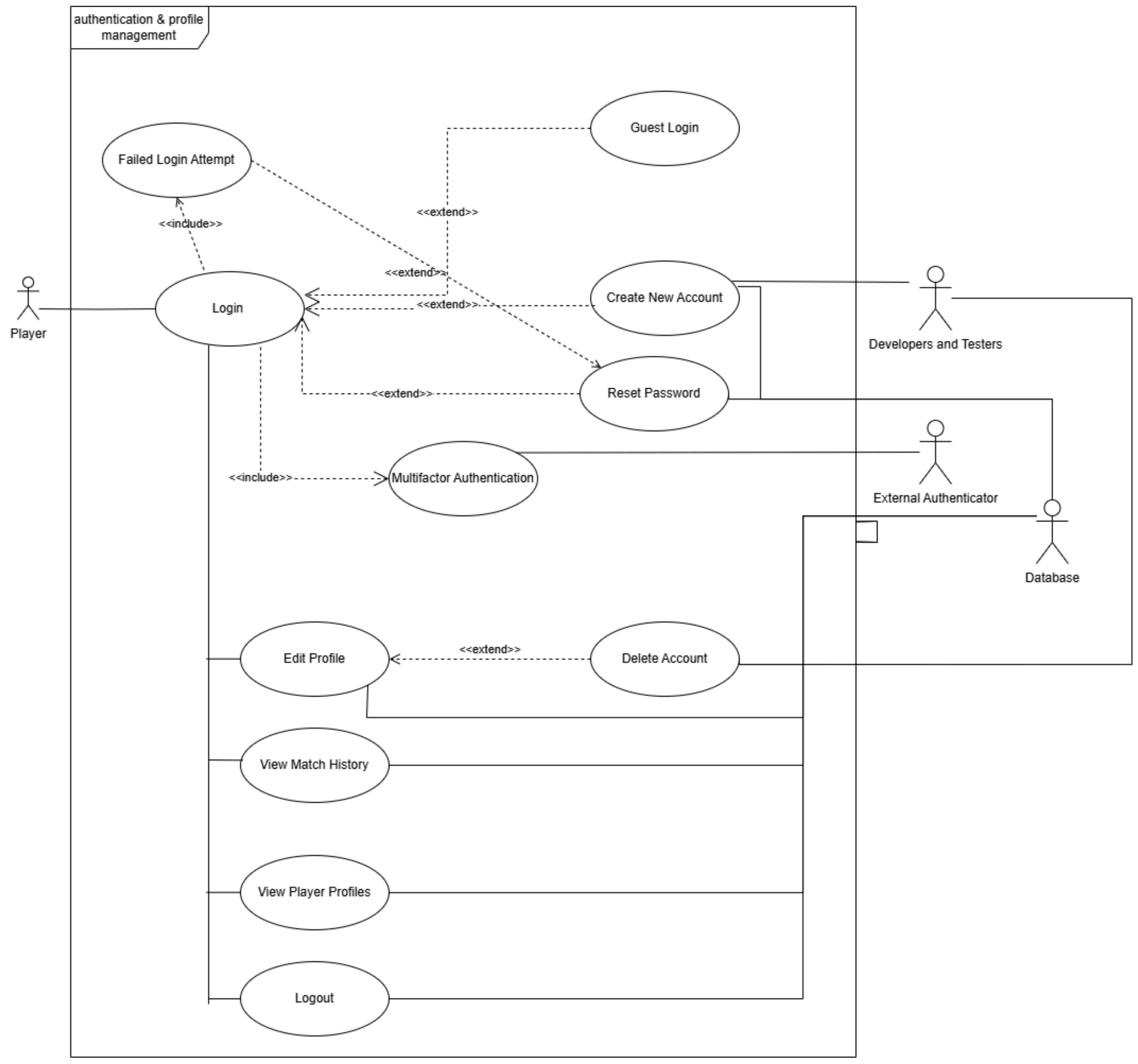
Authentication & Profile Team Use Cases & Diagram



Registering/Creating New Account

Primary Actor:
Game Player

Goal in Context:
Enable any person to create an account.

