Software Design Document (SDD)

Project Title:
Storify – Smart E-Commerce Platform

Document Title:
Software Design Document (SDD)

Course Name:
CSAI 203 – Introduction to Software Engineering

Instructor:
Dr. Mohamed Sami Rakha

Team Number:
Team #28

Team Members:
• Ahmed Tamer — ID: 20240145 • Omar Ahmed — ID: 202400354 • Samaa Khaled — ID: 202401280

Table of Contents

1. Introduction

1.1 Purpose of the Document

The purpose of this Software Design Document (SDD) is to provide a detailed and structured description of the system design for the *Storify – Smart E-Commerce Platform*. This document translates the functional and non-functional requirements defined in the SRS into a complete technical design that guides developers during implementation. It includes the architectural design, data design, user interface design, UML diagrams, and the application of the MVC pattern.

The SDD ensures that all system components are clearly defined and aligned with the project requirements, thereby minimizing ambiguity and supporting accurate development.

1.2 Scope of the Design Phase

This document covers all aspects required for designing the Storify system, including the system architecture, module-level design, database schema, user interface wireframes, and detailed UML diagrams (class diagrams and five sequence diagrams).

The design phase focuses on how the system will be built rather than what the system does.

It defines the internal structure of the application using the MVC design pattern, describes the interactions between system components, and identifies the technologies that will be used.

1.3 Intended Audience

This document is intended for:

- **Course Instructor:** To evaluate the system's design and ensure compliance with course standards.
- **Developers:** To serve as a blueprint for implementing the system architecture and modules.
- **Project Team Members:** To maintain a consistent understanding of the system structure.
- **System Designers and Reviewers:** To assess the quality, completeness, and correctness of the design.

1.4 Overview of the Contents

This document includes the following sections:

- **System Overview:** Provides a high-level description of the Storify system and its design goals.
- **Architectural Design:** Presents the monolithic architecture, system components, and the technology stack.
- **Detailed Design:** Includes the MVC model, class diagram, sequence diagrams, and the interaction between components.
- **UI/UX Design:** Presents the wireframes and layout of the main user interfaces.
- **Data Design:** Covers the database schema, ER diagram, file structure, and data dictionary.
- **Conclusion:** Summarizes the key design decisions and their importance to the system.

2. System Overview

2.1 Brief Description of the System

Storify is a smart e-commerce platform designed to provide users with an intuitive and seamless online shopping experience. The system allows customers to browse products, add items to a shopping cart, place orders, and make payments securely. On the administrative side, the system provides tools for managing products, inventory, orders, and user accounts efficiently.

The platform aims to enhance user satisfaction through fast performance, responsive design, and reliable data management, while ensuring the system can scale to accommodate future growth.

2.2 Key Design Goals and Constraints

Design Goals:

- **Scalability:** The system should support a growing number of users and products without performance degradation.
- **Modularity:** Components are separated logically using the MVC pattern to facilitate maintenance and updates.

- **Security:** User data and payment information must be securely stored and transmitted.
- **Usability:** The interface should be simple and intuitive for both customers and administrators.

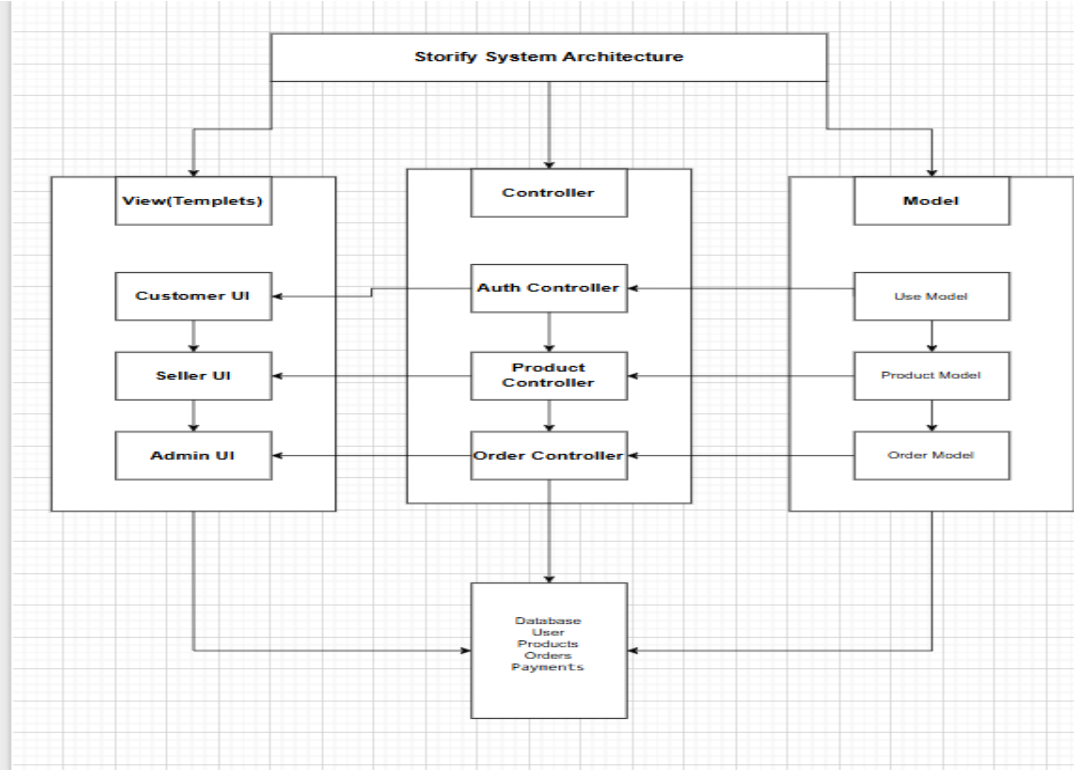
Constraints:

