**PageFlow: Bookstore Inventory and Sales Automation**

**Group 3**

**Created by Team: Kiana Lang, Gustavo Hernandez, Fernando Lugo Urena**

**09/07/2025**

Table of Contents

[1. Project Vision and Description 3](#_Toc208156318)

[2. Team Roles 4](#_Toc208156319)

[3. Collaboration Methodology 5](#_Toc208156320)

[4. The Definition of “Done” 6](#_Toc208156321)

[5. Product Design 7](#_Toc208156322)

[6. Sprint 1 Retrospective Summary Report 12](#_Toc208156323)

[7. Sprint 2 Retrospective Summary Report 13](#_Toc208156324)

[8. References 14](#_Toc208156325)

# Project Vision and Description

PageFlow aims to modernize the operations of a rare bookstore by replacing outdated, paper-based processes with a streamlined digital system. The system will support core functions such as inventory management, customer ordering, and secure authentication. The goal is to reduce manual workload, improve accuracy, and enhance customer satisfaction.

Key Capabilities:

* Inventory Management: Track book quantities and availability.
* Customer Ordering: Allow customers to browse and place orders.
* Authentication & User Roles: Secure login and role-based access.
* Dashboard & Reporting: Visualize inventory and sales metrics.
* Book Search: Fast and accurate search with autocomplete.
* Dev Environment Setup: CI/CD pipeline for reproducible builds.

# Team Roles

Product Owner and Database Engineer– Gustavo  
Defines product vision, manages the product backlog, and designs the database schema to ensure data integrity and backend support.

Scrum Master – Kiana  
Facilitates Scrum ceremonies, removes obstacles, and ensures the team follows Agile principles.

Frontend Developer and Backend Developer – Fernando  
Designs and implements both the user interface and backend logic for core features like authentication, ordering, and dashboard metrics.

# Collaboration Methodology

We follow the Scrum framework with iterative development through structured sprints.

Key Tools:

* Communication: Discord (for meetings and team chat)
* Version Control: GitHub (for source code management and collaboration)
* Project Management: Trello (for sprint boards and task tracking)
* Documentation: Google Docs (for shared project documentation)

Meeting Schedule:

* Weekly: Sprint planning and review sessions
* Daily: Asynchronous stand-ups via Discord chat
* End of Sprint: Sprint retrospectives

Workflow Overview:

* Tasks are assigned during sprint planning
* Progress is tracked in Trello
* Code changes are reviewed via GitHub pull requests before merging

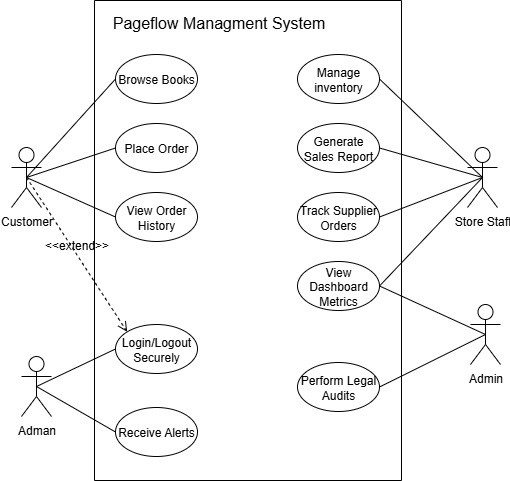
# The Definition of “Done”

A feature or task is considered "Done" when the following criteria are met:

* Code Complete & Reviewed: Code is complete, reviewed, and merged into the main branch.
* Tests Pass: All unit and integration tests pass successfully.
* Deployed to Staging: The feature is deployed to the staging environment.
* Functionality Verified: Functionality is verified against acceptance criteria.
* Documentation Updated: User guides and technical notes are updated.
* Product Owner Approved: The Product Owner has reviewed and approved the feature.
* No Critical Issues: No critical bugs or unresolved issues remain.

