Use Case: Signing Up and Setting Up an Account

Iteration: 1, Initial version.

Primary actor: User (New Player)

Goal in context:

- To allow new users to create an account and set up their profile to access the OMG platform.

Preconditions:

- The user has access to the OMG platform via a web browser or application. • The user is not already registered.

Trigger:

- The user wants to create an account to access the platform.

Scenario:

1. The user navigates to the OMG platform's sign-up page.
2. The system displays a registration form requesting:
    - Username
    - Email address
    - Password
    - Date of birth (for age restrictions)
3. The user fills in the required details and submits the form.
4. The system validates the input fields and checks for:
    - Unique username and email
    - Password strength
    - Proper format for email and date of birth
5. If validation is successful, the system sends a verification email to the user.
6. The user checks their email and clicks the verification link.
7. The system confirms the email and activates the account.
8. The user logs in for the first time using their credentials.
9. The system prompts the user to set up their profile, requesting:
    - Profile picture (optional)
    - Display name
    - Game preferences (optional)
    - Privacy settings
10. The user completes the profile setup and submits the information. 11. The system saves the user's preferences and redirects them to the main dashboard.

Exceptions:

1. Username or email already taken—system prompts the user to choose a different one.
2. Weak password—system suggests a stronger password.
3. Email not verified—system prevents login and reminds the user to verify their email.
4. Forgot password—user can request a reset before completing verification. 5. User skips profile setup—user can proceed to the dashboard and complete setup later.

Priority: Moderate priority, to be implemented in early development.

When available: First increment.

Frequency of use: Frequent.

Channel to actor: Via web browser or application.

Secondary actors:

- System administrator (for account verification issues).
- Customer support (for troubleshooting account creation).
- Database

Channels to secondary actors:

- System administration dashboard.
- Customer support ticket system.
- System

Open issues:

1. What mechanisms will be in place to prevent bots and fake accounts?
2. How can the system ensure strong password compliance without frustrating users? 3. Should the platform enforce two-factor authentication (2FA) for added security? 4. Will users have the ability to sign up using third-party authentication (Google, Facebook, etc.)?

Use Case: Logging in with an Existing Account

Iteration: 1, Initial version.

Primary actor: User (Returning Player)

Goal in context:

- To allow returning users to log in and access their OMG platform account.

Preconditions:

- The user has already registered and verified their account.
- The user has access to a web browser or application to log in.

Trigger:

- The user wants to log in to their existing account.

Scenario:

1. The user navigates to the OMG platform's login page.
2. The system displays a login form requesting:
    - Username or email
    - Password
3. The user enters their credentials and submits the form.
4. The system validates the credentials against stored user data.
5. If validation is successful, the system authenticates the user and grants access. 6. The system redirects the user to their dashboard, displaying their profile and game history.
7. If enabled, the system checks for saved preferences and applies them (e.g., dark mode, notifications).

Exceptions:

1. Incorrect username/email or password—system displays an error and prompts the user to try again.
2. Multiple failed login attempts—system may trigger a CAPTCHA or temporarily lock the account.
3. Forgotten password—user can request a password reset via email.
4. Account not verified—system prompts the user to verify their email before logging in. 5. System maintenance—if the platform is under maintenance, a message is displayed with an estimated availability time.

Priority: High priority, required for user access.
When available: First increment.

Frequency of use: Frequent.

Channel to actor: Via web browser or application.

Secondary actors:

- System administrator (for account recovery issues).
- Customer support (for troubleshooting login problems).
- Database

Channels to secondary actors:

- System administration dashboard.
- Customer support ticket system.
- System

Open issues:

1. Should the platform support social media logins (Google, Facebook, etc.)? 2. Will two-factor authentication (2FA) be required or optional for enhanced security? 3. How long should login sessions remain active before requiring reauthentication? 4. Should the system notify users of login attempts from new devices or locations?

Use Case: Joining a Match

Iteration: 1, Initial version.

Primary Actor: User (Player)

Goal in Context: To allow users to join a multiplayer match on the OMG

platform. **Preconditions**:

- The user is logged in to the platform.
- The user has a stable internet connection.
- The user has access to the game library and available matches.
- The match being joined is open and ready for new players.

Trigger:

- The user wants to join an available match either by browsing the game list or accepting an invitation.

Scenario:

1. The user navigates to the match selection screen, either through the game library or a game invite.
2. The system displays available matches, including information such as:
    - Game type
    - Number of players already in the match
    - Current game status (waiting for players, in-progress, etc.)
3. The user selects a match to join.
4. The system checks if the match is still open and if the user meets the necessary criteria (e.g., correct skill level, no restrictions).
5. If the match is available, the system allows the user to join the match and displays a waiting room or the game interface, depending on the game's state.
6. The system notifies the other players in the match that a new player has joined.
7. Once all players have joined, the system begins the game.