- **Technology Limitations:** The system uses a monolithic architecture, relying on specific technologies such as HTML/CSS/JS for frontend and MySQL for data storage.
- **Performance Requirements:** The system must maintain responsive page load times under normal traffic.
- **Development Timeline:** Design and implementation must adhere to the course project schedule.

3. Architectural Design

3.1 System Architecture Diagram

Storify adopts a **monolithic architecture**, where all system components—presentation, business logic, and data management—operate within one unified application.
The architecture follows the MVC pattern to ensure clear separation of responsibilities, ease of maintenance, and smoother development workflow.



3.2 Discussion of Architectural Style and Components

The system uses a **Monolithic Architecture**, in which all modules—including authentication, product management, orders, cart, and payment—are packaged and deployed as a single codebase.

This architecture simplifies:

- deployment
- debugging
- internal communication
- consistency across modules

It is ideal for academic projects of medium scope, ensuring stability and easier versioning.

The main components include:

- **View Layer:** Manages UI and user interactions.
- **Controller Layer:** Receives inputs, applies logic, and coordinates system flow.
- **Model Layer:** Manages data, rules, and communication with the database.
- **Database:** Stores products, users, orders, payments, and system data.

3.3 Technology Stack and Tools

The system is developed using the following technologies:

- **Frontend:** HTML, CSS, JavaScript
- **Backend:** Python and Flask framework
- **Database:** MySQL (or CSV files if opted)
- **Architecture Style:** Monolithic, MVC design pattern
- **Design Tools:**
    - Draw.io (UML diagrams)
    - Canva / Figma (Wireframes)
    - GitHub (version control)
- **Testing Tools:** Flask built-in test client, manual testing

4. Detailed Design

4.1 Model–View–Controller (MVC) Pattern

4.1.1 Description of the MVC Pattern

The Model–View–Controller (MVC) pattern is an architectural design approach that separates an application into three connected layers: the **Model**, **View**, and **Controller**.

This separation ensures modularity, maintainability, and scalability, which are essential for a large system like *Storify – Smart E-Commerce Platform*.

MVC was chosen for Storify because it:

- Supports **clear separation of concerns**, making each module easier to update.
- Improves **team productivity** by allowing parallel development of UI, logic, and routing.
- Enhances **maintainability**, especially with multiple user types (Customer, Seller, Admin).
- Reduces complexity by organizing the system into independent but cooperative layers.

4.1.2 Mapping of Project Components to MVC

MVC Layer	**Storify Components**	**Description**
Model	User, Customer, Seller, Admin, Product, Order, Payment, Notification	Represents system data, business rules, relationships, and operations such as order processing, product management, and authentication.
View	Web UI pages (HTML + CSS + JS), templates for login, home, product details, cart, checkout, analytics, seller	Handles the interface that users interact with, displaying data from the Model and capturing user input.

	dashboard, admin dashboard	
Controller	Flask route handlers (/login, /products, /add-product, /checkout, /admin/verify	Receives user requests, calls the appropriate Model functions, and returns the correct View to the user.