# Product Design

Use Case diagram

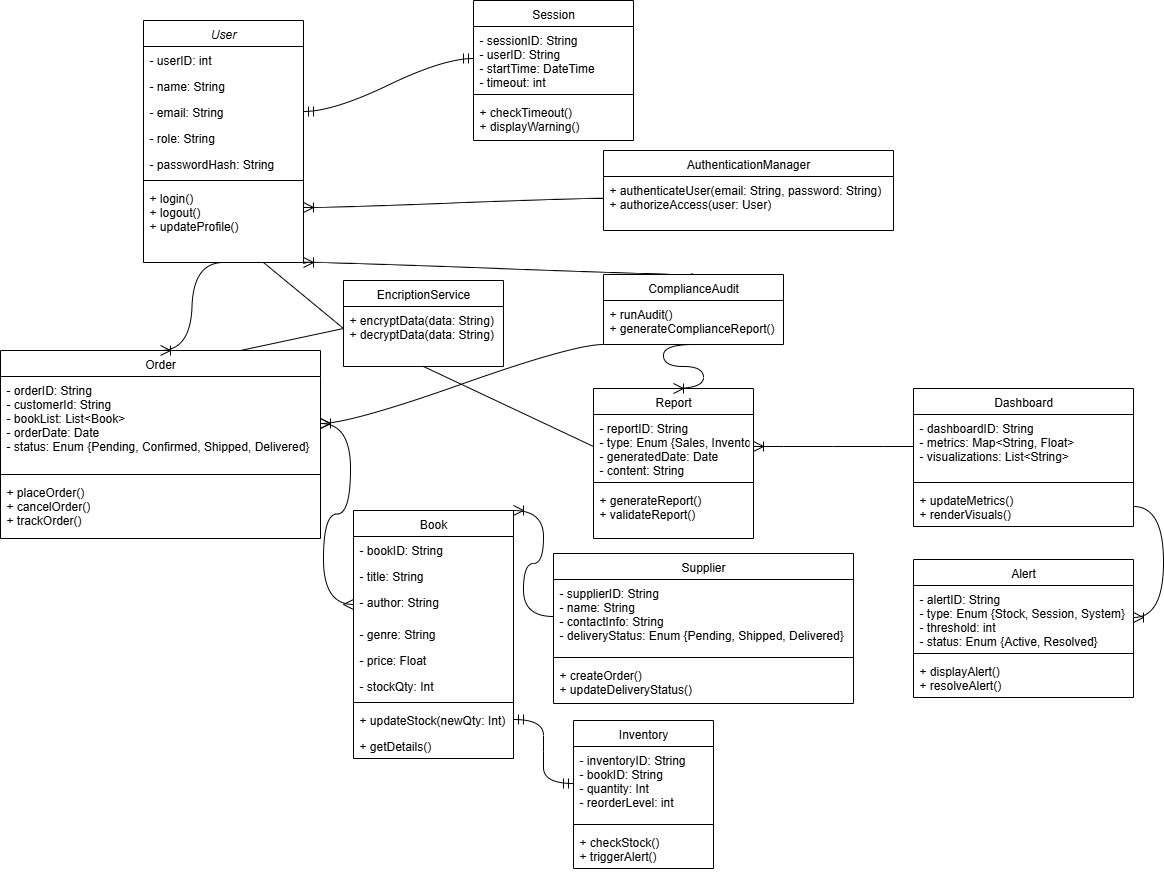


Illustrates the interactions between external actors (users and systems) and the system’s functionalities. It defines the system boundary and identifies key use cases such as browsing books, placing orders, managing inventory, and generating reports.

Key Elements:

* Actors: Customer, Staff, Admin, System
* Use Cases: Browse books, Place order, Manage inventory, Generate reports, Secure login, etc.
* Relationships: Includes extends to show functional dependencies and optional behaviors.
* System Boundary: Clearly defines what is inside the system and what is external.