Exceptions:

- **Match Full**: If the match is already full, the system notifies the user and suggests other available matches.
- **Match in Progress**: If the match has already started, the system informs the user that the game cannot be joined.
- **User Incompatible**: If the user does not meet the requirements (e.g., skill level mismatch or region restriction), the system will notify the user and suggest an alternative match. • **Connection Issues**: If the user loses connection while attempting to join, the system will prompt the user to try again or return to the match selection screen.

Priority: High priority, required for multiplayer functionality.

When Available: First increment.

Frequency of Use: Frequent, as joining matches is central to the gameplay.

Channel to Actor: Via web browser or application.

Secondary Actors:

- **Match Host**: The player who created the match.
- **System Administrator**: For managing backend issues, such as matchmaking or server stability.
- **Customer Support**: For troubleshooting connection or account issues related to joining matches.

Channels to Secondary Actors:

• **Match Host**: Notification system or in-game chat for match updates. • **System Administrator**: Admin dashboard for monitoring match server health and load. • **Customer Support**: Customer support ticket system for any gameplay-related issues.

Open Issues:

• Should there be a limit on how many matches a player can join in a certain period? • Will there be matchmaking restrictions (e.g., region-based, skill-based) for certain games?
• Should there be a mechanism for a player to leave a match once they have joined?

Use Case: Tracking Game History

Iteration: 1, Initial version.

Primary Actor: User (Player)

Goal in Context: To allow users to view their past game history, including wins, losses, and other relevant statistics.

Preconditions:

- The user is logged in to the OMG platform.
- The user has participated in one or more matches.
- The system has stored game data such as match results, player statistics, and game types.

Trigger:

- The user wants to view their game history to review their performance or analyze their past games.

Scenario:

1. The user navigates to their profile or dashboard on the OMG platform. 2. The system displays the user's game history section, showing a list of past games including:
    - Game title
    - Match date
    - Opponents
    - Game results (win/loss/draw)
    - Game duration
    - Performance metrics (e.g., score, ranking)
3. The user can filter their game history by date, game type, or performance (e.g., showing only wins or specific games).
4. The user can select a specific game from the list to view more detailed information, such as:
    - A detailed game breakdown (moves, actions, or interactions)
    - Leaderboard positions at the time of the match
    - Any achievements or milestones earned during the game
5. The system allows the user to export or share their game history via social media, email, or within the platform.

Exceptions:

- **No Game History**: If the user has not played any games yet, the system will display a message informing the user that they have no game history available.
- **Data Corruption**: If there's an issue with retrieving the game data, the system will display an error message and suggest the user contact customer support.
- **No Internet Connection**: If the user's connection is lost while trying to load their game history, the system will prompt the user to reconnect.

Priority: High priority, as it is an essential feature for player engagement and tracking progress.

When Available: First increment.

Frequency of Use: Frequent, as players often want to track their progress and review past matches.

Channel to Actor: Via web browser or application.

Secondary Actors:

- **System Administrator**: For ensuring the integrity and availability of game data.
- **Customer Support**: For assisting users with any issues related to accessing their game history.

Server

Channels to Secondary Actors:

- **System Administrator**: Admin dashboard for monitoring data storage and user requests.
- **Customer Support**: Customer support ticket system for troubleshooting issues related to game history retrieval

System.

Open Issues:

- How long should game history be stored? Should there be an archive or a limit on the number of games shown?
- Should users have the ability to delete or edit their game history?
- Will game history be shared across multiple devices or platforms (cross-platform play)?

Use Case: Sending a Friend Request

Iteration: 1, Initial version.

Primary Actor: User (Player)

Goal in Context: To allow users to send friend requests to other players in order to connect, play together, and interact on the OMG platform.

Preconditions:

- The user is logged into the OMG platform.
- The user has a valid account and is active on the platform.
- The user is able to search for or browse other players' profiles.

Trigger:

- The user wants to add another player to their friends list to connect and play together.

Scenario:

1. The user navigates to the profile or search section to find the player they wish to send a friend request to.
2. The system displays a list of available players or a search function for the user to enter a username or email.
3. The user selects the player they want to send a friend request to.
4. The system displays the selected player's profile, showing their current status (online/offline), games played, and other relevant information.
5. The user clicks on the "Send Friend Request" button on the profile.
6. The system sends the friend request to the selected player and notifies the user that the request has been sent successfully.
7. The system notifies the selected player of the incoming friend request, which they can accept or decline.
8. If the request is accepted, the system adds the users to each other's friends list, and they are able to see each other's online status and send game invites.

