# **Sam Shoes E-Commerce Web Application**

Contents

[**Sam Shoes E-Commerce Web Application** 1](#_Toc182748087)

[**1.** **Requirement Analysis** 2](#_Toc182748088)

[**1.** **System Design** 4](#_Toc182748089)

[**5.** **Designing Phase** 6](#_Toc182748090)

[**6.** **Development Phase** 10](#_Toc182748091)

[**7.** **Testing Phase** 15](#_Toc182748092)

[**8.** **Deployment and Maintenance** 19](#_Toc182748093)

**Project Description**

The Sam Shoes E-Commerce Application is a feature-rich web platform that was constructed with Bootstrap for the front end and Django for the back end. The application, which was created especially for shoe sales, serves administrators (store managers or owners) as well as end users (customers). It creates a smooth shopping experience by integrating key e-commerce elements like product administration, dynamic cart functionality, user authentication, and a checkout system.   
Users may easily manage their shopping carts, examine comprehensive product descriptions, and peruse an enormous shoe inventory because to this application's scalability and user-friendliness. Product inventory management tools, such as the ability to add, update, remove, and export product details in CSV format, are available on the administrator side.

**User Login and Signup**: Among the primary features are password encryption and secure authentication with session management.   
**Product Catalog**: Offers simple navigation by listing every item that is accessible along with search and filter capabilities.   
**Dynamic Cart Management** allows users to dynamically show the current total, add items to their cart, and modify quantities.   
**Product Details Page:** Provides comprehensive information about each product, including size, color, cost, title, and description.   
The checkout page provides a summary of the items selected, their quantities, and the total cost before the user completes the purchase.   
In order to enhance user experience and ensure appropriate operation on PCs, tablets, and mobile devices, the application makes use of **Bootstrap 5** for responsive design and aesthetics.

The application prioritizes both functional and non-functional needs to guarantee a system that is reliable, secure, and effective:   
**Security**: Uses industry-standard security protocols, such as input validation, encrypted passwords, and defense against common vulnerabilities like cross-site scripting (XSS) and SQL injection.   
Performance: Designed to ensure a seamless shopping and browsing experience by optimizing page loads.   
**Scalability**: It is appropriate for both small-scale organizations and larger enterprises because it is built to manage growing user traffic and product data.   
Sam Shoes offers a **user-friendly** and robust platform that is customized to meet your needs, whether you are a customer trying to buy your favorite shoes or an administrator overseeing inventory. The application's capabilities will be further enhanced in the future by adding features like payment gateway integration, tailored recommendations, and advanced analytics, which will better meet the ever-changing needs of the e-commerce sector.

## Requirement Analysis

**Functional Requirements**

1. **Verification of the user**   
     
   Strong user authentication features are included in the system to guarantee administrators' and customers' safe access. Through a signup procedure, users may create an account by providing basic information such as their username, email address, and password. Encrypted password hashing algorithms are used to safely store these credentials.  
   After registering, users may access their accounts through a login process that verifies the information they have supplied against a database. By securely preserving user activity, a session management system makes sure that users remain signed in until they decide to log out. Important functionality including product catalogs, cart management, and checkout are inaccessible to unauthenticated users.

In order to maintain a safe and well-regulated platform, administrators may also keep an eye on and control user behavior.

**Catalog of Products**   
  
Users may browse, search, and examine a variety of shoe goods in the product catalog, which is the central component of the e-commerce platform. Every product listing contains thorough details like:

* + Title
  + Description
  + Price (retail and wholesale)
  + Size
  + Color
  + Available Stock
  + Product Image

**Administrators** can carry out **CRUD** tasks since they have complete authority over the product catalog:   
**Add**: Admins have the ability to add new items with all required information.   
**Edit**: The catalog may be kept up to date by easily updating existing product data.   
**Delete**: If a product is discontinued or sold out, it may be deleted.   
**Export**: For reporting, backup, or inventory management needs, the complete product list may be exported to a CSV file.   
Users may easily locate the goods they require thanks to the catalog's user-friendly search and filtering features. Before making a purchase, clients may make educated judgments thanks to product descriptions and photographs.

**The functionality of the cart**:  
Users can dynamically manage their selections thanks to the cart capability. From the catalog or product details pages, users can add items to their cart by indicating the amount they would like. Users can view a list of all selected items, their quantities, and individual prices after adding things to their cart.

o Modify their order before checking out by updating the quantity of items in real time.   
o Take away things they don't want to buy.   
An asynchronous updating mechanism powers the cart functionality, guaranteeing real-time responsiveness. For instance, when a user click add to cart button a message with product title will be appeared with button and the cart icon dynamically displays the current item count whenever an item is added, removed, or altered in the cart. The purchasing experience is smooth and engaging thanks to this function.

1. **Checkout Page**  
   The checkout page serves as the final step in the purchasing process, summarizing the user’s selected items for review. It displays a detailed breakdown of the cart contents, including:
   * Product names and descriptions.
   * Quantities and unit prices.
   * Subtotals for each product.
   * Change Product Quantity using an input value and update button.
   * Delete the product from cart with delete button in front of each item.

The checkout page is designed to minimize errors, allowing users to make final adjustments to their cart if needed.

Although payment integration is not currently included, the checkout system is structured to allow easy future enhancements, such as adding payment gateways (e.g., PayPal, Stripe) or implementing order confirmation emails.

By addressing these requirements in detail, the system ensures a user-friendly and robust shopping experience, supporting both the needs of the end-users and the operational demands of the administrators.

**Non-Functional Requirements:**

1. Performance: The system should load pages within 2 seconds under normal load.
2. Scalability: The application should support concurrent users.
3. Security: Implement HTTPS, secure password storage, and input validation.

## System Design

**High-Level Design (HLD)**

1. **Database Design:**

**Users Table**: `id`, `username`, `email`, `password`

**Products Table**:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| title | description | size | color | retail price | Wholesale price | stock | image |
| Bravo Shoes | A unique shoes for your unique look | UK-42 | Black | 1200 | 1000 | 12 | Img |

1. **API Endpoints:**

'' (Index Route)

'admin/' (Admin Route)

   'signup/' (Signup)

   'loginPage/' (Login Page)

   'login' (Login)

   'logout' (Logout)

(Note: Below Require Authorizations to access the pages and data)

'store/home/' (Home)

'store/view-product/<int:product\_id>' (View Product)

'store/about-us' (About Us)

'store/blog' (Blogs)

'store/contact\_us' (Contact Us)

'store/add-to-cart' (Add to Cart)

**Low-Level Design (LLD)**

**Workflow for Adding Items to the Cart**

1**. User Interaction on the Product Page**

* + On the product details page, the user is presented with comprehensive information about the product, including its title, description, price, size, color, and availability.
  + To purchase the product, the user clicks the prominently displayed "Add to Cart" button. This button is intuitive and easily accessible, ensuring a hassle-free user experience.
  + If user want to add the same product for different quantity it would be easy by just clicking the add to button again.

2**. Request Generation and Backend Processing**

* + Upon clicking the "Add to Cart" button, a request is generated and sent to the server using an AJAX or fetch call. This request includes:
  + The product ID: Uniquely identifies the product the user wants to add.
  + The quantity: Specifies how many units of the product the user wants to purchase.
  + The user session data: Associates the cart item with the current user’s session, ensuring that the cart is personalized and secure.
* On the server side, the request is processed by the relevant view function in Django. The function performs the following actions:
* Retrieves the product details from the database using the provided product ID.
* Checks the availability of the product in stock. If the desired quantity exceeds the available stock, an error message is returned to the UI.
* Updates the user's cart in the session or database by adding the product and its quantity. If the product is already in the cart, the quantity is incremented accordingly.

3. **Real-Time UI Updates**

* Once the backend successfully processes the request, a response is sent back to the front end. This response typically includes:
* A confirmation message indicating that the item was successfully added to the cart.
* The updated cart count, reflecting the total number of items currently in the cart.
* On receiving this response, the user interface dynamically updates to reflect the changes without requiring a page reload. The cart icon, visible on the navigation bar across all pages but count only showing on home page , is updated to display the latest item count.

4. **Enhanced User Experience**

* To enhance usability, the system may also display a small popup or modal window confirming that the item has been added to the cart. This confirmation may include a quick view of the cart’s contents, providing users with an overview of their selections without navigating away from the product page.
* Users can choose to continue shopping or proceed directly to the cart page by clicking a "Cart" icon button in navbar.

5. **Data Persistence and Scalability**

* The cart data is stored securely in the user’s session or linked to their account if they are logged in. This ensures that the cart persists even if the user navigates to different pages within the application unless the user log out.

This detailed workflow ensures that adding items to the cart is both efficient and user-friendly, providing immediate feedback and maintaining the integrity of the shopping experience.