Pre-Conditions:

- Networking is running as needed.
- It's a human player and not a bot that is trying to overload the server.

Trigger:

- Player clicks the create new account after opening the platform.

Scenario:

1. Player loads the platform through a web browser.
2. Player clicks "Create An Account" after being prompted to login.
3. Player inputs sufficient details for Username, Password, Verify Password & Email.
4. If inputted invalid details where passwords don't match the requirements or if username already exists, it asks the user to perform actions again until valid details are put in.
5. When player's details are valid, it creates account for the player and asks the user to login.

Post-Conditions:

- The player has created an account with an username and passwords that meet the minimum requirements.
- The account details get added to the authentication database and thus allows the player to save progress in game and have a tracked identity in the platform.

Exceptions:

- Server malfunction due to overloading

Priority:

- High – Essential for all parties in the game.

When Available:

- First increment.

Frequency of Use:

- Frequent – Couple times every day or couple times every minute depending on the popularity of the platform.

Channel to Actor:

- Click of a button using touchscreen or mouse click.

Secondary Actors:

- Developers & Testers

Channels to Secondary Actors:

- Raw Code

Open Issues:

1. At what point should the server forcibly shut own the player(if the user is creating many a accounts using the same IP)?

User Login

Primary Actor:
Game Player

Goal in Context:
Allow a player to login.

Pre-Conditions:

- The system is accessible and online.
- The player has an already registered account.

Trigger:

- Player initiates login by clicking the "Login" button.
- Player selects "Guest Login"

Scenario:

Login:

1. Player opens the platform and is prompted to login
2. Player clicks the "login" button.
3. Players enters their username and password.
4. System validates their login credentials.
5. if credentials are valid, the player is granted access.
6. If credentials are invalid, an error message is displayed and the player is prompted to try again.
7. The system may temporarily lock the account if the player has multiple failed attempts.
8. If player does not have an account, they click "Create An Account" and are redirected to the creating an account page.

Guest Login:

1. Player clicks on "Guest Login"
2. System grants limited access, restricting certain featues like, saving progress, or accessing leaderboards.

Post-Conditions:

- After login, the player has access to their profile, and the game features.
- After guest login, the player has limited access.

Exceptions:

- Server downtime.
- Multiple incorrect credentials may result in a temporary lock.

Priority:

- High. Essential for security, and also for user access control.

When Available:

- First increment.

Frequency of Use:

- Frequent. Every time a player accesses the platform.

Channel to Actor:

- Web interface via mouse click or touch interaction.

Secondary Actors:

- Database

Channels to Secondary Actors:

- Code

Open Issues:

1. A possible "Remember Me" option for easier access.
2. Should multifactor authentication be implemented?
3. What specific limitations should be placed on guest users?

User Logout

Primary Actor:
Game Player

Goal in Context:
Allow a player to logout.

Pre-Conditions:

- The system is accessible and online.
- The player is logged in.

Trigger:

- Player initiates logout by clicking the "Logout" button.
- Automatic logout due to inactivity.

Scenario:

1. Player clicks the "logout" button.
2. System ends the session.
3. Player is redirected to the login page.

Post-Conditions:

- After logout, the session is ended, and the player must login again to access their profile.

Exceptions:

- Server downtime.
- Server error prevents proper session termination.

Priority:

- High. Essential for security, and also for user access control.

When Available:

- First increment.

Frequency of Use:

- Frequent. Everytime a player exits the platform.

Channel to Actor:

- Web interface via mouse click or touch interaction.

Secondary Actors:

- Database

Channels to Secondary Actors:

- Code

Open Issues:

1. After how long should the system automatically logout due to inactivity?
2. Should there be a confirmation prompt?

Edit Profile

Primary Actor:
Game Player

Goal in Context:
Allow user to make changes to profile when needed.

Pre-Conditions:

- Networking is running as needed.
- Player has an existing profile and wants to make changes to it.
- Player is logged in.

Trigger:

- Player clicks the edit profile button after logging in to existing account.

