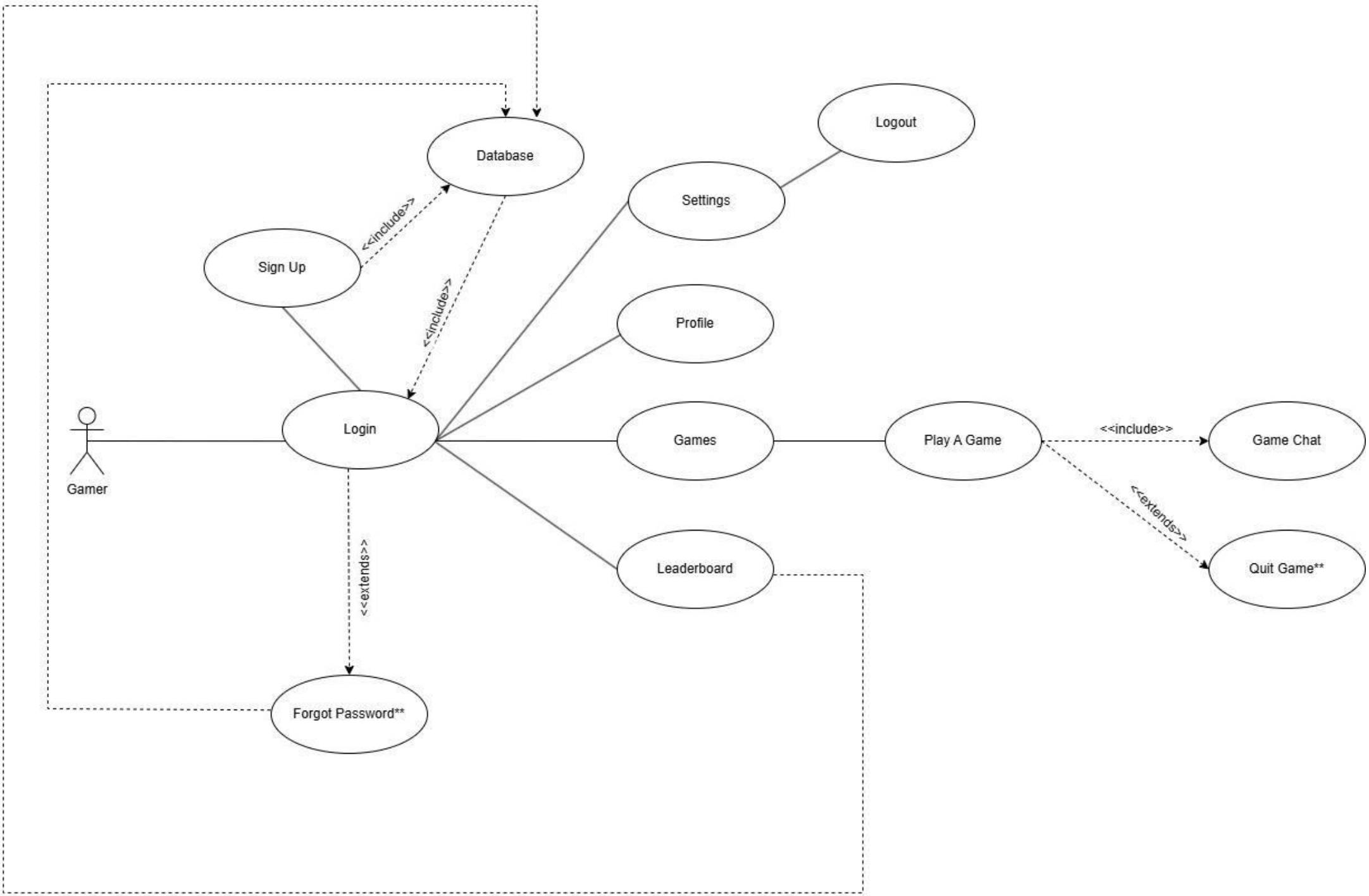
# USE CASE DIAGRAM - GUI



**Use Case:** Login

**Iteration:** 1

**Primary Actor:** Existing User

**Goal in Context:** Allow the existing user to authenticate and gain access to the system's features.

**Preconditions:**

1. The system is running and accessible.
2. The user has an existing account.

**Trigger:**

The user attempts to access the system's features.

