**Document 06 – Sprint 1 Report**

CS 4321 – Fall 2022

Contents

[1 Document Ownership 1](#_Toc114573489)

[2 Directions 1](#_Toc114573490)

[3 Class Diagram 1](#_Toc114573491)

[4 Retrospective 2](#_Toc114573492)

[5 System Tests 2](#_Toc114573493)

# Document Ownership

This document is contained in your GitHub repository in a folder named *docs*. At the end of the Sprint, Sections 3 and 4 will contain information you supply. Section 2 contains only directions on what to do on the *user\_stories….xlsx* spreadsheet. Section 5 only contains information about how you will write system tests, which will be written in the spreadsheet.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Team | 5 | | Team Member Names |  |
|  | |  |  | | --- | --- | | 1. | Nakota Szymoniak | | 2. | Chris Wood | | |  |  | | --- | --- | | 3. | Lee Owen | | 4. | Mun Yeul Lee | |

# Directions

On the *05\_user\_stories\_distribute.xlsx* spreadsheet:

1. Rename this spreadsheet: *user\_stories\_team\_X.xlsx*, where X is your team number.
2. Store it in your GitHub repository in a folder named *docs.*

On the *User Stories* tab:

1. Do not add or remove, or renumber any user stories.
2. At the end of the sprint it should be sorted on Priority.
3. For the *Acceptance Criteria* column (column G) write the acceptance criteria for each user story before you code it. You don’t need acceptance criteria for stories you have not completed; however, it is OK to write them.
4. For the *Code* column (column I), type “C” if the code is complete, “~” if the code is in progress. Leave blank if you have not started the code.
5. For the *System Tests* column (column J), type “C” if the system test(s) have been written (in the spreadsheet). Otherwise, leave blank. The System Tests will be written in separate tabs, details are in Section 3 below.
6. For the *Status* column (column K), type “C” if the system tests have been run and are all passing, type “N” if at least one of the system tests is not passing, leave blank if the system tests have not been conducted.
7. For the *Comments* column (column L), write any comments that are needed about the status of the code or tests.

# Class Diagram

|  |
| --- |
| **Deliverable**  At the conclusion of this sprint, you should make a class diagram using StarUML. You can make multiple diagrams at different levels of granularity, or just break it up. The diagram should be included in this document along with brief remarks to explain your design. The diagram(s) must be readable. Also, include the diagrams saved as image files (jpg or png) in your *docs* folder on GitHub. |

Diagram, schematic

Description automatically generated

The image should be in 4K so please zoom in.

The design has two main paths from store to product. A store has one inventory which has many products A store also has many customers which has many orders which has many products.

The three classes (store, inventory, and order) that interact directly with the GUI have respective controller classes that the GUI will use instead of interacting directly.

Classes with an ID (product, customer, store, and order) have an IDValidator.

Classes that use a map have MapPersistence to build them and later save them.

# Retrospective

1. Towards the end of this sprint, read this short page about what a software retrospective is and why it is important:

<https://searchsoftwarequality.techtarget.com/definition/Agile-retrospective>

1. Meet as a group and discuss the following questions and provide a group written response below. Write as much as is appropriate.
2. What worked well for us?

|  |
| --- |
| **Answer**  When we had meetings, they were very productive for analyzing different requirements and reaching the conclusions we needed to figure out our designs. Our UML and mock GUI were very useful references for both coding and filling out the user requirements spreadsheet. |

1. What did not work well for us?

|  |
| --- |
| **Answer**  Everyone on the team struggled with independent time management, and overall scheduling conflicts that made frequent meetings difficult. Communicating through text was very inefficient compared to the real meetings that we had.  These issues also lead to more problems like the entirety of the project getting pushed back as no one worked at the beginning, and caused more disparity between members understanding as one person would have to do large chunks at a time to then catch up. |

1. What actions can we take to improve our process going forward?

|  |
| --- |
| **Answer**  Holding each other accountable to make sure everyone works frequently by creating premade short-term deadlines as every one of us struggles to work without a crunch time.  Designating consistent days and times for meetings where we bring up issues instead of having a meeting over an issue. This will ensure they are always held frequently and will help with discussing more minor problems.  More frequent paired work will help us with accountability, introducing more perspectives, and ensure that more than just one member understands a specific implementation.  Doing more tests and updating the documentation as we go instead of at the end. |

# System Tests

In the *user\_stories\_team\_X.xlsx* spreadsheet create a tab for each user story. The tabs should be named: US-1, US-2, *etc.* where the number represents the priority of the user story. On each tab, specify the system test(s) in the format below. An example is provided on the US-1 tab (which should be removed before turning in). There should be at least on system test for each user story.

|  |  |
| --- | --- |
| **US Title** | [User story title] |
|  |  |
| **Test Num** | 1 |
| **Description** | [Brief description] |
| **Directions** | [List steps to conduct test so that it is reproducible] |
| **Expected Output** |  |
| **Comments** |  |
|  |  |
| **Test Num** | 2 |
| **Description** |  |
| **Directions** |  |
| **Expected Output** |  |
| **Comments** |  |