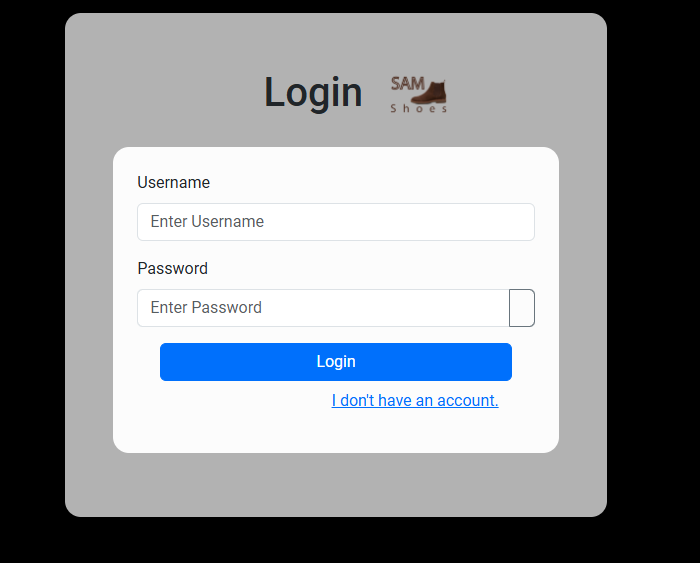
## Designing Phase

**UI/UX Design**

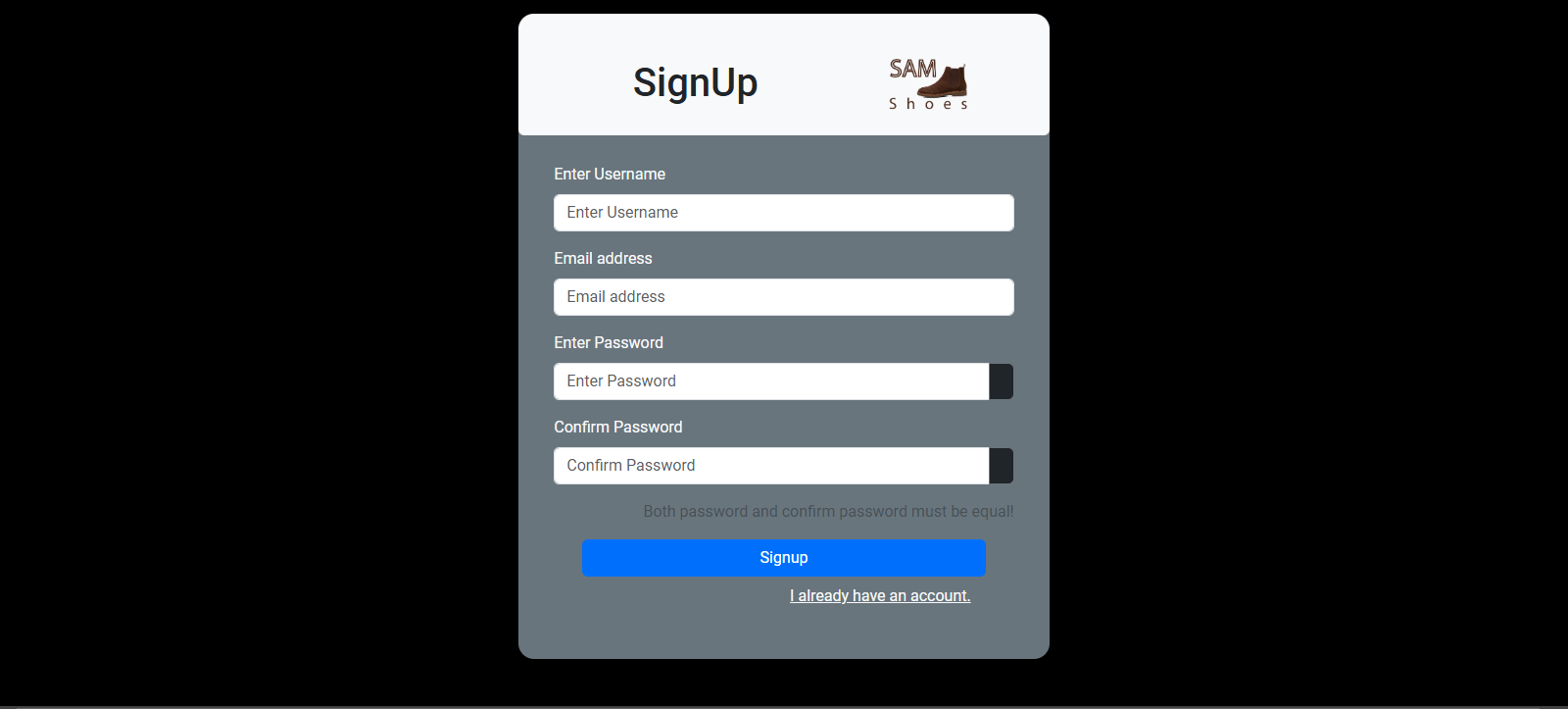
* 1. **Signup Page:** Signup
  2. **Login Page:** Login
  3. **Home Page**: Display featured products with navbar including menu items and cart.
  4. **Product Page**: Show details of each product with an add to cart button.
  5. **Cart Page**: Display selected items their quantities and each item total price.
  6. **Blog Page**: Showing Blogs
  7. **Contact Us Page**: Showing contact form.
  8. **Error Page**: An error Page for showing different kinds of errors.
  9. **About Us Page**: Showing about us info

**Tools for Designing:**

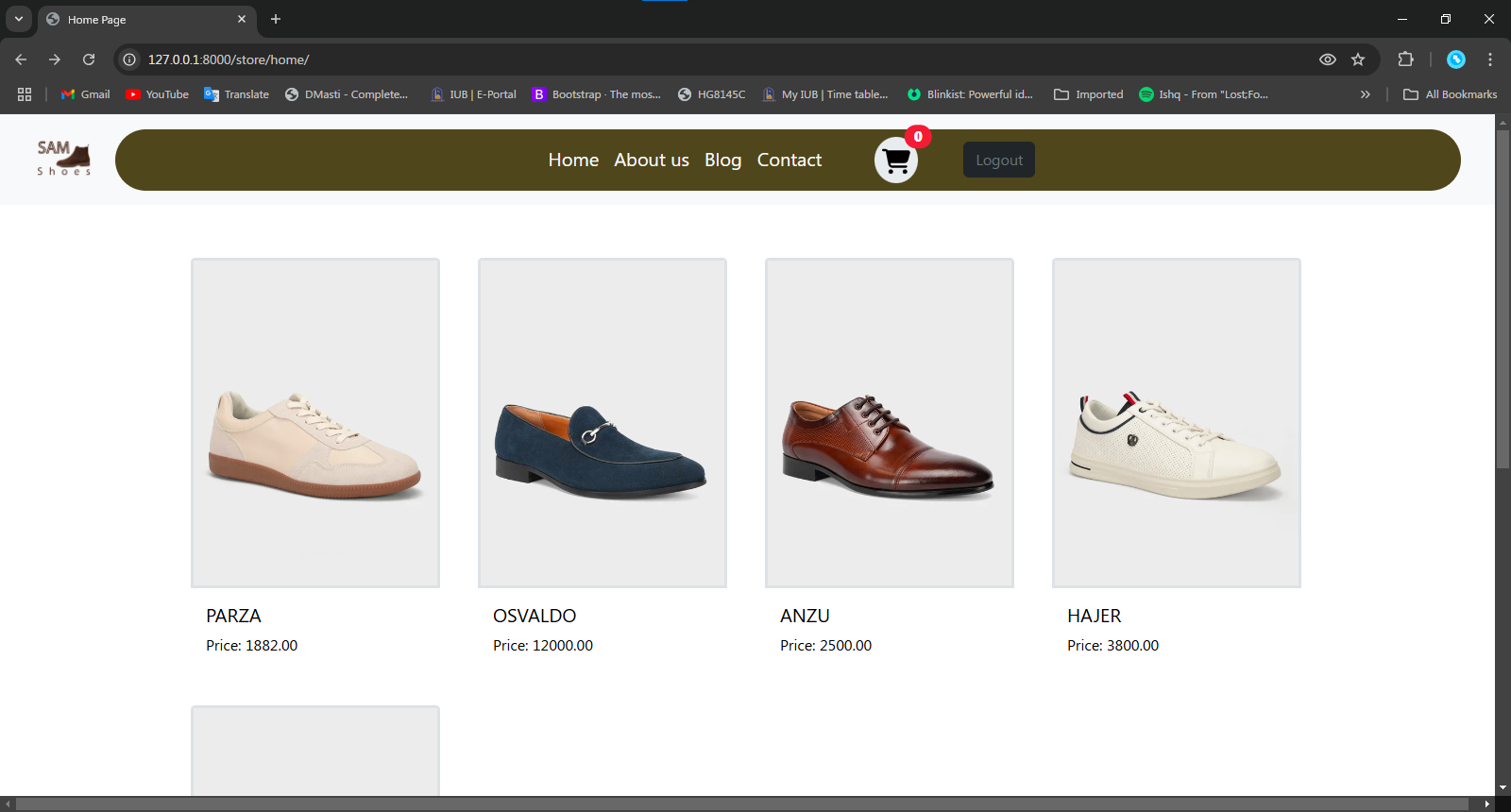
1. Figma
2. Adobe Photoshop
3. Adobe Illustrator

