Final Project - Asteria Trading (Auction)

Drew Conyers dtc888
Spring 2021
https://github.com/EE422C/spring-21-final-project-drewtconyers

Overview

- Asteria Trading is a cloud based art auction house in which users can place artwork up for auction or for purchase for other users to purchase and trade art.
- Developed using JavaFX and Microsoft Azure Database. Server is hosted using Azure.

Asteria Trading (USER INFORMATION)

- In this application it is important to click the refresh button to get updated information from the server.
- When loading in you will be prompted to login with a previously created account, to register an account, or proceed as a guest user.
    - When registering enter your full name and create a username which will be used on login.
    - Passwords will be hashed and saved for your security.
- The main screen will display artwork that has recently been put on auction or purchase.
    - The main screen contains all recent auction closures and which user won that auction.
    - There is also a full history of all bids on all pieces up for auction.
- To create a new piece of artwork to auction or sell enter the upload screen.
    - You will be prompted to enter the title of your piece, a photo of your piece, an optional description of the piece, the starting bid price, how long you would like the auction to last, and an optional purchase price for the piece.
    - When your auction expires if there was another bidder in the auction the highest bid price or the purchase price will be added to your funds and removed from the new owner.
- To purchase a piece you will need to add funds to your account under the Asteria Wallet tab in which you will enter the amount of USD you wish to add to your account.
- To actually purchase a piece go to the main screen to select a piece to bid on in new art.
    - Clicking a piece in new art will take you to the auction page in which you will be able to view all information about the selected piece. You will also be able to place a bid on the item up for auction or purchase the piece if that option is available.
    - Make sure to refresh the page to get updates about the auction and see who is the highest bidder.

    - If you are the highest bidder at when the auction expires the item will be placed in your gallery page.
- Selecting the gallery page will show all pieces you currently own, including pieces you have put up for auction for which will be transferred to new owners when your auction expires.
- Selecting the explore page will show all pieces on the Asteria database for you to view.

Asteria Trading (PROGRAMMER INFORMATION)

- Asteria Trading's server startup is very simple, just run the server executable file and the server will be run on your local machine.
    - Or you can host your server on a cloud provided such as Microsoft Azure. Create a web application and place the server jar file in the service.
    - Further instructions here.
- The Asteria database is being run on Microsoft Azure so there is no setup required for the database. The Azure database is the crucial piece of this program and all information is sent to the server to be processed and inserted into the database and then accessed by the client when requested.
- For the actual server program, the server will accept clients from sockets with a client handler for each client.
    - Each time a client chooses to look at a piece of new artwork or a new client login in there is a new connection to the server. This implementation allows the server to constantly refresh the auction information and ensure the database is always updated.
    - All information from the client is parsed from JSON serialization based on the clients request.
        - Requests include:
            - New upload of artwork into the database.
            - New bid on a piece on auction.
                - All new bids are handled in a thread safe way between clients with a queue lock to ensure only one bid is processed at a time.
            - Purchase of artwork.
- Again this application heavily relies on the database to keep all information from the clients.
- For the client side of the program, there is a new connection when a user logins and for each view of an artwork on auction. This ensures that the server is constantly updating the database.
    - You can change the PORT AND HOST in the Params class.
- There is a preloader screen to allow the program to load and execute all required SQL requests.

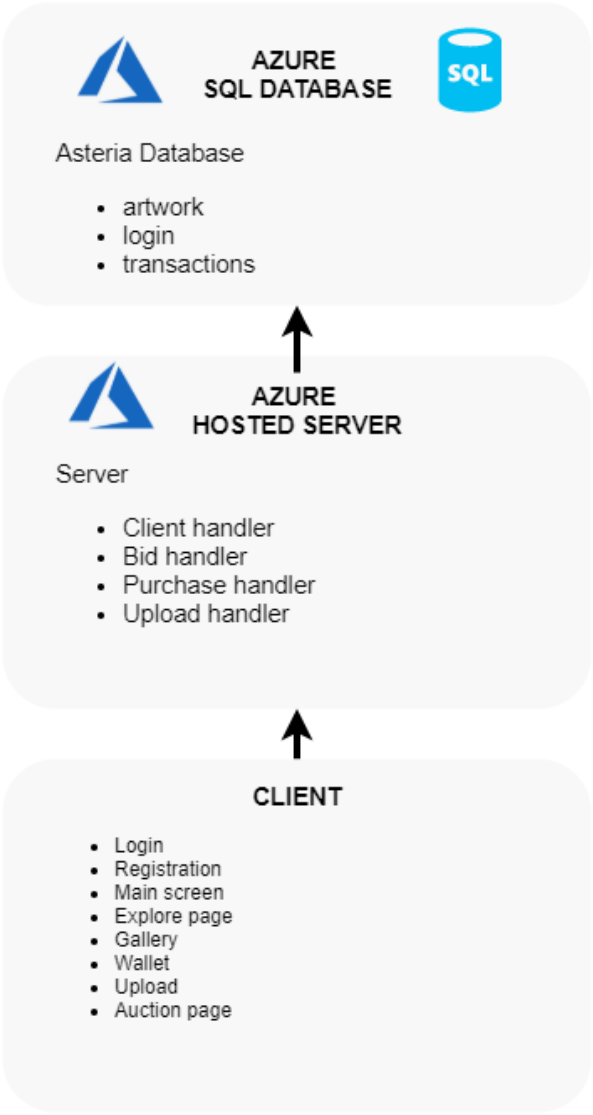
    - The pages that are loaded in are the login, main screen, explore page, and upload screen. These pages are uniform across all clients.
    - After the user logins in the gallery page and wallet page are loaded with the clients information.
- There is six pages once logged in:
    - Main screen:
        - The main screen holds all pieces that are on auction or can be purchased. Database requests are made to receive the information about the artwork, recently sold pieces, and all transactions made on the server. Click on a piece in new artwork to load the auction page.
        - When the client refreshes the page new SQL statements are executed to get the most recent information from the server.
    - Upload:
        - All required information is sent to the server and processed on the server.
        - Images converted to 64Bit string to save space on server.
    - Auction page:
        - The auction page sends bid and purchase information to the server to be processed.
    - Explore page:
        - When the application load in the explore page will show all pieces of art on the database by executing SQL statements to the database.
    - Gallery:
        - When the user logins in the gallery page will show all pieces of art on the owned by the user by executing SQL statements to the database.
    - Wallet:
        - Funds are updated directly into the database for the user.

