­­­­­­­­­­

**Project Title: Bookstore Management System**

**Group’s Number: 2**

**Created by: Kyle Spicer**

**Date: 5/9/2025**

**­­­­­­­­**

Table of Contents

[1. Project Vision and Description 3](#_Toc71466583)

[2. Team Roles and Assignment 4](#_Toc71466584)

[3. Team Collaboration Methodology 5](#_Toc71466585)

[4. The Definition of “Done” 6](#_Toc71466586)

[5. Product Design 7](#_Toc71466587)

[6. Sprint 1 Retrospective Summary Report 10](#_Toc71466588)

[7. Sprint 2 Retrospective Summary Report 11](#_Toc71466589)

[8. References 12](#_Toc71466590)

# Project Vision and Description

**Description:**

Our development team has been contracted by the owner of a rare bookstore to design and implement an automated system for accounting, inventory management, ordering, and sales. This new system will modernize the store’s operations by replacing the current paper-based workflow with a streamlined, computer-based solution. The inventory process—from initial cataloging of books to final purchase—will become significantly faster and more efficient.  
  
The system’s core functionalities will include:

|  |  |
| --- | --- |
| **Functionality** | **Description** |
| Inventory Management | Monitor, update, and manage book inventory in real-time |
| Sales Records | Automatically record/store transaction data for reporting and accounting purposes |
| Supplier Orders | Enable the bookstore to place and manage orders with book suppliers. |
| Customer Orders | Allow customers to browse available inventory and place orders. |

**Project Vision:**

The system is designed to enhance operational efficiency, minimize human error, and provide valuable insights through performance tracking, data collection, and reporting. Scalability is a core focus, allowing the platform to evolve with future feature additions and increased demand.

The initial release will be a cross-platform command-line utility developed in Python 3.10. It will use TCP sockets to support multiple simultaneous client connections, enabling interactions such as order placements from manufacturers and customers. Threading will be used to manage concurrent connections effectively. For data persistence, the system will initially rely on JSON files to track inventory and sales records. The first Release Candidate (RC1.0.0) will operate entirely through a command-line interface, with plans to introduce a graphical user interface in future versions.

# Team Roles

|  |  |  |
| --- | --- | --- |
| **Team Member Number** | **Role** | **Responsibilities** |
| Team Member 1  **Kyle Spicer** | Scrum Master | * Facilitate Agile practices and coach both the Product Owner and development team on Scrum principles. * Shield development team from external interruptions and distractions. * Identify and remove issues that hinder team progress * Foster collaborative environment that supports rapid and informed decision-making. * Conduct scrum practices: daily standups, sprint planning, reviews, and retrospectives. |
| Team Member 2  **Kyle Spicer** | Product Owner | * Define/prioritize product features based on stakeholder input and business value. * Serve as the primary liaison between stakeholders and the development team. * Manage/refine product backlog, ensuring clarity and alignment with project goals. * Monitor product performance and user feedback to inform future development. |
| Team Member 3  **Shaun White** | Developer 1 | * Collaborate with Product Owner, Scrum Master, and stake holders during sprint planning sessions * Translate backlog items into functional software features for the current sprint cycle. * Contribute to code development, documentation, and version control. * Participate in code reviews and quality assurance efforts. |
| Team Member 4  **Michael Oneal** | Developer 2 | * Lead creation of technical documentation, including UML diagrams, flowcharts, and architecture overviews. * Design and implement test cases aligned with product requirements and sprint goals. * Assist in software development and integration tasks. * Conduct continuous testing and debugging throughout the sprint. * Implement GUI for project |

# Collaboration Methodology

Effective and consistent communication will be critical to the success of our development process. Our team will utilize a tiered approach to collaboration tools to ensure accessibility, flexibility, and minimal disruption in the event of technical or scheduling issues.

**Communication Platform (Primary): Discord.** Discord will server as the central hub for daily/weekly communication. It provides persistent text and void channels for real-time collaboration, and the ability to organize meetings and share computers screens.

**Communication Platform (Secondary / Presentations): Google Meet or Zoom.** Google Meet or Zoom will be used for scheduled video meeting such as sprint planning, sprint reviews, and stakeholder presentations.

**Source Code Sharing: GitHub:** To manage, share, and maintain the project’s source code, the team will use GitHub as the primary version control and collaboration platform. GitHub allows robust support for Git-based workflows, enabling efficient tracking of changes, code reviews, and collaborative development.

