

GUI Use Case Descriptions

Use Case: Login Window

- **Use Case:** Accessing the 0mG platform by signing in, creating an account, or resetting a password.
- **Iteration:** 1st iteration
- **Primary Actor:** Player (a user accessing the platform).
- **Goal:** The player authenticates or registers to use the platform.
- **Preconditions:** The 0mG app is running, showing the login window.
- **Triggers:** The player launches the 0mG platform.
- **Main Scenario (Sign In):**
 1. The login window appears with fields for username and password, plus "Login," "Create Account," and "Forgot Password" buttons.
 2. The player enters "player" and "1234" and clicks "Login."
 3. The system checks the credentials (hardcoded: "player"/"1234").
 4. The home screen (GameMenu) displays with game and feature options.
- **Alternate Scenarios:**
 - **Sign-In Fails:** If credentials are invalid (e.g., "wrong"/"0000"), an error "Invalid username or password" appears, and the player retries.
 - **User Signs Up:**

1. The player clicks "Create Account," opening a dialog.
 2. They enter "newplayer" and "newpass123," then click "Create."
 3. If valid, the account is logged (demo console output) with "Account created successfully!" and the dialog closes.
 4. If empty, an error "Username and password cannot be empty" keeps the dialog open.
- **User Resets Password:**
 1. The player clicks "Forgot Password," opening a dialog.
 2. The system attempts to contact a backend email service (placeholder) for a reset email.
 3. The dialog shows a placeholder message and closes, returning to the login window.
 - **Post Conditions:**
 - Successful sign-in shows the home screen.
 - Sign-up creates an account and returns to login.
 - Password reset returns to login (placeholder).
 - **Exceptions:** App crashes if the login window fails to load; closing the app terminates the process.
 - **Priority:** High; essential for platform access.
 - **Frequency of Use:** High; used at session start.
 - **Channel to Actor:** GUI (login window interface).
 - **Secondary Actors:** Backend Email Service (placeholder for reset).
 - **Open Issues:** Should there be a limit on login attempts for security?

Use Case: User Signs Up (Create Account)

- **Use Case:** Creating an account on the 0mG platform.
- **Iteration:** 1st iteration
- **Primary Actor:** Player (a new user of the 0mG platform).
- **Goal in Context:** The player creates an account using the GUI to access the 0mG platform.
- **Preconditions:**
 - The 0mG platform is open and running.
 - The player doesn't already have an account.
- **Triggers:** The player clicks the "Sign-Up" or "Create Account" button on the login screen.
- **Scenario:**
 1. The player opens the 0mG platform.
 2. The login screen appears with an option to sign up.
 3. The player clicks "Sign Up" and sees a form asking for a username and password.
 4. The player enters their details and clicks "Create Account."
 5. The app confirms the account is created and sends them back to the login screen.
- **Post Conditions:**
 - The player now has an account.
 - The app is back at the login screen.

- **Exceptions:**
 1. If the username is taken, the app shows an error message, and the player tries again.
 2. If passwords don't match, the app asks the player to re-enter the password.
 3. If there is a system issue, the app says it couldn't create an account, and the player stays on the sign-up screen.
- **Priority:** High.
- **When Available:** Always available when the 0mG system is running.
- **Frequency of Use:** Once per new player.
- **Channel to Actor:** The app's GUI (sign-up screen).
- **Secondary Actors:** None.
- **Channel to Secondary Actors:** N/A.
- **Open Issues:**
 - Should we require an email?
 - Do we need specific password rules?

Use Case: User Signs In

- **Use Case:** Signing into the 0mG platform.
- **Iteration:** 1st iteration
- **Primary Actor:** Player (a registered user of the 0mG platform).

- **Goal in Context:** The player signs into the 0mG platform via the GUI to access multiplayer board game features.
- **Preconditions:**
 - The 0mG system is running on the player's device.
 - The player has an existing account with valid credentials.
- **Triggers:** The player initiates the 0mG system and selects the sign-in option.
- **Scenario:**
 1. The player starts the 0mG system.
 2. The GUI displays the sign-in screen with username and password fields.
 3. The player enters a username and password and clicks "Sign In."
 4. The GUI sends credentials to the Authentication stub and verifies them.
 5. If successful, the GUI transitions to the main menu with game options.
- **Post Conditions:**
 - The player is signed in.
 - The GUI shows the main menu.
- **Exceptions:**
 1. If the player enters invalid credentials, the GUI displays "Wrong username/password," and the player retries.
 2. If there is a system failure, the GUI shows "Unable to sign in," and the player stays on the sign-in screen.
- **Priority:** High.
- **When Available:** Always available when the 0mG system is running.
- **Frequency of Use:** Once per session.