****

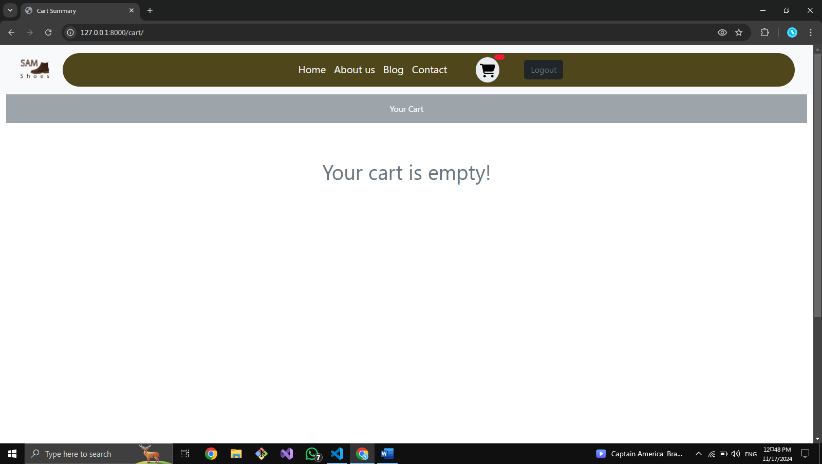
Login Page

****

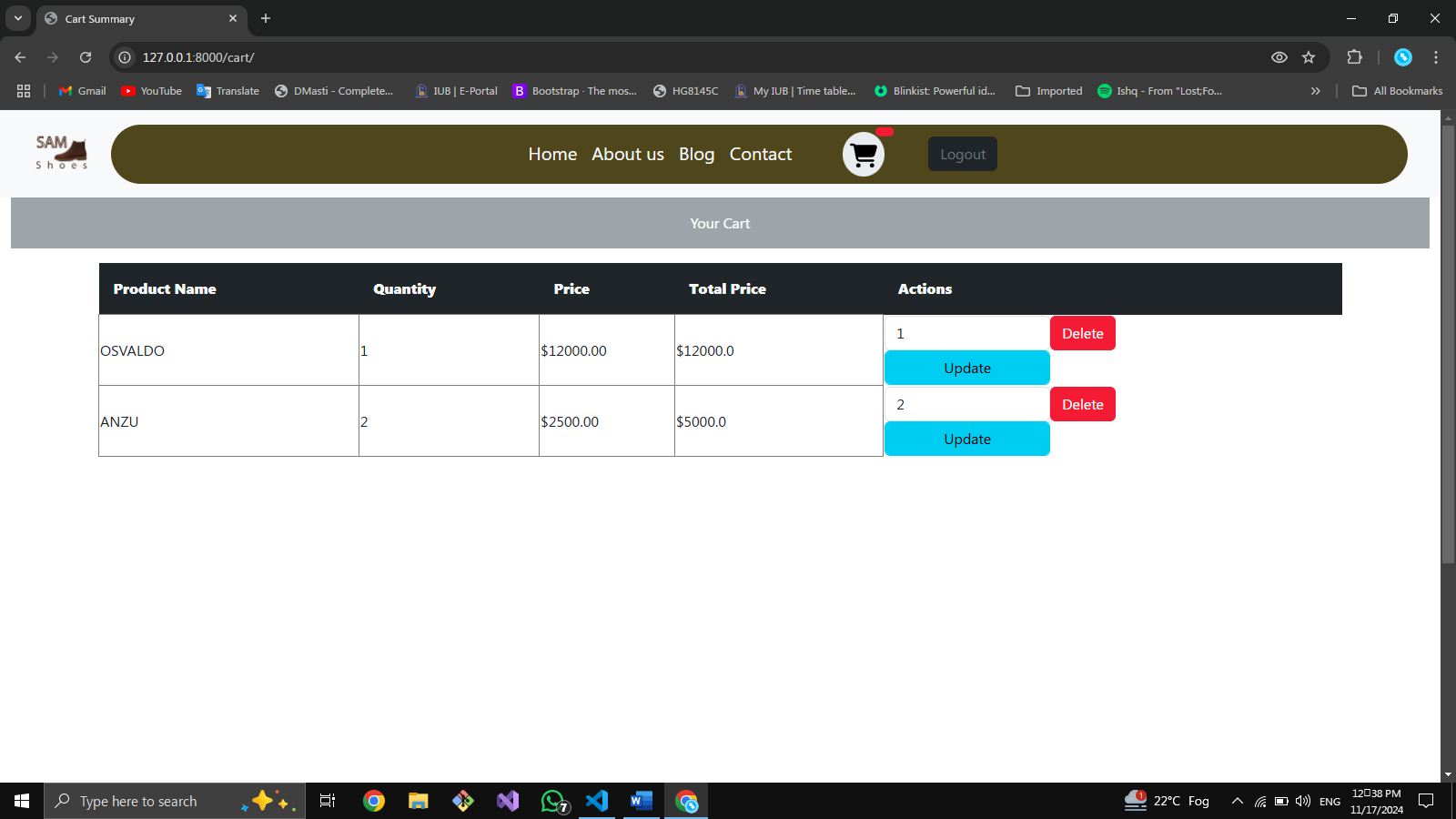
Signup Page



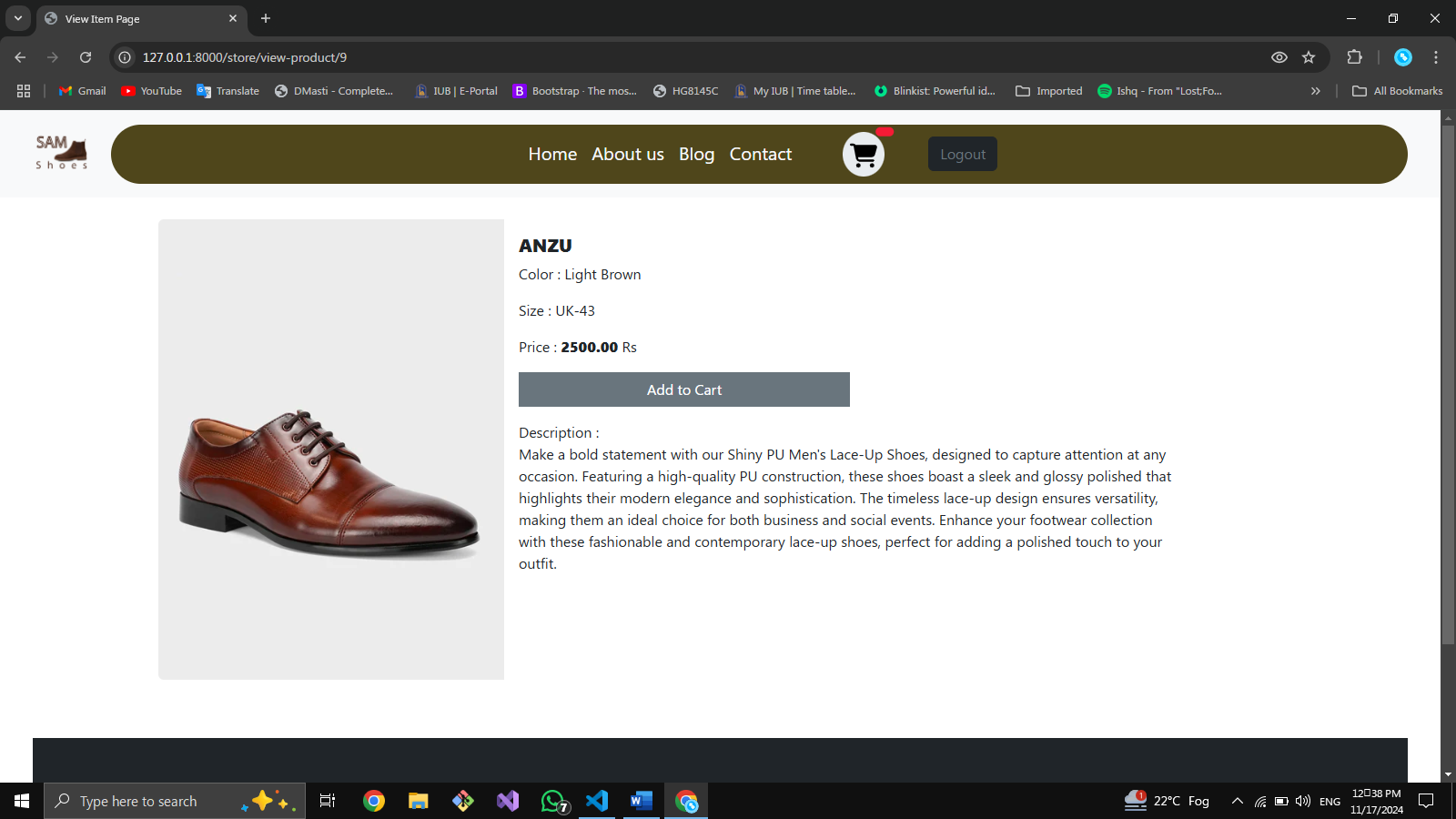
Home Pages



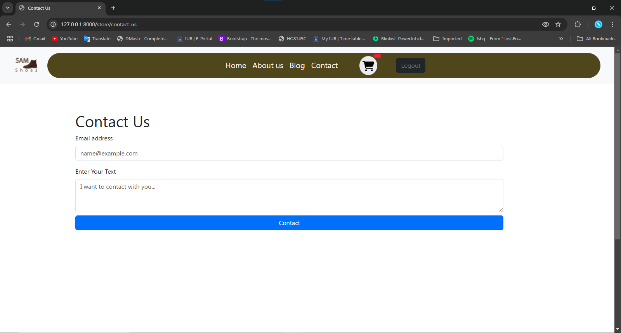
Cart Empty Page

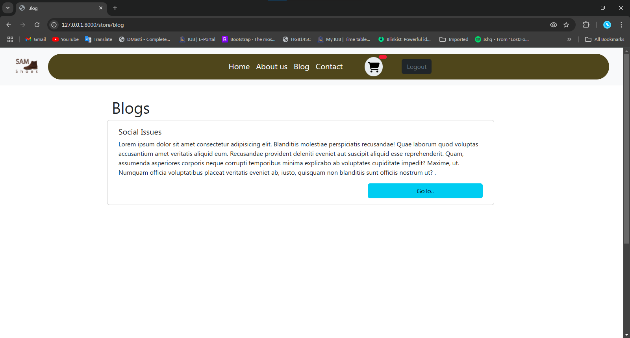


Cart Page



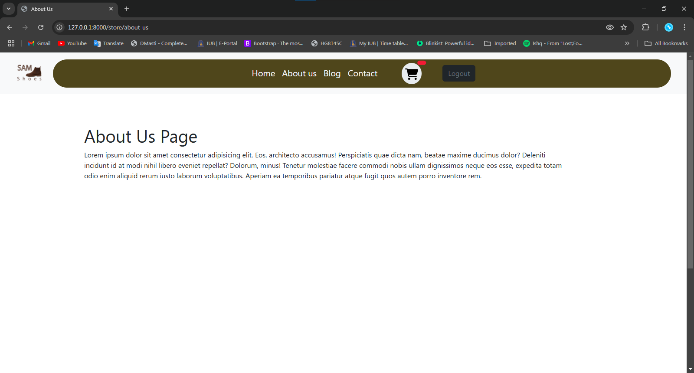
View Product Details Page





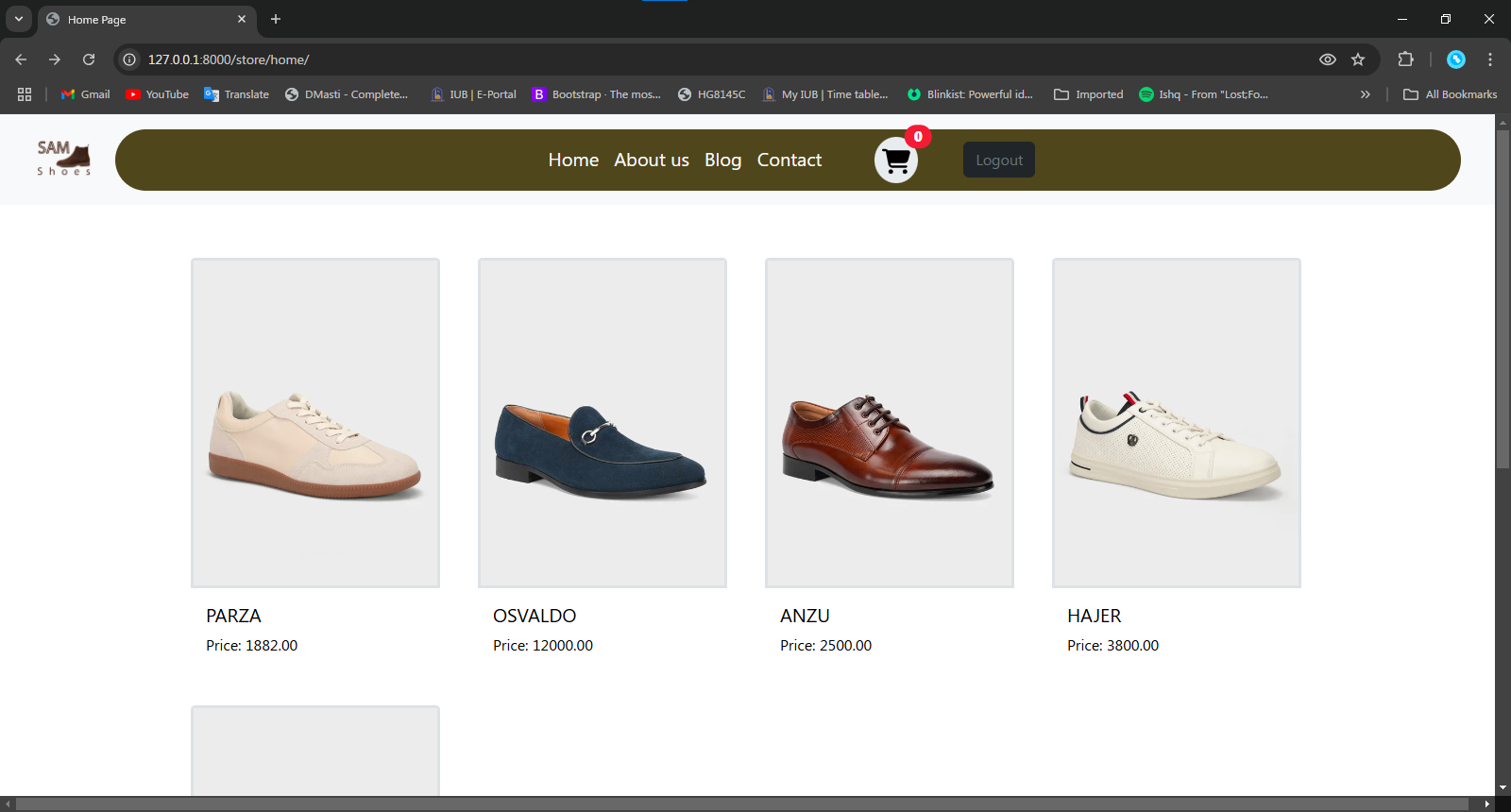
Contact Us page

Blogs Page



**Design Rationale:**

**Home Page**: Simple and clean layout for easy navigation.



**Navigations Includes:** Home, Blog, Contact Us, About Us, View Cart etc.

**Cart Icon**: Displays the current item count for user convenience.

## Development Phase

**Environment Setup**

**Requirements**

1. Python, Django, Bootstrap 5, SQlite

**Steps**

1. Initialize the Django project: django-admin startproject sam\_shoes.
