Selling Website Documentation

Introduction

The Selling Website is a simple full-stack web application built using Python (Django framework) with PostgreSQL as the database and HTML, CSS, and Tailwind CSS for the frontend design.

The goal of this project is to demonstrate how to create a basic e-commerce platform where users can log in either as sellers or buyers.

- **Sellers can log in to their accounts and (in future enhancements) manage their products.**
- **Buyers can log in to browse and view available products.**
- **All products are displayed on a dedicated product list page, which serves as the home page after login.**

The project also integrates Django's authentication system (login, logout, redirects) and is structured to be easily extended with additional features such as product management, cart, and checkout.

It is developed in Visual Studio Code and is designed to be beginner-friendly while showcasing a clean separation of backend (Django) and frontend (Tailwind CSS) components.

Project Overview

This project is a small e-commerce (selling) web application built using:

- **Python 3.x**
- **Django Framework**
- **PostgreSQL (via pgAdmin 4)**
- **HTML, CSS, Tailwind CSS for frontend**

It allows:

Sellers to log in and manage their products.
Buyers to log in and view product listings.
A simple shopping cart (optional) and checkout system.

Key Features

- **User authentication (signup, login, logout)**
- **Separate login pages for sellers and buyers**
- **Product list & detail pages**
- **Tailwind-styled frontend**
- **PostgreSQL database integration**
- **Admin interface for managing data**

System Requirements

Software

- **Python 3.10+ (tested on 3.13)**
- **Django 5.x**
- **PostgreSQL**
- **pgAdmin 4**
- **Visual Studio Code (VSCode)**

Hardware

- **Any system capable of running Python & PostgreSQL**
- **Internet connection for installing packages**

Technology Stack

Component	Technology
Backend	Python (Django)
Database	PostgreSQL (managed via pgAdmin)
Frontend	HTML, CSS, Tailwind
Server (dev)	Django development server
Deployment	Render (for production)

Installation & Setup

Create Virtual Environment

```
python -m venv venv venv\Scripts\activate
```

Install Dependencies

```
pip install django psycopg2-binary crispy-bootstrap5 crispy-tailwind
```

Create Django Project

django-admin startproject shop cd shop python manage.py startapp store

Configure Database (PostgreSQL)

DATABASES = { 'default': { 'ENGINE': 'django.db.backends.postgresql', 'NAME': 'your_db_name', 'USER': 'your_pg_user', 'PASSWORD': 'your_pg_password', 'HOST': 'localhost', 'PORT': '5432', } }

Run migrations:

python manage.py migrate

Created superuser:

python manage.py createsuperuser

Project Structure

```
Shop/
├── media/
├── shop/
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── store/
│   ├── migrations/
│   ├── templates/store/
│   │   ├── base.html
│   │   ├── welcome.html
│   │   ├── product_list.html
│   │   ├── product_detail.html
│   │   ├── seller_login.html
│   │   ├── buyer_login.html
│   │   ├── seller_dashboard.html
│   │   ├── buyer_dashboard.html
│   │   └── signup.html
│   ├── models.py
```

│ ├── views.py

│ └── urls.py

├── manage.py

└── Procfile

Implemented Features

Welcome Page

- **URL: /**
- **Shows buttons for Seller Login and Buyer Login**
- **Template: welcome.html**

Authentication

- **Separate login pages:**
    - **/seller-login/ → seller_login.html**
    - **/buyer-login/ → buyer_login.html**
- **Redirects to product list after login**
- **Logout link redirects back to welcome**

Product Management

- **Product model with fields: name, description, price, image**
- **Admin can add products in Django admin panel**
- **product_list.html shows all products**
- **product_detail.html shows individual product info**

