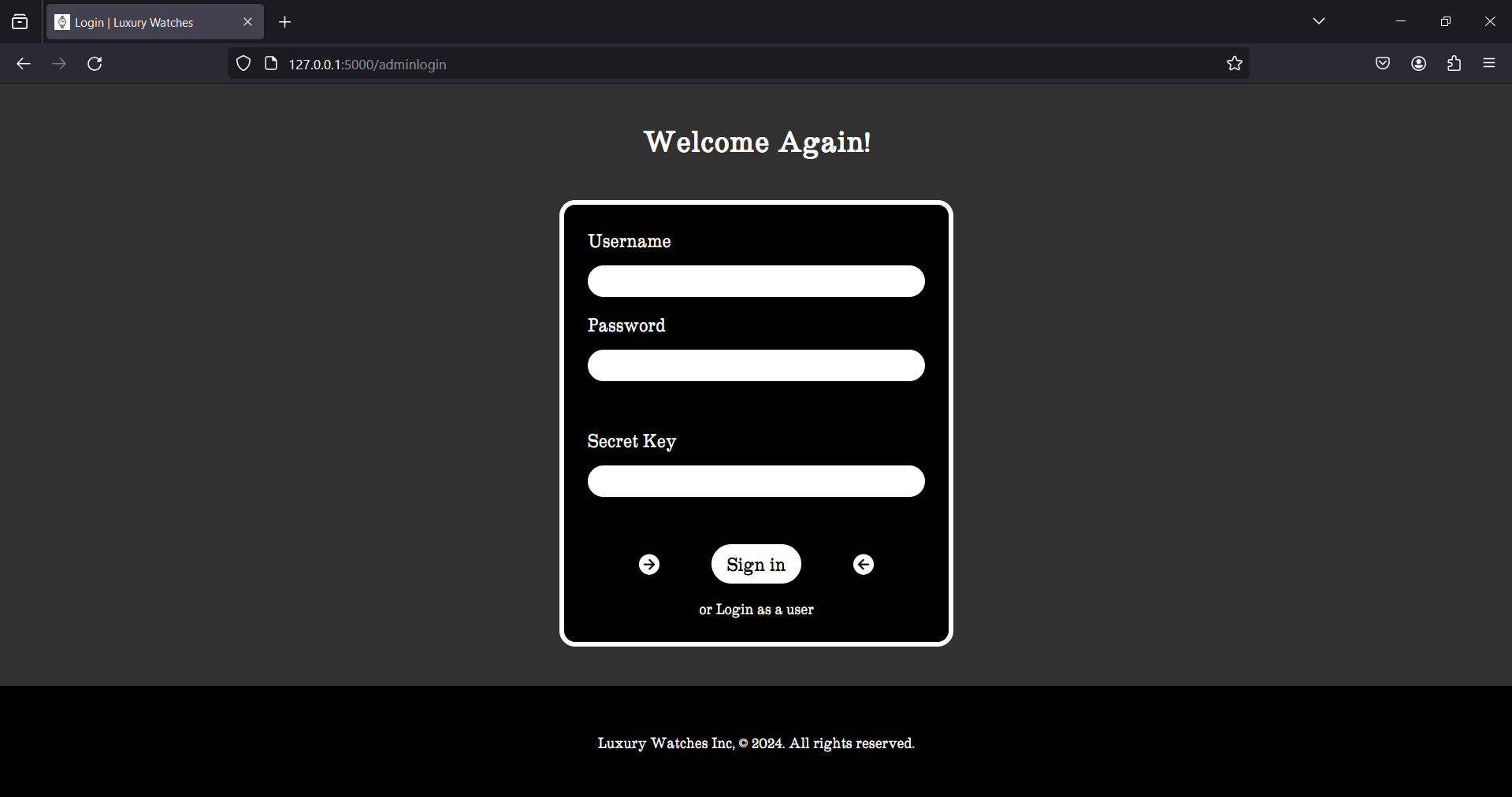
## **View Cart page:**



The user can see all of the items added to his cart in this page in addition to their total cost. The user can also checkout his payment through the checkout button.