- **Channel to Actor:** GUI (sign-in interface).
 - **Secondary Actors:** None.
 - **Channel to Secondary Actors:** N/A.
 - **Open Issues:**
 - Should we add a "Forgot Password" option?
 - Should we show a loading indicator during the stub call?
-

Use Case: User Resets Password

- **Use Case:** Resetting a forgotten password.
- **Iteration:** 1st iteration
- **Primary Actor:** Player (a registered user of the 0mG platform who has forgotten their password).
- **Goal in Context:** The player recovers access to their account by resetting their password using the GUI, allowing them to log back into the 0mG platform.
- **Preconditions:**
 - The 0mG platform is running and accessible on the player's device.
 - The player has an existing account with a valid registered email address.
 - The player is on the Login Screen and unable to sign in due to a forgotten password.
 - The platform's email system is operational for sending reset links.
- **Triggers:** The player clicks the "Forgot Password" link on the Login Screen.
- **Scenario:**

1. The player opens the 0mG platform and navigates to the Login Screen.
 2. The GUI displays fields for username and password, along with a "Forgot Password" link below the sign-in button.
 3. The player clicks "Forgot Password," and the GUI transitions to a Password Reset Screen with an email input field.
 4. The player enters their registered email address and clicks "Submit."
 5. The system validates the email, generates a unique reset link, and sends it to the player's email inbox. The GUI displays a message: "Check your email for a reset link."
 6. The player opens their email, finds the message from 0mG, and clicks the reset link, which opens a browser or app window with a New Password Screen.
 7. The player enters a new password (and confirms it if required) and clicks "Save Password."
 8. The system updates the player's account with the new password, and the GUI shows a confirmation: "Password updated successfully. Please log in."
 9. The player is redirected to the Login Screen, enters their username and new password, and signs in successfully.
- **Post Conditions:**
 - The player's account password is updated in the system.
 - The player regains access to the 0mG platform and is returned to the Login Screen to sign in.

- The reset link becomes invalid after use or expiration.
- **Exceptions:**
 1. If the entered email isn't tied to an account, the GUI displays "No account found with this email. Please try again or sign up."
 2. If the email is malformed (e.g., missing "@"), the GUI shows "Invalid email format" and prompts re-entry.
 3. If the player clicks the reset link after its expiration, the GUI displays "This link has expired. Request a new one."
 4. If the system can't send the email (e.g., server issue), the GUI shows "Unable to send reset link. Try again later."
 5. If the new password doesn't meet criteria (e.g., minimum length), the GUI displays an error message and prompts re-entry.
 6. If connectivity drops during submission, the GUI shows a notification: "Connection lost. Retry?"
- **Priority:** High; essential for account recovery and user retention.
- **When Available:** Always available when the system is running.
- **Frequency of Use:** Rare but critical when needed.
- **Channel to Actor:** GUI (Login Screen → Password Reset Screen → New Password Screen) and email client (for receiving the reset link).
- **Secondary Actors:** Backend email service.
- **Channel to Secondary Actors:** Internal API call to email service.
- **Open Issues:**
 - Should the reset link expire after a set time (e.g., 15 minutes, 24 hours)?

- Should there be specific password rules (e.g., 8+ characters, include a number)?
 - Should there be a limit on reset requests per hour to prevent abuse?
-

Use Case: User Waits for Opponent

- **Use Case:** Waiting in a lobby for another player to join a selected game on the 0mG platform.
- **Primary Actor:** Player (a user who wants to play a game).
- **Goal:** The player waits for an opponent to join their game session to start playing.
- **Preconditions:** The player has signed in, selected a game (e.g., Checkers), and chosen a lobby mode (e.g., Quick Play, Create Lobby, or Join Lobby).
- **Triggers:** The player enters a lobby after selecting a game and mode.
- **Main Scenario (Waiting in Lobby):**
 1. The player selects Checkers and Quick Play, entering a lobby.
 2. The system displays a waiting screen with a "Waiting for Opponent" message and a cancel option.
 3. Another player joins the lobby via matchmaking.
 4. The system starts the Checkers game, showing the game board to both players.
- **Alternate Scenarios:**
 - **No Opponent Joins:**