Scenario:

1. Player loads the platform through a web browser.
2. Player is able to log in to the system.
3. Player clicks "Edit Profile" after logging in.
4. Player is able to edit username or profile picture. They are also able to change email address or password with proper verification.
5. If invalid details detected the user performs actions again until valid details are put in.
6. When player's details are valid, it changes the desired information.

Post-Conditions:

- The player has successfully edited or changed their information.
- The database is updated as required.

Exceptions:

- Server malfunction due to overloading

Priority:

- Medium.

When Available:

- Second Iteration.

Frequency of Use:

- Only when desired by user.

Channel to Actor:

- Click of a button using touchscreen or mouse click.

Secondary Actors:

- User Database.

Channels to Secondary Actors:

- Code

Open Issues:

1. Security measures for user verification?
2. Require password input to edit profile?

Multifactor Authentication (MFA)

Primary Actor Game Player

Goal in Context

- Requires users to provide an addtional verification step beyond standard username/password to increase account security

Pre-Conditions

- the system is accessible and online
- the played has a registered account and a valid username/password
- MFA feature is enabled on the platform

Trigger -PLayer attempts to log in with correct username and password, prompting the system to initiate the MFA process

Scenario

1. Player enters valid username and password on the login screen.
2. System validates these credentials.
3. Upon successful validation, the system requests a second factor (e.g., code sent to email, SMS, authenticator app).
4. Player inputs the one-time code or approves the request via an authenticator app.
5. System verifies the code or approval.
6. If the second factor is correct, the player is fully logged in.
7. If the second factor is incorrect or not entered within a certain time frame, the system denies access and prompts the user to try again.

Post-Conditions

- Upon successful MFA, the player gains access to their profile and game features.
- If MFA fails, the player remains locked out until they provide the correct second factor.

Exceptions

- Delivery delay or failure for the one-time code (e.g., email or SMS not received).
- Authenticator app malfunction or time-synchronization errors causing valid codes to be rejected.

Priority

- Medium

When Available

- Third iteration

Frequency of Use:

- Every login attempt if MFA is enforced by default.
- Optional if the player chooses to enable it.

Channel to Actor

- Click of a button using touchscreen or mouse click.

Secondary Actor

- External authentication service (e.g., authenticator app)

Open Issues

- Handling of backup authentication methods when the user cannot access their primary second factor?
- Storage and management of user MFA preferences?

Viewing Match History

Primary Actor Game Player

Goal in Context

- To enable the user to view his/her match history and other stats related to it

Pre-Conditions

- the system is accessible and online
- the user is registered and has a username

Exceptions The user has not played any previous match is using the platform for the first time.

Scenario

1. Player enters valid username and password on the login screen.
2. System validates these credentials.
3. The player clicks on match history
4. the match history is visible to the player through a user interface.
5. if the player has no match history, the UI shows a 0/NA or an error window letting the user know the absence of the match history

Post-Conditions

- The player is able to view the matches, their serial number and the stats related to the match.

Priority

- High- Match History is crucial in mathcmaking and keeping the stats of the player

When Available

- first iteration

Frequency of Use:

- Optional: if the player wants to use it
- high for the system for the process of matchmaking

Channel to Actor

- Click of a button using touchscreen or mouse click.

Secondary Actor

- Database

Open Issues

- what stats should the player be able to see?
- should the stats that the system can utilize be the same as the stats that the player can see.

Failed Login Attempt

Primary Actor

- Game Player

Goal in Context

- Highlight the scenario where a player fails to provide valid login credentials and the resulting system behaviour

Pre-conditions

- The system is accessible and online.
- The player has a registered account.
- The player is attempting to log in.

Trigger

- Player enters incorrect login credentials (e.g., username or password) during the login process.

Scenario

1. Player clicks the "Login" button.
2. Player inputs their username and password incorrectly.
3. System checks the credentials against the stored user data.
4. System detects invalid credentials.
5. System displays an error message indicating the login has failed.

Post-Conditions

- The player is not granted access to the system if credentials remain invalid.
- The system may lock the account if the failed attempts exceed the allowed number.

Exceptions

- Data inconsistency or database error causing valid credentials to be rejected.