Dashboards

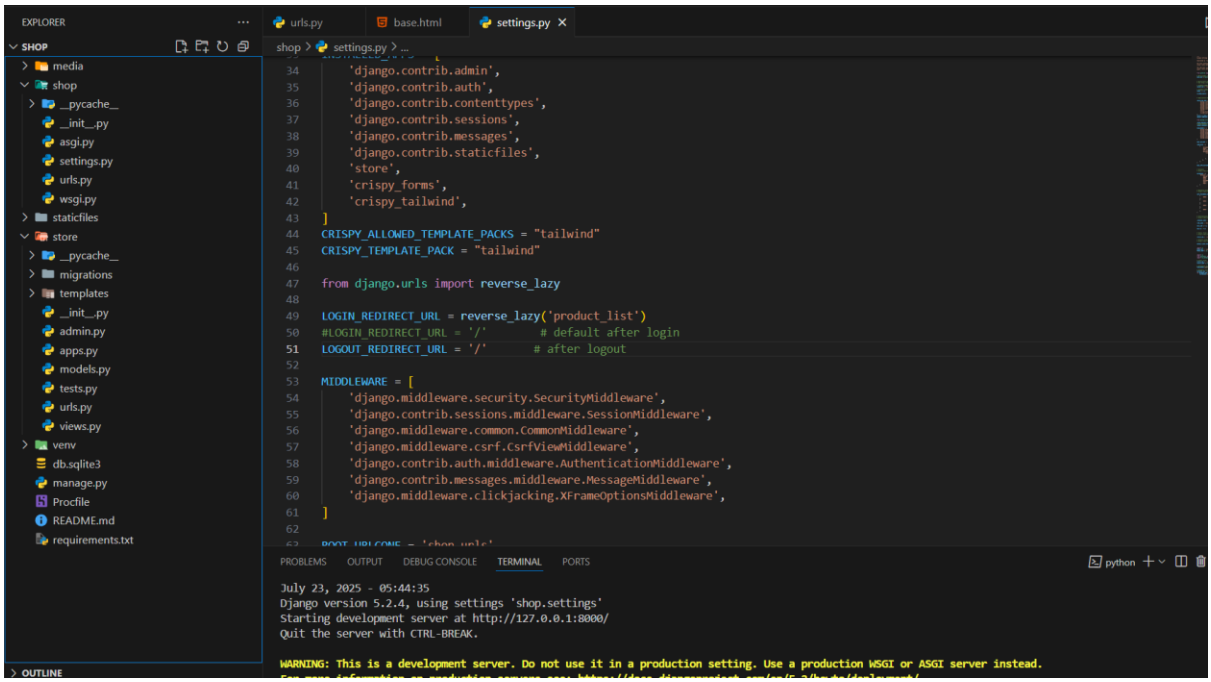
- **Seller dashboard (future scope: add/manage products)**
- **Buyer dashboard (future scope: manage orders)**