**Scenario:**

1. The user enters their username and password.
2. The system verifies the credentials against the database.
3. If the credentials are valid, the user is taken to the main menu.
4. If the credentials are invalid, the system displays an error message and prompts the user to try again.

**Post Conditions:**

The user is logged in and can access the system's functionalities, or, if the credentials were invalid, an error message is displayed and the user remains logged out.

**Exceptions:**

1. The user enters an incorrect username or password, prompting an error message and a retry request.
2. The system displays a message indicating that login is unavailable.
3. The user's account does not exist; the system prompts the user to try again, and after successive failures, suggests signing up instead.
4. A system timeout or other technical issues occur.

**Priority:** High – Logging in is necessary to access other system functionalities.

**When Available:** Within the first development iteration.

**Frequency of Use:** Once per session (or as needed if the session times out).

**Channel to Actor**: The user interacts with the system via the GUI using a keyboard and mouse/touchscreen.

**Secondary Actors**: The database (for verifying credentials).

**Channel to Secondary Actors**: The system interacts with the database via database requests.

**Open Issues**:

1. Should there be a password recovery option?
2. Should multi-factor authentication be implemented?

**Use Case**: Sign Up

**Iteration:** 1

**Primary Actor:** New User

**Goal in Context:** Allow a new user to create an account.

**Preconditions:**

1. The system is powered on and accessible.
2. The user is on the sign-in page or has chosen to create a new account.

**Trigger:**

The user selects the "Sign Up" option.

**Scenario:**

1. The user selects the "Sign Up" option.
2. The system presents a registration form with fields for username/email, password, and other required information.
3. The user fills in the form.
4. The system validates the entered information (e.g., checks for a unique username and password strength).
5. If the information is valid, the system creates a new user account and stores it in the database.
6. The user is prompted to log in with their new credentials.

**Post Conditions:**

A new user account is created, and the user can now log in with the newly created credentials.

**Exceptions:**

1. The username or email already exists.
2. The password does not meet the requirements.
3. There is an error or issue connecting to the database.
4. A system timeout or other technical issues occur.

**Priority:** High – Sign-up is essential for growing the user base and providing access to new users.

**When Available:** Within the first development iteration.

**Frequency of Use:** Once per new user.

**Channel to Actor:** The user interacts with the system via a graphical user interface (GUI) using a keyboard and mouse/touchscreen.

**Secondary Actors:** The database (for storing user accounts).

**Channel to Secondary Actors:** The system interacts with the database via database requests.

**Open Issues:**

1. Should there be an email verification process?
2. Should a CAPTCHA be included to prevent bots?

**Use Case:** Forgot Password

**Iteration:** 1

**Primary Actor:** Existing User

**Goal in Context:** Allow the existing user to reset their password.

**Preconditions:**

1. The system is powered on and accessible.
2. The user has an existing account linked to a valid email address.

**Trigger:**

The user clicks on the "Forgot Password" link on the login screen.

**Scenario**:

1. The user selects the "Forgot Password?" option.
2. The system displays a password recovery form requesting the registered email address (or username).
3. The user enters their email address and submits the form.
4. The system verifies whether the email is associated with an account.
5. If the email is valid, the system sends a password reset link to the user's email via the email subsystem stub.
6. The system displays a confirmation message indicating that reset instructions have been sent.

**Post Conditions**:

The user receives an email containing a password reset link with instructions for creating a new password.

**Exceptions**:

1. If the entered email is not linked to any account, the system displays an error message prompting the user to check their input.

2. If the email cannot be sent due to network or system issues, the system notifies the user and suggests retrying later.

**Priority:** Medium – Essential for account recovery and maintaining user access, though not a core gameplay function.

**When Available**: Within the first development iteration.

**Frequency of Use**: Infrequent – only used when a user forgets their password.

**Channel to Actor**: The user interacts with the system via a graphical user interface (GUI) using a keyboard and mouse/touchscreen.

**Secondary Actors**: The email subsystem (stubbed to simulate sending the reset instructions).

Channel to Secondary Actors: N/A

**Open Issues:**

1. Should there be a validity period for the reset link?
2. Should additional security measures (e.g., security questions) be implemented before allowing a password reset?

**Use Case**: Games

**Iteration**: 1

**Primary Actor**: User

**Goal in Context**: Allow the user to view all available games.

**Preconditions**:

The user is successfully logged in.

