Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Уральский федеральный университет имени первого Президента России Б.Н.Ельцина» (УрФУ)
ИРИТ-РТФ
Базовая кафедра «Аналитика больших данных и методы видео анализа»

Grade on project _____
Project supervisor _____SAIF M.A.___
Commission's members _____
_____

Defense date _____21.06.2025_____

Report on the project

Design and Development of an Arabic E-Commerce Platform Using Django Framework

Student: Sabar Lebate _____
(Signature)

Group #: RIM-140930

Yekaterinburg
2025

Contents

Executive Summary

SabarStor is a modern, bilingual e-commerce web application (focused on Arabic) developed using the Django framework. The project is designed to provide a user-friendly online store with support for an Arabic right-to-left (RTL) layout, product categorization, shopping cart functionality, user authentication, and a responsive design.

The platform allows users to register, browse products by category, view detailed product information, and manage shopping carts. Administrators can manage product listings, categories, and orders through the Django admin panel. The application integrates with Cloudinary for secure and scalable image storage and is deployed online using Render, ensuring ease of access and performance.

To simplify deployment and development, Docker and Docker Compose are configured. Although the application uses SQLite for simplicity, Docker's configuration allows for scalability. Logging mechanisms are also integrated for system monitoring, and a set of unit tests were written to ensure the reliability of key features such as user registration, login, and shopping cart behavior.

SabarStor's successful delivery demonstrates proficiency in developing, designing, deploying, and testing integrated systems. The project meets academic and professional standards, providing a solid foundation for future enhancements, such as payment integration, user reviews, and inventory tracking.

Introduction

Project Description

The SabarStor project is a modern e-commerce web application specifically designed for Arabic users using the Django framework. It allows users to browse products by category, view product details, and add items to their cart, as well as log in and create a new account. The project was developed to provide a simple, flexible, and secure online shopping experience for Arabic users, with RTL support and a responsive design that fits all screen sizes.

The system relies on an SQLite database in the development environment, while images are hosted using Cloudinary. The project has been successfully uploaded to the Render platform, enabling users to interact with it online.

The Problem That the Project Solves

With the growing trend toward digital commerce, many small businesses and users suffer from a lack of ready-made Arabic-language software solutions that are simple and easy to use. Some systems also lack support for right-to-left orientation, which can cause browsing difficulties for Arabic users.

The SabarStor project addresses these challenges by providing an e-commerce platform that:

- Fully supports the Arabic language.
- Provides a flexible and seamless user interface.
- Allows users to easily manage their accounts and interact with products.
- Scalable and deployable on cloud servers.

Target Audience

The target audience for this project includes:

- Arab users who want to shop online in their native language.
- Small business owners and local stores who need a simple digital solution to showcase their products.

- Students and new developers who want to study an e-commerce project using an open-source, customizable programming language.
- Anyone looking for a simple e-commerce system that can be easily deployed and modified as needed.

System Design

The SabarStor system consists of a set of organized and integrated components that contribute to a flexible user experience and efficient management of products and users. The system's internal architecture is designed using the Django framework, which supports the MVC (Model-View-Controller) architecture, with an emphasis on separating business logic from presentation.

1. Application Structure

The project consists of the following applications:

accounts/: Responsible for user registration, login/logout, and profile management.

store/: Contains the logic for displaying products, categories, product details, and the shopping cart.

templates/: Contains HTML templates for all pages, designed in Arabic and with RTL support.

static/: Contains CSS and JavaScript files for customizing the overall design.

2. Database Design

A SQLite database was used during development, and the main tables include:

CustomUser: Registered users. A custom user form was adopted to support additional fields in the future.

Category: Product categories (clothing, electronics, etc.).

Product: Contains data for each product (name, description, image, price, category).

CartItem: Cart items associated with the user.

3. User Interface

The interface was designed using Bootstrap 5 (RTL version) to provide a modern user experience in Arabic. Interface Features:

- Responsive design that works on phones and tablets.
- A top navigation menu with links to key pages.
- New login and registration pages are carefully designed with consistent colors.
- The product display and shopping cart interface is clear and easy to use.

4. Backend and Authentication

The Django Authentication System was used to create accounts and log in.

- Used a CustomUser instead of the default User.
- Protected sensitive pages with @login_required.
- Set up logging messages to record all sensitive operations (registration, login, and page access).

5. Integration with External Services

- Cloudinary was used to store product images professionally without relying on the local file system.
- Docker was configured to facilitate running the application in an isolated environment.
- The project was deployed via Render to be permanently available to users online.

Implementation

The SabarStor project implementation involves programming all system components in a fully integrated manner using the Django framework, along with modern tools such as Docker and Cloudinary. This section details the implementation of the most important parts of the application.

1. User Registration & Authentication

- A custom registration form was developed using CustomUserCreationForm.

- The login, registration, and personal dashboard pages were created using elegant Arabic HTML templates.
- All data passes through a validation filter before being saved to the database.
- Protection against unauthorized access was enabled using @login_required.

```
logger = logging.getLogger(__name__)

# (تتطلب تسجيل دخول) لوحة التحكم ✅
/accounts/dashboard/
@login_required  2 usages  Sabar Lebate
def dashboard(request):
    logger.info(f"👤 Dashboard accessed by user: {request.user}")
    return render(request, template_name: 'accounts/dashboard.html')
```

2. Product Listing & Shopping Cart

A Product model is created containing a name, image, description, price, and category.

- The images are uploaded to Cloudinary.
- The user can add, edit, or delete products to the cart.
- The cart is only associated with registered users.

```
# إضافة منتج إلى السلة
/add/{product_id}/
def add_to_cart(request, product_id):  1 usage  Sabar Lebate
    product = get_object_or_404(Product, id=product_id)
    cart = Cart(request)
    cart.add(product)
    return redirect('cart_detail')
```

3. Dashboard & Analytics

- The user dashboard displays personal account details and order history.
- The manager displays products by category and number of views.
- Important events such as registration, login, logout, and orders are logged.

```python
import logging

logger = logging.getLogger(__name__)
# (تتطلب تسجيل دخول) لوحة التحكم ✅
/accounts/dashboard/
@login_required  2 usages  Sabar Lebate
def dashboard(request):
    logger.info(f"Dashboard accessed by user: {request.user}")
    return render(request, template_name: 'accounts/dashboard.html')
```

4. Docker Setup

The project is configured to run within a containerized environment using Docker and Docker Compose, making it easy to run in any environment without the need for manual configuration.