2. Integrate Bootstrap 5 for frontend styling.
3. Configure static file management using the WhiteNoise package for efficient development and deployment.

**Feature Implementation**

**Authentication Module**

**Login and Signup Views**:  
The authentication module utilizes Django's built-in authentication system for secure and efficient user management. This includes the login, logout, and signup functionalities, all integrated into the application.e.g.

python

* from django.contrib.auth import login, authenticate
* from django.shortcuts import render, redirect etc.

def **login**(request):

if request.method == 'POST':

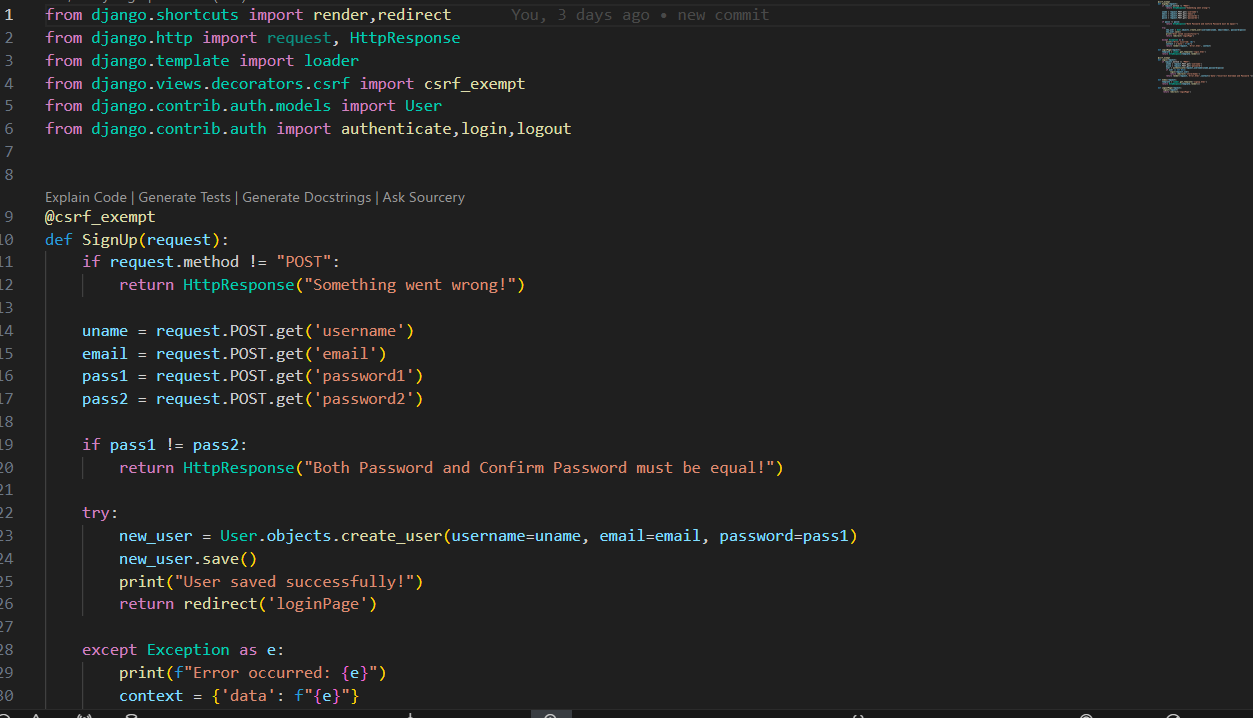
# handle login

def **signup**(request):

if request.method == 'POST':

# handle signup

Example screenshots:

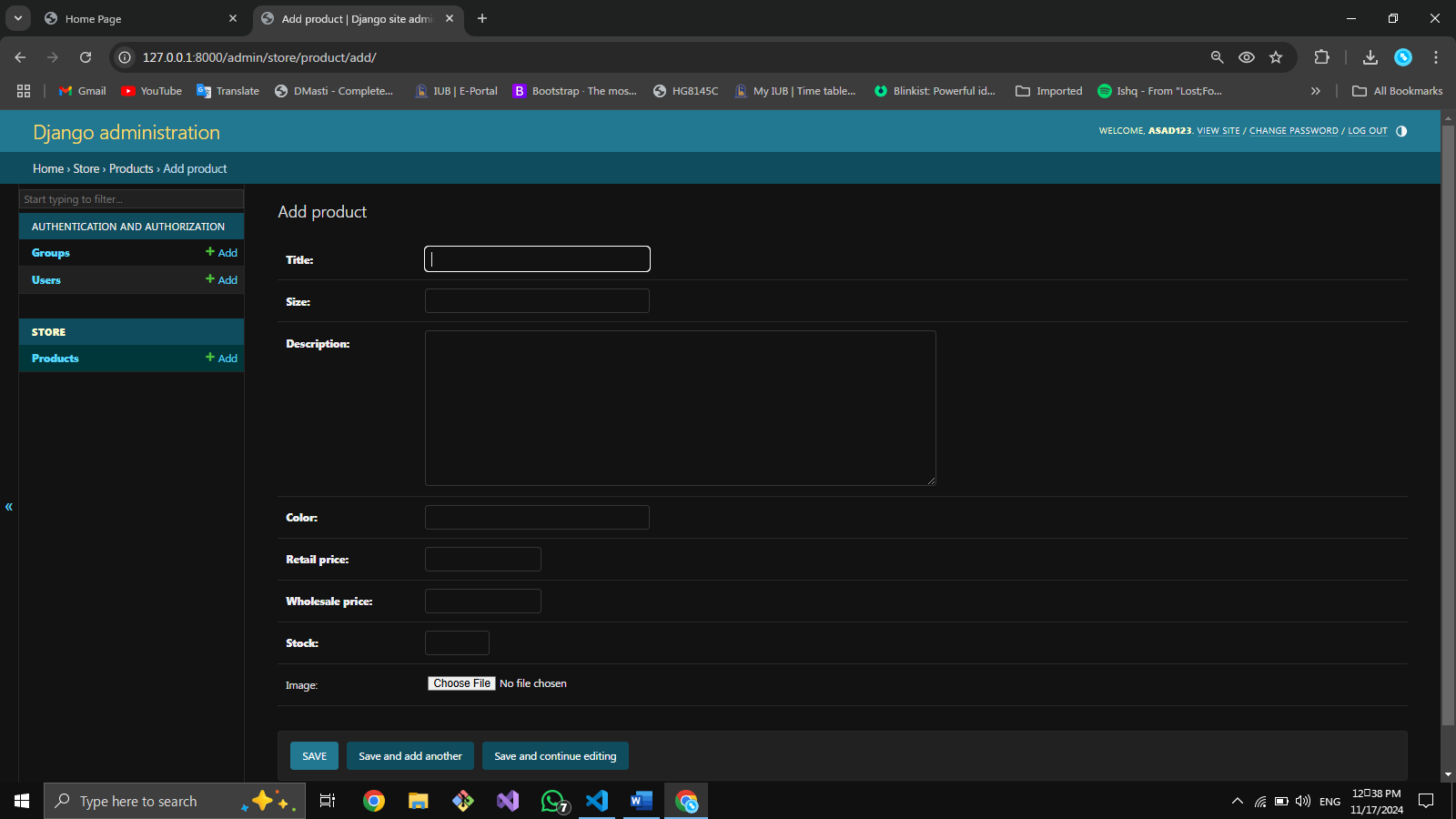


**Product Management**

**Database Model for Product:**

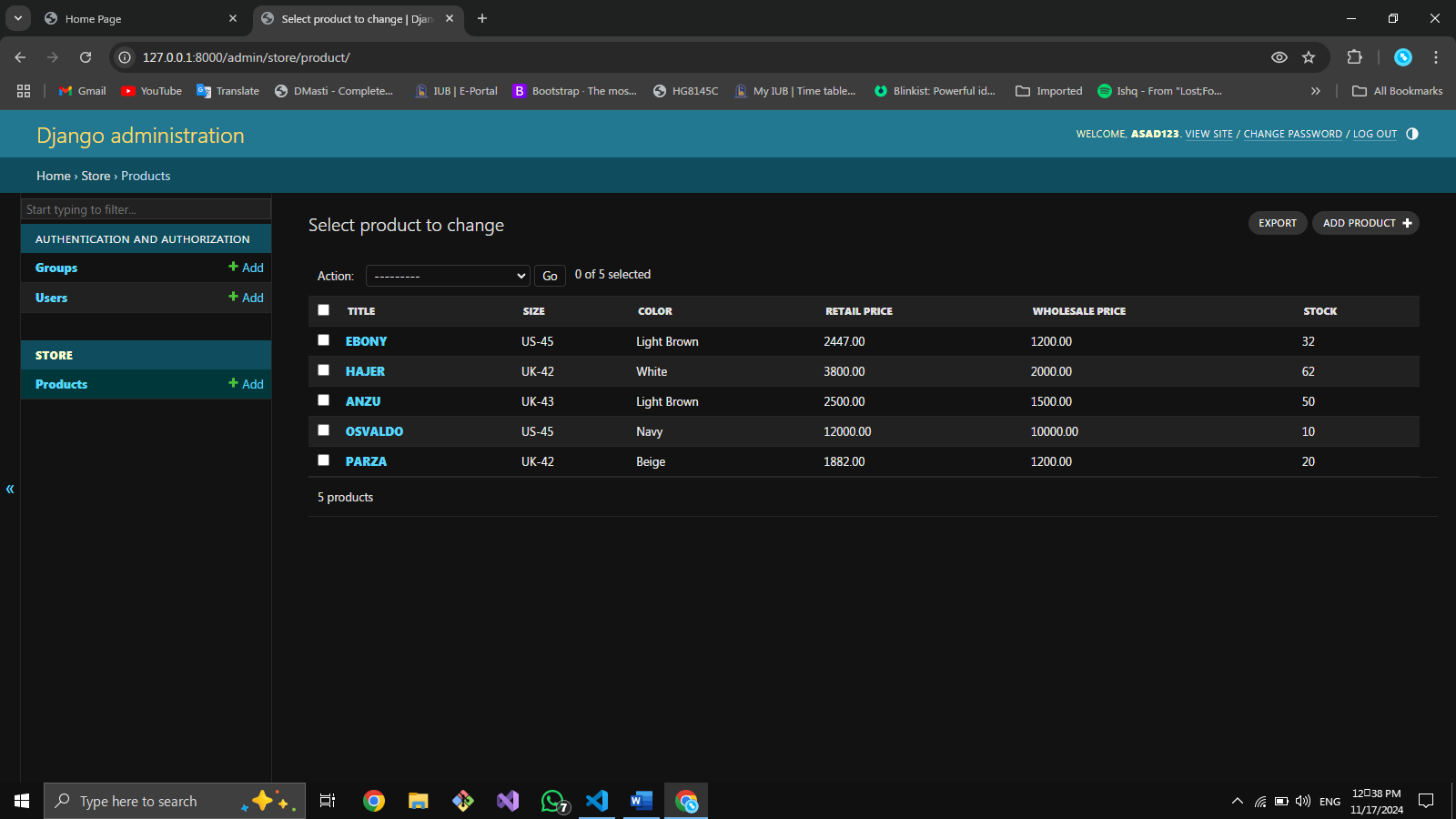
Firstly, I designed a database model using Django models to make a table in database with its attributes and constraints in SQLite.

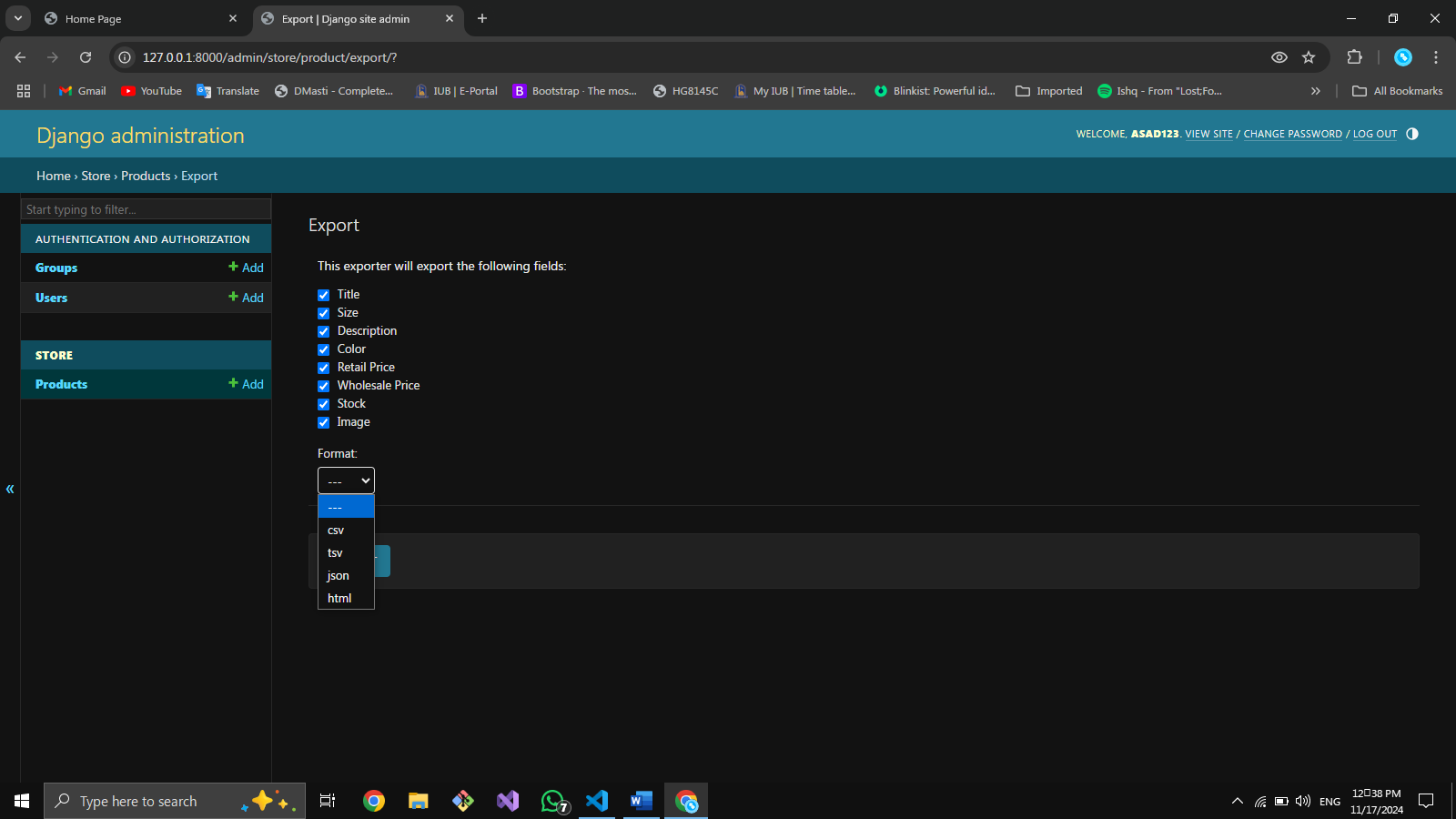
* **Add/Remove Products**: Admin can add, edit, and remove products via Django admin interface.