**Trigger:**

The user logs in successfully.

**Scenario:**

1. The user logs in successfully.
2. The system displays a list of available games.

3. The user browses the list and selects a game to join or test.

**Post Conditions**:

The user selects a game and can play it.

**Exceptions:**

1. The user is unable to log in.
2. There are network connection issues.
3. A system timeout or other technical issues prevent the start of the gaming session.

**Priority**: High – Viewing the game library is essential for providing gameplay.

**When Available**: Within the first iteration.

**Frequency of Use**: Multiple times per user, depending on their engagement.

**Channel to Actor**: The user interacts with the system via a graphical user interface (GUI) using a keyboard and mouse/touchscreen.

**Secondary Actors**: Game Server (for managing game sessions and matchmaking).

Channel to Secondary Actors: The system interacts with the game server via network communication.

**Open Issues**: N/A

**Use Case**: Play A Game

**Iteration**: 1

**Primary Actor**: User

**Goal in Context**: Allow the user to join a game session.

**Preconditions**:

The user is logged in and has selected a game to play.

**Trigger**:

The user selects a game to play.

**Scenario**:

1. The user selects the game to play.
2. The system adds the user to the selected game session.
3. The system matches the player with someone of a similar skill level.
4. The user is taken to the game interface.

**Post Conditions**:

The user is added to the selected game session and can play the game.

**Exceptions**:

1. No active game sessions are available.
2. The selected game session is full.
3. There are network connection issues.
4. A system timeout or other technical issues prevent the start of the gaming session.

**Priority**: High – Joining a game is a core functionality for multiplayer interaction.

**When Available**: Within the first iteration.

**Frequency of Use**: Multiple times per user, depending on their engagement.

**Channel to Actor**: The user interacts with the system via a graphical user interface (GUI) using a keyboard and mouse/touchscreen.

**Secondary Actors**: Game Server (for managing game sessions and matchmaking).

**Channel to Secondary Actors**: The system interacts with the game server via network communication.

**Open Issues**:

1. Should there be requirements (e.g., rank, invite-only) to join certain games?
2. What happens if the game starts while the user is joining?

**Use Case**: Game Chat

**Iteration**: 1

**Primary Actor**: User

**Goal in Context**: Allow users to communicate with each other during a game session.

**Preconditions**:

The user is logged in and has successfully joined a game session.

**Trigger**:

The user wants to send a message to other players.

**Scenario**:

1. The user selects a game to play from the library.
2. The user successfully joins a game session.
3. The user accesses the chat interface.
4. The user types a message.
5. The user sends the message to their opponent(s).
6. The message is displayed to all players in the game session.

**Post Conditions**:

The message is sent and displayed to the other players.

**Exceptions**:

1. Network connection issues prevent the message from being sent.
2. A system timeout or other technical issues occur.

**Priority**: Medium – Game chat enhances social interaction and communication during gameplay, but it

is not required for gameplay.

**When Available**: Within the first iteration.

**Frequency of Use**: Multiple times per user during a game session.

**Channel to Actor**: The user interacts with the system via a graphical user interface (GUI) using a keyboard and mouse/touchscreen.

**Secondary Actors**: The game server (for relaying chat messages).

**Channel to Secondary Actors**: The system interacts with the game server via network communication.

**Open Issues**:

1. Should private messages or voice chat be implemented in future iterations?
2. Should there be filters for inappropriate or offensive words and terms?

**Use Case**: Quit Game

**Iteration**: 1

**Primary Actor**: User

**Goal in Context**: Allow users to exit the game application.

**Preconditions**:

1. The system is running.

**Trigger**:

The user selects "Quit Game."

**Scenario:**

1. The user selects "Quit Game" from the menu.
2. The system prompts for confirmation.
3. The user confirms, and the system closes the application.

**Post Conditions**:

The application exits successfully.

**Exceptions**:

1. The system fails to close due to an error.
2. The user cancels the quit action before confirming.

**Priority**: High – Essential for user control.

**When Available**: Within the first development iteration.

**Frequency of Use**: Occasionally.

**Channel to Actor**: The user interacts with the GUI via a keyboard and mouse/touchscreen.

**Secondary Actors**: None.

**Channel to Secondary Actors**: None.

**Open Issues**:

1. Should there be a warning if the user quits with unsaved progress?

**Use Case**: Leaderboard

**Iteration**: 1

**Primary Actor**: User

**Goal in Context**: Allow the user to view the leaderboard and see player rankings.

**Preconditions**:

The user is successfully logged in.