- **Dockerfile:** Used to create a custom image that installs the requirements and runs the project.
- **docker-compose.yml:** Used to run the application's core service without any external databases, as the project relies on the built-in SQLite database.

```yaml
version: '3.9'

services:
  web:
    build: .
    command: gunicorn sabarstor.wsgi:application --bind 0.0.0.0:8000
    volumes:
      - .:/app
    ports:
      - "8000:8000"
    env_file:
      - .env
```

```
# استخدم صورة Python الرسمية
FROM python:3.10-slim
# تعيين مجلد العمل داخل الحاوية
WORKDIR /app
# نسخ وتثبيت الحزم
COPY requirements.txt .
RUN pip install --upgrade pip
RUN pip install --no-cache-dir -r requirements.txt

# نسخ باقي ملفات المشروع
COPY . .
# إعداد متغيرات البيئة
ENV PYTHONDONTWRITEBYTECODE 1
ENV PYTHONUNBUFFERED 1
# جمع الملفات الثابتة وتشغيل الترحيلات
RUN python manage.py collectstatic --noinput
RUN python manage.py migrate

# فتح المنفذ
EXPOSE 8000

# الأمر الافتراضي لتشغيل الخادم
CMD ["sh", "-c", "python manage.py migrate && gunicorn sabarstor.wsgi:application --bind 0.0.0.0:8000"]
```

5. System Testing

A series of system tests were conducted to ensure that all project functions were working properly and meeting the specified requirements. These tests included the following aspects:

1. Registration and Login Test

Purpose: Ensure that the user can create a new account and log in successfully.

Result: New users were successfully registered, and login was tested using both valid and invalid data.

Status: Passed

```python
from django.test import TestCase
from django.urls import reverse
from accounts.forms import CustomUserCreationForm
from django.contrib.auth.models import User

class UserRegistrationTests(TestCase):
    def test_register_view_accessible(self):
        """صفحة التسجيل تفتح بشكل صحيح ✅"""
        response = self.client.get(reverse('register'))
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'registration/register.html')

    def test_user_registration_success(self):
        """يمكن تسجيل مستخدم جديد بنجاح ✅"""
        response = self.client.post(reverse('register'), {
            'username': 'newuser',
            'password1': 'SuperSecure123!',
            'password2': 'SuperSecure123!',
        })
        self.assertEqual(response.status_code, 302)  # Redirect to login or dashboard
        self.assertTrue(User.objects.filter(username='newuser').exists())
```

2. Logout Test

Purpose: Ensure that the user can log out correctly.

Result: Logout was successful and redirected to the appropriate page.

Status: Passed

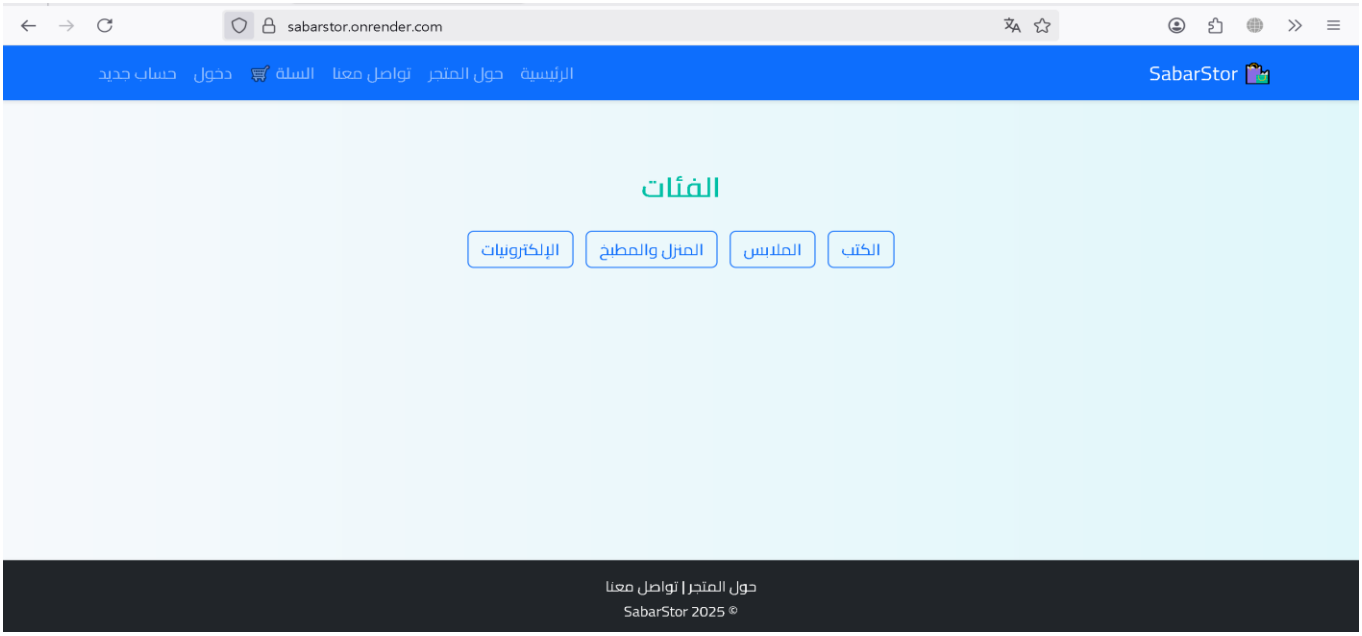
```python
class LogoutTest(TestCase):

    def test_user_can_logout(self):
        # تسجيل دخول المستخدم أولاً
        CustomUser.objects.create_user(username='logoutuser', password='logoutpass123')
        self.client.login(username='logoutuser', password='logoutpass123')

        # تنفيذ تسجيل الخروج
        response = self.client.get(reverse('logout'))

        # التأكد من إعادة التوجيه لصفحة تسجيل الدخول
        self.assertRedirects(response, reverse('login'))

        # التأكد من أن المستخدم لم يعد مسجلاً في الجلسة
        response2 = self.client.get(reverse('dashboard'))
        self.assertRedirects(response2, f"{reverse('login')}?next={reverse('dashboard')}")
```

