**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 | 1 | | Team Member Names |  |
|  | |  |  | | --- | --- | | 1. | Hudson Bowen | | 2. | Bazil Ashfaq | | |  |  | | --- | --- | | 3. | Melissa Torres | | 4. | Roosevelt James | | 5. |  | |

Video Link(s) (See [Section 4](#_Video_Demo)):

|  |
| --- |
|  |

# 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. |

The main classes in our overall design are the AuctionSystem.java, AuctionSystemView.java, AuctionSystemController.java, and AuctionStatePersistence.java. The AuctionSystem.java is our model, and it holds all of the information regarding any auctions, bid histories, categories, seller’s commission, and buyer’s premium. AuctionSystemController.java is our controller. This class interprets inputs from the user and notifies our model and our view when they need to change. Our view is the AuctionSystemView.java class. This class contains all the GUI elements needed to display the system’s information. Our AuctionStatePersistence.java class saves and loads the information stored in our model. We used a Push Model for our MVC architecture, so our view has many event handlers which collect relevant information and send them to the controller when a user interacts with the system. The controller then interprets the information and notifies the model and view on how they should change.

A screenshot of a computer screen

Description automatically generated

A diagram of a flowchart

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. |

For data persistence, all the information related to the categories, auctions, bid histories, seller’s commission, and buyer’s premium are saved to a text file whenever one of the values changes. When the controller (AuctionSystemController.java) updates the model (AuctionSystem.java), it runs the static saveState method in the AuctionStatePersistence class which saves all of the changes to the system into a text file titled “system\_state.txt”. When the system first starts and whenever it exits testing, the controller runs the static loadState method which reads the saved text file and restores the system to its last saved state.

The format of our system\_state.txt file is as follows:

CATEGORIES

(category)

(category)

ITEMS

(item name)|(item weight)|(item description)|(item category)|(item condition)|(item tag1)|(item tag2)|(item tag3)|(item start date)|(item end date)|(item Buy It Now price)|(current winning bid for item)|(whether or not the item is active)

END\_BID\_HISTORY

(item name)|(item weight)|(item description)|(item category)|(item condition)|(item tag1)|(item tag2)|(item tag3)|(item start date)|(item end date)|(item Buy It Now price)|(current winning bid for item)|(whether or not the item is active)

END\_BID\_HISTORY

BUYERS\_PREMIUM

(buyer’s premium)

SELLER\_COMMISSION

(seller’s commission)

Here is a small example:

CATEGORIES

Electronics

ITEMS

Phone|3|Small phone|Electronics|New|tag 1|||2024-12-11T14:46:52.716991400|2024-12-12T10:00|300.000000|120.000000|true

END\_BID\_HISTORY

BUYERS\_PREMIUM

5.0

SELLER\_COMMISSION

7.2

# 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. |

Video 1:  <https://youtu.be/EcTnl27lsDU>

Video 2: <https://youtu.be/zNbjgQDUOis>