4.1.3 Responsibilities of Model, View, and Controller

Model Responsibilities

- Represent system entities (User, Product, Order, Payment, Notification).
- Enforce business rules (stock checking, seller verification, product approval).
- Manage all data storage, retrieval, and validation.
- Support the background Checker module for stock monitoring.
- Ensure consistency of user roles and permissions (Customer, Seller, Admin).

View Responsibilities

- Display user interfaces for browsing, purchasing, managing products, and admin operations.
- Show dynamic content such as product lists, order details, and notifications.
- Collect user input (filters, cart updates, queries, product submission forms).
- Provide responsive, user-friendly pages for all actors.

Controller Responsibilities

- Handle incoming HTTP requests from Customers, Sellers, and Admins.
- Call the appropriate Model operations (e.g., add product, validate login, place order).
- Select and render the correct View with updated data.
- Coordinate system functionalities such as checkout, authentication, and notifications.
- Ensure secure access based on role (RBAC-like behavior).

4.1.4 Interaction Between Components

The following describes how Storify components interact during system operations:

1. **User Action → Controller**
   A Customer searches for a product; a Seller adds an item, or an Admin verifies a listing.

2. **Controller → Model**
   The Controller requests data or operations from the Model
   (e.g., retrieve products, update stock, validate user credentials, store a new order).

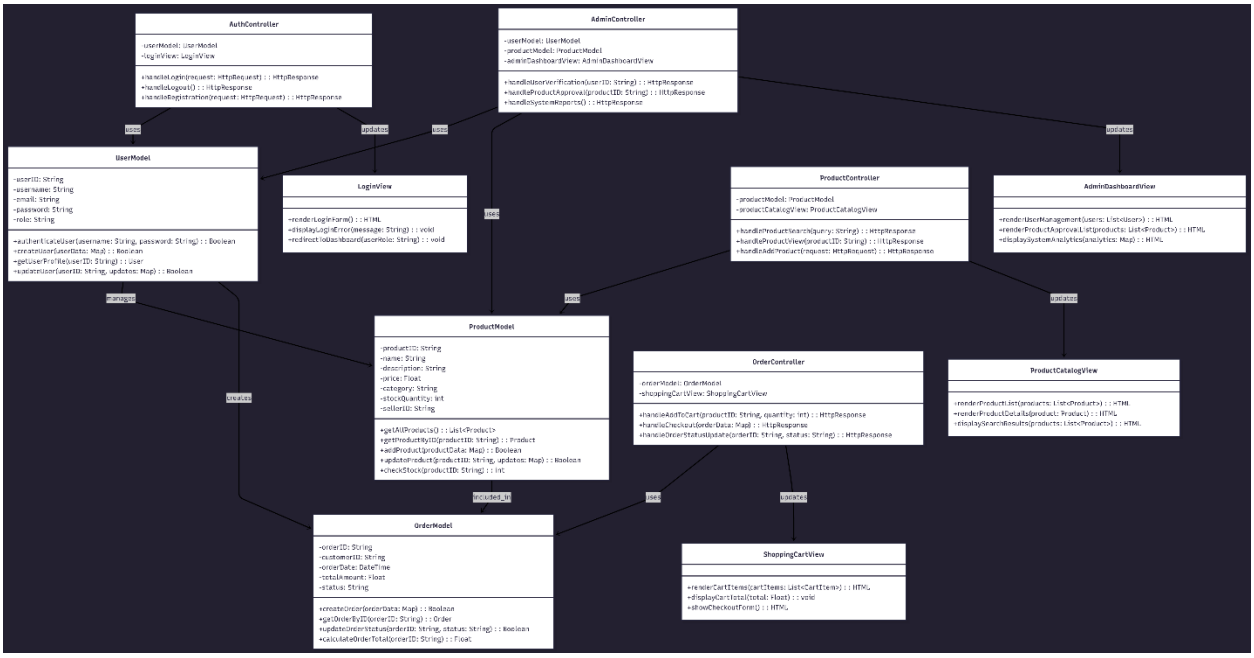
3. **Model → Controller**
   The Model returns structured data such as product lists, order confirmations, or error results.

4. **Controller → View**
   The Controller renders the appropriate template (home page, cart, seller dashboard, admin panel) and injects the Model data into it.

5. **View → User**
   The UI displays updated information—products, notifications, payment confirmation, etc.

