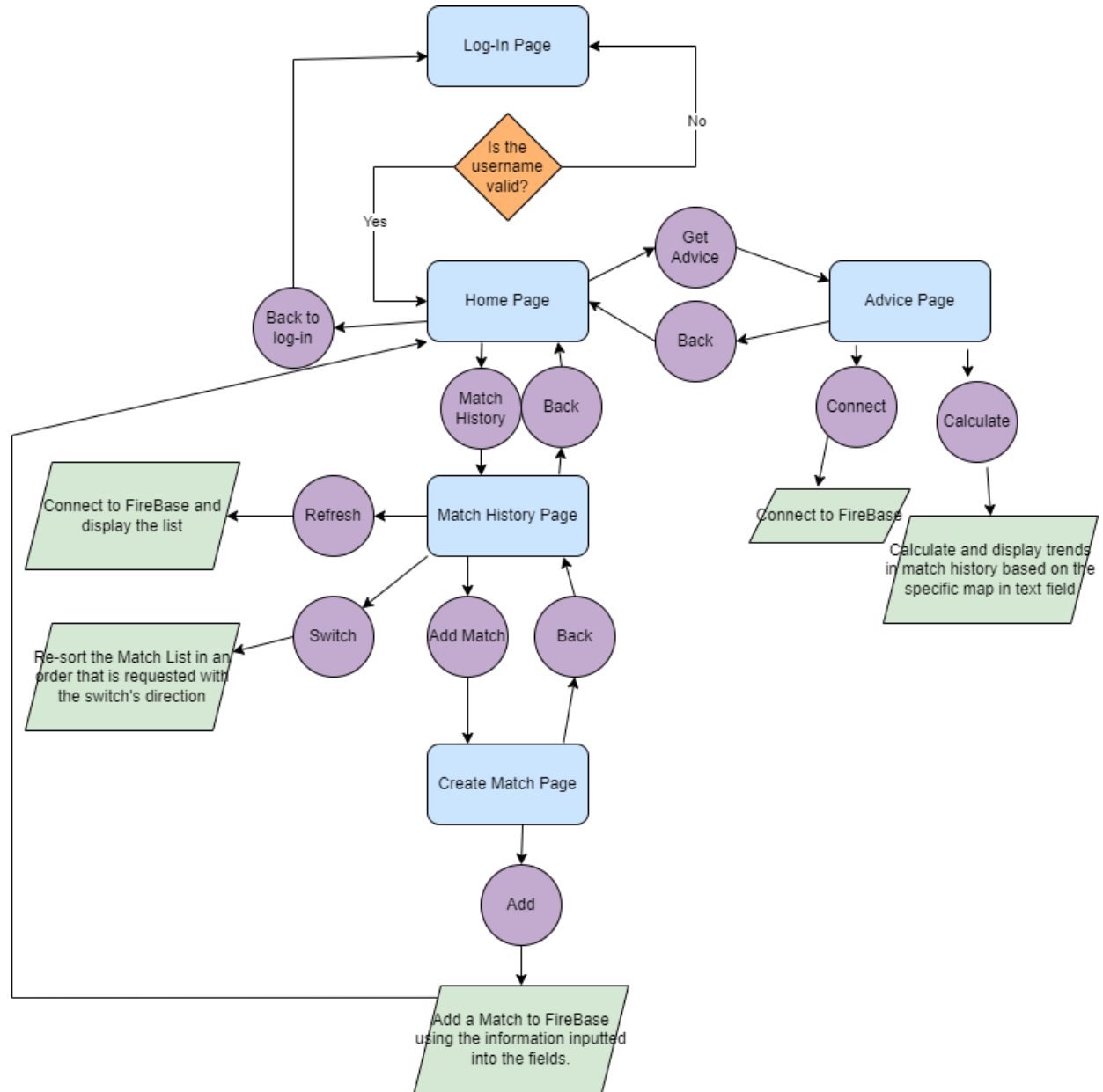


## Criterion B

### Flowchart(s):

*(See Appendix for Previous Versions)*



### Algorithms:

One of the more complicated algorithms the program uses is a way of calculating data trends for a specific game map. I completed this by calculating the win loss ratio of each agent (game character) getting played on the map and reporting the agent with the best ratio and the agent with the worst ratio. The steps to this algorithm are as follows:

Step	Details
1	Create a HashMap storing the agents as the key and an ArrayList<Integer> of its win and loss count. This will store the agents and their win and loss records that are used on the map we are testing for.
2	Using a for-loop, we go through every match in the linked list of all the matches.
3	For every match, we check to see if it is using the specific game map we are testing for.
3.1	If it is the game map we are testing for, we check to see if the HashMap already has data stored for the agent in this match.
3.2	If there is no data yet for this agent, we create an object in the HashMap with one win or one loss.  If there is data already, we add either a win or a loss to the existing data.
4	After fully populating the HashMap with agents and their win-loss records, we have to find the best ratio and the worst ratio:  Create variables holding the best and worst agent names and ratios.
5	Set the best and worst agent variables to "jett," while setting their win-loss ratios to -999 and 999, showing extreme examples so they are easily replaced. These are just placeholders and will be immediately replaced by the first agent who is tested.