1. No opponent joins within a timeout period (e.g., 60 seconds).
 2. The system shows a message: "No opponent found. Try again?" with options to retry or exit.
 3. The player chooses to exit, returning to the home screen.
- **Player Cancels Waiting:**
 1. The player clicks "Cancel" while waiting.
 2. The system exits the lobby and returns the player to the home screen.
 - **Opponent Joins for Different Game:** If the player selected Connect Four, the system starts Connect Four instead of Checkers (same flow, different game).
- **Post Conditions:**
 - If an opponent joins, the game starts (e.g., Checkers game board displayed).
 - If no opponent joins or the player cancels, they return to the home screen.
 - **Exceptions:** If the matchmaking server is down, the system shows an error: "Cannot connect to lobby."
 - **Priority:** Medium; required for multiplayer gameplay.
 - **Frequency of Use:** Moderate; used when the player chooses multiplayer modes.
 - **Channel to Actor:** GUI (lobby waiting screen).
 - **Secondary Actors:** Other Players (who join the lobby).
 - **Open Issues:** Should there be a visible timer for the waiting period?

Use Case: User Selects Game

- **Use Case:** Choosing a board game from the available game library.
- **Iteration:** 1st iteration
- **Primary Actor:** Player
- **Goal in Context:** The player selects a game they wish to play, initiating the process of joining a match.
- **Preconditions:**
 - The player is logged into their account.
 - The system displays a list of available games.
 - Game thumbnails and descriptions are preloaded.
- **Triggers:** The player interacts with the game selection screen.
- **Scenario:**
 1. The player navigates to the Game Selection screen.
 2. The system displays a list of available board games (e.g., Checkers, Connect Four, Tic-Tac-Toe).
 3. The player selects a game by clicking or tapping on its thumbnail.
 4. The system loads the selected game's details, including available rooms and game settings.
 5. The player confirms their selection and is redirected to the Matchmaking Lobby or Game Setup screen.
- **Post Conditions:**

- The selected game is loaded, and the player is ready to join or create a match.
 - The system updates the player's recent game history.
- **Exceptions:**
 1. If the player selects a game that is temporarily unavailable, an error message is displayed.
 2. If the system fails to load game details, the player is prompted to try again.
 3. If there is a network issue, the player is notified and returned to the main menu.
 - **Priority:** High; essential for accessing and playing games.
 - **When Available:** Any time after login.
 - **Frequency of Use:** High; occurs in every session before gameplay starts.
 - **Channel to Actor:** GUI (Game Selection screen).
 - **Secondary Actors:** None.
- **Open Issues:**
 - Should the game selection screen have a recommended game feature?
 - Should unavailable games be grayed out, hidden, or marked with a notification?

Use Case: Creating a New Game Lobby

- **Use Case:** Creating a private or public game room.
- **Iteration:** 1st iteration

- **Primary Actor:** Player
- **Goal in Context:** The player sets up a new game session and invites others to join.
- **Preconditions:**
 - The player has selected a game.
 - The system is connected to the matchmaking service.
- **Triggers:** The player chooses to create a new game.
- **Scenario:**
 1. The player selects “Create Game” from the Game Setup screen.
 2. The system prompts the player to select game settings (e.g., time limit, board size, AI).
 3. The player sets the game to public or private.
 4. If private, the player invites friends by sharing a room code.
 5. The system creates the game lobby and waits for other players.
 6. The player clicks “Start Game” once enough players have joined.
- **Post Conditions:**
 - A game session is successfully created.
 - Invited players receive notifications to join.
 - The system updates the database with the new game session.
- **Exceptions:**
 1. If the player disconnects before starting, the lobby is deleted.
 2. If a private game invite fails, the system retries sending it.

3. If no players join within a set time, the system offers to convert it to an AI match.
- **Priority:** High; core functionality for multiplayer games.
 - **When Available:** Any time after selecting a game.
 - **Frequency of Use:** Moderate; occurs when players want to host a match.
 - **Channel to Actor:** GUI (Game Lobby interface).
 - **Secondary Actors:** Invited players.
 - **Open Issues:**
 - Should private game lobbies expire after a certain time?
 - Should players be able to customize game rules before starting?

Use Case: Joining an Existing Game

- **Use Case:** Joining a public or private game room.
- **Iteration:** 1st iteration
- **Primary Actor:** Player
- **Goal in Context:** The player joins an active multiplayer game session.
- **Preconditions:**
 - The player is logged into the platform.
 - There are active game lobbies available.
- **Triggers:** The player selects “Join Game” from the game menu.
- **Scenario:**
 1. The player navigates to the Join Game menu.