Results and Output

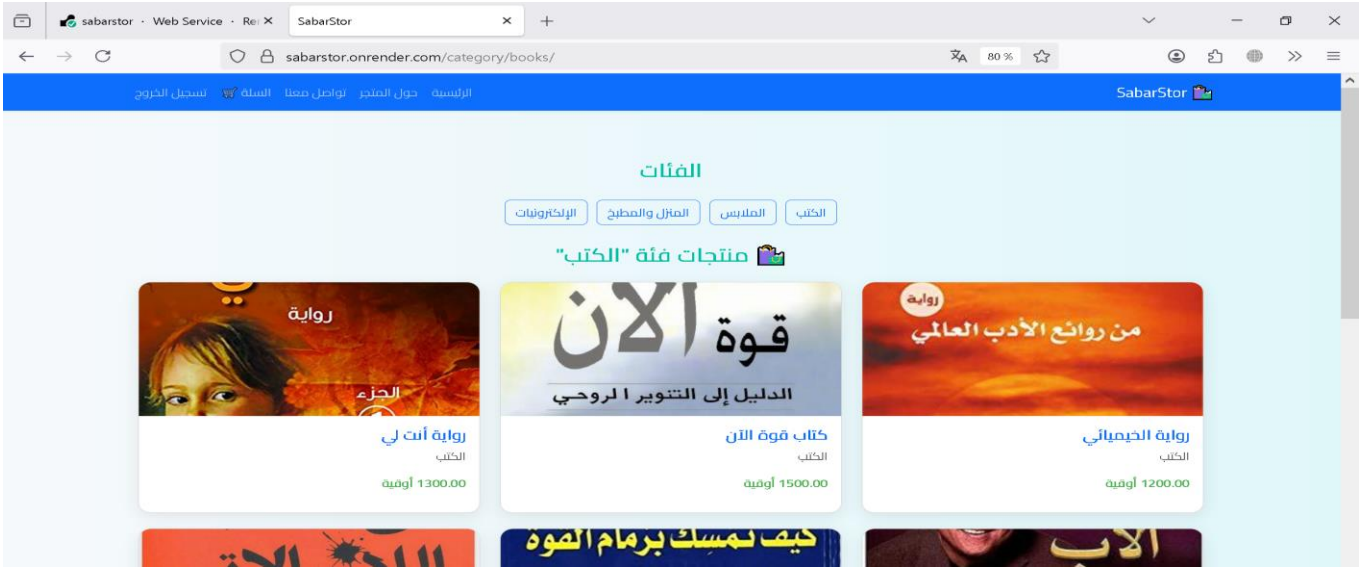
During the implementation of the SabarStor project, a set of results and outputs were achieved that reflect the success of the e-commerce application, as follows:

1. Main User Outputs

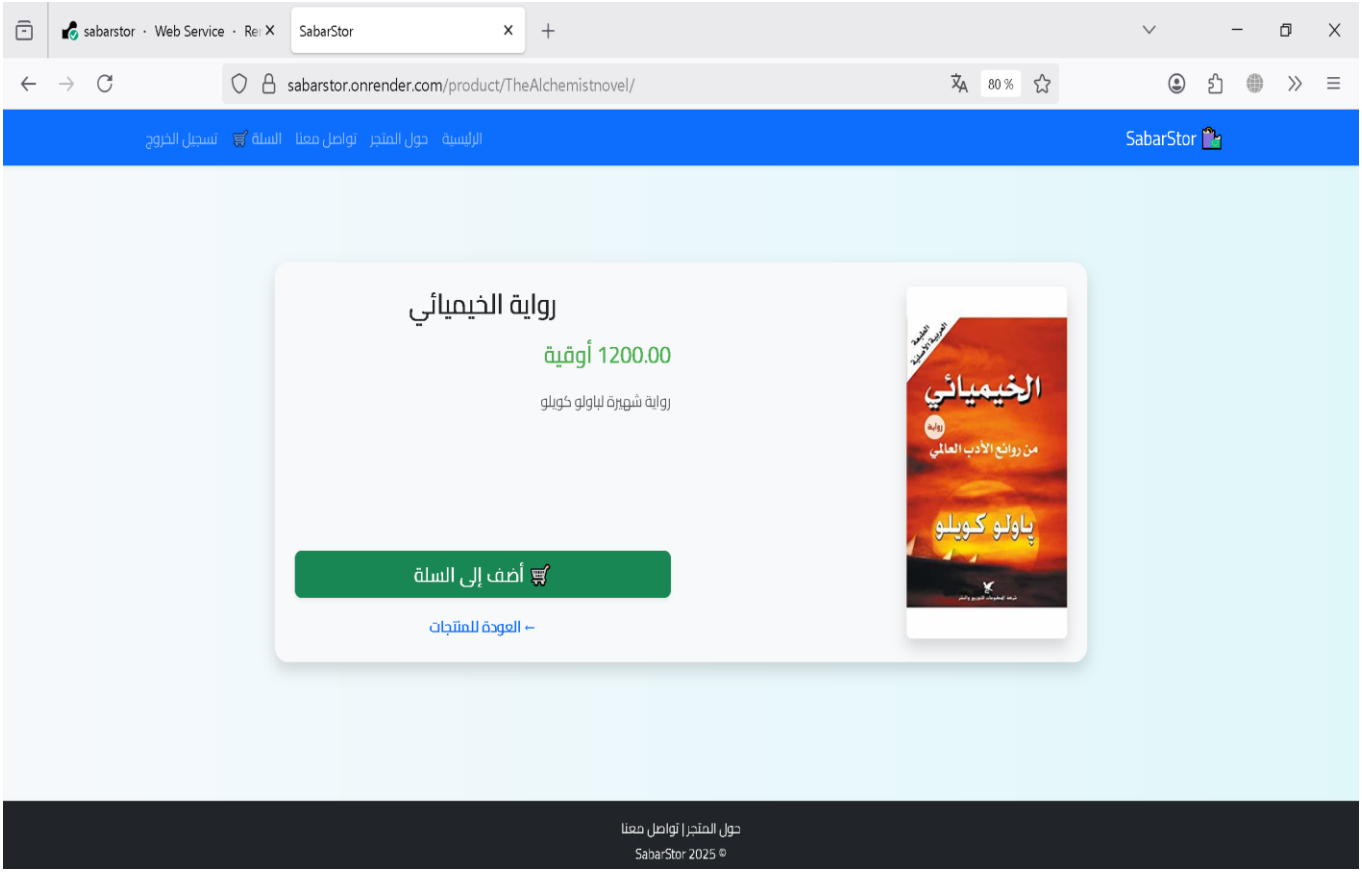
- Attractive Arabic right-to-left (RTL) interface using Bootstrap 5.