4.2 UML Diagrams

4.2.1 Detailed Class Diagram

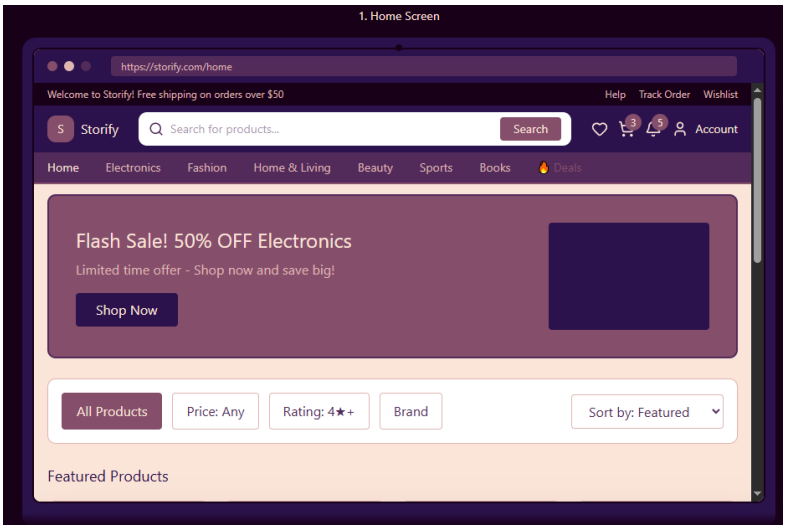


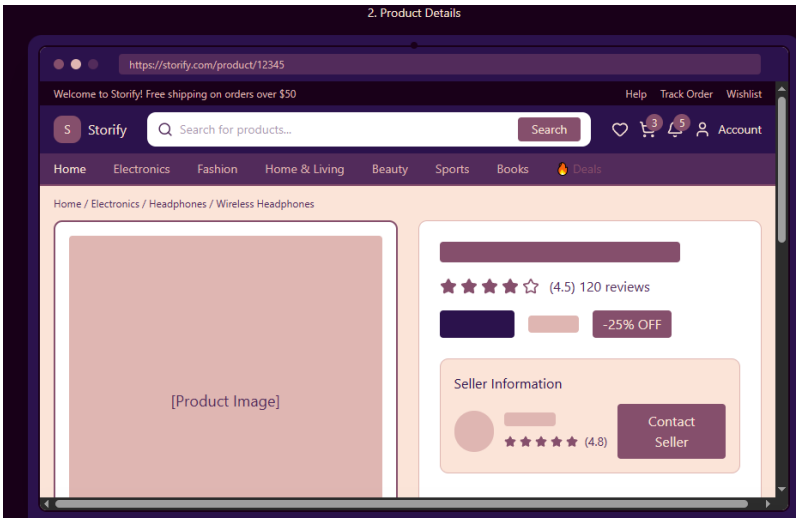
4.3 UI/UX Design

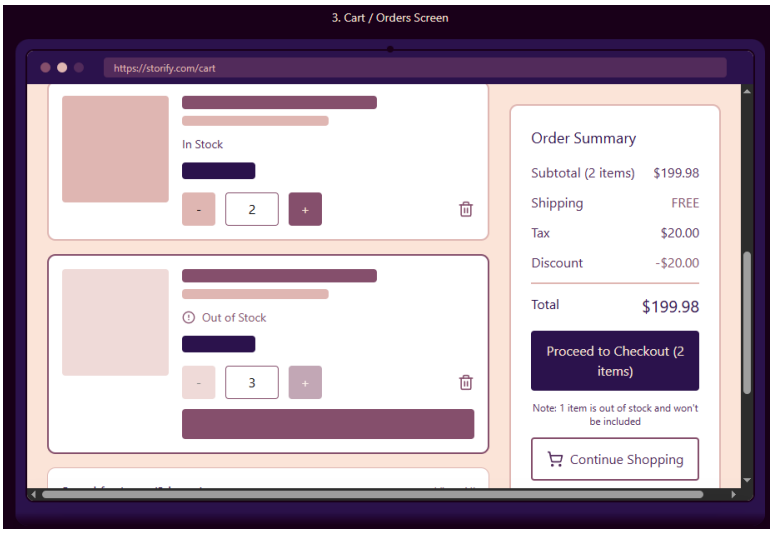
4.3.1 Wireframes / Mockups

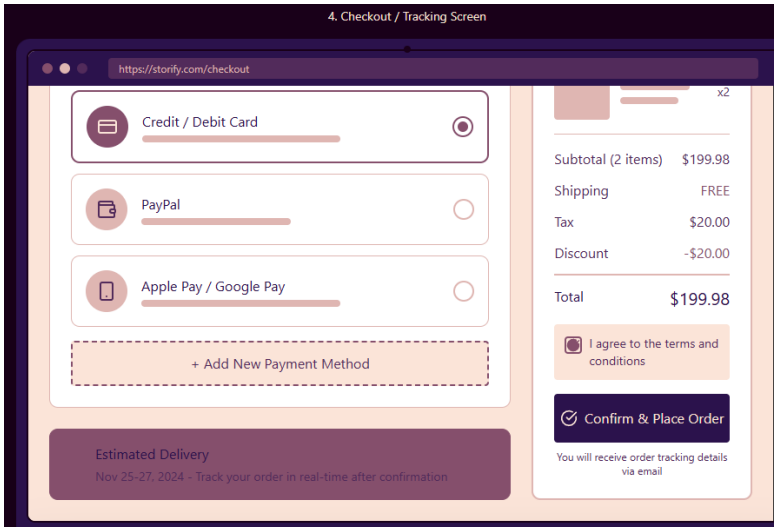
The following wireframes show the main pages of the Storify platform for customers, sellers, and admins. They highlight layout, navigation, and core functions to ensure a clear and user-friendly interface.

Customer View:

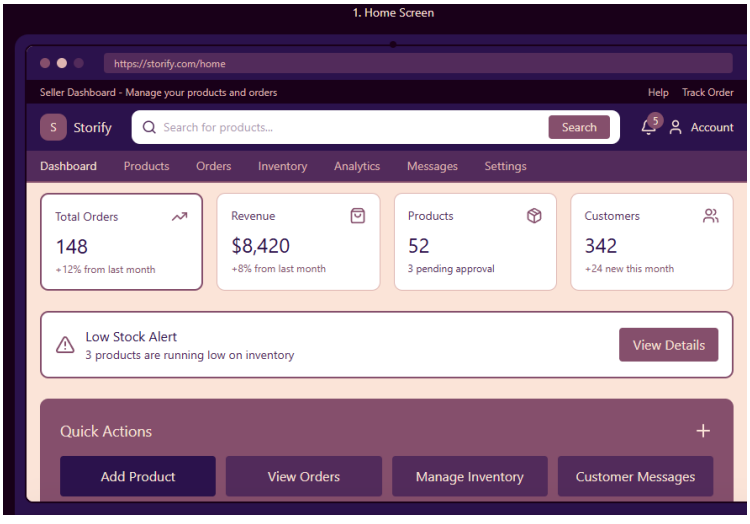

1. Home Screen
https://storify.com/home
Welcome to Storify! Free shipping on orders over $50
Help
Track Order
Wishlist
Storify
Search for products...
