Zewail City of Science, Technology and Innovation

University of Science and Technology

School of Computational Sciences and Artificial Intelligence

CSAI 203 - Fall 2025

Introduction to Software Engineering

< Storify >

<Phase 4: Core Functionality Prototype>

Team Number: #28

Team Members:

• Ahmed Tamer — ID:202401457

• Omar Ahmed — ID:202400354

• Samaa Khaled — ID:202401280

Representative Contact info:

s-omar.elgazar@zewail.edu.eg

1. Introduction

This document presents the progress achieved in **Phase 4** of the *Storify* project.
The objective of this phase is to implement the **core functionality** of the system using the **MVC architecture** and CSV files as the data layer.

During Phase 4, the focus was on:

- Preparing structured CSV data files
- Implementing Models
- Developing Repositories for data access
- Implementing Controllers (Blueprints) for system logic
- Creating Views (HTML Templates) to display the data
- Ensuring the system runs without errors and supports the main user roles:
    - Customer
    - Seller
    - Admin

---

2. Project Architecture

Storify follows an **MVC (Model–View–Controller)** architecture.

2.1 Models (M)

Models represent the data structure of core system entities such as:

- User
- Product
- Cart
- CartItem
- Order
- OrderItem
- Notification

Each model maps directly to fields stored in CSV files, following a clean object-oriented structure.

2.2 Views (V)

Views are implemented using **HTML Jinja templates**.

Examples include:

- index.html
- login.html, signup.html
- products.html, product_details.html
- cart.html, checkout.html
- orders.html
- Seller Dashboard templates
- Admin Dashboard templates

Each template receives dynamic data passed by the controllers and renders user-specific content.

2.3 Controllers (C)

Controllers manage the application logic and connect Repositories to Templates.

Main controllers include:

- auth_controller.py
- user_controller.py
- product_controller.py
- cart_controller.py
- order_controller.py
- seller_controller.py
- admin_controller.py

Each controller handles its own Blueprint and routes.

---

3. Data Layer (CSV Files)

All system data is stored using CSV files:

User Data

- users.csv
  Contains all user accounts with fields:
  id, username, email, password, role

Product & Cart Data

- products.csv
- carts.csv
- cart_items.csv

Order Data

- orders.csv
- order_items.csv

Notifications

- notifications.csv
  Used mainly for Seller/Admin updates.

Each CSV file is structured and filled with sample test data to allow complete system testing.

---

4. Repositories

Repositories are responsible for **reading and writing CSV data**.

Implemented repositories include:

- user_repository.py
- product_repository.py
- cart_repository.py
- order_repository.py
- notification_repository.py
- repository_factory.py

Each repository includes essential methods such as:

- get_all()

- get_by_id()
- create()
- update()
- (plus specialized methods depending on the entity type)

This layer ensures clean separation between the logic and the data source.

---

5. Controllers & Main Features

5.1 Authentication

Features implemented:

- Login
- Signup
- Session creation
- Role-based redirects (Customer → profile, Seller → dashboard, Admin → admin panel)

5.2 Customer Features

- View products
- View product details
- Add to cart / remove from cart
- View cart
- Checkout (Order creation)

5.3 Seller Features

- Seller dashboard
- View products belonging to the seller
- View seller-specific orders
- View notifications (optional feature)

5.4 Admin Features

- Admin dashboard

- View and verify sellers
- View and verify products
- View system reports (simple CSV-based reporting)

All features are connected end-to-end:
CSV → Repository → Controller → Template

---

6. Templates (Views)

All required templates were created, formatted, and connected to controllers.

Template examples:

- /templates/index.html
- /templates/login.html
- /templates/signup.html
- /templates/products.html
- /templates/cart.html
- /templates/orders.html

Seller templates:

- seller/dashboard_seller.html
- seller/seller_products.html
- seller/seller_orders.html

Admin templates:

- admin/admin_dashboard.html
- admin/verify_sellers.html
- admin/verify_products.html
- admin/reports.html

Each template dynamically displays data passed from controllers.

---

7. System Testing

At the end of Phase 4:

- The system runs without errors
- All core features operate end-to-end
- CSV files load correctly
- Controllers properly return data to views
- Role-based restrictions work
- Navigation flows (Login → Dashboard → Features) are fully functional

---

8. Summary of Completed Work

During Phase 4, the following major components were fully implemented:

8.1 Data Layer (CSV)

All CSV files required for system logic have been created and filled with test data:

- users.csv
- products.csv
- carts.csv, cart_items.csv
- orders.csv, order_items.csv
- notifications.csv

8.2 Models

Models were implemented for:

- User
- Product
- Cart + CartItem
- Order + OrderItem
- Notification

8.3 Repositories

Repositories now handle reading and writing CSV data:

- UserRepository

- ProductRepository
- CartRepository
- OrderRepository
- NotificationRepository
- RepositoryFactory

8.4 Controllers (Blueprints)

Core system logic has been implemented across:

- Authentication (login, signup, logout)
- User Controller (profiles)
- Product Controller
- Cart Controller
- Order Controller
- Seller Controller
- Admin Controller

All controllers use session handling and role-based access control.

8.5 Views/Templates

All required templates for customers, sellers, admins, and general pages were implemented.

9. Pending Work (To Be Completed in Next Phases)

Although Phase 4 completes 50% of the system's core functionality, several features remain incomplete and will be covered in the next development phases. These pending items are outlined below:

• Advanced Reports System
This feature has not been implemented yet. It will be added in the next phase to allow the admin to generate system reports.

• Product and Seller Approval Logic
The admin can currently view sellers and products, but the actual approval or rejection actions have not been implemented.

• Improved UI Styling
CSS styling is still basic. More professional styling will be added later, potentially using a CSS framework.

• Enhanced Notifications
The notification system is partially implemented. Only sellers receive notifications for now; additional types of notifications will be added.

• Input Validation and Error Handling
Form validation and error messages will be improved to ensure better user experience and cleaner system behavior.

• Security Enhancements
Session protection and access control will be extended to prevent unauthorized access and improve system security.

10. Conclusion

Phase 4 (Core Functionality) has been successfully completed.

The system now supports:

- Authentication
- User roles
- Product browsing
- Cart operations
- Orders
- Admin and Seller dashboards
- Notifications
- A complete MVC structure

This provides a strong foundation for the next development phases, including enhanced validation, UI improvements, and additional business logic.