Class Diagram

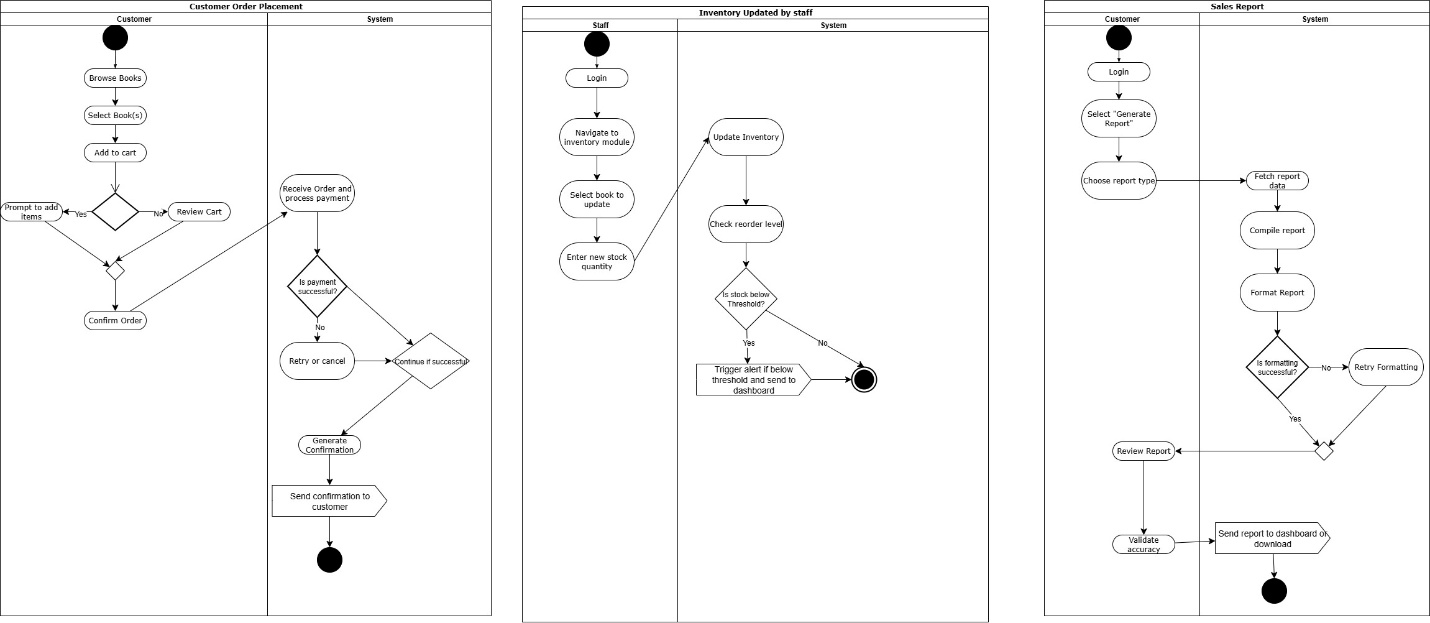


The class diagram outlines the structure of the system’s core components and defines the static structure of the system by showing classes, their attributes, methods, and relationships. It helps in understanding the data model and object-oriented design of the system.

Key Elements:

* Classes: User, Book, Order, Inventory, Supplier, Report, Alert, Dashboard, Session, AuthenticationManager, EncryptionService, ComplianceAudit
* Attributes & Methods: Each class includes visibility modifiers (+, -, #) for public, private, and protected members.
* Relationships: Includes associations like one-to-many, many-to-many, and utility class usage.

Activity Diagram



The activity diagram illustrates the workflow of a customer placing an order and helps visualize the dynamic workflows of the system, showing how activities are carried out step-by-step. Swimlanes are used to separate responsibilities between actors and system components.