Exceptions:

- **Friend Request Already Sent**: If the user has already sent a friend request to the player, the system will display a message stating that the request has already been sent. • **Player Already in Friends List**: If the user and the selected player are already friends, the system will display a message indicating that they are already connected. • **User Blocked**: If the selected player has blocked the user, the system will inform the user that they cannot send a friend request.
- **Request Declined**: If the player declines the friend request, the system will notify the user of the declined status.

Priority: Low

When Available: First increment.

Frequency of Use: Moderate, as players typically send friend requests to people they want to play with or communicate with more regularly.

Channel to Actor: Via web browser or application.

Secondary Actors:

- **System Administrator**: For monitoring and resolving any issues related to user accounts and the friend request system.
- **Customer Support**: For assisting users with issues related to sending or receiving friend requests (e.g., blocked users, incorrect functionality).

Channels to Secondary Actors:

- **System Administrator**: Admin dashboard for managing system errors and resolving issues with the friend request feature.
- **Customer Support**: Customer support ticket system for troubleshooting and resolving any user complaints or issues related to friend requests.

Open Issues:

- Should there be a limit on how many friend requests a user can send in a day to prevent spam?
- Should friend requests expire if not accepted within a certain period?
- Should the system provide an option to customize friend request messages, or should it be a simple request notification?

Use Case: Quitting a Game

Iteration: 1, Initial version.

Primary Actor: User (Player)

Goal in Context: To allow users to quit a game they are currently playing on the OMG platform, whether due to disconnecting, finishing the game, or choosing to exit early.

Preconditions:

- The user is actively participating in a game on the OMG platform.
- The game is currently running, and the user has control over their game actions. • The system allows players to leave or quit during gameplay (depends on game type and settings).

Trigger:

- The user wants to quit the game either to stop playing, exit due to a technical issue, or leave for personal reasons.

Scenario:

1. The user decides to quit the game while it is in progress.
2. The system displays a confirmation prompt asking the user if they are sure they want to quit the game.
    - This prompt includes options: **"Yes"** to confirm quitting or **"No"** to cancel and continue playing.
3. If the user selects **"Yes"**, the system proceeds to quit the game.
    - The user's game data (score, progress, etc.) is either saved or discarded, based on game rules and settings.
    - If applicable, the system updates the match status (e.g., marks the game as incomplete, or declares a surrender or forfeiture).
    - The user is returned to the main game menu, dashboard, or match selection screen.

4. If the user selects **"No"**, the game continues, and the user remains in the match. 5. If the user experiences a disconnect (internet loss, crash, etc.), the system attempts to reconnect the player, and if unsuccessful, the game ends, recording the user as a quitter or incomplete participant, depending on game type.

Exceptions:

- **Game State Lock**: If quitting is not allowed due to game rules (e.g., competitive ranked matches), the system will notify the user that quitting is not permitted.
- **Unsaved Progress**: If quitting results in unsaved progress or incomplete matches, the system will warn the user that they may lose any unsaved data.
- **Connection Issues**: If the game quits unexpectedly due to connection problems, the system will try to reconnect the player. If unsuccessful, the game will be marked as incomplete, and the user may be penalized or asked to reconnect.
- **Forced Quit by Server**: If the game server crashes or is shut down, the system will automatically end the game for all players, notifying the users of the issue.

Priority: Medium priority, as quitting is a common part of gameplay but may need to be controlled for certain match types (e.g., competitive modes).

When Available: First increment.

Frequency of Use: Moderate, as players occasionally quit games for various reasons.

Channel to Actor: Via web browser or application.

Secondary Actors:

- **Match Host**: If the user is the host of the game, their quitting may affect the game status for all players.
- **System Administrator**: For monitoring technical issues, including connection problems or server shutdowns that may cause game quitting.
- **Customer Support**: For assisting users with issues related to quitting games or recovering unsaved progress.

Channels to Secondary Actors:

- **Match Host**: In-game chat or notification system for informing other players if the match is ending or disrupted.
- **System Administrator**: Admin dashboard for managing server health, player connection status, and game match integrity.
- **Customer Support**: Customer support ticket system for resolving issues related to quitting games, recovering lost progress, or handling disputes over quitting.

Open Issues:

- Should there be a penalty or consequence for players who quit mid-game in competitive or ranked matches?
- How should the system handle quitting in cooperative games (e.g., does it impact other players)?
- Should there be a confirmation step to prevent accidental quitting during intense gameplay?