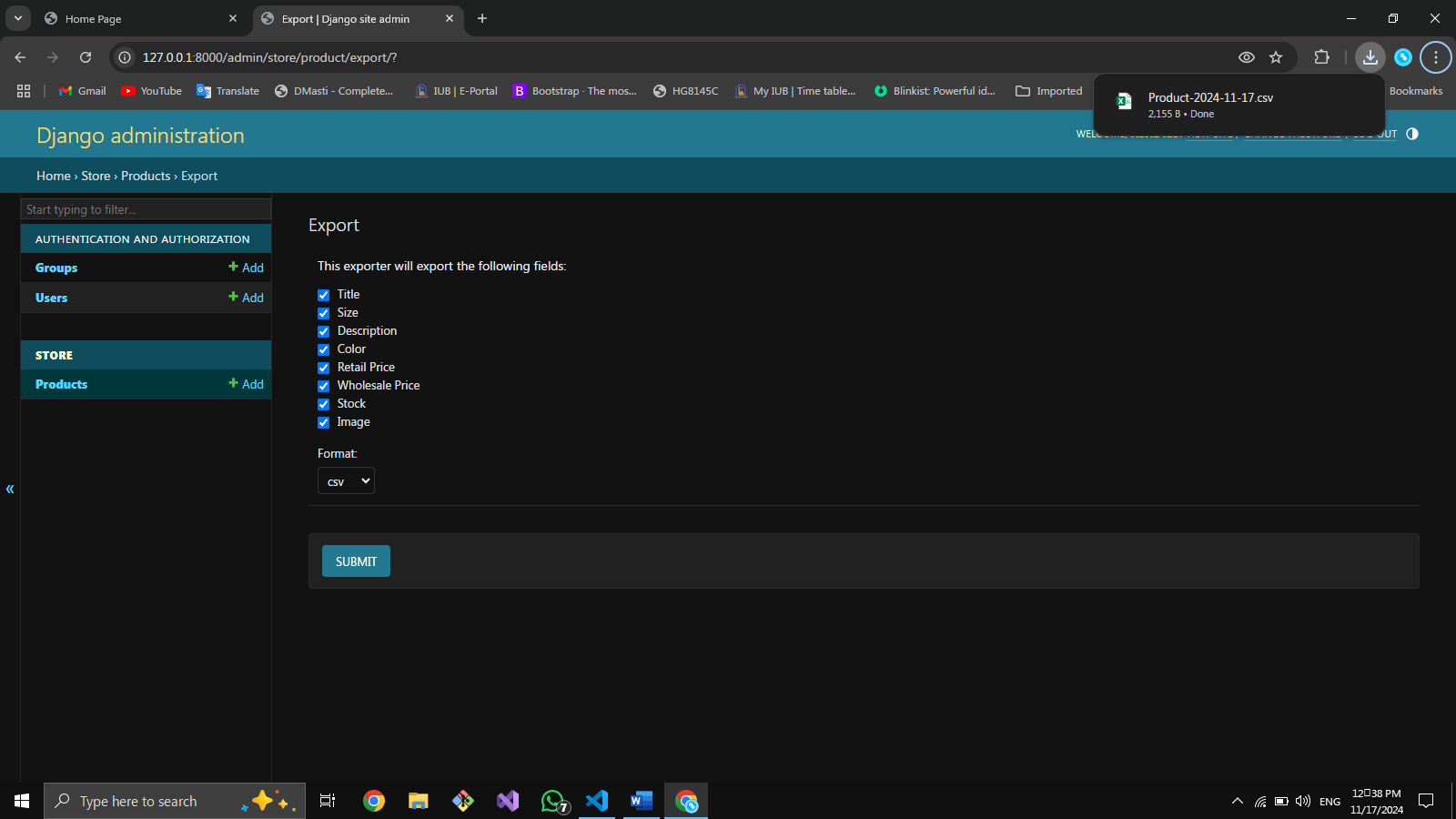


Admin Panel for product Management

* **CSV Export**: Export product list to CSV using Django’s CSV response.