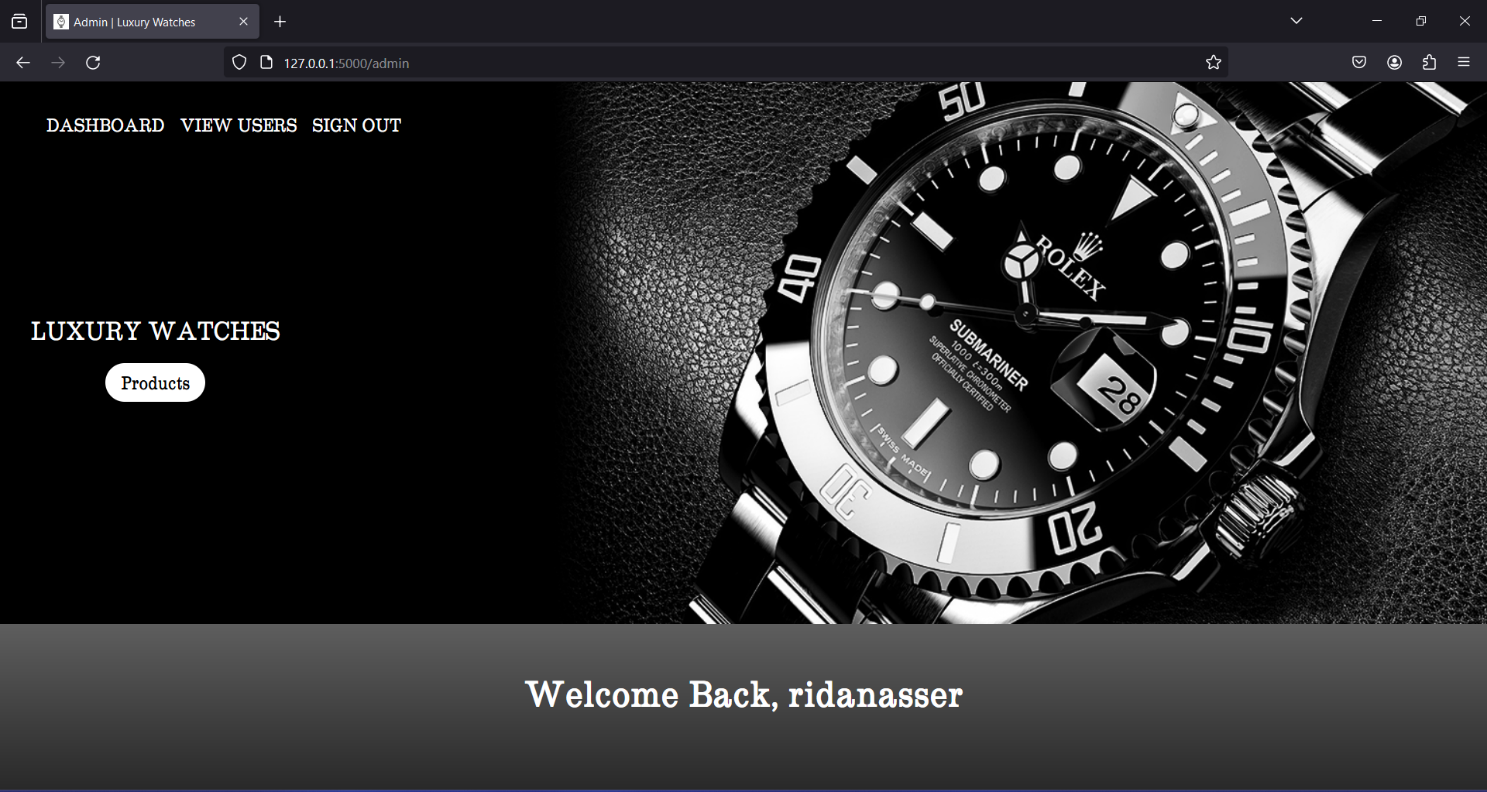
# **Admin Functionality Breakdown**

## **Admin Login:**



There is only one admin account. The admin can enter his username, his password, and the application’s **secret key** to login.

