Storify E-Commerce Web Application

Team Members

- **Name 1:** Omar Ahmed
- **Name 2:** Ahmed Tamer
- **Name 3:** Samaa Khaled

Team Number: #28

GitHub Repository: Elgazar2005/Storify-

1. Introduction

Storify is a web-based e-commerce application developed using the Flask framework. The system allows customers to browse products, manage shopping carts, and place orders, while sellers manage products and orders, and administrators oversee the platform.

The project follows a layered architecture using **Controllers, Repositories, and Blueprints** to ensure maintainability and scalability.

2. System Architecture

The application is structured using the MVC-inspired pattern:

- **Controllers (Blueprints)**
  Handle routing and request/response logic.
- **Repositories**
  Manage data access and business logic.
- **Templates (Jinja2)**
  Responsible for UI rendering.
- **Flask App (**app.py**)**
  Registers blueprints and initializes the application.

Main Components

- Authentication Controller
- Customer Controller
- Product Controller
- Cart Controller
- Order Controller
- Seller Controller
- Admin Controller

3. Functional Requirements

The system implements the following functionalities:

Customer

- User registration and login
- Browse products
- View product details
- Add/remove items from cart
- Place orders
- View order history

Seller

- Seller dashboard
- Add and manage products
- View orders related to their products
- Receive notifications

Admin

- Admin dashboard

- View all users
- Verify sellers
- Verify products

4. Non-Functional Requirements

- Modular and scalable code structure
- Session-based authentication
- Clear separation of concerns
- Easy deployment and configuration
- Readable and maintainable codebase

5. Deployment

- Framework: **Flask**
- Environment: **Python Virtual Environment**
- Run Command: **python app.py**

```
python app.py
```

- Application runs locally on:

http://127.0.0.1:5000

6. Testing Evidence

6.1 Unit Tests (3 Required)

Test 1: User Repository – Create User

- Ensures users can be created successfully.

Test 2: Product Repository – Get All Products

- Verifies that product list retrieval works correctly.

Test 3: Cart Repository – Add Item to Cart

- Confirms items are added with correct quantity.

6.2 Integration Test (1 Required)

Test: Product List Route

- Endpoint: /products
- Verifies HTTP 200 response
- Confirms rendered product data

6.3 Test Report Summary

Test Name	**Type**	**Result**
User Creation Test	Unit	Pass
Product Retrieval Test	Unit	Pass
Cart Add Test	Unit	Pass
Products Route Test	Integration	Pass

7. Documentation

Technical Documentation

- Flask Blueprints for modular routing
- Repository pattern for data handling
- Jinja2 templates for views
- Session management for authentication

User Documentation

- Users can sign up and log in
- Customers can shop and place orders
- Sellers manage products and orders

- Admins supervise the platform

8. Presentation & Demo

- Live demonstration of:
    - User login
    - Product browsing
    - Cart and checkout
    - Seller dashboard
    - Admin dashboard
- Each team member participated in the presentation
- System demonstrated successfully during class

9. Conclusion

The Storify project successfully meets all functional and non-functional requirements.
It demonstrates a complete e-commerce workflow with clean architecture, proper testing, and deployment of readiness.

Submitted by:
Storify Development Team
Course: Introduction Software Engineering
Year: 2025