2. The system displays a list of available game lobbies.
 3. The player selects a game lobby.
 4. If the game is **public**, the player joins immediately.
 5. If the game is **private**, the system prompts for a room code.
 6. Upon successful entry, the player enters the game lobby and waits for the match to start.
- **Post Conditions:**
 - The player is placed into the selected game room.
 - **Exceptions:**
 1. If the game is full, the player receives an error message.
 2. If the room code is incorrect, the system prompts the player to re-enter it.
 3. If there is a connection issue, the system notifies the player.
 - **Priority:** High; core multiplayer functionality.
 - **When Available:** Any time after selecting a game.
 - **Frequency of Use:** High; occurs when joining an active session.
 - **Channel to Actor:** GUI (Game Lobby interface).
 - **Secondary Actors:** Other players in the lobby.
 - **Open Issues:**
 - Should players be able to spectate games?
 - Should the system auto-suggest open games based on player skill level?

Use Case: User Waits for Opponent / user couldn't join

- **Use Case:** Waiting in a lobby for another player to join a selected game, but no one joins.
- **Primary Actor:** Player (a user who wants to play a game).
- **Goal:** The player waits for an opponent to join their game session, but no opponent is found.
- **Preconditions:** The player has signed in, selected a game (e.g., Checkers), and entered a lobby via a mode (e.g., Quick Play).
- **Triggers:** The player enters a lobby after selecting a game and mode.
- **Main Scenario (No Opponent Joins):**
 1. The player selects Checkers and Quick Play, entering a lobby.
 2. The system shows a waiting screen with a "Waiting for Opponent" message and a cancel option.
 3. No opponent joins within a timeout period (e.g., 60 seconds).
 4. The system displays a message: "No match found. Try again?" with options to retry or exit.
 5. The player chooses to exit, returning to the home screen.
- **Post Conditions:**
 - If no opponent joins and the player exits, they return to the home screen.
 - If the player retries, they remain in the lobby.
- **Exceptions:** If the matchmaking server fails, the system shows an error: "Cannot connect to lobby."
- **Priority:** Medium; part of multiplayer gameplay.

- **Frequency of Use:** Moderate; occurs in multiplayer modes when opponents are unavailable.
 - **Channel to Actor:** GUI (lobby waiting screen with message and options).
 - **Secondary Actors:** Other Players (none join in this scenario).
 - **Open Issues:** Should the timeout period be adjustable?
-

Use Case: Matchmaking (Quick Play)

- **Use Case:** Automatically matching players for a game.
- **Iteration:** 1st iteration
- **Primary Actor:** Player
- **Goal in Context:** The system automatically finds a match with available players.
- **Preconditions:**
 - The player is logged in.
 - The system has other players looking for a match.
- **Triggers:** The player selects “Quick Play” from the main menu.
- **Scenario:**
 1. The player selects “Quick Play”.
 2. The system searches for available players of similar rank.
 3. The system matches players into a new lobby.
 4. The players confirm readiness.
 5. The game starts.

- **Post Conditions:**
 - The player is placed into a match.
- **Exceptions:**
 1. If no opponents are found, the system offers to wait or play against AI.
 2. If matchmaking times out, the player is returned to the menu.
- **Priority:** High; essential for seamless multiplayer experience.
- **When Available:** Anytime after selecting a game.
- **Frequency of Use:** High.
- **Channel to Actor:** GUI (Matchmaking system).
- **Secondary Actors:** Other players in the matchmaking queue.
- **Open Issues:**
 - Should players be allowed to cancel matchmaking once started?
 - Should there be a ranked matchmaking option?

Use Case: In-Game Chat System

- **Use Case:** Sending and receiving chat messages during a game.
- **Iteration:** 1st iteration
- **Primary Actor:** Player
- **Goal in Context:** To enable real-time text-based communication between players during an active game session.
- **Preconditions:**
 - The player is inside a game session.

- The chat system is enabled for the game.
- **Triggers:** The player sends a message in the in-game chat.
- **Scenario:**
 1. The player types a message in the chat input box.
 2. The system sends the message to the game server.
 3. The message appears in the chat window for all players.
 4. The system timestamps and logs the chat message.
- **Post Conditions:**
 - The message is visible to all game participants.
 - The system records the message for moderation if needed.
- **Exceptions:**
 1. If the player sends a message that violates community guidelines, it is flagged or blocked.
 2. If there is a **network issue**, the message is queued and sent when reconnected.
 3. If **chat is disabled**, the player is notified that messaging is unavailable.
- **Priority:** Medium; enhances user experience but is not core to gameplay.
- **When Available:** During an active game session.
- **Frequency of Use:** Moderate; varies based on player interaction.
- **Channel to Actor:** Keyboard input, touchscreen.
- **Secondary Actors:** Other players in the game.
- **Open Issues:**
 - Should messages be stored permanently or disappear after a game ends?