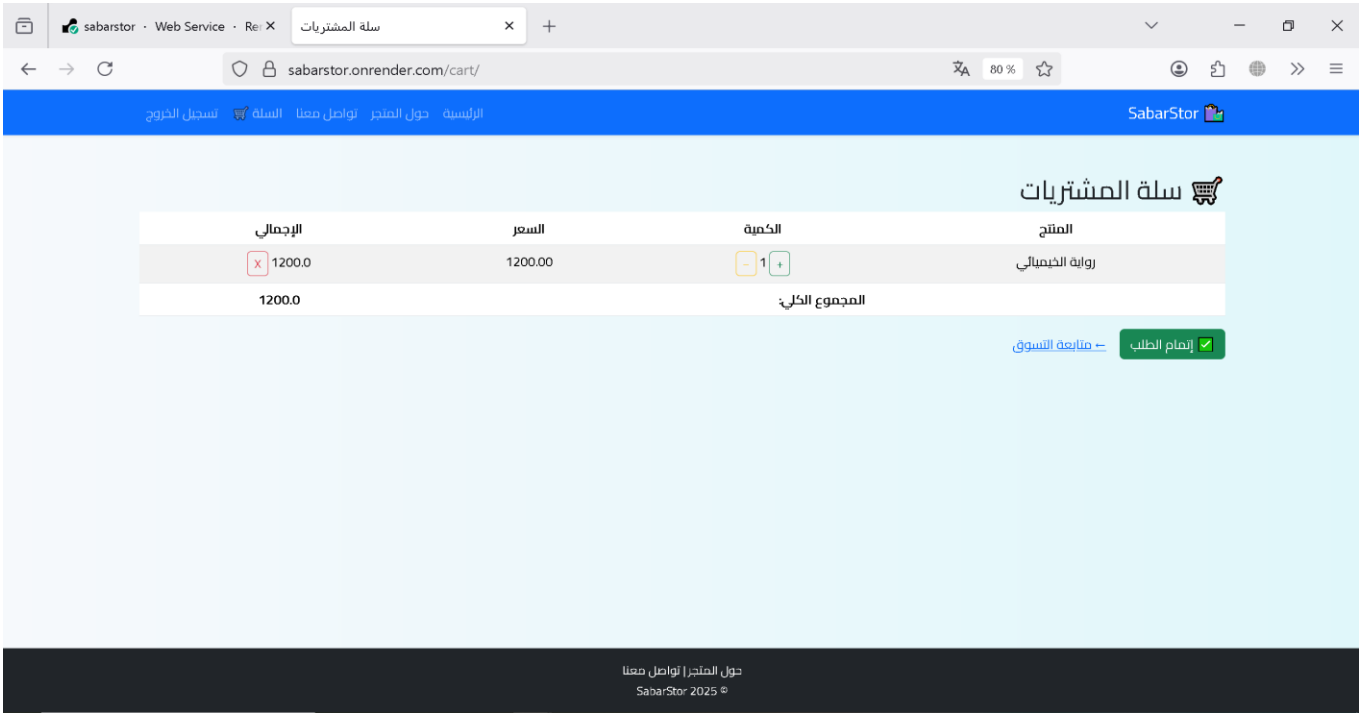
- Display products by category in an organized and easy-to-browse manner.



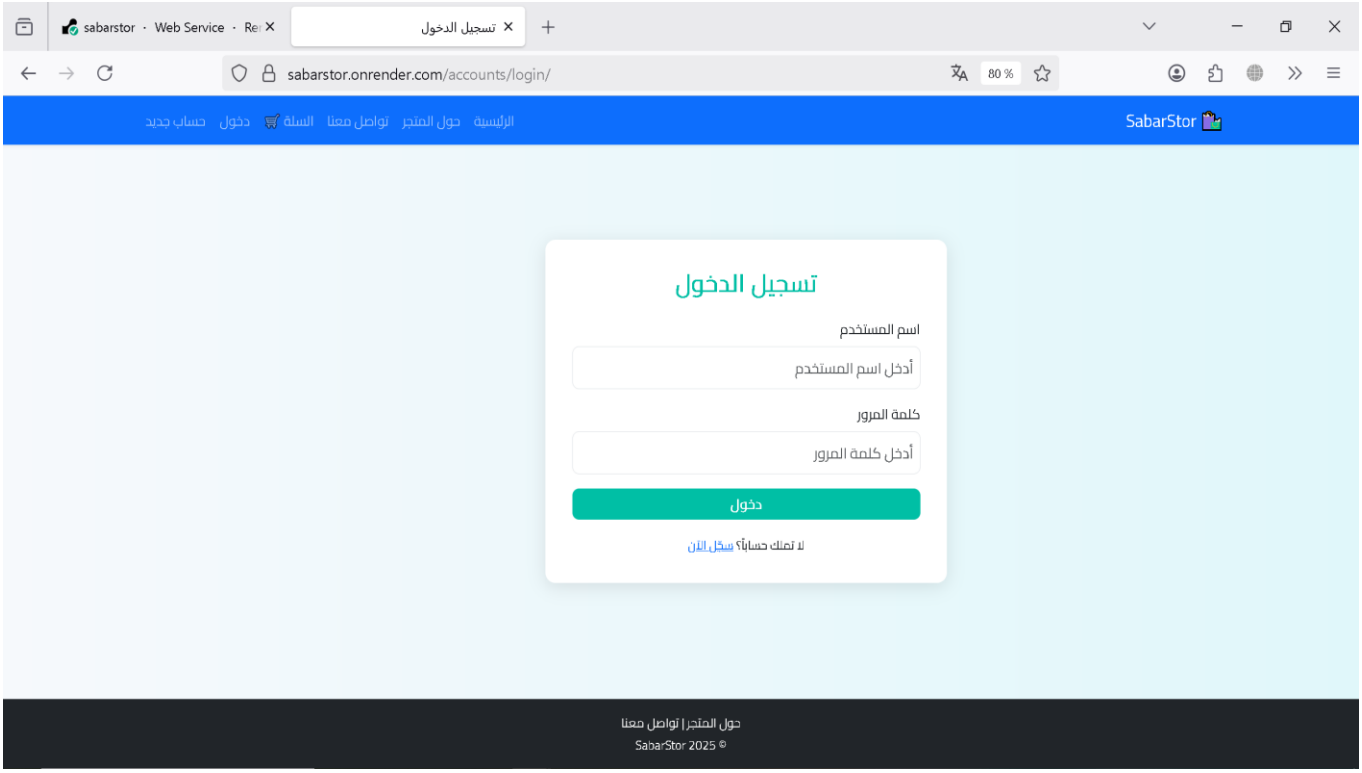
- Detailed product pages including price, image, and description.