## **Admin Panel:**

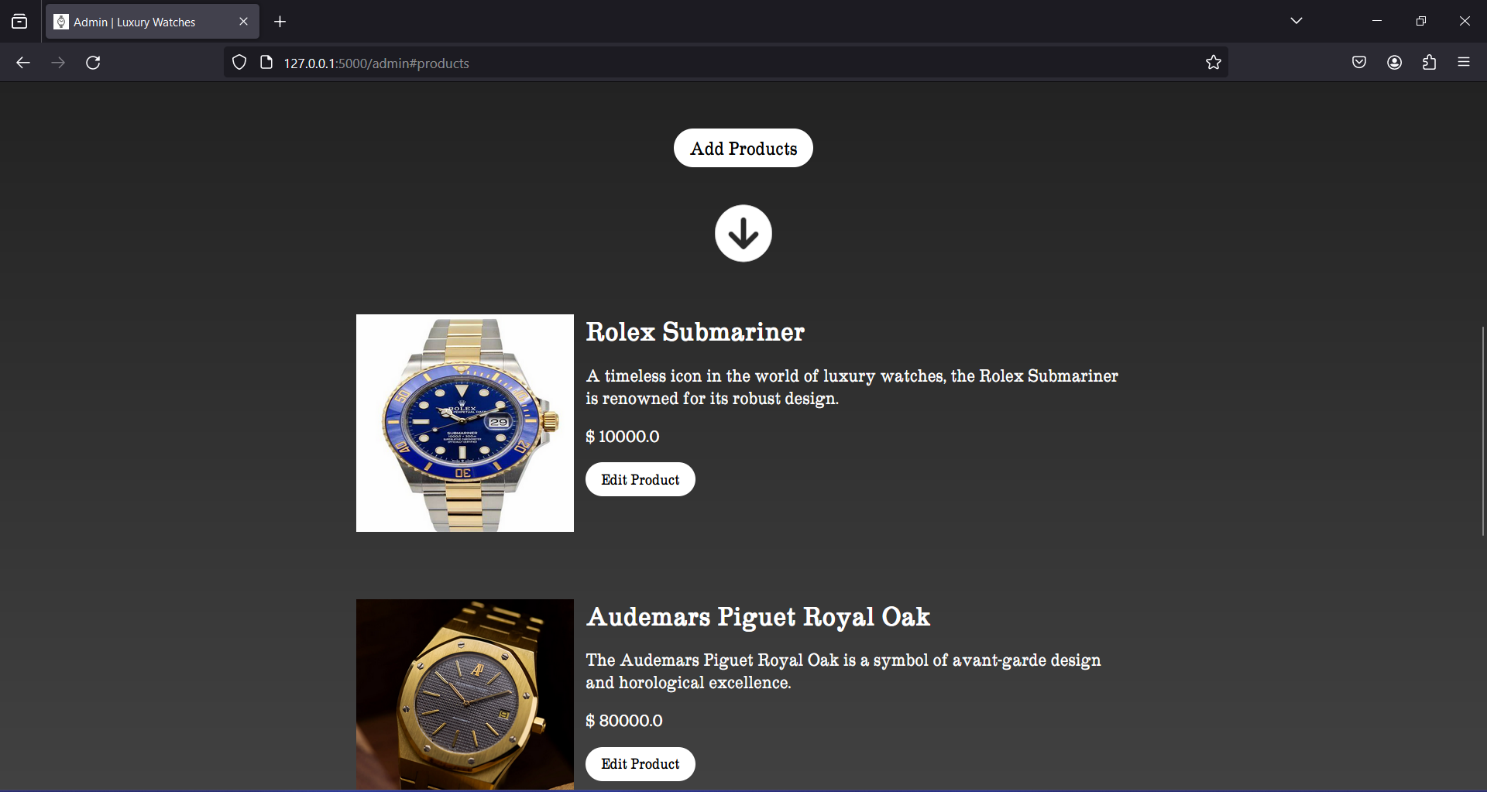


Once logged in, the admin will be redirected to the Admin Panel. He can use the top menu to navigate through this page.

It contains a ‘VIEW USERS’ section – mentioned later in the report, and the ‘DASHBOARD’ section and the ‘SIGN OUT’ button we mentioned before.

The Admin Panel also contains a ‘PRODUCTS’ section where the admin can control the products.

