**Sprint Report**

Contents

[1 Document Ownership 1](#_Toc183351081)

[2 Class Diagram 1](#_Toc183351082)

[3 Data Persistence 1](#_Toc183351083)

[4 Video Demo 2](#_Toc183351084)

# Document Ownership

This document is contained in your GitHub repository in a folder named *docs*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Team | 3 | | Team Member Names |  |
|  | |  |  | | --- | --- | | 1. | Braeden Barnett | | 2. | Tyler Young | | |  |  | | --- | --- | | 3. | Zyen Atkins | | 4. | Josh Reed | | 5. | Easton Conkin | |

Video Link(s) (See [Section 4](#_Video_Demo)): <https://www.youtube.com/watch?v=qQ_NS3PxV7c>

|  |
| --- |
|  |

# Class Diagram

|  |
| --- |
| **Deliverable**  At the conclusion of this sprint:   * Write a narrative explaining your overall design including how you implemented MVC. This should be accompanied by at least one accurately drawn class diagram. You can make multiple diagrams at different levels of granularity. * **The diagram(s) should be included in this document.** The diagram(s) must be readable. * The diagram(s) should also be saved as image files (jpg or png only) in your *docs* folder on GitHub. |

<<<Narrative goes here, followed by diagrams. Diagrams should be full page, each on a separate page. Turn the page landscape if appropriate. >>>

Our team’s overall design included three different folders which adhered to the MVC architecture. The first folder was named controllers which held the controller classes that oversaw handling the user selecting a value from a drop-down list, clicking a button, etc.

The next folder called models held the classes that were responsible for calculating the business logic of the program so whenever a user would want to see their seller report for example then a class in the models’ folder would be called to calculate the values to get the total sum.

Our third folder named views oversaw the displaying of the UI portion of the program as well as updating it with new information whenever a user decided to do something.

A screenshot of a computer program

Description automatically generated

# Data Persistence

|  |
| --- |
| **Deliverable**  At the conclusion of this sprint, write a brief narrative explaining:   * How you went about data persistence, classes involved, *etc.* * The format of data file(s) used for data persistence, including a brief example of each file. |

<<<Example files and narrative go here>>>

I initially was going to use JSON to store data. However, I wanted to venture a bit out of my comfort zone, and instead opted to use .dat files. I had never experimented with this type of data before, and it was an overall seamless process. By using Java's DataInputStream, I was able to achieve a simple functionality where users could save their Buyer's Premium/Seller's Commission values in the SettingsConfig class. The controller would simply grab the int objects from the view, store them in a .dat file, and save it to the user's personalized .dat file. From this point forward, the user can click "Load" to have their GUI remember the previous values. It was a simple step in the right direction, but I knew it would soon turn into a large leap as I learned .dat files support "old version" integration, meaning users can revert their settings to a previous version

# Video Demo

|  |
| --- |
| **Deliverable**  When your project is complete, create a video that demo’s your User Stories and provide the link in [Section 1](#_Document_Ownership). Requirements:   * You can make a single video, or, if needed, 2 or 3. * The total length should be whatever is needed to accomplish the agenda below. * Preferably, post your video(s) on YouTube. * Agenda:  1. Explain your design. You should use class diagrams sized so that the relevant portions fill as much of the screen as possible. As part of this, explain how your design implements MVC. 2. Choose one User Story and step through the code as if it were being executed. You’ll start by showing the code where the appropriate event handler responds to the user. Next, to whatever it calls, etc, explaining as you go. 3. Explain how you handled data persistence showing code and data files. 4. Explain how you implemented the ability to start and stop time (User Stories 14 & 15) showing code. 5. For each completed User Story: 6. State the number of the User Story, and then state the User Story itself. 7. Demonstrate it with your software. If you have multiple system tests, you can demo all of them, or just the main (success) one. You can decide on the basis of time that you have. |