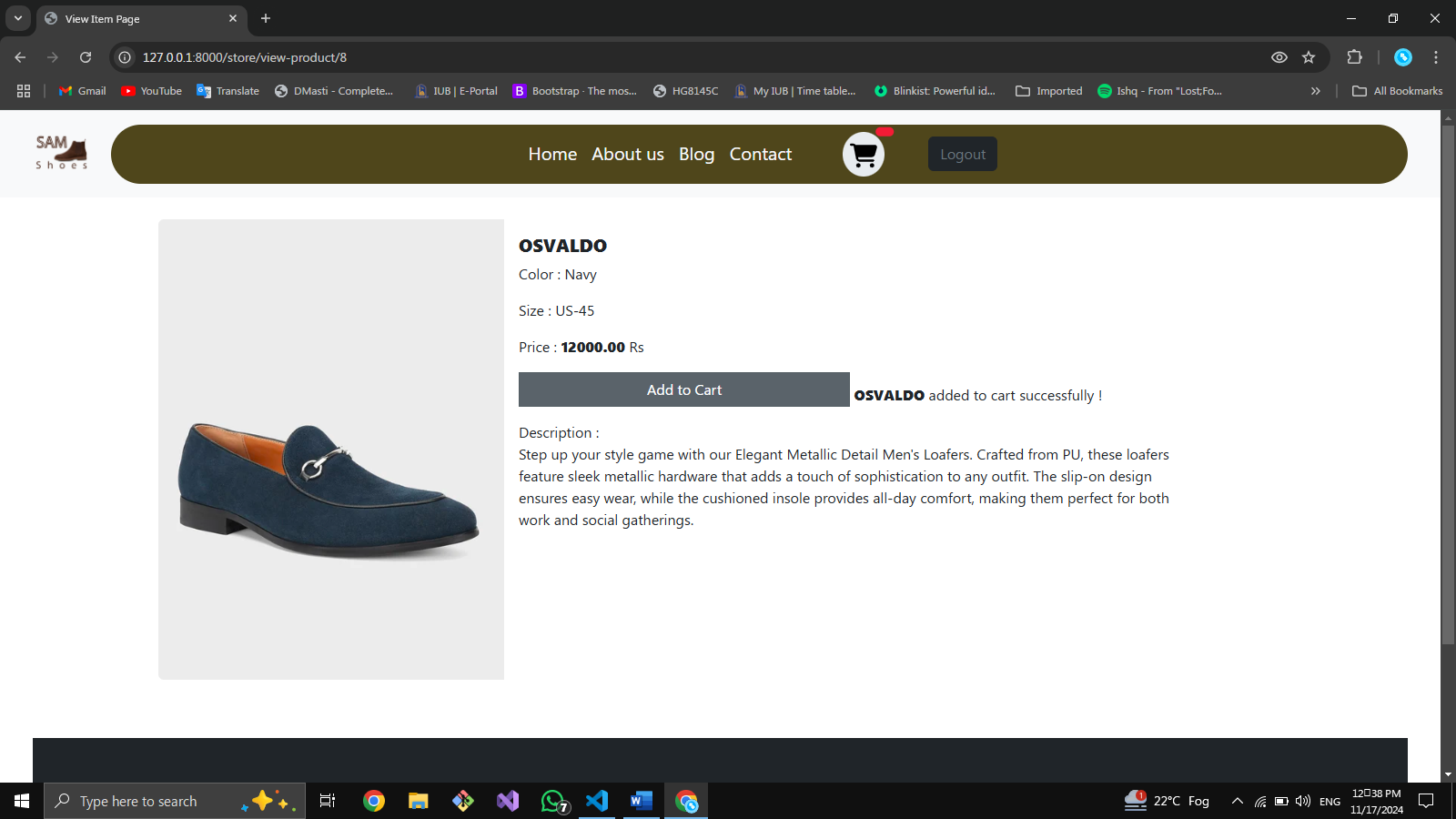


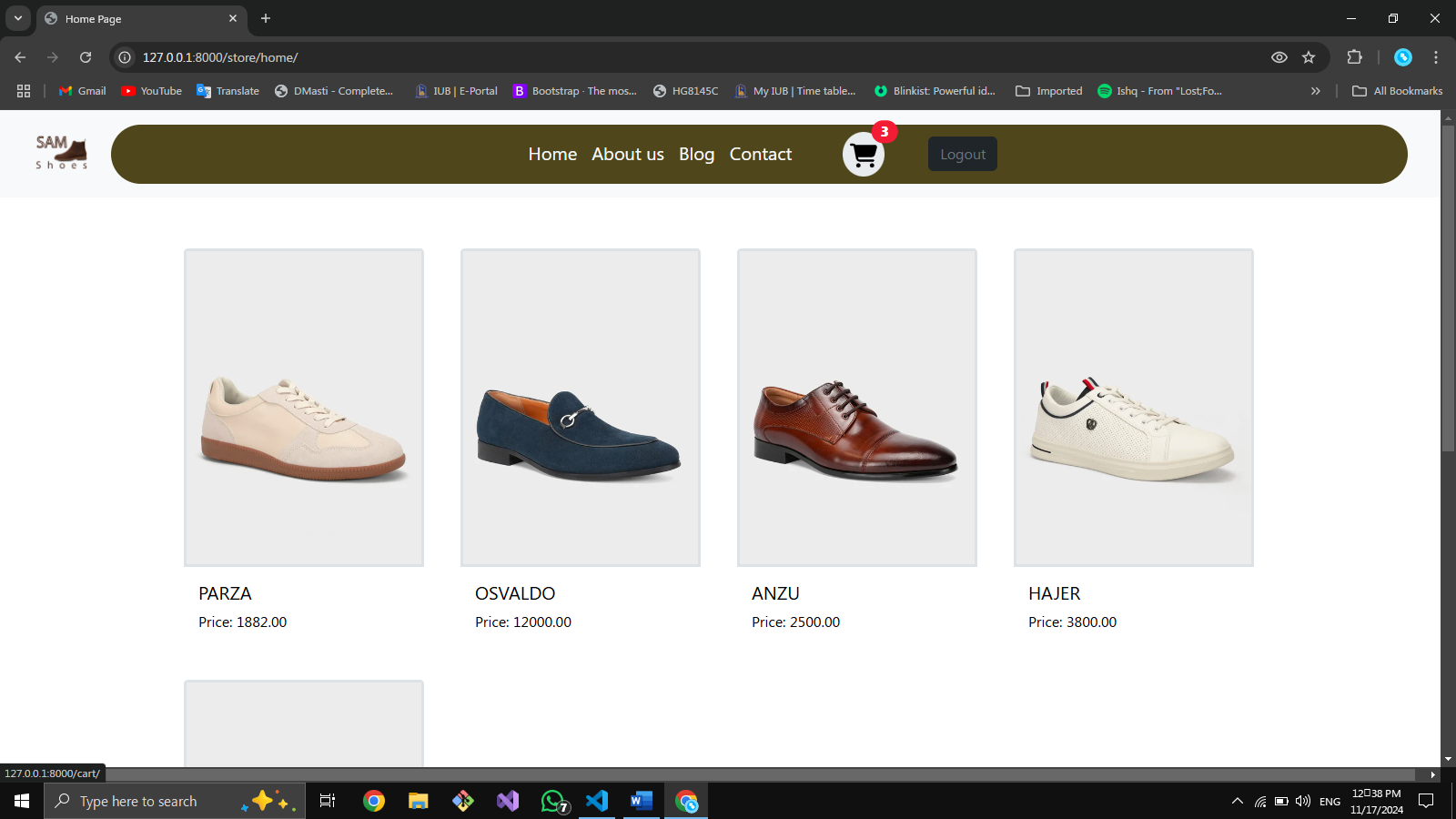


**Cart and Checkout**

* **Cart Functionality**: updates cart item count.

Clicking a product will redirect to view\_item page from where we add that product to cart with messge like “OSVALDO added to cart successfully !” below

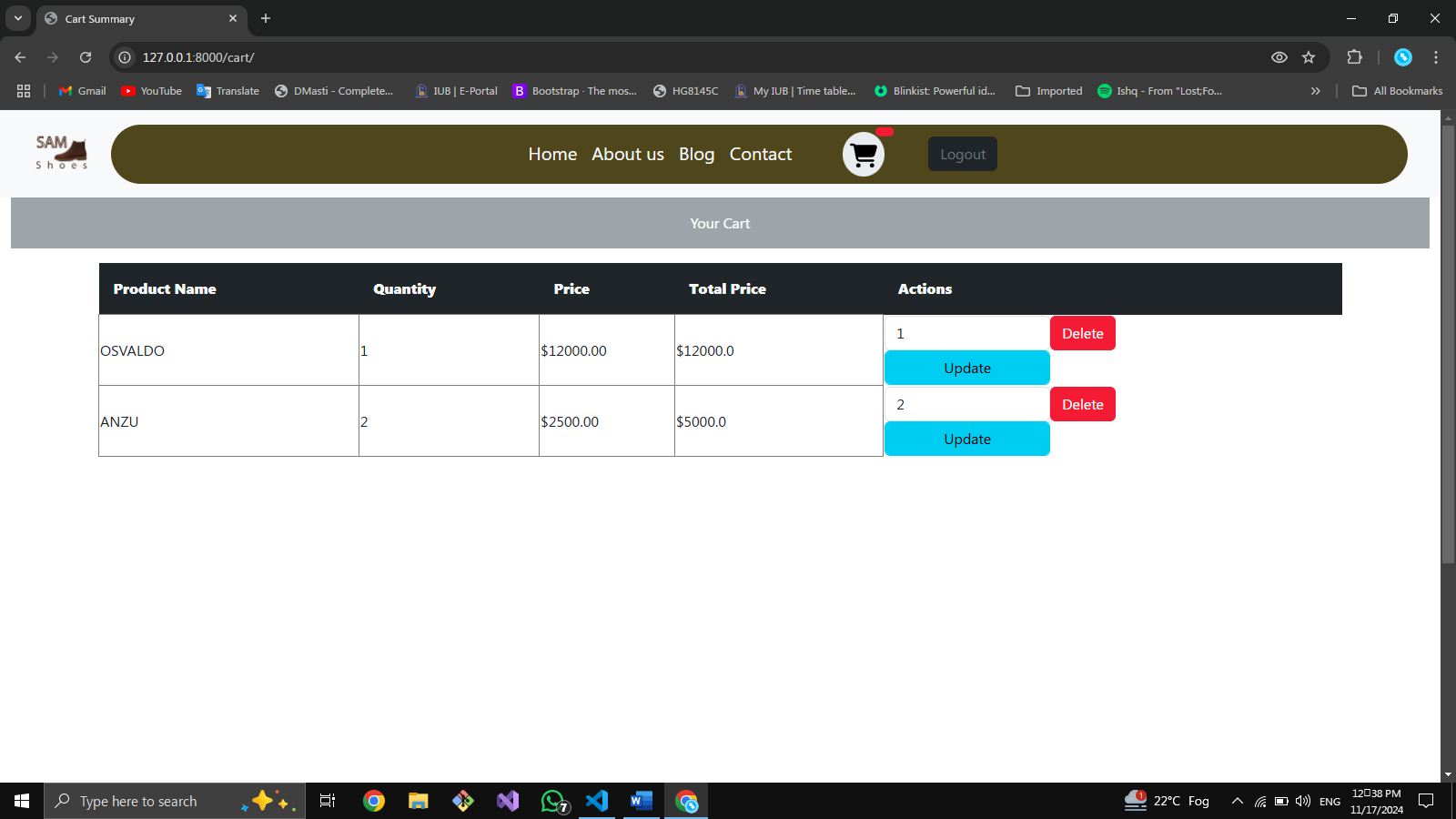




Cart count updated on home page

Now click on cart icon and it will redirect you to cart page showing your items in cart.

* **Checkout View**: Displays selected products with quantities and total price.



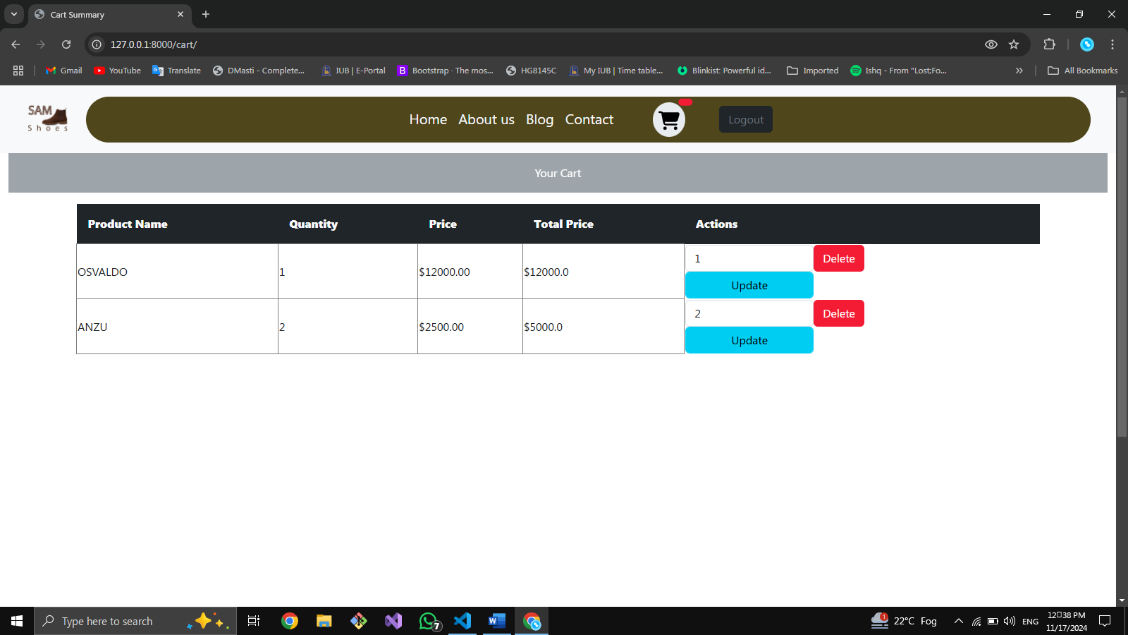
You can Update items quantity and delete them in real time showing your cart updated on each click.