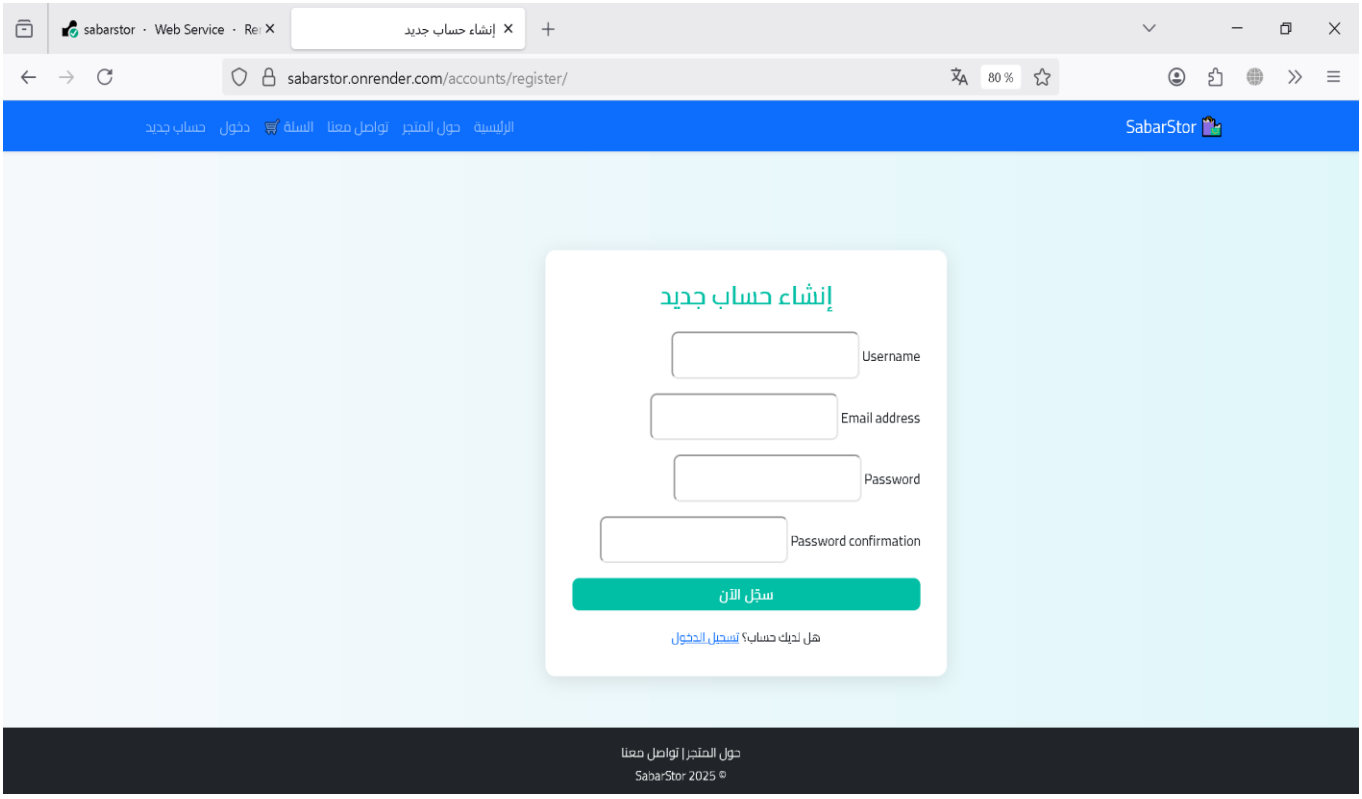


- Ability to add products to the cart and track them interactively.

