**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 | 5 | | Team Member Names |  |
|  | |  |  | | --- | --- | | 1. | Kathleen Mock | | 2. | Christian McCambridge | | |  |  | | --- | --- | | 3. | Miguel Uriostigue | | 4. | William Adair | | 5. | Carl Hamer | |

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

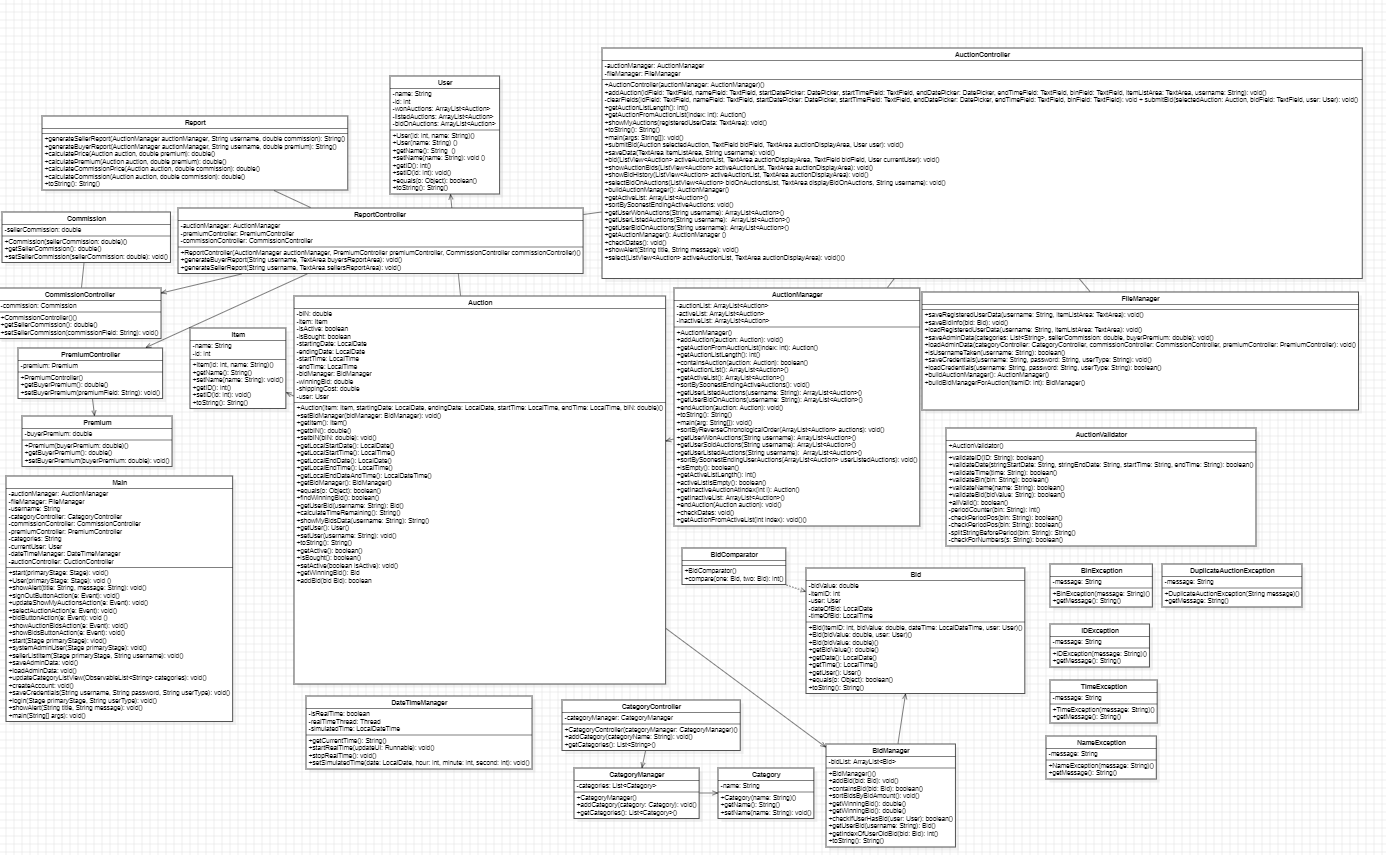
|  |
| --- |
| <https://valdostaedu-my.sharepoint.com/personal/wcadair_valdosta_edu/_layouts/15/AccessDenied.aspx?Source=https%3A%2F%2Fvaldostaedu-my%2Esharepoint%2Ecom%2Fpersonal%2Fwcadair_valdosta_edu%2FDocuments%2FSoftware%20Engineering%20Project%2Emp4%3Fct%3D1733936847622%26or%3DOWA-NT-Mail%26cid%3D3a7f0d8f-bc76-c4f3-7ee3-b3fc838cf57a%26nav%3DeyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&correlation=d78d6ca1-a0b2-7000-3b0f-1b500a652d83&Type=item&name=4cb62aca-0cea-450a-8df3-68c5740a4fd1&listItemId=14&listItemUniqueId=3f436c76-31cf-40fb-8f6f-1ae2a0daf8c4> |