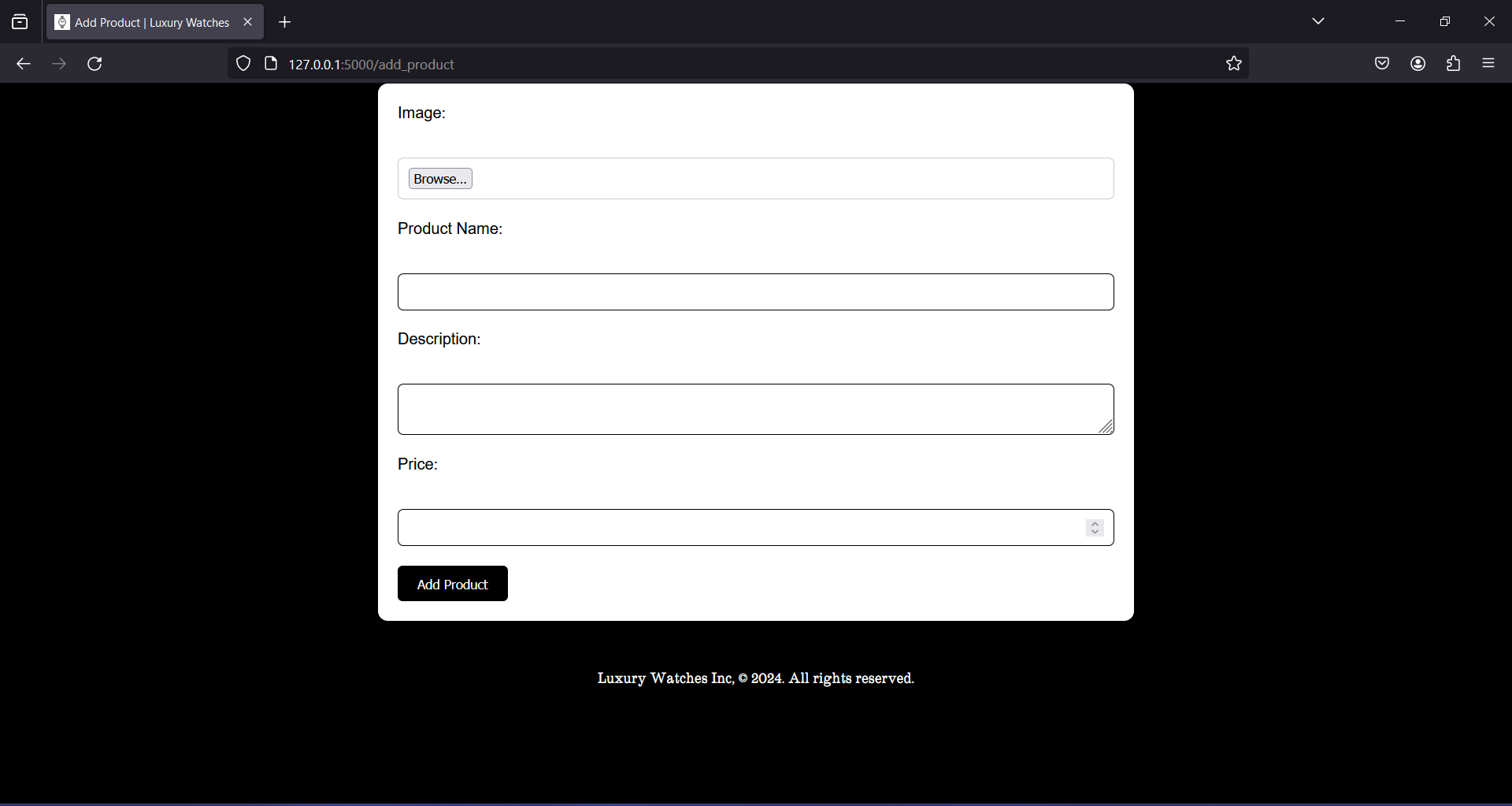
‘PRODUCTS’:



The Admin Panel also contains a ‘PRODUCTS’ section where the admin can control the products.

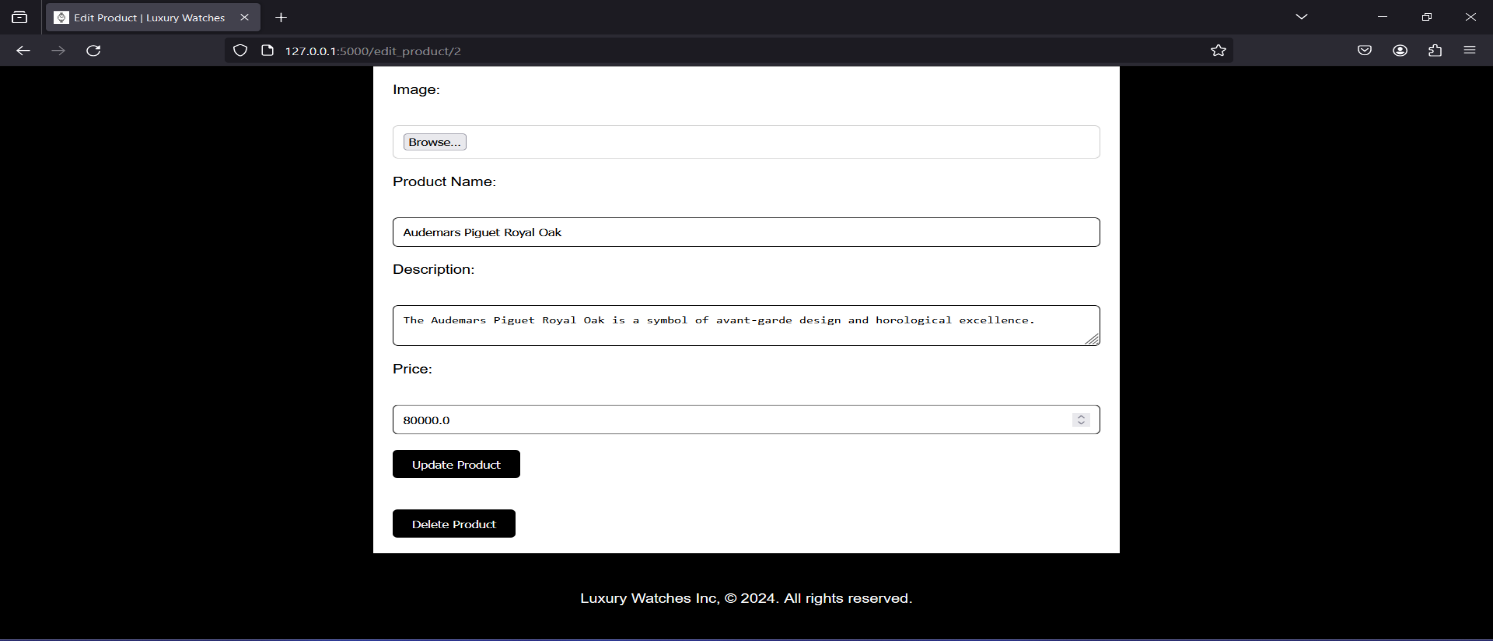
This section enables the admin to add, edit, and delete products.