6	<p>Using an entrySet,, cycle through the HashMap and check if the win-loss ratio is better than the current best or worse than the current worst, and replace accordingly.</p> <p>[Error handling] = if a win-loss ratio includes zero losses, don't calculate the ratio because dividing by zero will crash the app. Just pass the string as "undefeated."</p>
7	<p>Display the best and worst agents on this map along with their win-loss ratios to the app screen.</p>

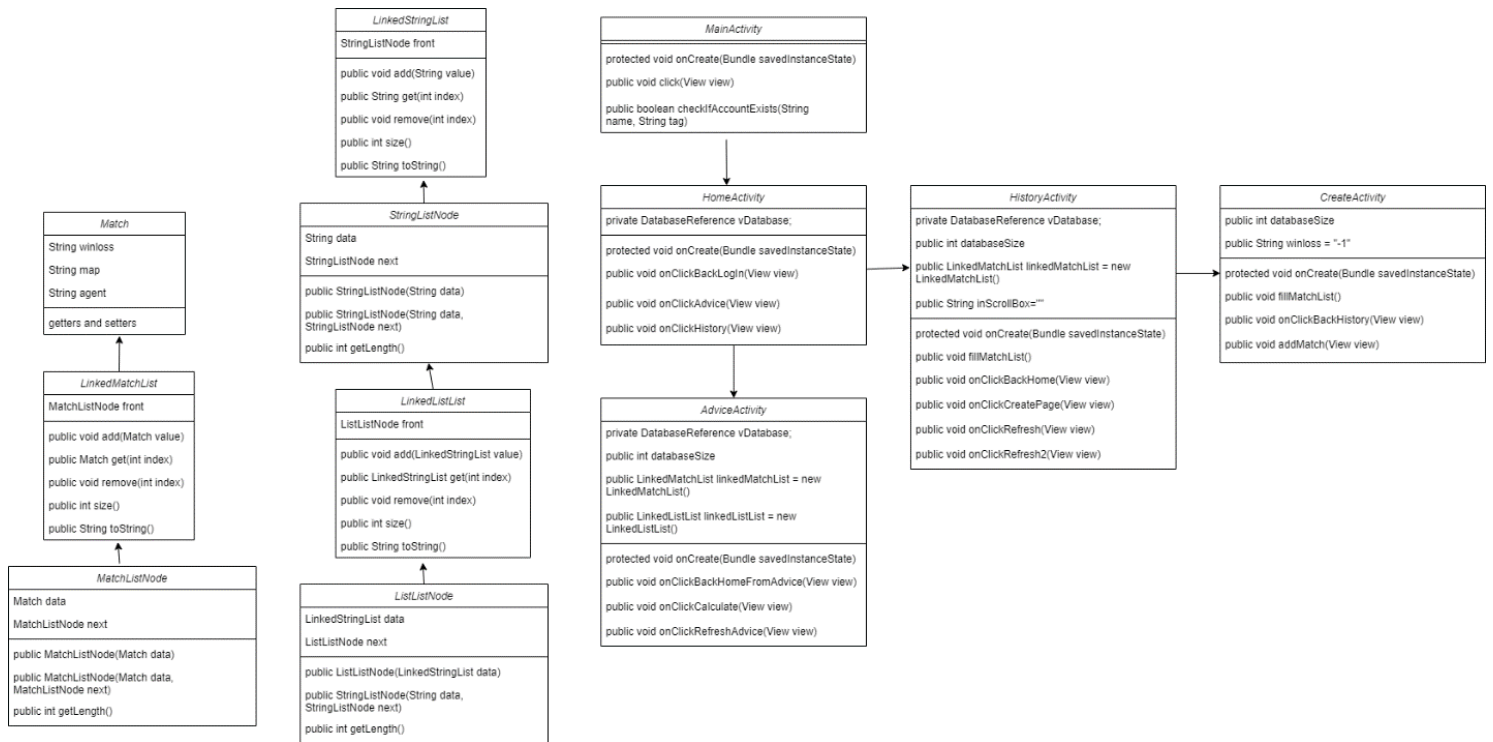
### **Data Structures:**

I chose to store my data in Google FireBase. I chose FireBase because it allows protection of data, and it has a simple data design to store data using a hierarchical structure with children. This allows me to store my client's match history and the variables attached to each of them.