- Should players be able to mute individual opponents?
-

Use Case: In-Lobby Chat

- **Use Case:** Sending and receiving messages in a game lobby before the match starts.
- **Iteration:** 1st iteration
- **Primary Actor:** Player
- **Goal in Context:** To enable real-time communication between players before a game begins.
- **Preconditions:**
 - The player is inside a game lobby.
 - Chat functionality is enabled for the lobby.
- **Triggers:** The player sends a message in the in-lobby chat.
- **Scenario:**
 1. The player types a message in the in-lobby chat box.
 2. The system sends the message to the game lobby server.
 3. The message appears in the chat window for all players in the lobby.
 4. Other players can respond in real time.
- **Post Conditions:**
 - The message is visible to all lobby participants.
 - Players can communicate before the match starts.
- **Exceptions:**

1. If the player sends a message that violates community guidelines, it is flagged or blocked.
 2. If there is a network issue, the message fails to send, and the system prompts a retry.
 3. If chat is disabled, the player is notified that messaging is unavailable.
- **Priority:** Medium; enhances social interaction.
 - **When Available:** Anytime while in a game lobby.
 - **Frequency of Use:** Moderate to high.
 - **Channel to Actor:** GUI (Chat interface).
 - **Secondary Actors:** Other players in the game lobby.
 - **Open Issues:**
 - Should players be able to send pre-set messages (e.g., "Ready to start!")?
 - Should in-lobby chat persist after the game starts, or should it clear?

Use Case: User Plays Checkers

- **Use Case:** Engaging in a game of Checkers.
- **Iteration:** 1st iteration
- **Primary Actor:** Player
- **Goal in Context:** The player competes in a Checkers game against an opponent to win or enjoy the match.
- **Preconditions:**
 - The player is signed in and has joined a Checkers game **lobby**.

- **Triggers:** The game starts after both players confirm readiness.
- **Scenario:**
 1. The game board appears on the screen with each player's pieces.
 2. The first player makes a move by selecting and moving a piece.
 3. The system validates the move and updates the board.
 4. The opponent takes their turn.
 5. The game continues until one player captures all opponent pieces or a draw is declared.
 6. The game ends, and the GUI displays the results.
- **Post Conditions:**
 - The game ends, and results are recorded.
 - Players can view their updated stats and ranking.
- **Exceptions:**
 1. If a player makes an invalid move, the system prompts a retry.
 2. If the opponent disconnects, the system offers an AI substitute or declares the remaining player the winner.
 3. If a network issue occurs, the system attempts reconnection.
- **Priority:** High; core game functionality.
- **When Available:** Any time after joining a Checkers game lobby.
- **Frequency of Use:** High; occurs whenever Checkers is played.
- **Channel to Actor:** GUI (Game interface).
- **Secondary Actors:** Opponent (another player).
- **Open Issues:**

- Should there be a turn timer (e.g., 30 seconds per move)?
 - Should there be an option to propose a draw?
-

Use Case: User Plays Connect Four

- **Use Case:** Engaging in a game of Connect Four.
- **Iteration:** 1st iteration
- **Primary Actor:** Player
- **Goal in Context:** The player competes in a Connect Four game to win by aligning four discs in a row.
- **Preconditions:**
 - The player is signed in and has joined a **Connect Four game lobby**.
- **Triggers:** The game starts after both players confirm readiness.
- **Scenario:**
 1. The Connect Four grid appears on the screen.
 2. The player drops a disc into one of the columns.
 3. The system validates the move and updates the board.
 4. The opponent takes their turn.
 5. The game continues until a player aligns four discs in a row, column, or diagonal.
 6. The game ends, and the GUI displays the results.
- **Post Conditions:**
 - The game ends, and results are recorded.

- Players can view their updated stats and ranking.
- **Exceptions:**
 1. If a column is full, the player is prompted to pick another.
 2. If the opponent disconnects, the system offers an AI substitute or declares the remaining player the winner.
 3. If a network issue occurs, the system attempts reconnection.
- **Priority:** High; core game functionality.
- **When Available:** Any time after joining a Connect Four game lobby.
- **Frequency of Use:** High; occurs whenever Connect Four is played.
- **Channel to Actor:** GUI (Game interface).
- **Secondary Actors:** Opponent (another player).
- **Open Issues:**
 - Should there be a turn timer?
 - Should there be an option to propose a draw?