## **Add Products:**



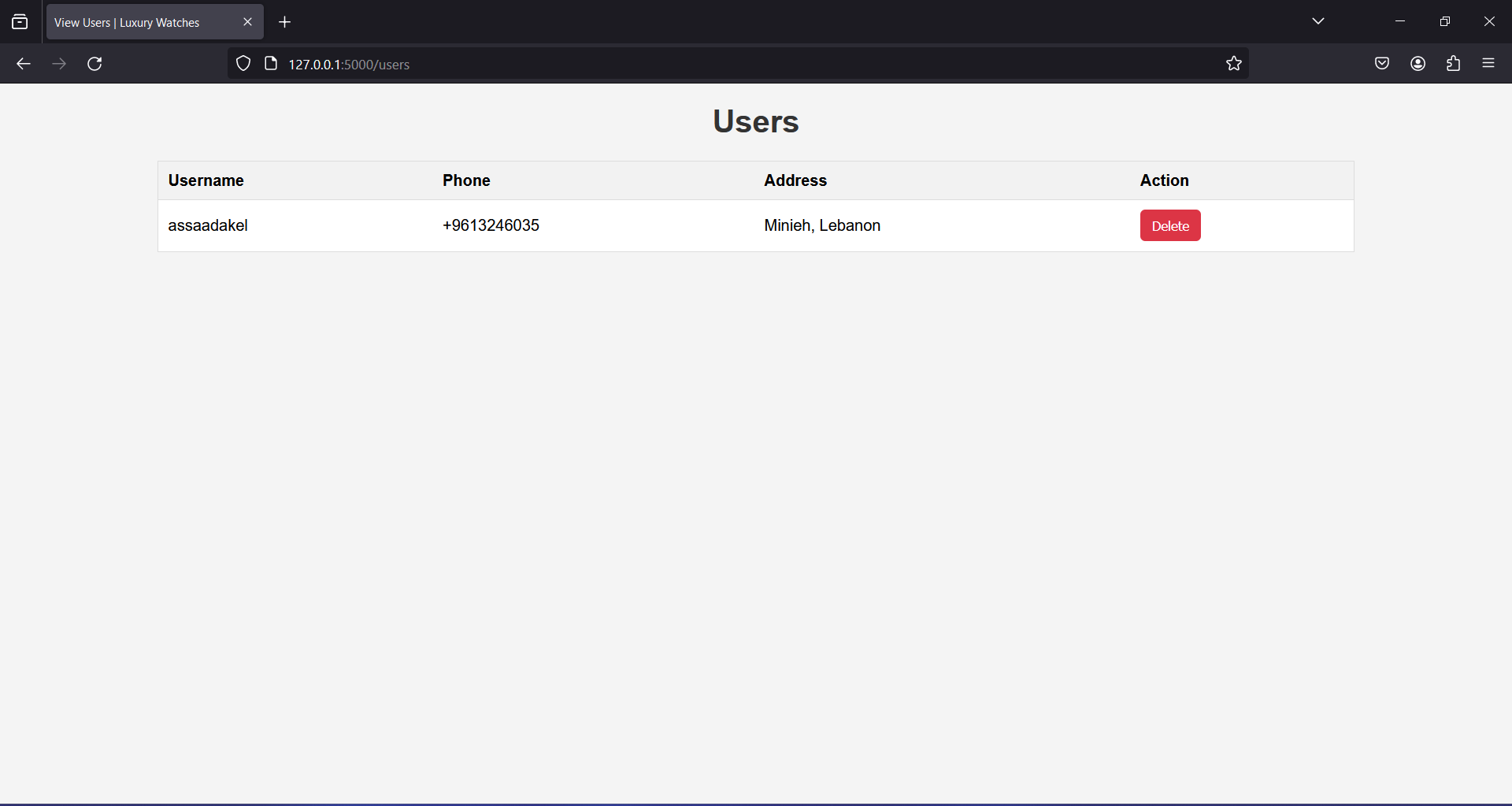
Once clicked on the ‘Add Products’ button, the admin will be redirected to this page. He can now choose an image, a product name, a description, and set the product’s price. Then after clicking on ‘Add Product’, the product will be added to the **Admin Panel** and to the **User** page.