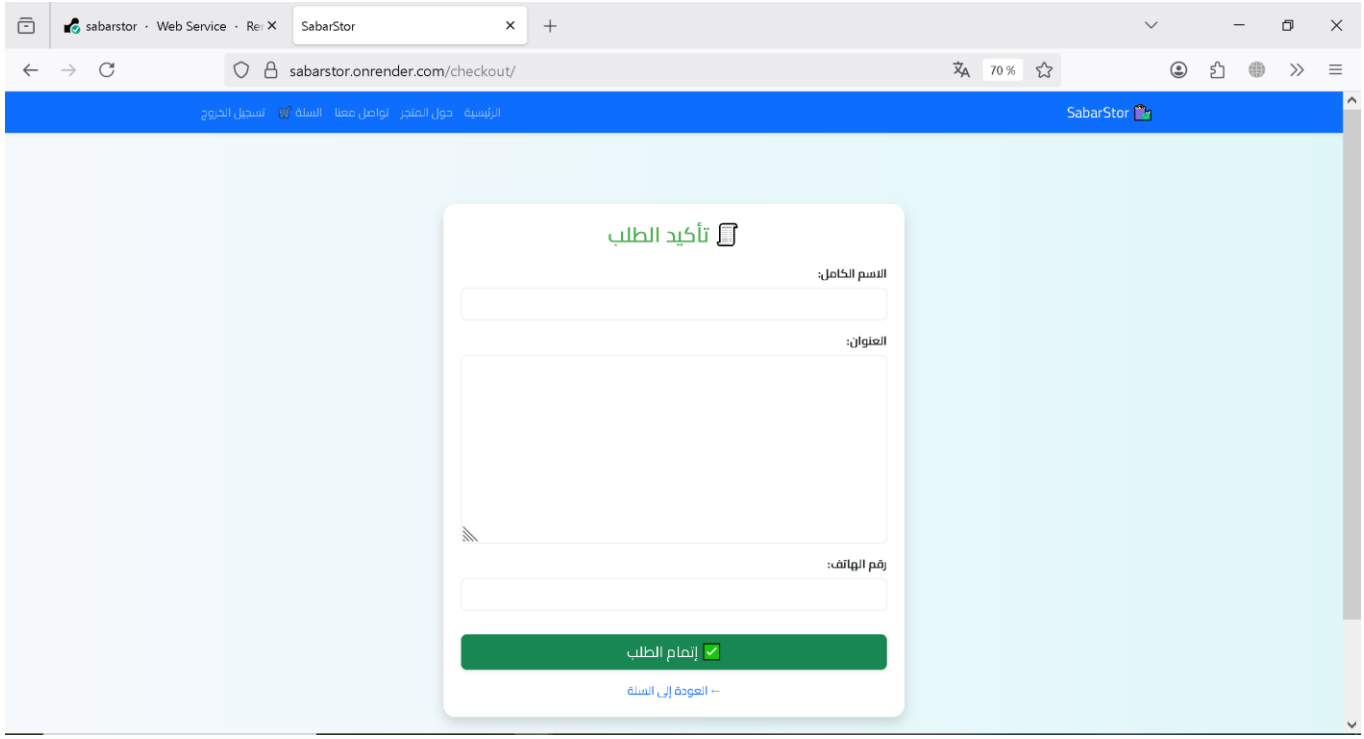


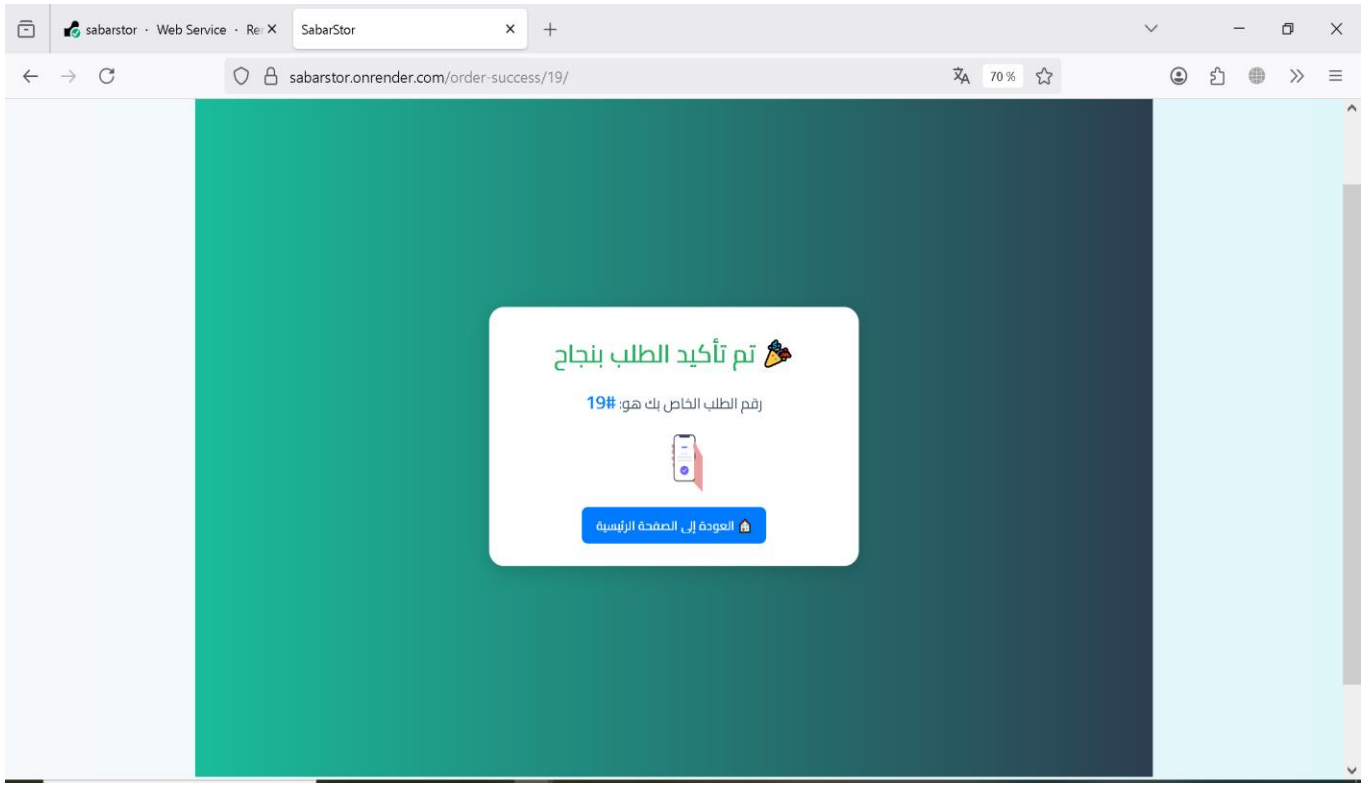
- New login and registration system with modern design.

