Preface

After reading through your team's use case descriptions in relation to the project requirements file provided by the instructor, I must say that I am genuinely amazed by the level of details and planning your team has put into this project. Without a doubt, I can say that your team goes above and beyond in some aspects of the project requirements. For example, the usage of OTP is certainly something that may not be "required" as per say but your use case diagrams file has it as a feature. The level of details behind each use case is also commendable as the interdependence of use cases is clearly and correctly shown in your use case diagram. With all this being said there are still two features that I propose as they can be interpreted to be part of the project requirements.

Feature Request: Delete Account From Database & System

Reasoning For Request: In the project document > under the requirements section >in the Multiplayer Interface section > there is a bullet point that reads "Users should be able to log in, create profiles, and manage their accounts". Generally, when one is able to "manage" their account, it also implies that they have the ability to delete their account. Since "delete account" or anything relating to this has not been found in the use case descriptions or diagram, I propose this as a feature.

High Level Description:

Use Case: User Account Deletion

Iteration: 2

Primary Actor: User

Goal in Context: Allow a user to delete an account from the database and the system by providing necessary credentials for verification.

Preconditions:

• The user device is powered on.

• The user is logged into their account.

• The user navigates to the profile management/account management page.

• The website is operating without any technical issues.

• The user has a valid email account

Triggers:

• The user selects the ‘Delete Account’(Formatted In Red Font) option from the profile management page.

Scenario:

1. The user selects the "Delete Account" option on the profile management page. .

2. The system prompts the user to confirm if they actually want to confirm with the confirm not being the default option.

3. The user selects confirm.

4. The system prompts the user to input their current password and their associated email account.

5. The system checks if the password and email match what is stored in the database from when the user registered the account.

6. If valid, the system sends an account deletion verification email with a one-time password.

7. Otherwise, users will have to re-enter their password and email until it matches what is stored in the database.

8. The user enters the one-time password into the given prompted textbook.

9. The system deletes all data related to the user from the database.

10. The user is redirected to the sign-in page and they receive an email saying their account has been deleted.

Postconditions:

• The user successfully deletes their account and related data from the database.

• The login credentials(username) are now available to use for new registrations.

• The user cannot login again with the deleted credentials.

Exception:

• The inputted password doesn’t match the database password.

• The inputted email doesn’t match what is in the database.

• The inputted one time password doesn’t match what the system sent out.

Priority: Medium (Dead Accounts and Banned Accounts should be removed from the system as to conserve resources)

When Available: Always accessible on the profile management page.

Frequency of Use: Once per user account.

Channel to Actors: Web browser, system UI, email access.

Secondary Actors: System Administrator.

Channel to Secondary Actors: Database Management/(Auth Manager)

Open Issues:

• Possibly recovering the account after it gets deleted from the database.

Expected Impacts:

The implementation of this design will mostly impact the authentication and profile management division as they will simply have to incorporate ways to remove data from the database as opposed to the current ways of simply adding data or replacing data(resetting passwords). In implementing this feature, the GUI division will also have to add this feature to their interface. Other than these two minimal changes there aren't any other major changes that may drastically impact the system.

Suggestions for Implementation:

- Add a method named deleteAccount() under the ProfileManager section of the class diagram of the authentication team.
    - This method can have other sub/helper methods like getEmail() and verifyEmail(), getPassword() and verifyPassword() getOTP() and verifyOTP() which all help with backend work of deleting the account.
- This use case description will likely connect to your “user dashboard” use case which will include “Delete Account”. There is also a likely chance that this method will also connect to an already existing use case of OTP verification. You can simply use the code for OTP verification to do one aspect of this use case.

Feature Request: Search For Any Player & Challenge Such Player

Reasoning For Request: In the project document > under the requirements section >in the Graphical User Interface(GUI) section > there is a bullet point that reads “View and challenge other players by searching their profile”. Although your team has something called the Use Case: Viewing Player Profile(in GUI use case descriptions file) and the Use Case: Matchmaking with a Friend(in Matchmaking use case descriptions file) which meet some of the requirements of the bullet point mentioned in the project document, it doesn’t meet the general requirement of 1)being able to search for **any user** and be able **challenge them directly from their profile**. As such, I propose this use case called Use Case: Search For Any Player & Challenge Such Player.

High Level Description:

Use Case: Search For Any Player & Challenge Such Player

Iteration: 2

Primary Actor: Any User

Goal in Context: Allow any user to search for another user through search and challenge them to a game.(doesn’t need for users to be “friends”),

Preconditions:

• The challenging user and challenged users both have their devices on.

• The challenging user and challenged user are both logged into their accounts.

• The challenging user can search for any player and challenge them.

• The other user is online and isn’t currently in a game.

Triggers:

• The challenging user searches for the username of the other user they want to challenge and their other player's profile shows up on the page. The challenging user then clicks "challenge" on the profile of the other user.

Scenario:

1. A challenging user logs on their account and searches the username of the player they want to challenge who is also online and is not playing any other games.

2. The challenged player receives a challenge request from the challenging player and has the options to accept or decline & block requests for 30 mins(this can be arbitrary)

3. If the challenged player accepts, then a game begins and follows the regular procedure. If the challenged player declines, then it places the challenging user on a 30 second timer before they can send the same user another challenge request. If the challenging player blocks requests from the challenging player then they can't be challenged again. Instead the challenging player simply gets a notification saying they can't challenge the other player.

Postconditions:

• The challenged user accepts the challenging users requests to a game and the game begins.

• The challenged user rejects the challenging users requests to a game and it stops the challenging user from sending any more requests for a period of time.

Exception:

• The challenged user isn't online or accepting challenges

• The challenging user is being restricted by the system

Priority: Medium

When Available: Always accessible on the profile management page

Frequency of Use: Common

Channel to Actors: Web browser, system UI,.

Secondary Actors: none

Channel to Secondary Actors: none

Open Issues:

•What if one user spams challenge requests to other players, should there be a block function or something that limits the amount of times a user can challenge another player during a period of time.?

Expected Impacts:

The implementation of this system will not be too difficult as it will essentially combine the code of the GUI team and the matchmaking team where instead of a user being only restricted to their friends, they can search and challenge anyone that is also online and is accepting requests. This is not intended to remove any sections from the code you may already have, instead it only removes the restrictions.

Suggestions for Implementation:

- For GUI team and matchmaking team to have a meeting to discuss the effects of implementing this feature to their already existing code. Since the GUI team has the features called Viewing Player Profile & User Searches for a friend and since the matchmaking team has the feature called Matchmaking with a friend - a lot of the code will presumably remain the same and the rest will be merged.
- Both the GUI Team & Matchmaking will likely have this in their use case diagram as they will incorporate different sections of this use case description/feature.