## **Edit & Delete Products:**



When clicking the ‘Edit Product’ button, the admin will be redirected to this page where he can edit or delete the selected product. Any choice he’ll take will appear on the **Admin Panel** and on the **User** page.

## **View & Delete Users:**



In the ‘View Users’ page mentioned before, the admin can see all the registered users’ information, and can also choose to delete them from the database.

# **User Interface (UI) Details**

The user interface of the Flask e-commerce web application for the Luxury Watches Store is designed to provide an intuitive and visually appealing experience for both customers and administrators.

## **Login Page:**

The login page features a clean and minimalist design. Users are prompted to enter their credentials in clearly labeled input fields. Error messages are displayed prominently in case of invalid login attempts.

## **Registration Page:**

The registration page allows new users to sign up for an account. It presents input fields for username, password, phone number, and address in a structured layout. Feedback messages inform users about successful registrations or validation errors.

## **Product Management Pages:**

Pages for adding, editing, and deleting products offer intuitive forms for inputting product details. Admins can upload product images, specify names, descriptions, and prices, and save changes with minimal effort.

## **Shopping Cart:**

The shopping cart page displays a summary of items added by the user. Each product is listed with its image, name, description, and price. The total price of the items in the cart is prominently displayed, facilitating quick checkout decisions.

## **Contact Form:**

A dedicated contact form allows users to send inquiries to the store owner. The form prompts users to enter their name, email address, and message, providing a straightforward means of communication.

# **Business Logic Overview**

The Flask e-commerce web application incorporates robust business logic to facilitate seamless interaction between users and the store's inventory. Key aspects of the business logic include:

## **User Authentication:**

Secure login and registration mechanisms ensure that only authorized users can access the platform. Password validation and encryption enhance security, while session management tracks user activity throughout the session.

## **Product Management:**

The application enables administrators to efficiently manage the store's inventory. Admins can add, edit, and delete products, ensuring that the catalog remains up-to-date with the latest offerings. Product details are stored in a relational database, allowing for easy retrieval and manipulation.

## **Shopping Cart Operations:**

Users can add products to their shopping carts and view the contents before proceeding to checkout. The application validates user actions, such as adding duplicate products or attempting to access the cart without logging in, to maintain data integrity and security.

## **Email Communication:**

The contact form facilitates communication between users and the store owner. Upon submission, the application sends email inquiries to the designated recipient using SMTP servers from popular email providers. Error handling ensures that email delivery failures are appropriately handled.

# **Conclusion**

In conclusion, the Flask e-commerce web application represents a significant achievement in leveraging modern web technologies to create a dynamic and scalable platform for luxury watch enthusiasts. With a solid foundation in place, the application is poised for continued evolution and success in the competitive e-commerce landscape. However, it was not without its challenges:

## **Challenges Faced:**

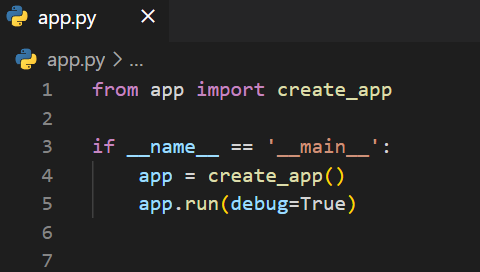
The project introduced new technologies and concepts, requiring a steep learning curve for implementation. Debugging errors and ensuring the seamless integration of various components posed additional challenges. Continuous learning and adaptation were essential to overcoming these obstacles.