Search
Account
Home
Electronics
Fashion
Home & Living
Beauty
Sports
Books
Deals
Flash Sale! 50% OFF Electronics
Limited time offer - Shop now and save big!
Shop Now
All Products
Price: Any
Rating: 4★+
Brand
Sort by: Featured
Featured Products

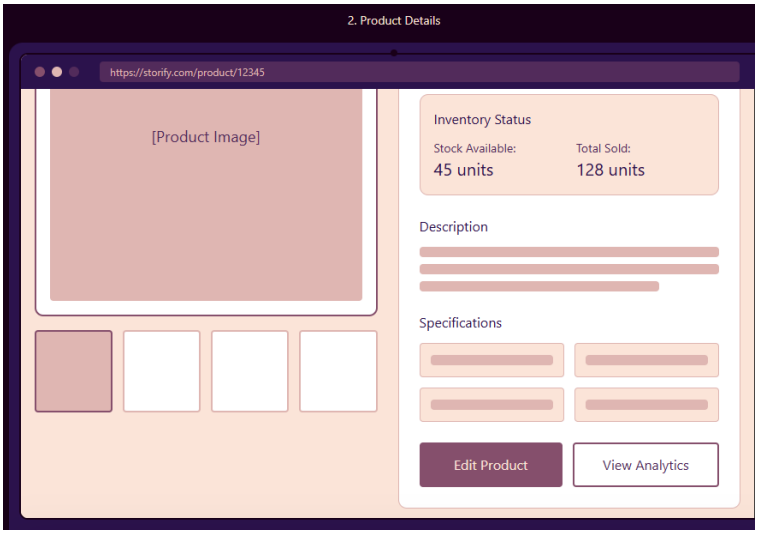

2. Product Details
https://storify.com/product/12345
Welcome to Storify! Free shipping on orders over $50
Help
Track Order
Wishlist
Storify
Search for products...
Search
Account
Home
Electronics
Fashion
Home & Living
Beauty
Sports
Books
Deals
Home / Electronics / Headphones / Wireless Headphones
[Product Image]
(4.5) 120 reviews
-25% OFF
Seller Information
(4.8)
Contact Seller