Use Case: User Plays Tic-Tac-Toe

- **Use Case:** Engaging in a game of Tic-Tac-Toe.
- **Iteration:** 1st iteration
- **Primary Actor:** Player
- **Goal in Context:** The player competes in a Tic-Tac-Toe game to win by aligning three markers in a row.
- **Preconditions:**

- The player is signed in and has joined a Tic-Tac-Toe game lobby.
- **Triggers:** The game starts after both players confirm readiness.
- **Scenario:**
 1. The Tic-Tac-Toe grid appears on the screen.
 2. The player places an X or O in an available square.
 3. The system validates the move and updates the board.
 4. The **opponent** takes their turn.
 5. The game continues until a player aligns three markers or the board is full.
 6. The game ends, and the GUI displays the results.
- **Post Conditions:**
 - The game ends, and results are recorded.
 - Players can view their updated stats and ranking.
- **Exceptions:**
 1. If a square is already taken, the system prompts the player to choose another.
- **Priority:** High.
- **Open Issues:**
 - Should Tic-Tac-Toe have larger grid variations (e.g., 5x5)?

Use Case: Viewing Player Profile

- **Use Case:** Allowing a player to view their own or another player's profile.
- **Iteration:** 1st iteration

- **Primary Actor:** Player
- **Goal in Context:** The player can access and view profile details, including game stats, win/loss records, and ranking.
- **Preconditions:**
 - The player is logged into the 0mG platform.
 - The system has stored profile data.
- **Triggers:** The player selects “Profile” from the main menu or clicks on another player’s name.
- **Scenario:**
 1. The player navigates to their own or another player’s profile.
 2. The system retrieves and displays profile details such as username, avatar, ranking, and past game history.
 3. The player can scroll through past match results and statistics.
- **Post Conditions:**
 - The profile details are displayed successfully.
 - The player can return to the main menu.
- **Exceptions:**
 1. If profile data is unavailable, an error message is shown.
 2. If network issues prevent loading, the system displays an offline mode message.
- **Priority:** Medium; enhances player engagement.
- **When Available:** At any time after logging in.
- **Frequency of Use:** Moderate.

- **Channel to Actor:** GUI (Profile page).
 - **Secondary Actors:** None.
 - **Open Issues:**
 - Should players be able to customize their profiles (e.g., change avatar, bio)?
 - Should the profile page include social features like “Friend Requests” or “Follow”?
-

Use Case: User Searches for a Friend

- **Use Case:** Finding other players on the platform.
- **Iteration:** 1st iteration
- **Primary Actor:** Player
- **Goal in Context:** To allow players to find and challenge others or view their profiles.
- **Preconditions:**
 - The player is logged in.
 - The system has a player database.
- **Triggers:** The player enters a username in the search bar.
- **Scenario:**
 - The player navigates to the Search feature
 - 2. The system displays a search bar where the player enters a username.
 - 3. The system fetches matching results.

4. The player selects a user to view their profile or send an invite.
- **Post Conditions:**
 - The search results are displayed.
 - The player can interact with found users.
 - **Exceptions:**
 1. If no matches are found, an error message is displayed.
 2. If the search fails due to connectivity, the system prompts a retry.
 - **Priority:** Medium; improves social interaction.
 - **When Available:** After login.
 - **Frequency of Use:** Moderate.
 - **Channel to Actor:** GUI (Search interface).
 - **Secondary Actors:** Other players.
 - **Open Issues:**
 - Should there be filters (e.g., ranked players, friends only)?
 - Should players be able to block others?

Use Case: User Views Leaderboard

- **Use Case:** Displaying player rankings.
- **Iteration:** 1st iteration
- **Primary Actor:** Player
- **Goal in Context:** To allow players to see the top-ranked users and their win/loss records.

- **Preconditions:**
 - The player is logged into the platform.
 - The system has stored ranking data.
- **Triggers:** The player selects “Leaderboard” from the menu.
- **Scenario:**
 1. The player navigates to the Leaderboard menu.
 2. The system retrieves and displays a list of top-ranked players.
 3. The player can scroll, filter, or search for specific users.
 4. The player can click on a username to view their profile.
- **Post Conditions:**
 - The leaderboard is displayed successfully.
- **Exceptions:**
 1. If rankings fail to load, an error message is shown.
 2. If the player enters an invalid username in search, the system notifies them.
- **Priority:** Medium; enhances competition and engagement.
- **When Available:** Any time after login.
- **Frequency of Use:** Moderate to high.
- **Channel to Actor:** GUI (Leaderboard screen).
- **Secondary Actors:** None.
- **Open Issues:**
 - Should leaderboards have multiple categories (e.g., weekly, monthly, all-time)?