Priority

- High

Availability

- First Iteration

Frequency of Use

- Occasional – Occurs only when the user enters incorrect credentials.

Channel to Actor

- Web interface via mouse click or touch interaction.

Secondary Actors

- Database

Channel to Secondary Actor

- Code

Open Issues

1. How many failed attempts should be allowed before locking the account?
2. Should the lock duration be fixed or dynamic based on repeated offenses?

Password Reset

Primary Actor:
Game Player

Goal in Context:
Allow a player to reset their password if they forget it.

Pre-Conditions:

- The system is accessible and online.
- The player has a registered account.

- The player has access to the registered email.

Trigger:

- Player clicks on "Forgot Password" on the login page.

Scenario:

1. Player clicks on "Forgot Password"
2. System prompts the player to enter their registered email address.
3. Player enters the email and submits a request.
4. System verifies if the email is registered to an account.
5. If the email is valid. the system sends a password reset link to the email.
6. Player clicks on the password reset link. Which leads them to a password reset page.
7. Player enters a new password.
8. System verifies the new password against security requirements.
9. If valid, the password is updated and the player is notified.
10. Player can now login using the new password.

Post-Conditions:

- The password is updated, and the player is notified.
- System prevents the old password from being used.

Exceptions:

- Email entered does not match a registered email.
- Reset link expires after a certain time.
- The new password does not meet security requirements.
- System fails to send the reset email.

Priority:

- High. Essential for account recovery.

When Available:

- First increment.

Frequency of Use:

- Medium. Only for when players forget their passwords.

Channel to Actor:

- Web interface via mouse click or touch interaction.

Secondary Actors:

- Database

Channels to Secondary Actors:

- Code

Open Issues:

1. How long should the reset link remain valid?
2. Should MFA be required?
3. Should there be a limit to the number od resets with in a certain timeframe?

Delete Account

Primary Actor:
Game Player

Goal in Context:
Enable any person to delete their account from the profile management page.

Pre-Conditions:

- Network is running as needed.
- Person is logged into their account

Trigger:

- Player clicks the delete account button from their profile management page.

Scenario:

1. Player logs in to their account.
2. Player goes the profile management page.
3. Player clicks on the delete account button.
4. Player gets asked to confirm if they want to delete account.
5. If players confirms deletion, they are asked to input their password again and if the passwords matches with the database then player is officially logged and account gets deleted

Post-Conditions:

- The person gets logged out of the account and lands to the homepage.
- The person gets a sidebar notification saying account deleted.

Exceptions:

- database/server malfunction.

Priority:

- Mid – not essential to the function of the platform.

When Available:

- Second increment.

Frequency of Use:

- moderate – once in a while.

Channel to Actor:

- Click of a button using touchscreen or mouse click under the profile management page.

Secondary Actors:

- Developers, Testers, Database

Channels to Secondary Actors:

- Code

Open Issues:

1. Should we let players recover account during a period of time if it's already deleted from database?

View Player Profile

Primary Actor:
Game Player

Goal in Context:
Allow user to view other players profile.

Pre-Conditions:

- Networking is running as needed.
- Player has an existing profile.
- Player is logged in and has started a game.
- Player wants to view opponent profile.

Trigger:

- Player clicks on the opponents name/ profile picture.

Scenario:

1. Player loads the platform through a web browser.
2. Player is able to log in to the system.
3. Player selects a game from the library and is matched with an opponent.
4. Player wishes to view opponent information.
5. Player clicks on the opponent username or profile picture.
6. Player is able to see information like ranking, games played etc.
7. Player is able to successfully exit out of the opponent profile and can continue the game as desired.

Post-Conditions:

- The player has successfully viewed other player’s profile.

Exceptions:

- Server malfunction.
- Malfunction in code/matchmaking.

Priority:

- Medium.

When Available:

- Second Iteration.

Frequency of Use:

- Only when desired by user.

Channel to Actor:

- Click of a button using touchscreen or mouse click.

Secondary Actors:

- Database

Channels to Secondary Actors:

- Code

Open Issues:

- Extent of information the player is able to see?