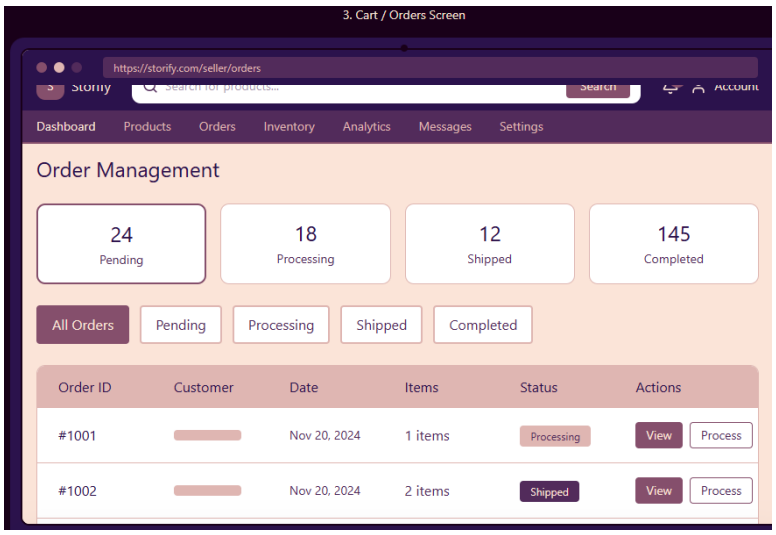

3. Cart / Orders Screen
https://storify.com/cart
In Stock
2
Out of Stock
3
Order Summary
Subtotal (2 items)
$199.98
Shipping
FREE
Tax
$20.00
Discount
-$20.00
Total
$199.98
Proceed to Checkout (2 items)
Note: 1 item is out of stock and won't be included
Continue Shopping

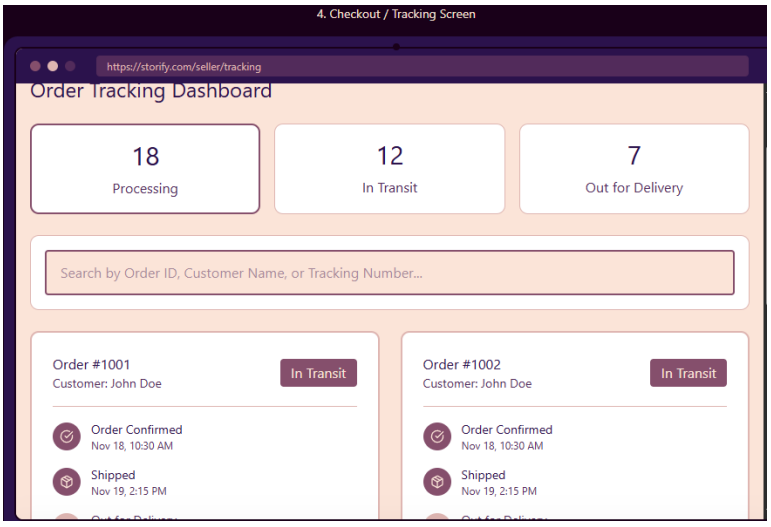

4. Checkout / Tracking Screen
https://storify.com/checkout
Credit / Debit Card
PayPal
Apple Pay / Google Pay
+ Add New Payment Method
Estimated Delivery
Nov 25-27, 2024 - Track your order in real-time after confirmation
Subtotal (2 items)
$199.98
Shipping
FREE
Tax
$20.00
Discount
-$20.00
Total
$199.98
I agree to the terms and conditions
Confirm & Place Order
You will receive order tracking details via email

Seller View:

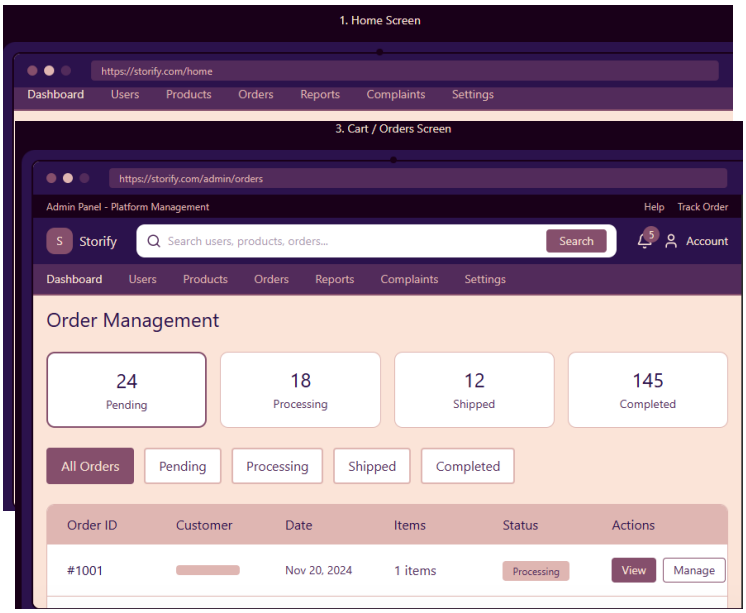

1. Home Screen
https://storify.com/home
Seller Dashboard - Manage your products and orders
Help
Track Order
Storify
Search for products...
Search
Account
Dashboard
Products
Orders
Inventory
Analytics
Messages
Settings
Total Orders
148
+12% from last month
Revenue
$8,420
+8% from last month
Products
52
3 pending approval
Customers
342
+24 new this month
Low Stock Alert
3 products are running low on inventory
View Details
Quick Actions
Add Product
View Orders
Manage Inventory
Customer Messages


2. Product Details
https://storify.com/product/12345
[Product Image]
Inventory Status
Stock Available:
45 units
Total Sold:
128 units
Description
Specifications
Edit Product
View Analytics

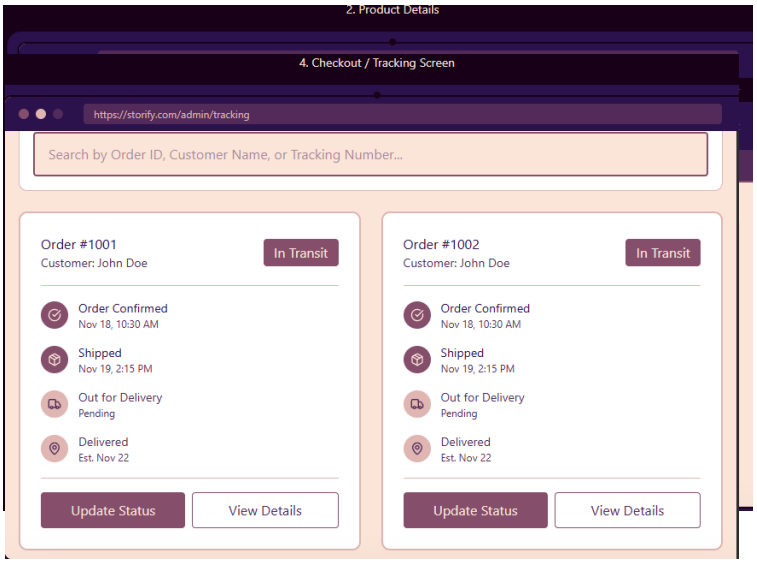




Admin View:



4.4



Data Design

4.4.1 Database Schema / ER Diagram (Description)

The database of the Storify e-commerce system is designed to support users, products, orders, payments, and notifications in a scalable and structured manner.

The Entity–Relationship (ER) model includes six main entities: **User, Product, Order, OrderItem, Payment, and Notification**.

Each entity stores essential information and maintains relationships that ensure data integrity and efficient system operations.

• **User–Product Relationship:**
A Seller can create multiple products. This establishes a **one-to-many** relationship between *User (Seller)* and *Product*.

• **User–Order Relationship:**
Each Customer may place multiple orders. This forms a **one-to-many** relationship between *User (Customer)* and *Order*.