- Should leaderboard stats include recent match history?
-

Use Case: User Views Game Stats

- **Use Case:** Reviewing detailed **game-specific** statistics.
- **Iteration:** 1st iteration
- **Primary Actor:** Player
- **Goal in Context:** The player can track performance in different games.
- **Preconditions:**
 - The player is logged in.
 - The system has stored match history and performance data.
- **Triggers:** The player selects “Game Stats” from the profile menu.
- **Scenario:**
 1. The player navigates to the Profile page and selects Game Stats.
 2. The system displays statistics for each available game.
 3. The player can filter stats by game or timeframe.
- **Post Conditions:**
 - The player’s stats are displayed.
- **Exceptions:**
 1. If stats fail to load, an error message is displayed.
- **Priority:** Medium; useful for self-improvement and competition.
- **When Available:** Any time after login.
- **Frequency of Use:** Occasional.

- **Channel to Actor:** GUI (Profile screen, Game Stats section).
 - **Secondary Actors:** None.
 - **Open Issues:**
 - Should stats include historical trends (e.g., monthly rankings)?
 - Should there be a global ranking based on total performance?
-

Use Case: User Views Match History

- **Use Case:** Reviewing past game matches.
- **Iteration:** 1st iteration
- **Primary Actor:** Player
- **Goal in Context:** The player can track past performance and review opponents.
- **Preconditions:**
 - The player is logged in.
 - The system has stored match history.
- **Triggers:** The player selects “Match History” from the profile menu.
- **Scenario:**
 1. The player navigates to the Profile page and selects Match History.
 2. The system displays a list of past matches with dates, opponents, and results.
 3. The player can click on a match to view detailed results.
- **Post Conditions:**
 - The player’s match history is displayed.

- **Exceptions:**
 1. If no matches are found, a message is displayed: “No matches played yet”.
- **Priority:** Medium; valuable for competitive players.
- **When Available:** Any time after login.
- **Frequency of Use:** Occasional.
- **Channel to Actor:** GUI (Profile screen, Match History section).
- **Secondary Actors:** None.
- **Open Issues:**
 - Should match history include game replays or chat logs?
 - Should there be a filter by opponent or game mode?

Use Case: Changing Settings

- **Use Case:** Allowing a player to customize system settings.
- **Iteration:** 1st iteration
- **Primary Actor:** Player
- **Goal in Context:** The player can change UI preferences, sound settings, and other customizable features.
- **Preconditions:**
 - The player is logged into the platform.
 - The system has a settings module.
- **Triggers:** The player clicks “Settings” from the main menu.

- **Scenario:**
 1. The player selects “Settings” from the main menu.
 2. The system displays adjustable options such as sound volume, UI theme, and chat preferences.
 3. The player modifies settings and clicks “Save”.
 4. The system applies changes and confirms the update.
 - **Post Conditions:**
 - The settings are updated and stored.
 - The player’s preferences persist in future sessions.
 - **Exceptions:**
 1. If the settings fail to save, an error message is displayed.
 2. If the system crashes mid-change, settings revert to defaults.
 - **Priority:** Medium; important for customization.
 - **When Available:** Any time after login.
 - **Frequency of Use:** Low to moderate.
 - **Channel to Actor:** GUI (Settings menu).
 - **Secondary Actors:** None.
 - **Open Issues:**
 - Should players be able to reset settings to default?
 - Should there be advanced settings (e.g., colorblind mode, text-to-speech)?
-

Use Case: User Adjusts Settings

- **Use Case:** Customizing platform preferences, such as sound, notifications, and UI layout.
- **Iteration:** 1st iteration
- **Primary Actor:** Player
- **Goal in Context:** The player can modify settings to personalize the **gaming experience**.
- **Preconditions:**
 - The player is signed in.
 - The settings module is accessible.
- **Triggers:** The player selects “Settings” from the main menu.
- **Scenario:**
 1. The player selects the settings gear icon in the top-right corner of the main menu.
 2. The GUI displays sliders or toggles for:
 - Music volume
 - Sound effects
 - Notifications
 - Chat visibility
 3. The player adjusts the settings and clicks "Save".
 4. The GUI applies changes and **returns to the main menu**.
- **Post Conditions:**
 - The player’s preferences are updated and applied across the platform.
- **Exceptions:**

1. If the save operation fails, the GUI displays “Settings failed to save. Try again.”
 2. If there is a network issue, the system shows a “Connection lost. Retry?” notification.
- **Priority:** Medium; enhances user experience.
 - **When Available:** Any time after login.
 - **Frequency of Use:** Occasional.
 - **Channel to Actor:** GUI (Settings screen via main menu).
 - **Secondary Actors:** None.
 - **Open Issues:**
 - Should settings sync across devices?
 - Should there be a dark mode option?