# 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. >>>

Narrative: For several classes, we used controller classes and manager classes. When the user changes information in the system such as pressing a button to submit information, the controller class modifies both the model classes and GUI, rather than the model class modifying the GUI directly.



# 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>>>

Narrative: When the application is first run, the user sees a login page. After the user logs into the system, the application reads in data by using the auction controller and file manager class. File manager reads through a text file initially created when a valid auction is saved for the first time. Subsequent saved valid auctions are appended to this text file. A separate text file is created for the first time when a bid is submitted for an auction. The bids saved on this text file are linked to the auction as it saves the item ID of the auction that was bid on. File manager initially loops through the list with the entries for auctions. When all the information is pulled such as auction name, ID, username, etc., an Auction is created, then the file manager loops through the bid list to find all the bids matching the ID for the newly created Auction, adding it to the auction’s BidManager. This auction is then added to an AuctionManager. AuctionController is the class that calls for file manager to read in data so that the system can display and interact with it in Main.

When application is run, users can log in or create an account to log in. When an account is created, their username, password, and user type that they selected is saved by file manager to a text file. When logging into accounts, file manager reads in the data for the user information and the system checks to see if what the user has input as values match the values listed in the data. If there is a match, user can log into their account.

System admin data for buyers premium and sellers commission is also saved in a text file by file manager when system admin presses save. When system admin logs back in, file manager reads the system admin data which is displayed on the page. When changes are made to the buyers premium and sellers commission, the data in the file is edited by file manager.

Example of admin\_data.txt:

Categories  
- shoes  
- pants  
- shirts  
  
Seller Commission: 30.0  
Buyer Premium: 29.0

Example of credentials\_txt:

system:admin:System Admin  
alice32:test:Registered User  
gr90:test:Registered User  
amy:test:Registered User  
josh:test:User

Example of registered\_user\_data.txt:

alice32: ID: 1, Name: basketball, Start date: 2024-12-03, start time: 21:21:21, End date: 2024-12-12, end time: 10:00:30, BIN: $10, active: true  
alice32: ID: 2, Name: baseball, Start date: 2024-12-01, start time: 22:22:22, End date: 2024-12-11, end time: 10:00:10, BIN: $0, active: true  
gr90: ID: 3, Name: book, Start date: 2024-12-02, start time: 21:21:21, End date: 2024-12-20, end time: 21:21:21, BIN: $30, active: true

Example of bid\_info.txt

Item ID: 1, bid amount: $5.00, date: 2024-12-11, time: 10:30:41, user: gr90  
Item ID: 1, bid amount: $6.00, date: 2024-12-11, time: 10:31:10, user: amy  
Item ID: 3, bid amount: $15.00, date: 2024-12-11, time: 10:31:20, user: amy

# 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. |