## Testing Phase

**Test Cases**

**Unit Tests**

1. **Login Functionality**
   * Validate that a user can log in successfully with correct credentials.
   * Ensure errors are displayed for invalid username or password combinations.
   * Test session initiation after a successful login.
2. **Cart Operations**
   * Verify that items can be added to the cart and persist in the session.
   * Ensure items can be removed accurately, with the cart updating dynamically.
   * Check that cart totals adjust correctly based on item additions or deletions.



Cart Page with Update Delete Operations

**Integration Tests**

1. **Product Management and Cart Interaction**
   * Test end-to-end functionality: adding products to the cart, updating quantities, and ensuring accurate data display in the cart view.
   * Validate that items added to the cart are visible during the checkout process, with pricing and quantities accurately calculated.
   * Ensure that unauthorized users are redirected to the login page when accessing the cart or checkout views.

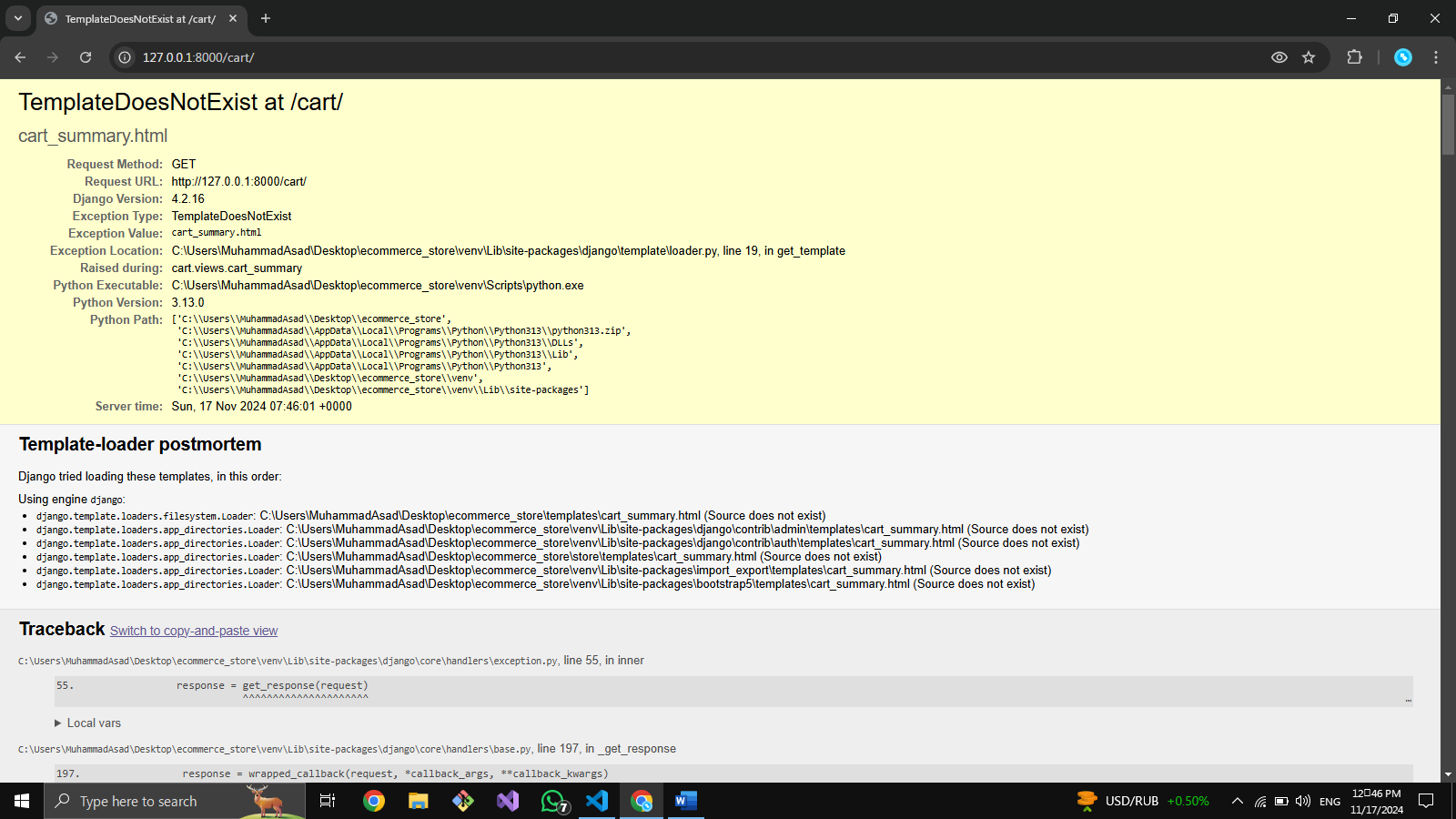
## Error Handling and Troubleshooting

### Testing Phase

* **Error Description**

An error occurred when I access the cart related info or route

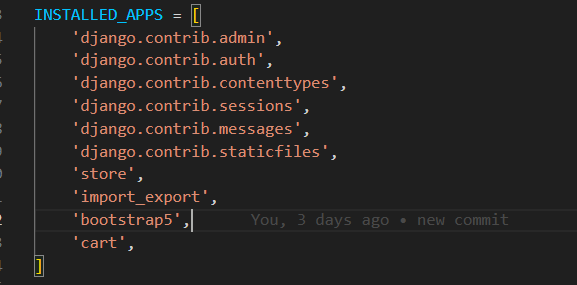




I

* **Solution**

Resolved this after almost 2 hours of debugging as the problem is occurred in following as I have removed the listed cart app in installed apps in settings.py



This testing framework ensures robust and reliable core functionality, providing a seamless user experience.

**Test Cases for User**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Description | Expected Outcome | Result |
| TC\_U \_01 | Signup | Signup page loads on start | Pass |
| TC\_U \_02 | Signup Validations | Input validations for signup like password, confirm password and email validations | Pass |
| TC\_U \_03 | If already account | If already account then redirect to login page of clicking “I already have an account”. | Pass |
| TC\_U \_04 | Login Page | Login Page Loads correctly | Pass |
| TC\_U \_05 | Login Validations | Redirect to Home page if valid user | Pass |
| TC\_U \_06 | User can’t access home page without validations | User redirects to login Page if wants to access other pages which require login | Pass |
| TC\_U \_07 | Home | Shows navbar, products and footer | Pass |
| TC\_U \_08 | View Product Details | Shows that product details on clicking that product in home products section | Pass |
| TC\_U \_09 | About us page | Goes to about us page | Pass |
| TC\_U \_10 | Blog Page | Goes to Blog Page | Pass |
| TC\_U \_11 | Contact Us page | Goes to Contact us page | Pass |
| TC\_U\_12 | Add to cart Message | On Add to cart button click a message will be displayed showing product added | Pass |
| TC\_U \_13 | Add to cart items-count | Count increased in cart icon on home page | Pass |
| TC\_U \_14 | View Cart | On cart icon click view car page displays | Pass |
| TC\_U \_15 | Empty Cart | If cart empty it shows empty cart message on cart page | Pass |
| TC\_U \_16 | Products in cart | If Products in cart they will be displayed in columns with update Input field and Update and Delete button | Pass |
| TC\_U \_17 | Delete Cart item | Delete respective item in real time and updates cart page | Pass |
| TC\_U \_18 | Update Cart Item | Update items numbers and total price in real time updating cart page | Pass |
| TC\_U \_19 | Log Out | On logout button clicked user session flushed and user log out with redirecting to login page. | Pass |
| TC\_U \_20 | Error Page | If any error happened then it will not crash the application rather displaying user an info message | Pass |

**Test Cases for Admin**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID | Description | Expected Outcome | Result |
| TC\_A\_01 | Admin Panel | On /admin route access admin panel by login | Pass |
| TC\_A\_02 | Adding Users | Add user with its access management | Pass |
| TC\_A\_03 | Adding products by clicking add product | Add product page with relevant input fields to add product | Pass |
| TC\_A\_04 | Export Products in CSV format | On export Button Clicked a page shows to check info to be exported and also checks the required file format to be exported and on clicking submit password | Pass |

## Deployment and Maintenance

**Deployment**

**Platform**: Hosted on [hosting provider, e.g., Heroku, AWS].

**Deployment Steps**: Install dependencies, set up environment variables, deploy using `git push` or deployment command.

**Maintenance Plan**

**Bug Fixes**: Address user-reported issues monthly.

**Feature Updates**: Quarterly review for feature improvements.