I implemented Linked Lists for my object Match to store the Match History. This was to store my Match History all in one place and keep it accessible. For sorting the Match History, I used stacks to utilize their "Last-In-First-Out" quality to reverse the order of the Match History.

I used HashMaps to store agents and their corresponding win-loss records. This helped me easily access and compare agent's win-loss ratios because of the ease of using keys. I did a similar task in a different part of my app where I used 2D Linked Lists to assign game map names with full descriptions. I used this 2D Linked List to search for a game map name, and display the corresponding description when it was found.

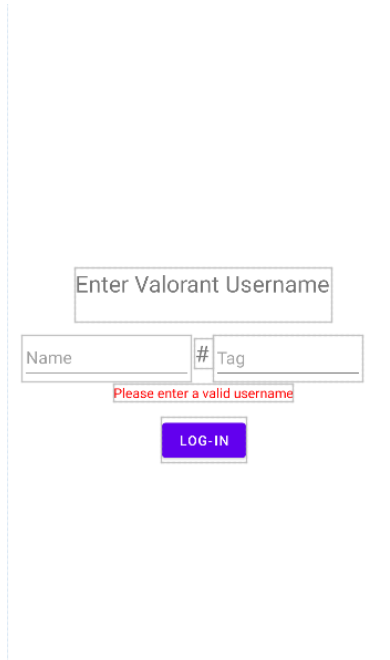
## UML Diagram (How Classes Interact):



## UI Flows:

*(See Appendix for Previous Versions of UI)*

### Log-In Page:



The Log-In Page UI mockup is centered within a light gray rectangular frame. It features a large text input field at the top with the placeholder text "Enter Valorant Username". Below this is a two-part input field: the left part is labeled "Name" and the right part is labeled "Tag" with a "#" symbol between them. A red error message, "Please enter a valid username", is positioned directly below the "Name" input field. At the bottom of the form is a purple button with the text "LOG-IN" in white capital letters.

### Home Page:

BACK TO  
LOG-IN

ValorantDB

GET ADVICE

MATCH HISTORY

## Match History Page:

BACK

Match History

ADD MATCH

Matches will show up here!

REFRESH FROM  
DATABASE

REFRESH LIST

Oldest to Newest



Newest to Oldest

## Create Match Page:

BACK

Add Match

Loss



Win

Map

Agent

ADD

## Advice Page:

CONNECT

BACK

Map

CALCULATE

Map Description:

### Test Plan:

Item to Test	Details	Success Criterion
A working log-in system	<p>The first page to the program has fields requiring the username for the user. If the fields have a valid username when the “log-in” button is pressed, the program will allow the user into the home page.</p> <p><b>This is successful if the username “poa#fps” is entered and the program changes to the home page of the app.</b></p> <p>-[Error handling] If an invalid username is entered, red text will appear requesting a valid</p>	6

	username and access to the app will be denied.	
The program has a user-friendly UI that is easy to follow.	The program will have one-to-two-word buttons that clearly explain what they do. These buttons will navigate between pages seamlessly. <b>This is successful if throughout the test, I navigate the program with ease and point out the working buttons.</b>	5
Displays match history	The match history of my client is the data that is being used. On the Match History page, this is supposed to be grabbed from Google FireBase, converted into a Linked List, then displayed on the app. <b>This is successful if the match history is displayed into the scroll list after the refresh button is pressed.</b>	2
Allows for sorting the match history list.	There is a page showing the match history and there is a switch on the page that controls how the list is sorted. <b>This is successful if I flip the switch twice to see if the match history is sorted accordingly.</b>	3
Allows for matches to be added to the database.	There is a page allowing the user to add matches to the database. <b>This is successful if I add a match and see it update in the match history list.</b>	4
Gives specific advice for specific maps.	The advice page gives advice when a specific map is entered. <b>This is successful if I enter a map and then accurate trends in the gameplay are reported on the screen.</b>	1



Displays game map descriptions.	The advice page gives descriptions of maps as it is entered. <b>This is successful if I enter a map and the corresponding description displays on the screen.</b>	8
Saves and loads information on Google FireBase for the app	The app uses Google FireBase to store my client's match history. <b>This is successful if the program saves a match that is created after restarting the app.</b>	7