Use Case: Exiting the Platform

- **Use Case:** Closing the 0mG platform.
- **Iteration:** 1st iteration
- **Primary Actor:** Player
- **Goal in Context:** The player closes the application safely.
- **Preconditions:**
 - The 0mG platform is open and running.
- **Triggers:** The player selects “Exit” from the menu or closes the app.
- **Scenario:**

1. The player clicks “Exit” in the menu.
2. The system asks for confirmation (optional).
3. The player confirms, and the app safely logs them out.

- **Post Conditions:**

- - The app closes successfully.
 - The player is logged out.

- **Exceptions:**

- 1. If the system crashes before closing, session data may be lost.
- 2. If the player is mid-game, the app warns about losing progress.

- **Priority:** Low; basic functionality.

- **When Available:** Always.

- **Frequency of Use:** Every session.

- **Channel to Actor:** GUI (Menu button).

- **Secondary Actors:** None.

- **Open Issues:**

- Should there be a “Save and Quit” option?
- Should players be warned before exiting mid-game?

Use Case: Receiving Notifications

- **Use Case:** Alerting players of game invites, friend requests, or match updates.
- **Iteration:** 1st iteration
- **Primary Actor:** Player

- **Goal in Context:** To ensure players are informed of important game-related events.
- **Preconditions:**
 - The player is logged in.
 - The system has notifications enabled.
- **Triggers:** The system sends a notification alert.
- **Scenario:**
 1. The player receives a notification for an event (e.g., friend request, game invite).
 2. The player clicks on the notification.
 3. The system redirects the player to the relevant page.
- **Post Conditions:**
 - The player is informed of the event.
- **Exceptions:**
 1. If the notification fails to load, an error message is displayed.
- **Priority:** Medium; enhances user engagement.
- **When Available:** Any time after login.
- **Frequency of Use:** High.
- **Channel to Actor:** GUI (Notification system).
- **Secondary Actors:** Other players.
- **Open Issues:**
 - Should players be able to mute notifications?
 - Should the system store past notifications for review?

Use Case: Game Session

- **Use Case:** Playing a game (e.g., Checkers) with an opponent on the 0mG platform.
- **Primary Actor:** Player (a user engaged in gameplay).
- **Goal:** The player participates in and completes a game session with an opponent.
- **Preconditions:** The player has signed in, selected a game, waited for an opponent, and entered the game session.
- **Triggers:** An opponent joins the lobby, and the system starts the game.
- **Main Scenario:**
 1. The player selects Checkers, waits for an opponent, and enters the game session.
 2. The system displays the game board with a chat option and turn indicator.
 3. The player makes moves (e.g., moving a checker piece) while the opponent responds.
 4. The game ends when a player wins (e.g., the player captures all opponent pieces).
 5. The system announces the winner and transitions to the game over screen.
- **Post Conditions:** The game ends, and the player sees the game over screen or returns to the main lobby.

- **Exceptions:** If the game server fails, the system shows an error: "Game connection lost."
 - **Priority:** High; core to the gaming experience.
 - **Frequency of Use:** High; occurs during every multiplayer game.
 - **Channel to Actor:** GUI (game board with chat and controls).
 - **Secondary Actors:** Other Players (the opponent)
 - **Open Issues:** Should there be a save option for unfinished games?
-

Use Case Description: Game Over (Returns to Main Lobby)

- **Use Case:** Concluding a game session and returning the player to the main lobby on the 0mG platform.
- **Primary Actor:** Player (a user who has finished a game).
- **Goal:** The player views the game result and returns to the main lobby to start a new session.
- **Preconditions:** The player has completed a game session (win, loss, draw, or quit).
- **Triggers:** The game ends, triggering the game over screen.
- **Main Scenario:**
 1. The player wins a Checkers game, and the system displays the game over screen with "You Win!" and a "Return to Lobby" button.

2. The player clicks "Return to Lobby."
 3. The system updates the player's stats (e.g., wins) and shows the main lobby with game selection options.
- **Alternate Scenarios:**
 - **Loss or Draw:** If the player loses or the game ends in a draw, the screen shows "You Lose!" or "Draw!" with the same return option.
 - **Player Skips Screen:** The player can click "Return to Lobby" immediately, bypassing stats review.
 - **Network Issue:** If the connection fails, the system returns to the lobby with an error: "Stats not saved due to connection loss."
 - **Post Conditions:** The player is back in the main lobby, ready to select a new game.
 - **Exceptions:** If the lobby fails to load, the system shows an error: "Cannot return to lobby."
 - **Priority:** Medium; ensures smooth transition after gameplay.
 - **Frequency of Use:** High; occurs after every game.
 - **Channel to Actor:** GUI (game over screen with button).
 - **Secondary Actors:** None.
 - **Open Issues:** Should the game over screen include a replay option?
-

Use Case Diagram: GUI