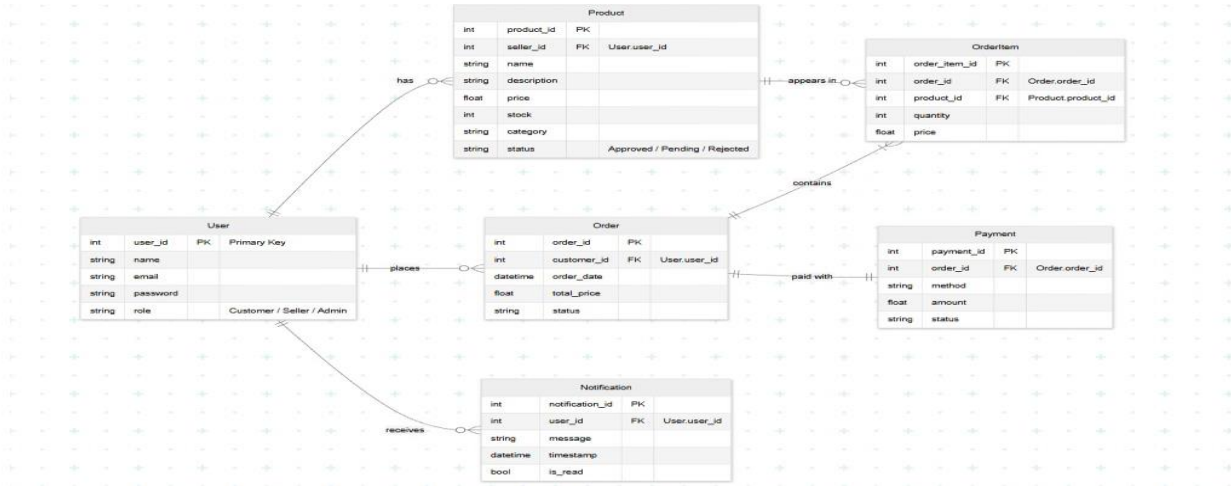
• **Order–OrderItem Relationship:**
Each order contains one or more purchased products. This is a **one-to-many** relationship between *Order* and *OrderItem*.

• **Product–OrderItem Relationship:**
A Product may appear in many orders. This is a **one-to-many** relationship from *Product* to *OrderItem*.

• **Order–Payment Relationship:**
Each order has exactly one payment record, forming a **one-to-one** relationship.

• **User–Notification Relationship:**
Each user may receive many notifications, giving a **one-to-many** relationship.

This database structure ensures consistency, supports efficient queries, and allows the system to scale as more users, sellers, and orders are added.



4.4.3 Data Dictionary

Below is the complete data dictionary for the Storify system.
It defines each table, field name, data type, and its purpose.

Table: User:

Field	Type	Description
user_id	INT (PK)	Unique identifier for each user
name	VARCHAR (100)	Full name of the user
email	VARCHAR (150)	User's login email (unique)
password	VARCHAR (255)	Hashed password
role	ENUM('customer','seller','admin')	Specifies the user type

Table: Product:

Field	Type	Description
productid	INT (PK)	Unique product ID
seller_id	INT (FK → User.user_id)	The seller who owns the product
name	VARCHAR(150)	Product name
description	TEXT	Detailed product information
price	DECIMAL(10,2)	Product price
stock	INT	Quantity available in inventory
category	VARCHAR(100)	Product category

Table: Order

Field	Type	Description
order_id	INT (PK)	Unique ID for the order
customer_id	INT (FK → User.user_id)	Customer who placed the order
order_date	DATETIME	Date and time of order
total_price	DECIMAL(10,2)	Total price of the order

Table: OrderItem

Field	Type	Description
order_item_id	INT (PK)	Unique ID for the order item

order_id	INT (FK → Order.order_id)	The order this item belongs to
product_id	INT (FK → Product.product_id)	Product being purchased
quantity	INT	Number of units ordered
price	DECIMAL(10,2)	Unit price at purchase time

Table: Payment

Field	Type	Description
payment_id	INT (PK)	Unique payment ID
order_id	INT (FK → Order.order_id)	The related order
method	ENUM('card','cash')	Payment method
amount	DECIMAL(10,2)	Paid amount
status	ENUM('pending','completed','failed')	Payment status

Table: Notification

Field	Type	Description
notification_id	INT (PK)	Unique notification ID
user_id	INT (FK → User.user_id)	User receiving the notification
message	TEXT	Notification text
timestamp	DATETIME	Time sent
is_read	BOOLEAN	Read/unread flag

5. Conclusion

5.1 Summary of the Design Phase

The design phase defined how Storify will be built using the MVC pattern for modularity and clear separation of concerns. Key modules—including authentication, product management, orders, notifications, and admin control—were mapped to MVC layers.

We created UML diagrams (class and sequence diagrams) to represent system structure and interactions, developed UI/UX wireframes for main screens, and defined the database schema and data dictionary. This design provides a solid foundation for development, scalability, and maintenance.