Use case: Forgotten Password Reset

Iteration: 1

Primary Actor: User (Player)

Goal in context:

• To allow users to access their account long enough to create a new password.

Preconditions:

• The user has access to the OMG platform via a web browser or an application. • The user has a registered account with an verified username, email and password.

Trigger:

• The user wants to access their account but can't give the password they registered with.

Scenario:

1. The user navigates to the 'forgot password'screen from the OMG login screen.
2. The screen prompts for:
    a. Email
    b. Username
3. User enters their information.
    a. The system checks both inputs for valid formatting
    b. The system checks the username/email pairing against stored user data. 4. If the input is valid, a temporary password is sent to the provided email address. a. The temporary password is deleted from the database 15 minutes after being sent, or right after the user logs in.
5. The system prompts the user to create a new password.
    a. System checks the password's validity.
6. The new password is added to the database.

Post conditions:

• The user has a new password that they can log into their account with.

Exceptions:

1. Username or email not recognized.
    - User is asked to check that they are entering the correct information.
2. Email not associated with username.
    - User is asked to enter a different email.
3. New password isn't formatted in a valid way.
    - System suggests ways to make a stronger password.

Priority: High priority. This feature restores access to the platform in the case that a user forgets their password.

When available: First iteration

Frequency of use: Intermittent

Channel to actor: ‘Forgot password’screen, via web browser or application.

Secondary actors:

- User data database: database containing usernames, emails and passwords.

Channel to secondary actors:

- User data database: application server for returning database information to the client.

Open issues:

1. Should there be a limit for password reset requests?
2. Should there be a time-out between reset requests?

Use Case: Deleting an Existing Account

Iteration: 1, Initial version.

Primary actor: User (Registered Player)

Goal in context:

- To allow users to permanently delete their OMG platform account along with associated data.

Preconditions:

- The user is logged into their account.
- The user has access to account settings where the deletion option is available.

Trigger:

- The user decides to permanently delete their account.

Scenario:

1. The user navigates to the OMG platform's account settings.
2. The system displays the option to delete the account.
3. The user selects the delete account option.
4. The system prompts the user with a confirmation message warning about data loss.
5. The user confirms the deletion request.
6. The system asks the user to re-enter their password for security verification. 7. The system validates the password and proceeds with the deletion process. 8. The system removes all user data, including profile information, game history, and preferences.
9. The system logs the user out and displays a confirmation message stating the account has been deleted.

Exceptions:

1. Incorrect password entered—system notifies the user and prompts them to try again. 2. User changes their mind—system provides an option to cancel the deletion request before final confirmation.
3. Pending transactions or disputes—if the user has ongoing purchases, refunds, or disputes, the system prevents account deletion until they are resolved.
4. System error—if deletion fails due to technical issues, the system notifies the user and logs the issue for review.

Priority: High priority, necessary for compliance with user privacy rights.
When available: Second increment.

Frequency of use: Infrequent.

Channel to actor: Via web browser or application.

Secondary actors:

- System administrator (for account recovery within a grace period if applicable). • Customer support (for handling disputes or issues related to account deletion).

Channels to secondary actors:

- System administration dashboard.
- Customer support ticket system.

Open issues:

1. Should there be a grace period (e.g., 30 days) during which users can recover their account before permanent deletion?
2. How should account deletion requests be handled for users with active subscriptions or premium features?
3. Should users receive a final email confirming their account deletion? 4. What level of data anonymization (if any) should be applied instead of full deletion to comply with legal requirements?

Use case: Logging out of account

Iteration: 1

Primary Actor: User (Player)

Goal in context:

- To allow users to log out of their account.

Preconditions:

- The user has a registered and verified account.
- The user is currently logged into their account from a web browser or application. • The user is not currently in a match.

Trigger:

- The user wants to log out of the account they are currently using.

Scenario:

1. The user initiates logging out through their dashboard.
2. System acts for confirmation from the user.
3. User confirms.
4. The system sends the logout request to the server.
5. The server confirms logout success and session termination.
6. System redirects user to the OMG platform's login page.

Post conditions:

- The user is logged out of their account and taken back to the login page.

Exceptions:

1. Connection interruptions: If the log out attempt is interrupted by connectivity issues, the system will display a message for the user, and log them out.

Priority: High priority. Logging out protects user information and privacy and notifies the system of session termination.

When available: First iteration.

Frequency of use: Frequent

Channel to actor: OMG platform dashboard via web browser or

application. **Secondary actors**: Server

Channel to secondary actors:

- Server: Client-server communication over the network

Open issues:

- N/A