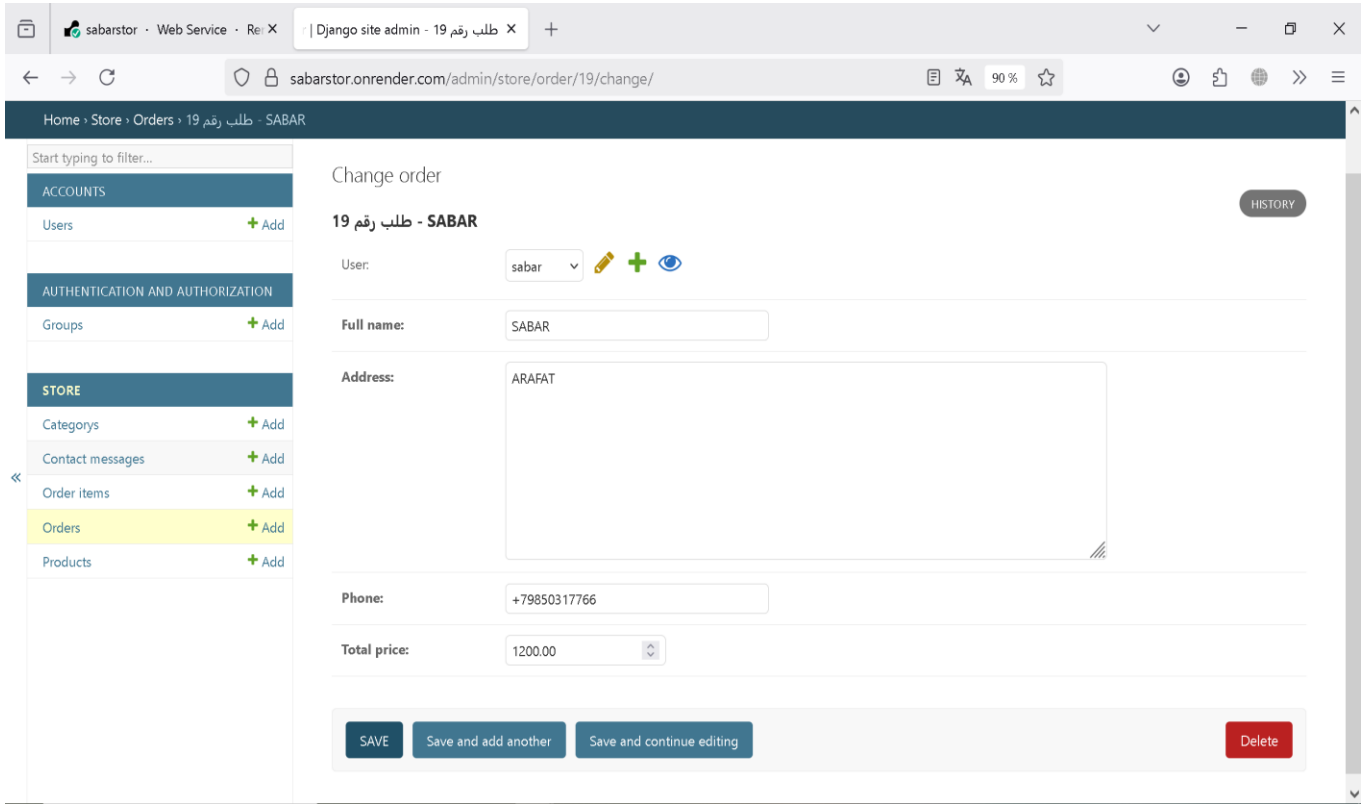




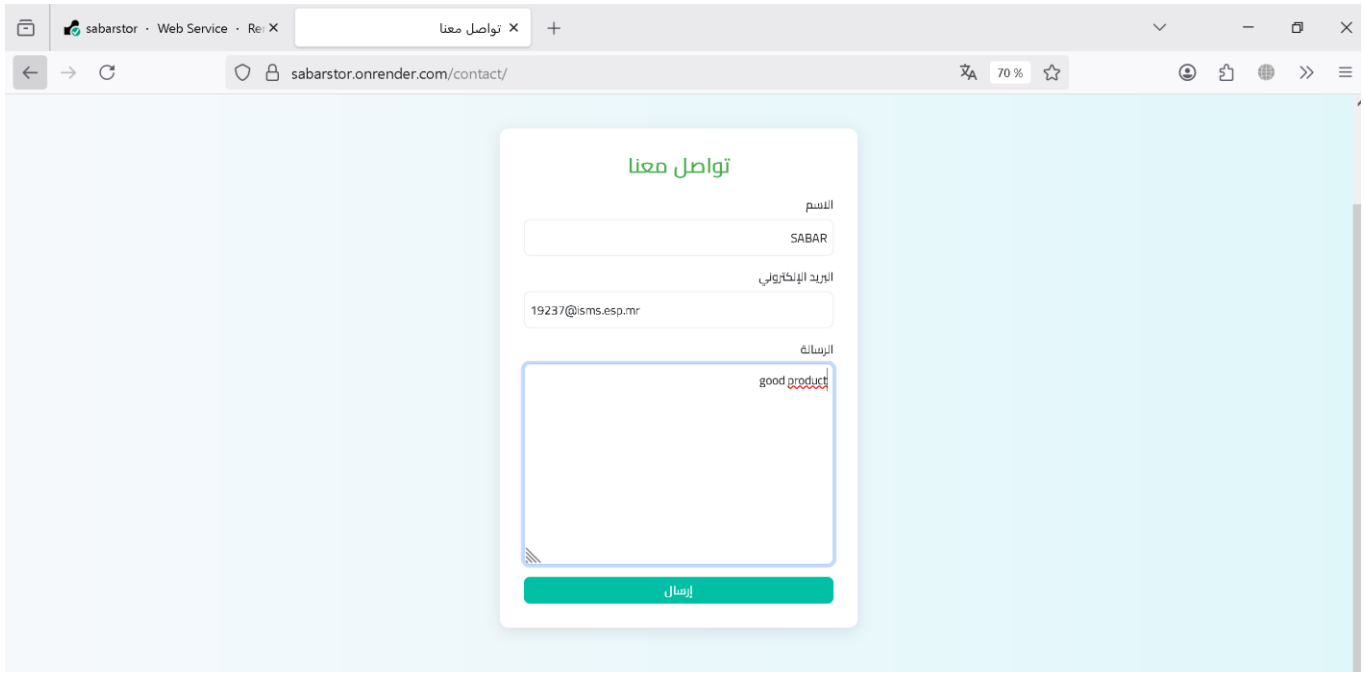
- Order completion with sending information to the database.