**Trigger**:

The user clicks on the "Leaderboard" button.

**Scenario**:

1. The user is logged in.
2. The user clicks on the "Leaderboard" button.
3. The system retrieves the leaderboard data from the database.
4. The system displays the leaderboard with player rankings.

**Post Conditions**:

The user is able to view the leaderboard with up-to-date rankings.

**Exceptions**:

1. The leaderboard data is not available due to a database connection error, a database error, or a system timeout.

**Priority**: Medium – Viewing the leaderboard can enhance competition and engagement but is not mandatory for gameplay.

**Frequency of Use**: Multiple times per user, depending on their interest.

**Channel to Actor**: The user interacts with the system via a graphical user interface (GUI) using a keyboard and mouse/touchscreen.

**Secondary Actors**: The database (for player ranking information).

**Channel to Secondary Actors**: The system interacts with the database via database requests.

**Open Issues**:

1. Should there be filtering or sorting options for different leaderboard categories?

**Use Case**: Profile

**Iteration**: 1

**Primary Actor**: User

**Goal in Context**: Allow users to view and update their profile information.

**Preconditions**:

1. The user is logged in.

**Trigger**:

The user selects "Profile" from the main menu.

**Scenario**:

1. The user selects "Profile."
2. The system retrieves profile data from the database.
3. The system displays profile details (e.g., username, avatar, game stats).
4. The user updates the information if desired.
5. The system saves the changes.

**Post Conditions**:

The user's profile is updated successfully.

**Exceptions**:

1. The system cannot retrieve or save profile data due to technical issues.
2. The user enters invalid data (e.g., a username that is already taken).

**Priority:** Medium – Enhances user experience.

**When Available**: Within the first development iteration.

**Frequency of Use**: Occasionally.

**Channel to Actor**: The user interacts with the system via a GUI using a keyboard and mouse/touchscreen.

**Secondary Actors**: The database (to store user preferences).

**Channel to Secondary Actors**: The system interacts with the database via queries.

**Open Issues:**

1. Should users be able to delete their profile?
2. Should profile updates require verification?

**Use Case**: Settings

**Iteration**: 1

**Primary Actor**: User

**Goal in Context**: Allow users to configure system and game preferences.

**Preconditions**:

1. The user is logged in.

**Trigger**:

The user selects "Settings" from the main menu.

**Scenario**:

1. The user selects "Settings."
2. The system displays available options (audio, controls, graphics, notifications).
3. The user modifies the settings and confirms the changes.
4. The system saves the new settings.

**Post Conditions**:

The new settings are applied successfully.

**Exceptions**:

1. The system fails to save changes due to a technical issue.
2. The user cancels the changes before saving.

**Priority**: Medium – Enhances user experience.

**When Available**: Within the first development iteration.

**Frequency of Use**: Occasionally.

**Channel to Actor**: The user interacts with the GUI via a keyboard and mouse/touchscreen.

**Secondary Actors**: The database (to store user preferences).

**Channel to Secondary Actors**: The system interacts with the database via queries.

**Open Issues**:

1. Should settings be stored locally or in the cloud?

2. Should there be a reset-to-default option?

**Use Case**: Logout

**Iteration:** 1

**Primary Actor**: User

**Goal in Context**: Allow users to securely end their session.

**Preconditions**:

1. The user is logged in.

**Trigger**:

The user selects "Logout."

**Scenario**:

1. The user selects "Logout."
2. The system asks for confirmation.
3. The user confirms the logout.
4. The system ends the session and redirects the user to the login screen.

**Post Conditions**:

The user is logged out and must log in again to access features.

**Exceptions**:

1. The system fails to end the session due to a technical issue.

**Priority:** High – Ensures security.

**When Available**: Within the first development iteration.

**Frequency of Use**: Once per session.

**Channel to Actor**: The user interacts with the GUI via a keyboard and mouse/touchscreen.

**Secondary Actors**: The database (to update session status).

**Channel to Secondary Actors**: The system interacts with the database via queries.

**Open Issues**:

1. Should there be an option to stay logged in?