# The Definition of “Done”

For our team, the “Done” means a task, feature, or deliverable is complete, reviewed, tested, and ready to be used or demonstrated without further modifications. We agreed that each sprint (week) and feature must meet specific criteria before it’s considered finished.

**Sprint-Level Criteria:**

* Team communication and collaboration efforts were consistent throughout the sprint. (Stand-ups, planning, retrospectives held.)
* The chose project idea was discussed, agreed upon, and approved.
* Initial planning and setup tasks were completed, including:
  + Writing project description and identifying scope.
  + Clearly defining team roles and responsibilities
  + Outlining collaboration tools and development practices
  + Submitting the finalized Week 1 project document.

**Task:**

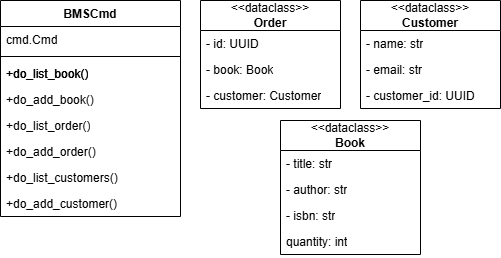
* Task has been fully implemented or addressed according to the backlog.
* Code is pushed to GitHub repository and follows agreed-upon coding standards
* Task has been tested appropriately
* Peer review or team confirmation has been completed.
* Related documentation, diagrams, or notes have been updated.
* Task is marked as complete in the team’s tracking system (GitHub issues or shared checklist)

**Features:**

* Inventory Management:
  + A functioning Python class that can load, clean, and display inventory data from any .csv file.
  + Inventory data must be organized and readable within the program.
* Sales Records:
  + Python functions to record, retrieve, and generate reports based on sales data.
  + All sales data must be saved to and read from the .csv file.
* Manufacturer Orders and Customer Orders:
  + Use a unified Python class named **Orders** capable of handling both types of orders using a shared database file.
  + Functions should include retrieving, deleting, and managing order for both customer and supplier roles.

# Product Design

**Bookstore Management System Class Diagrams (drawio)**

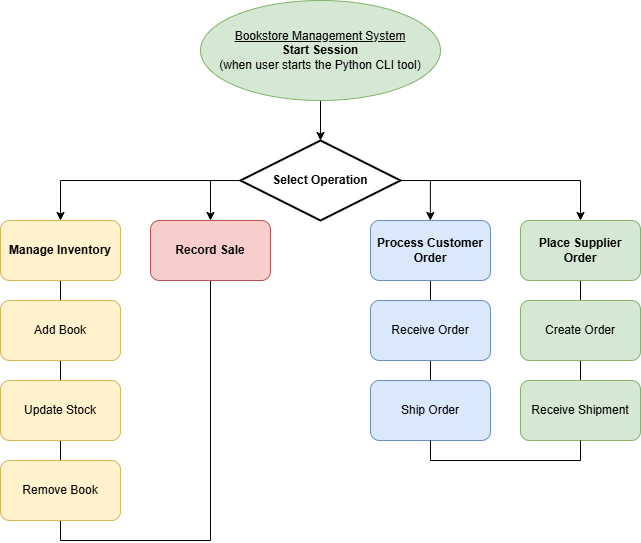
****

**Classmate’s contribution:**

A diagram of a customer

AI-generated content may be incorrect.

**Bookstore Management System Flow Chart (drawio)**

****

**Security and Non-Functional Design Notes**

|  |  |
| --- | --- |
| **Topic** | **Design Consideration** |
| Authentication | (Future Sprints) Will add user login functionality for staff and customers to secure access to system features. |
| Concurrency | Server uses Python’s threading module to manage multiple client socket connections simultaneously, enabling real-time multi-user access. |
| Data Integrity | Future implementation of locks/mutexes will protect file I/O from race conditions. Utilizing proper locking mechanisms to minimize data races between competing threads. |
| Persistence | JSON files are save immediately after every write operation to ensure consistency and recoverability. |
| Platform Support | CLI tool is developed in Python 3.10 and will be cross compatible with Windows/Unix/macOS, etc. |
| Scalability | Modular architecture supports future expansion. |

# Sprint 1 Retrospective Summary Report

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

# Sprint 2 Retrospective Summary Report

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

# References

# Nutcache. (2021, June 9). Agile, Scrum and sprint: What’s the difference? Nutcache Blog. <https://www.nutcache.com/blog/agile-scrum-sprint/>