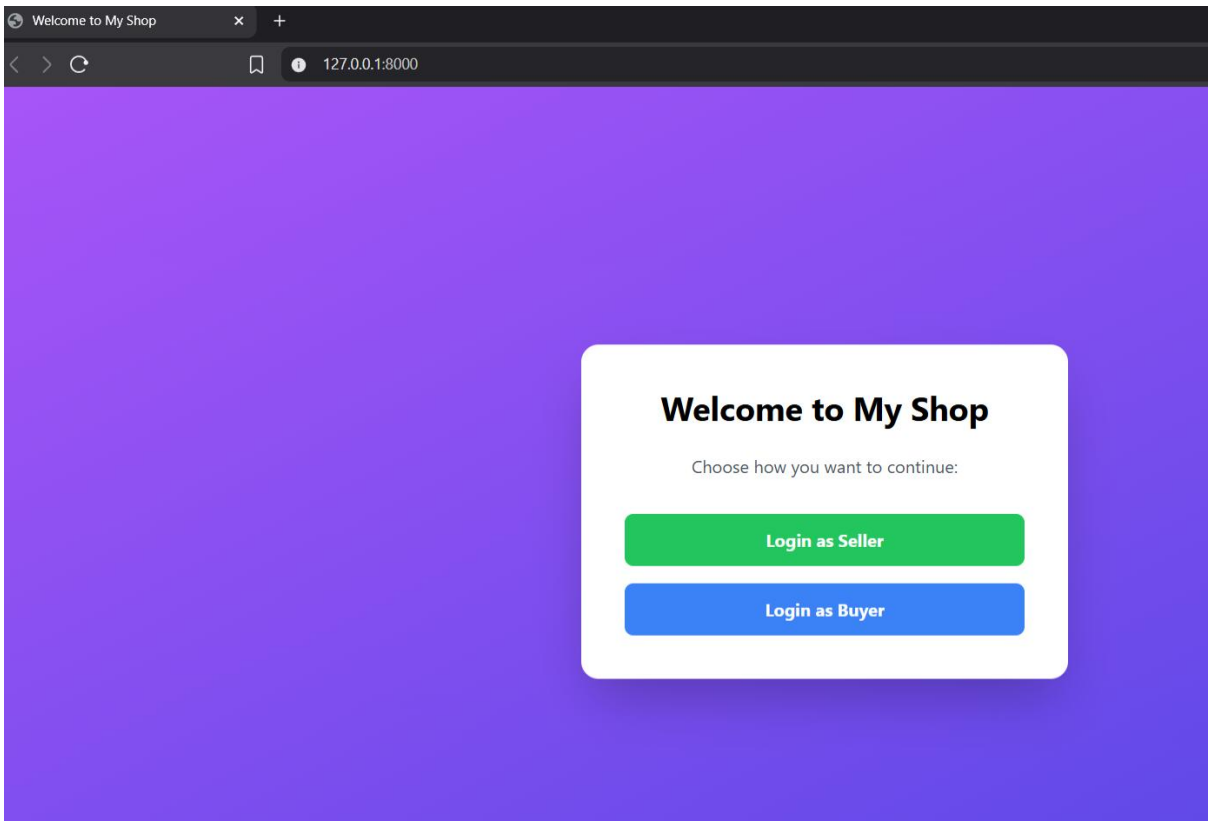
Deployment

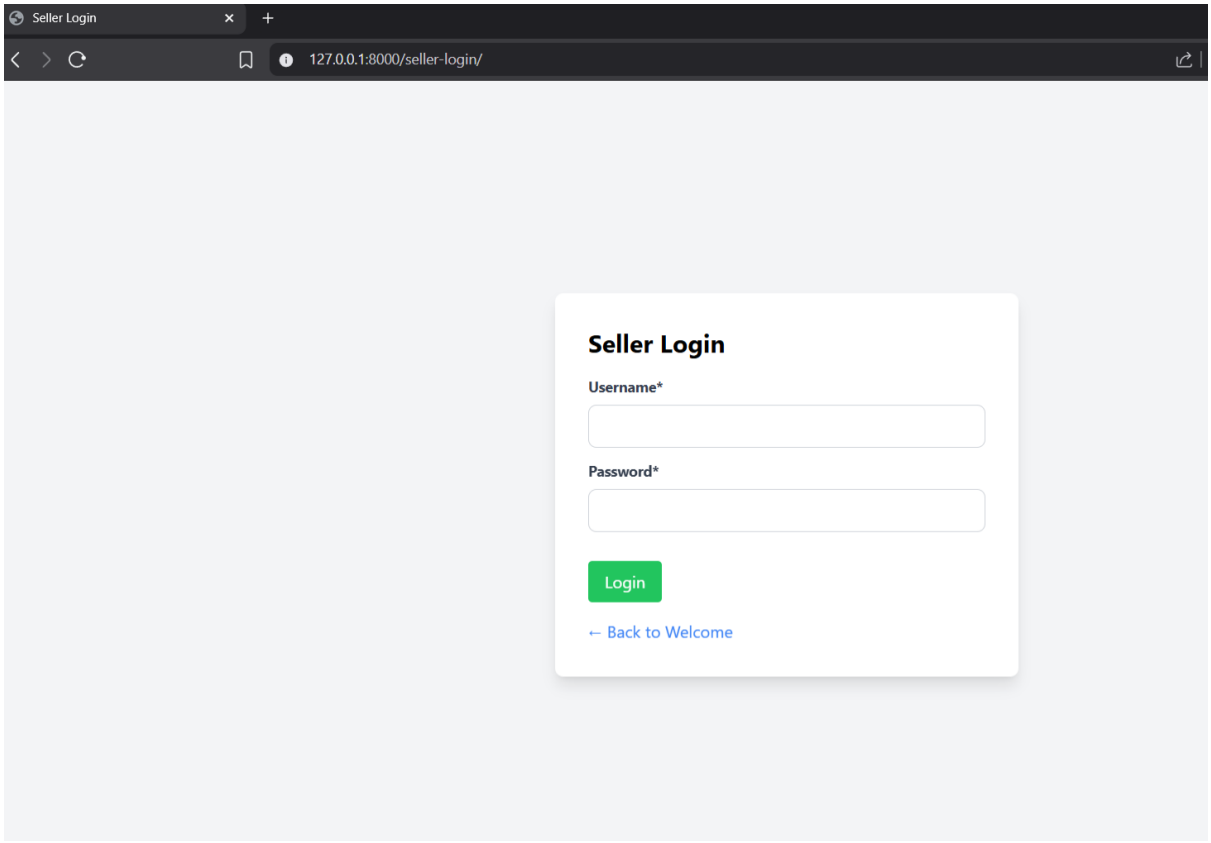
We deployed on **Render**:

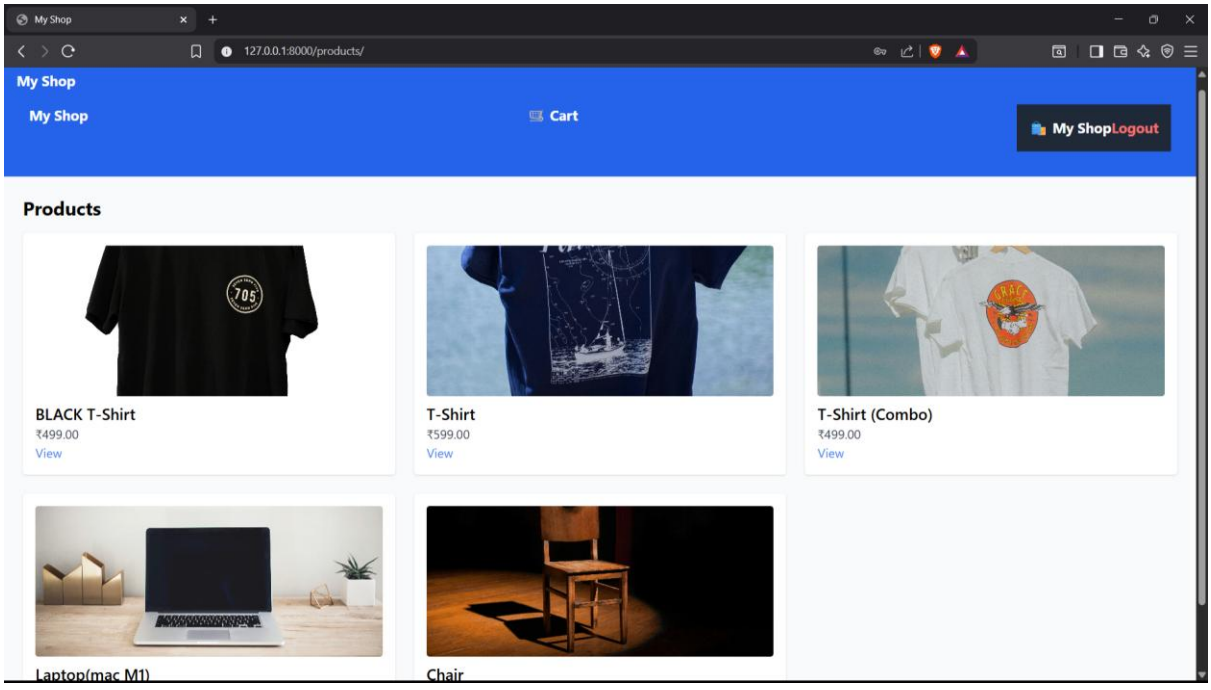
- Created a free Render account
- Linked GitHub repository
- Added PostgreSQL service in Render
- Created Procfile:

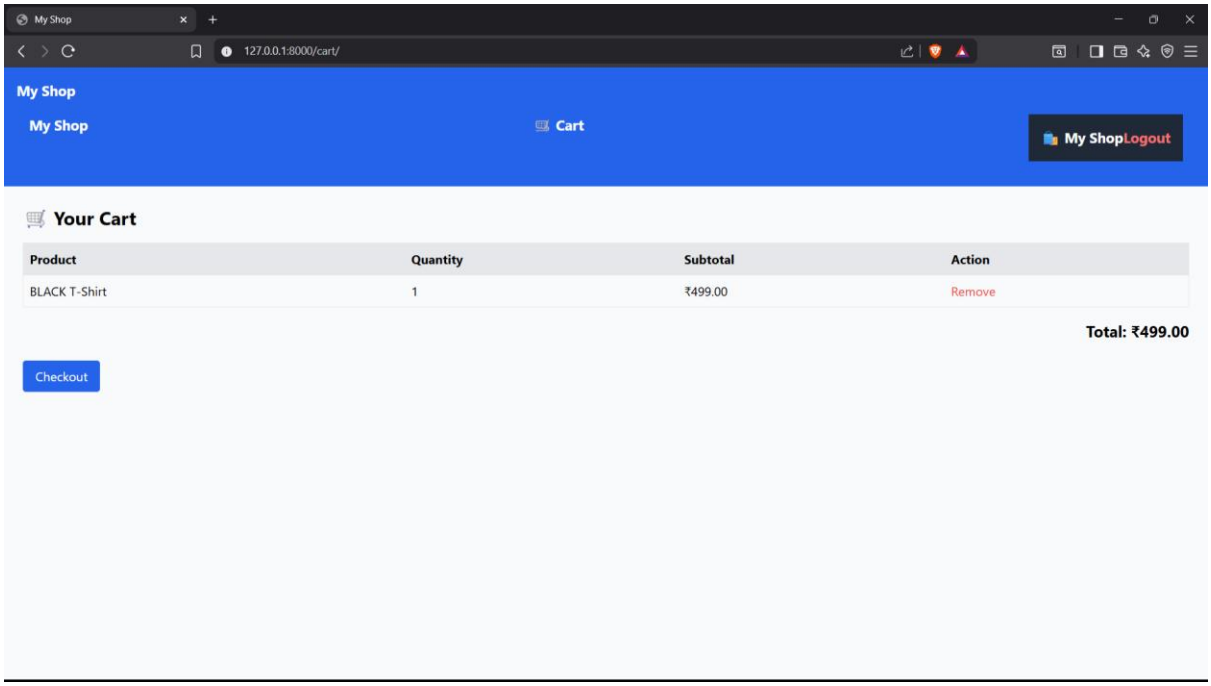
Screenshots

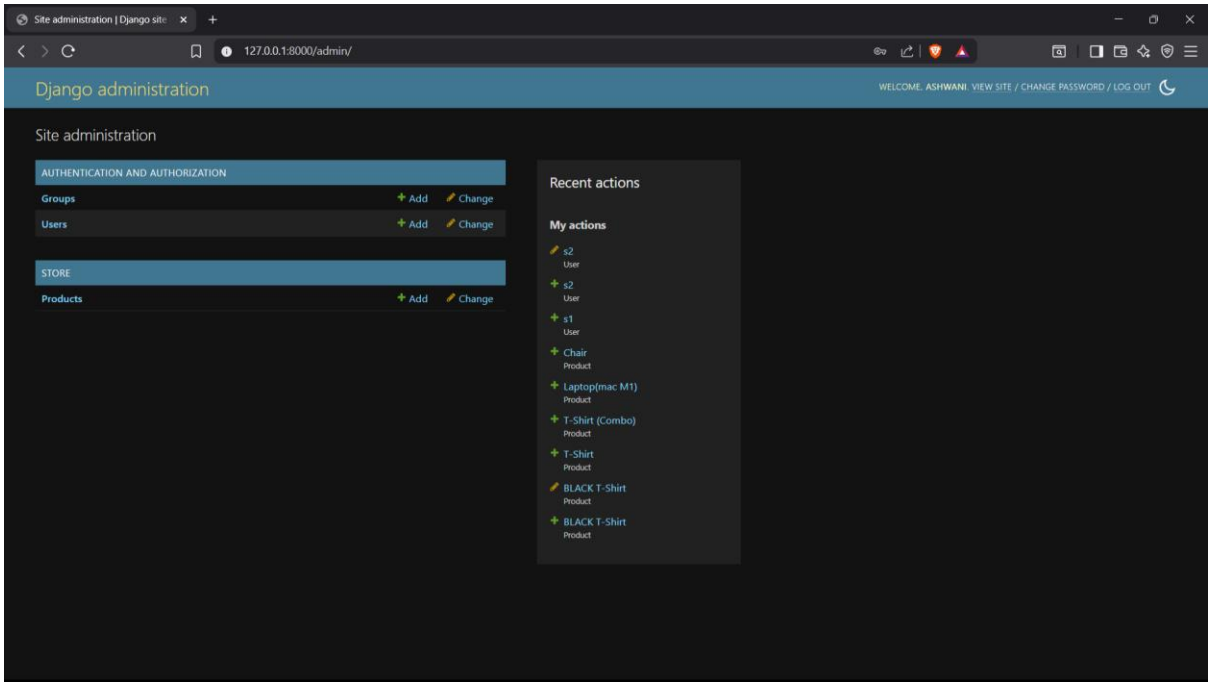


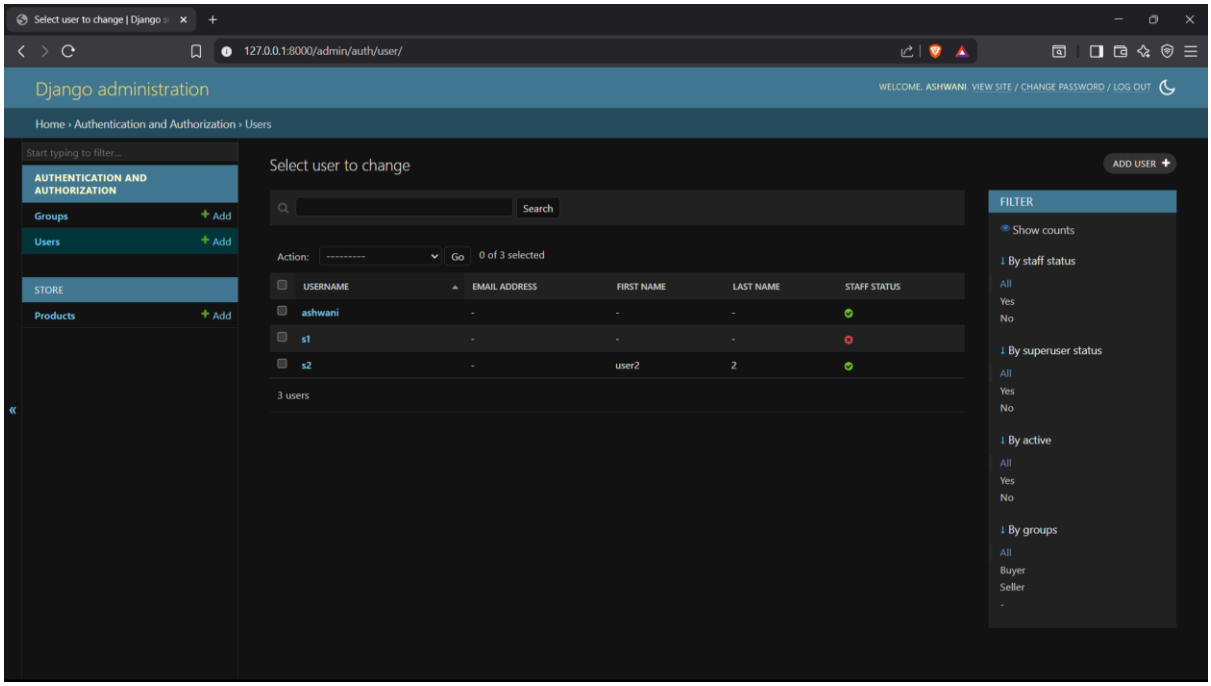


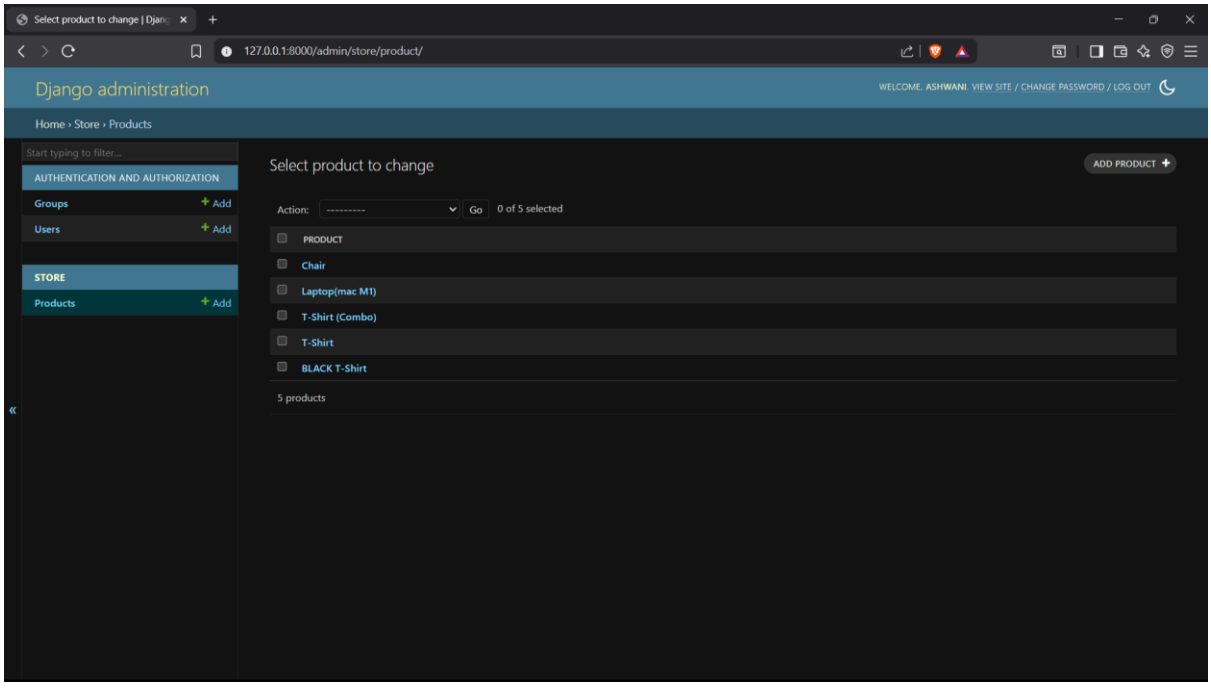

Seller Login
127.0.0.1:8000/seller-login/
Seller Login
Username*
Password*
Login
← Back to Welcome


My Shop
127.0.0.1:8000/products/
My Shop
My Shop
Cart
My ShopLogout
Products
BLACK T-Shirt
₹499.00
View
T-Shirt
₹599.00
View
T-Shirt (Combo)
₹499.00
View
Laptop(mac M1)
Chair


My Shop
127.0.0.1:8000/cart/
My Shop
My Shop
Cart
My ShopLogout
Your Cart
Product
Quantity
Subtotal
Action
BLACK T-Shirt
1
₹499.00
Remove
Total: ₹499.00
Checkout


Site administration | Django site
127.0.0.1:8000/admin/
Django administration
WELCOME, ASHWANI. VIEW SITE / CHANGE PASSWORD / LOG OUT
Site administration
AUTHENTICATION AND AUTHORIZATION
Groups
Add
Change
Users
Add
Change
STORE
Products
Add
Change
Recent actions
My actions
s2
User
s2
User
s1
User
Chair
Product
Laptop(mac M1)
Product
T-Shirt (Combo)
Product
T-Shirt
Product
BLACK T-Shirt
Product
BLACK T-Shirt
Product





Conclusion

This project demonstrates how to build a **full-stack Django web application** with:

- Authentication
- PostgreSQL database
- Tailwind CSS frontend

It serves as a solid foundation to expand into a full-fledged e-commerce system.