Diagrams Included:

1. Customer Order Placement

* Swimlanes: Customer, System
* Flow: Login → Browse → Add to Cart → Confirm → Payment → Confirmation

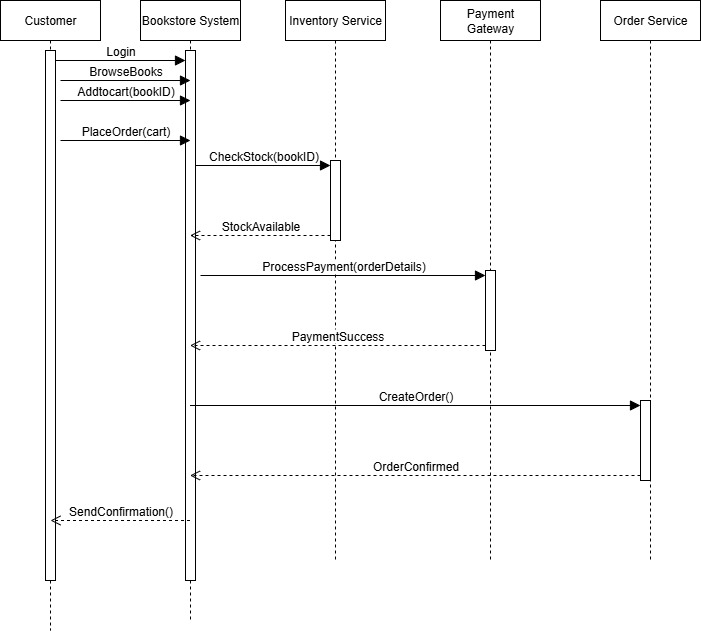
1. Inventory Update by Staff

* Swimlanes: Staff, System
* Flow: Login → Select Book → Update Stock → Check Threshold → Trigger Alert

1. Sales Report Generation

* Swimlanes: Staff, System
* Flow: Login → Select Report → Fetch Data → Format → Validate → Send

Sequence Diagram



Shows how objects interact in a time sequence to complete a specific process. It’s useful for understanding message flow and timing between components.

Diagram Included:

1. Customer Order Placement

* Participants: Customer, Bookstore System, Inventory Service, Payment Gateway, Order Service
* Flow: Login → Browse → Add to Cart → Place Order → Check Stock → Process Payment → Confirm Order → Send Confirmation

Collaboration Diagram

A diagram of a customer

AI-generated content may be incorrect.

Collaboration Diagram Description

The collaboration diagram above illustrates the interactions tween the customer and the online store front. The functions that take place on the store front are depicted by arrows. Once customers have moved into the store, they can begin their search or customization. They gain the ability to add the items to the shopping cart at this point. Once there are items in the shipping cart, they gain access it the checkout function and can complete their order.

# Sprint 1 Retrospective Summary Report

|  |
| --- |
| **Things That Went Well** |
| - Three members of the team responded promptly to the initial attempt to gather the team.  - We successfully held early conversations and assigned project roles through engaged and respectful dialogue.  - We established a foundational plan for responsibilities and begin building the initial backend code and database structure.  - The team showed a willingness to collaborate and use tools like GitHub and Discord.  - The Sprint 1 objective was achieved. |
| **Things That Could Have Gone Better** |
| - Two group members did not respond to the first round of messages and were slow to connect.  - Although both members eventually joined, they have remained disengaged from daily check-ins and sprint tasks.  - Better designs for the UI. |
| **Things That Surprised Us** |
| - One teammate (Fernando) stepped up to help with both frontend and backend responsibilities.  - Initial backend development moved faster than expected, allowing us to test key features earlier than planned.  - Communication in Discord has been more effective than expected for asynchronous check-ins.  - It took a long time to set up and complete VS Code updates and dependencies than expected. |
| **Lessons Learned** |
| - Early communication is essential. We should document all roles and expectations at the start of the sprint to avoid delays.  - Keeping the development environment updated and with latest tools is important to begin a project. |

# Sprint 2 Retrospective Summary Report

|  |
| --- |
| **Things That Went Well** |
|  |
| **Things That Could Have Gone Better** |
|  |
| **Things That Surprised Us** |
|  |
| **Lessons Learned** |
|  |

# References