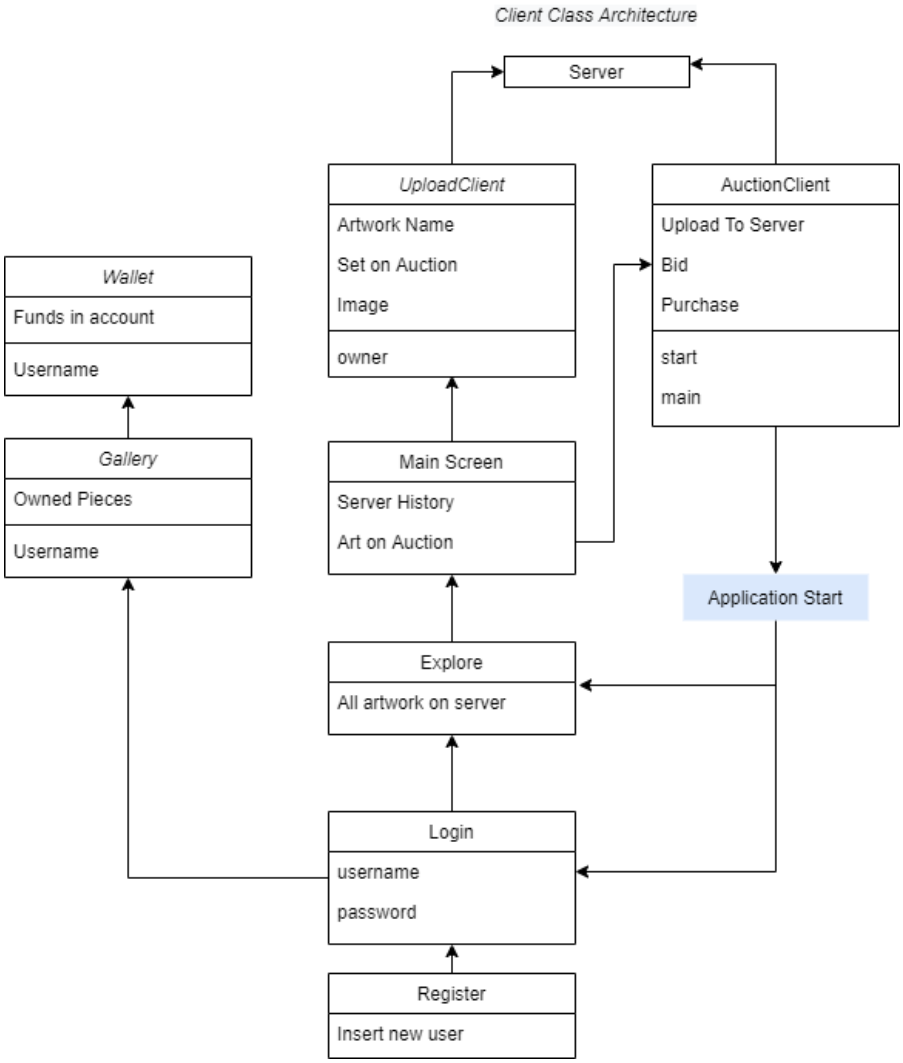
Features

- Minimum starting price must be greater than $0 for auction
- Duration for the auction of each item is set separately
- Purchase option for user
- Bid history of all items (who made the bid, if the item has been sold and to who)
- Descriptions of items
- Non volatile history of auctions and customer activity
- Clean GUI
- Countdown clocks for auctions
- Azure cloud server to host auction
- Password hashing
- Azure SQL Database
- Ability to view owned pieces
- Ability to see all pieces on server
- User wallet

- Converting images to 64Bit Strings to save space on server
- User credit transfer

Software Diagram





Citations

JavaFX CSS Reference Guide
https://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html

JavaFX Documentation
https://docs.oracle.com/javase/8/javafx/api/toc.htm

Difference between Dates
https://stackoverflow.com/questions/1970239/in-java-how-do-i-get-the-difference-in-seconds-between-2-dates

Updating Labels on a Thread
https://stackoverflow.com/questions/47150884/label-not-being-automatically-updated-with-thread-in-javafx

Fitting Images to Pane
https://stackoverflow.com/questions/27894945/how-do-i-resize-an-imageview-image-in-javafx/27894962

SQL to Get Last Entry in Table
https://docs.microsoft.com/en-us/sql/t-sql/functions/last-value-transact-sql?view=sql-server-ver15

Converting Image to Base64
https://stackoverflow.com/questions/49011802/converting-javafx-image-to-from-base64

Base64 String to Image
https://www.baeldung.com/java-base64-image-string