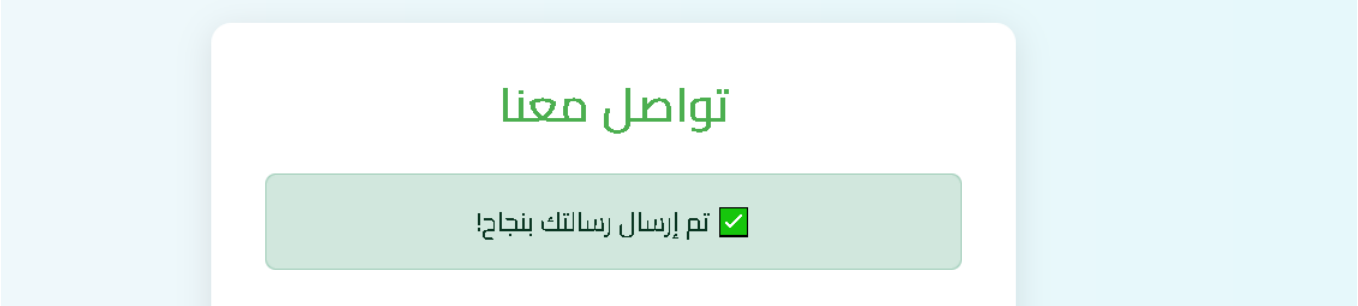


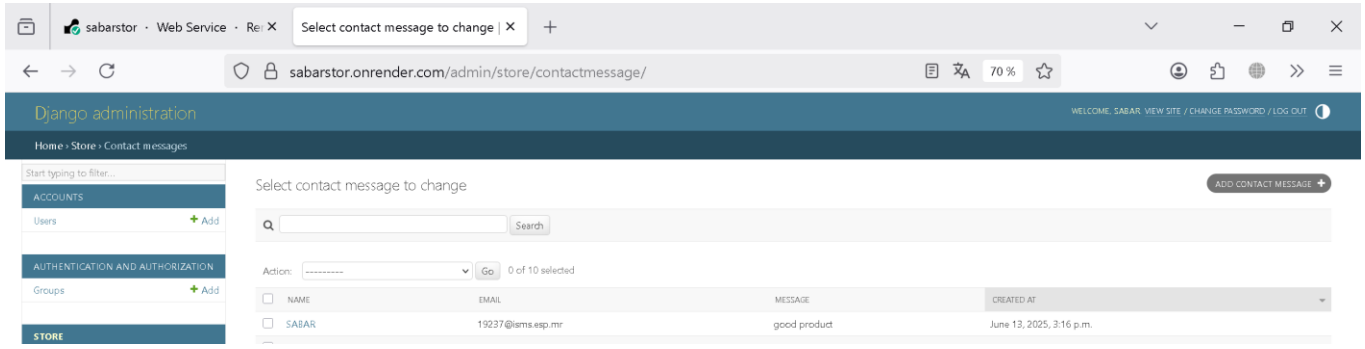




- An effective communication model that stores messages in a database with success notification.







3. Deployment and Operational Outputs

The project was deployed to the Render platform via the Dockerfile and docker-compose.yml files.

The website is running successfully at the following link:

https://sabarstor.onrender.com

The Docker environment was adopted to facilitate running the application in an isolated and reliable production environment.

Analysis & Evaluation

1.Achieved Project Objectives

The primary objectives identified at the beginning of the project have been achieved, including:

- Providing an Arabic platform for displaying products and e-commerce.
- Improving the user experience through an attractive and easy-to-use design.

- Enabling users to complete purchases in a simple and clear manner.
- Supporting the cart and displaying order details up to the checkout stage.

2.Strengths

- Design responsive interfaces.
- Using Django, which provides a powerful and efficient database management system.
- Clear separation of tasks between different applications within the project.
- Using Cloudinary to store images, ensuring fast loading and efficient storage.
- Using Docker to provide a stable and easy-to-deploy environment.
- Including tests to ensure quality output.
- Full Arabic language support in RTL format.

3.Challenges Overcome

- Some errors occurred during user registration and logout due to modifications to the CustomUser model, and these were successfully resolved.
- Some elements were not displayed in the interfaces due to conflicting static files. This was resolved by reorganizing the files and clearing the cache.
- Difficulty deploying the project for the first time due to the absence of some variables in the .env file. This was resolved through careful documentation of the requirements.

4.Suggested areas for improvement:

- Add a user rating and rating system for products.
- Link the platform to an actual electronic payment gateway.
- Provide an administrator control panel to manage products and orders.
- Improving the user interface by adding interactive effects (JS/AJAX).

Conclusion

The SabarStor project is a practical example of a fully integrated web application for the Arabic user, combining ease of use with the essential functionality of a modern e-store. Developing this project using the Django framework provided a deep understanding of how to build a system based on the MVC architecture that supports multiple users, shopping sessions, and dynamic content management. The adoption of technologies such as Cloudinary for image storage and Docker for easy deployment demonstrated the effectiveness of cloud solutions and container tools in improving the development experience and ensuring stability in production environments. The project has proven successful through its successful testing and practical results that reflected the achievement of the desired goals, making it a solid foundation for future expansions that include more advanced integrations and user experience improvements.

Future Recommendations

- Integration with online payment gateways: To enable real-world online purchases.
- Expanded administrative functionality: By adding an admin dashboard to manage products, orders, and users.
- Develop a companion mobile app: Using React Native or Flutter to provide a better user experience on mobile devices.
- Increased automated testing: To ensure app stability and cover various usage scenarios.
- Integration with analytics tools: Such as Google Analytics to understand user behavior and improve performance.

References

1. Django Documentation. (2024). *The Web framework for perfectionists with deadlines.* Retrieved from: https://docs.djangoproject.com/en/5.2/
2. Bootstrap RTL. (2024). *Right-to-left support for Bootstrap 5.* Retrieved from: https://bootstrap.rtlcss.com
3. Docker Documentation. (2024). *Developing and Deploying Applications with Docker.* Retrieved from: https://docs.docker.com/
4. Cloudinary. (2024). *Media Management and Optimization.* Retrieved from: https://cloudinary.com/documentation
5. Gunicorn Documentation. (2024). *Green Unicorn: Python WSGI HTTP Server for UNIX.* Retrieved from: https://docs.gunicorn.org/
6. GitHub Docs. (2024). *Getting started with Git and GitHub.* Retrieved from: https://docs.github.com/
7. Python Official Documentation. (2024). *Python 3.10+ Reference.* Retrieved from: https://docs.python.org/3/
8. SQLite Documentation. (2024). *Lightweight SQL database engine.* Retrieved from: https://www.sqlite.org/docs.html