## **Future Enhancements:**

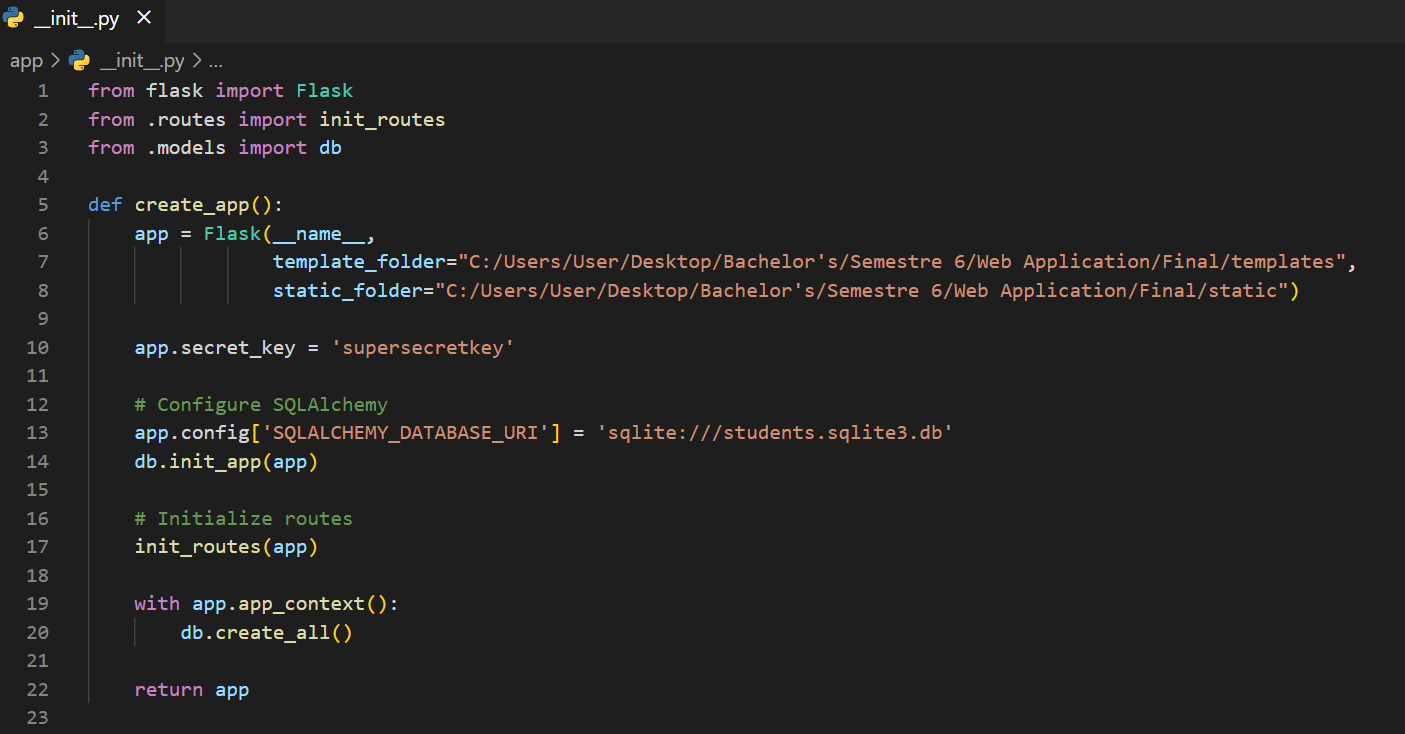
Integration of payment systems such as PayPal, Visa, or Stripe will enable secure and convenient transactions. Enhancements to the overall web application, including performance optimizations, UI refinements, and additional features, will further elevate the user experience and drive business growth.

# **Appendix (Code Snippets)**

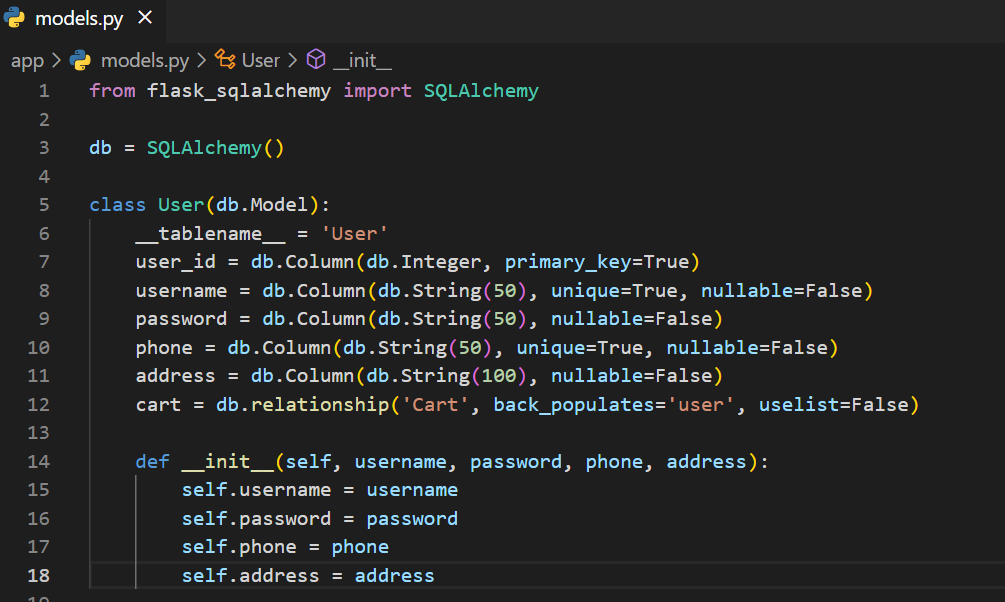
app.py:

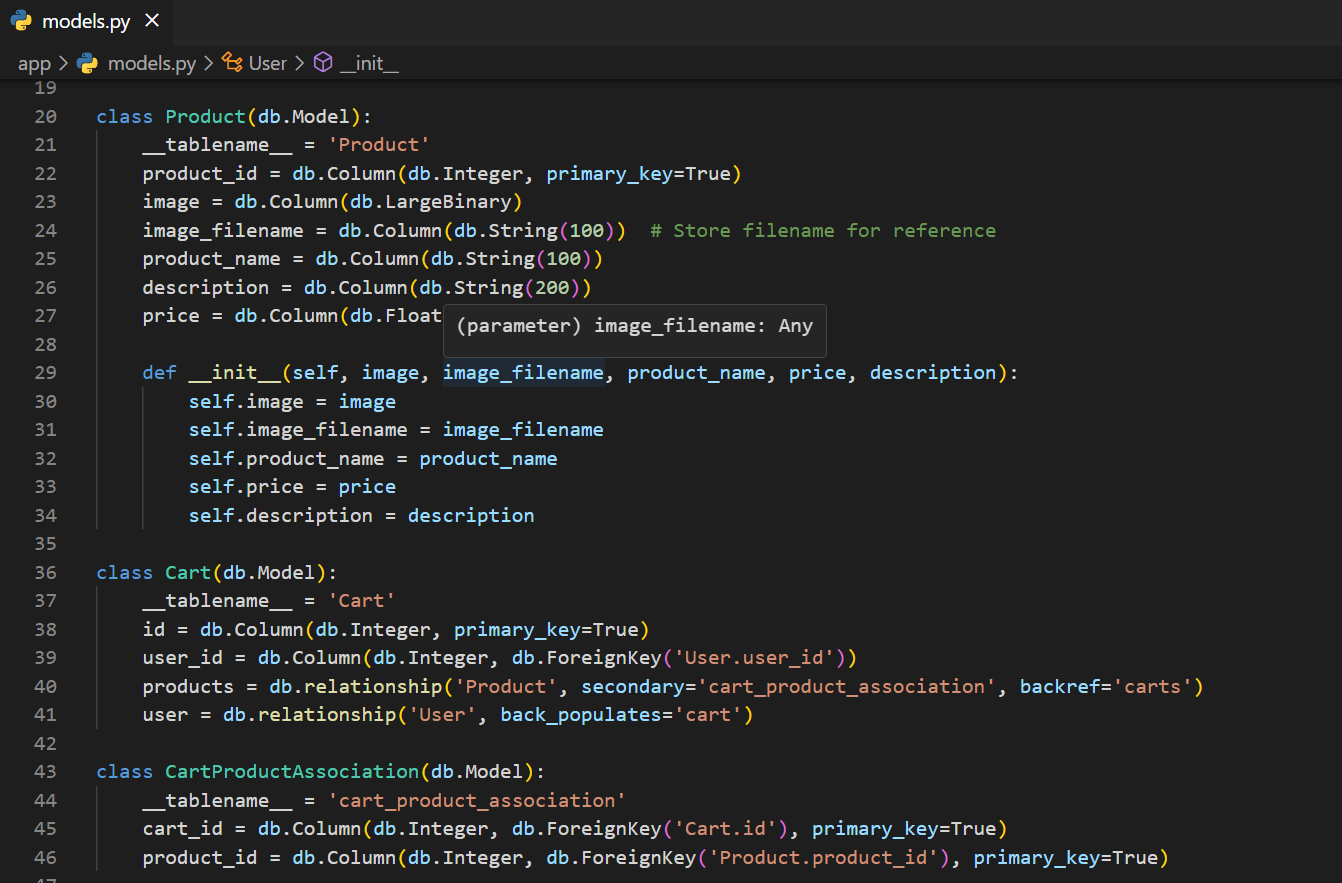
****

\_\_init\_\_.py:



models.py:





routes.py:

