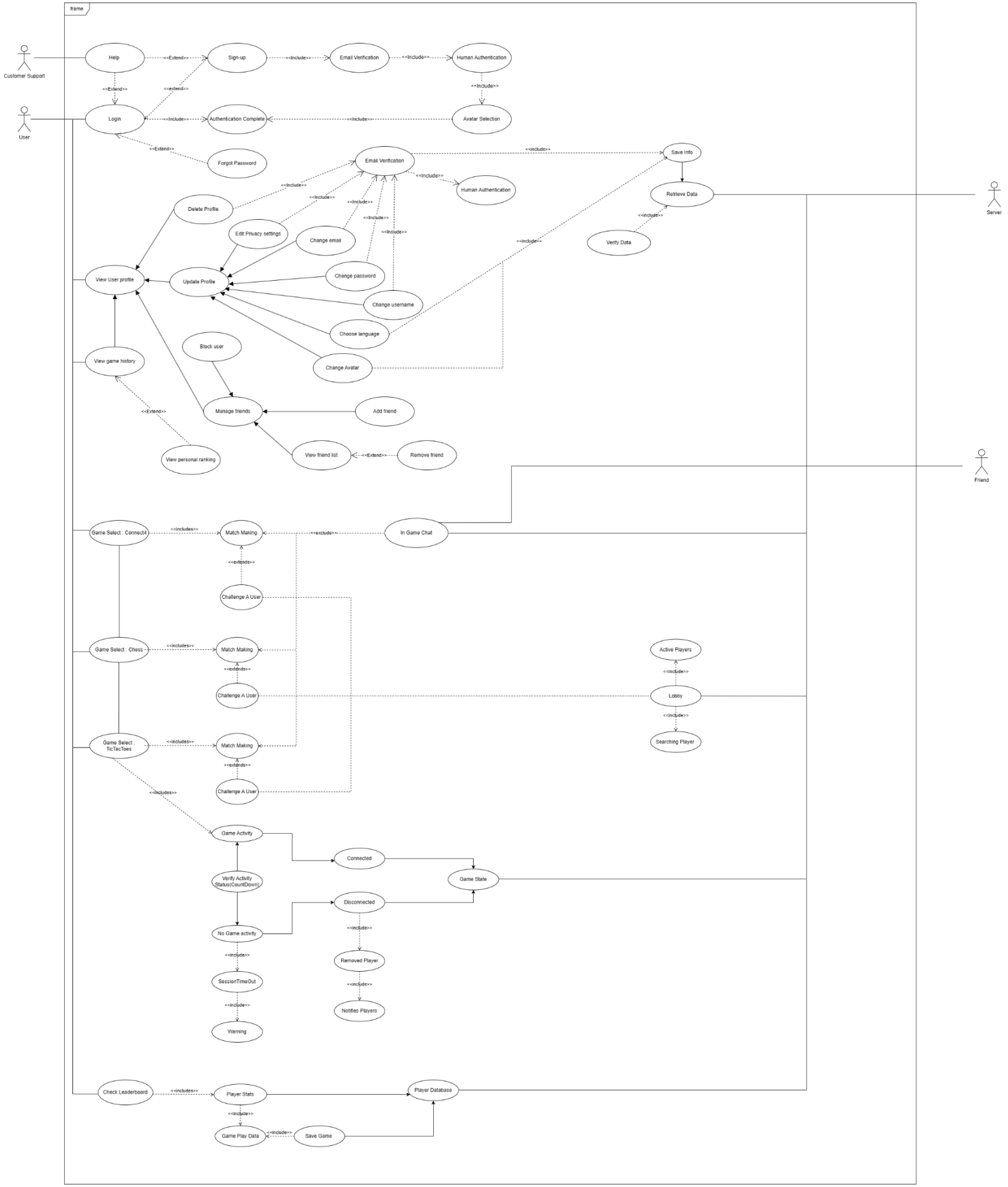
# SENG 300 – Project Plan

## Table of Contents


frame
Customer Support
User
Server
Friend
Help
Sign-up
Email Verification
Human Authentication
Login
Authentication Complete
Avatar Selection
Forgot Password
Save Info
Retrieve Data
Verify Data
Delete Profile
Edit Privacy settings
Change email
View User profile
Update Profile
Change password
Change username
Choose language
Change Avatar
Block user
View game history
Manage friends
Add friend
View friend list
Remove friend
View personal ranking
Game Select : Connect4
Match Making
In Game Chat
Challenge A User
Game Select : Chess
Game Select : TicTacToes
Active Players
Lobby
Searching Player
Game Activity
Connected
Disconnected
Game State
Verify Activity Status(CountDown)
No Game activity
Removed Player
SessionTimeOut
Notifies Players
Warning
Check Leaderboard
Player Stats
Player Database
Game Play Data
Save Game
<<Extend>>
<<Include>>
<<includes>>
<<extends>>
<<exclude>>

# Authentication & Profile Use Cases

## Sign Up

Iteration: 1, last modification: March 05, 2025

| Description | **Creating a new user account** |
|---|---|
| **Primary Actor** | New User |
| **Goal in Context** | Allow the user to successfully register an account to access the platform (OMG). |
| **Preconditions** | 1. The user has access to the platform<br>2. The user chooses to sign up for an account. |
| **Trigger** | User selects the “Sign Up” option. |
| **Scenario** | 1. User enters required details – username, email, password, confirm password, date of birth, and completes “Are you human?” verification.<br>2. The system validates all inputs (unique username, valid email, strong password, matching passwords, correct verification).<br>3. If valid, the system creates the account and sends a verification email.<br>4. Once email is confirmed, the user can login to their new account. |
| **Exceptions** | 1. The input is invalid (duplicate username, invalid email, weak password, mismatched password, failed verification).<br>2. Verification email is not received by the user or expired verification email needs to be resent. |
| **Priority** | High priority – Signing up for an account is the first step in the process and users cannot access the platform without an account. |
| **Frequency of Use** | Frequent (every time a new user joins the platform) |
| **Channel to Actor** | Online Multiplayer board Game platform (OMG), text fields, buttons. |
| **Secondary Actors** | 1. Database (validation, account creation)<br>2. Email service provider (verification email) |
| **Open Issues** | 1. Can AI/bots surpass the human verification system?<br>2. How secure are the personal details such as email and date of birth?<br>3. How would the system deal with temporary emails and identify them? |

# Login

Iteration: 1, last modification: March 05, 2025

| Description | **Authorizing an existing user to access their account.** |
|---|---|
| **Primary Actor** | Existing User |
| **Goal in Context** | Allow the user to login to the platform (OMG). |
| **Preconditions** | 1. The user has an existing account on the platform.<br>2. The user has access to the platform and login page. |
| **Trigger** | User selects the “Login” option. |
| **Scenario** | 1. User enters username and password.<br>2. User may select “Stay Logged In” option.<br>3. The system validates credentials by referring to the database.<br>4. If valid, the user is permitted to access their account. |
| **Exceptions** | 1. The credentials do not match, thereby prompting an error message.<br>2. If the user does not remember their password, they can reset it using the “Forgot Password” option and this would allow them to reset it using an email. |
| **Priority** | High priority – Users must be able to log in to access their accounts and use the platform. |
| **Frequency of Use** | Very Frequent (Users log in regularly to access their accounts) |
| **Channel to Actor** | Online Multiplayer board Game platform (OMG), text fields, buttons. |
| **Secondary Actors** | 1. Database (credential validation)<br>2. Email service provider (password reset email) |
| **Open Issues** | 1. Would the “Forgot Password” verification email have an expiry time?<br>2. What would the user do if they forgot their username?<br>3. If they are allowed to stay logged in to their account, at what point would the system automatically log them out for additional security measures? |

# Update Profile

Iteration: 1, last modification: March 03, 2025

| | |
|---|---|
| **Description** | **Allows the user to modify their profile details, including privacy settings, and avatar when necessary** |
| **Primary Actor** | Existing player |
| **Goal in Context** | Let the user edit personal details (privacy settings, avatar, etc.) so they can keep their profile information up-to-date and control what others see. |
| **Preconditions** | 1. User is authenticated (via **Human Authentication**).<br>2. The system has the user's existing profile data available. |
| **Trigger** | The user selects "Update Profile" (e.g., a button or menu option) from their profile page or account settings. |
| **Scenario** | 1. The system retrieves and displays the user's current profile information.<br>2. The user modifies any fields they want to change, such as name, location, or avatar (**includes** "Change Avatar").<br>3. The user adjusts privacy settings if desired (**includes** "Edit Privacy settings").<br>4. If the user changes their email address, the system **extends** "Email Verification" to confirm the new email.<br>5. The user confirms or saves their changes.<br>6. The system updates the user's profile in the database and returns them to the updated profile view. |
| **Exceptions** | 1. **Invalid input or network error** – The system rejects invalid data (e.g., an improperly formatted email) or fails to save due to a connectivity issue, prompting the user to correct or retry.<br>2. **Email verification failure** – If the user does not confirm the new email address, the system reverts to the old email. |
| **Priority** | High – Maintaining an accurate and secure profile is critical for user satisfaction. |
| **Frequency of Use** | Occasional – Typically only when users need to change their details or update privacy preferences. |
| **Channel to Actor** | Web or mobile application interface (e.g., a settings page). |
| **Secondary Actors** | 1. **Email Verification** (triggered only if the user changes their email).<br>2. **Database** (to store and retrieve updated profile data) |

| | |
|---|---|
| **Open Issues** | 1. How to synchronize updated profile details with other modules (e.g., Leaderboard, Matchmaking).<br>2. Whether to require re-authentication for sensitive changes (e.g., changing email). |

# Delete Profile

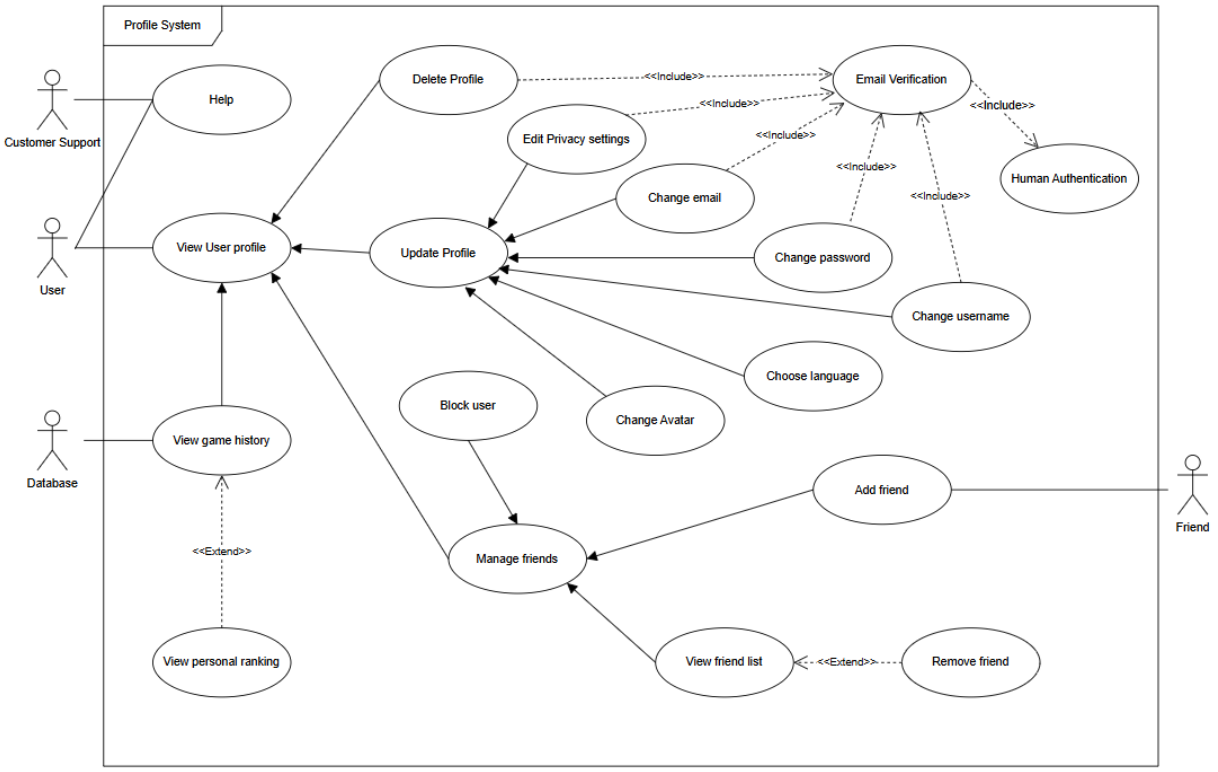
Iteration: 1, last modification: March 04, 2025

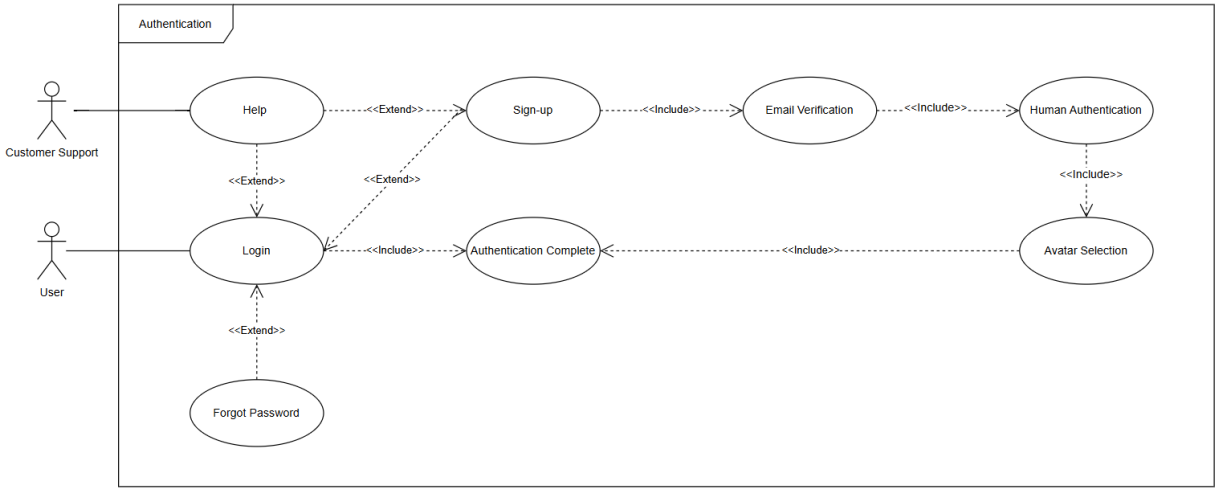
| Description | **Permanently remove the user's account and all associated data.** |
|---|---|
| **Primary Actor** | User |
| **Goal in Context** | Permanently remove the user's account and all associated data. |
| **Preconditions** | 1. User is authenticated and intentionally chooses to delete their account. |
| **Trigger** | The user selects "Delete Profile" from account settings. |
| **Scenario** | 1. System prompts for confirmation.<br>2. User confirms deletion.<br>3. System deletes or deactivates the account data.<br>4. User receives a success message and log out automatically. |
| **Exceptions** | 1. If system fails to delete data, user is notified and can retry.<br>2. If user changes mind, they can cancel the operation. |
| **Priority** | Medium to High |
| **Frequency of Use** | Rare (only if user wants to leave the platform) |
| **Channel to Actor** | Web or mobile application interface (account settings page) |
| **Secondary Actors** | 1. Data Retention Policies: Partial anonymization vs full deletion<br>2. Grace Period: Whether to offer a grace period before final deletion.<br>3. Linked Data: Handling game history, friend lists, or other references to the user's account.<br>4. Linked Data: Handling game history, friend lists, or other references to the user's account. |
| **Open Issues** | 1. Data Retention Policies: Partial anonymization vs full deletion.<br>2. Grace Period: Whether to offer a grace period before final deletion.<br>3. Linked Data: Handling game history, friend lists, or other references to the user's account. |

# Add or Remove Friends

Iteration: 1, last modification: March 07, 2025

| | |
|---|---|
| **Description** | **Allows a user to manage their friend list by adding new friend(s) or removing existing one(s)** |
| **Primary Actor** | User |
| **Goal in Context** | Enable the user to maintain a list of friends for easy assess to challenges or direct communication. |
| **Preconditions** | 1. User is authenticated.<br>2. System can search or retrieve other users' profiles. |
| **Trigger** | The user selects "add friend" or "remove friend" from their friend list. |
| **Scenario** | 1. User search for selects a friend's profile.<br>2. User clicks "add friend" or "remove friend"<br>3. System updates user's friend list in the database<br>4. System displays update status<br>5. message (success or error) |
| **Exceptions** | 1. If the target user is not found, the system displays an error |
| **Priority** | Medium |
| **Frequency of Use** | Occasional – depends on how often users add or remove connections |
| **Channel to Actor** | Web or Mobile interface (friend list) |
| **Secondary Actors** | 1. Friend Managing System which handles friend relationship in the database |
| **Open Issues** | 1. Handling large friend lists efficiently<br>2. Real – time updates of friend status<br>3. Potential – Privacy settings (user disables friend requests) |


Profile System
Customer Support
Help
Delete Profile
Email Verification
Edit Privacy settings
Human Authentication
Change email
User
View User profile
Update Profile
Change password
Change username
Choose language
Block user
Change Avatar
Database
View game history
Add friend
Friend
Manage friends
<<Extend>>
View personal ranking
View friend list
Remove friend
<<Include>>


Authentication
Customer Support
Help
Sign-up
Email Verification
Human Authentication
Login
Authentication Complete
Avatar Selection
User
Forgot Password
<<Extend>>
<<Include>>

# Game Logic Use Cases

## Move A Piece

Iteration: 2, last modification: March 07, 2025

| **Description** | **Move a Piece** |
|---|---|
| **Primary Actor** | Player. |
| **Goal in Context** | Move a valid piece to a valid position or capture an opponent's piece while enforcing chess rules, updating the board, handling captures, and checking for win/draw conditions. |
| **Preconditions** | 1. The game is in progress with a properly initialized board.<br>2. It is the current player's turn.<br>3. The selected piece belongs to the player. |
| **Trigger** | A player selects a piece and attempts to move it |
| **Scenario** | 1. The player selects a piece (an object instance).<br>2. The system verifies that the piece belongs to the player.<br>3. The piece object retrieves its possible legal moves based on its move attribute and current position.<br>4. The player selects a target position.<br>5. The system checks if the target position is within the valid moves.<br>• *Iteration:* The piece object iterates through its movement rules and interacts with the board state.<br>6. If the move is invalid, the piece object blocks it, and an error message is displayed.<br>7. If the move is valid:<br>• The piece updates its position attribute.<br>• If an opponent's piece is on the target square, it is captured and removed from play.<br>• The move is recorded in the move history.<br>**Iteration Details.** |

| | |
|---|---|
| | Each piece iterates through its own legal moves based on its *move* attribute.<br><br>The board verifies interactions (capturing, check conditions). |
| **Exceptions** | 1. **Invalid Move Attempt:** The move is outside the legal range; it is blocked and an error is shown.<br>2. **Turn Violation:** A player attempts to move when it's not their turn; the move is blocked.<br>3. **Checkmate/Stalemate Detection:** If a game-ending condition is met, the system declares the result and ends the game. |
| **Priority** | High – Fundamental game functionality. |
| **Frequency of Use** | Every turn. |
| **Channel to Actor** | Internal game logic, triggered by the player. |
| **Secondary Actors** | 1. **Game Engine:** Validates moves and enforces rules. |
| **Open Issues** | 1. Should move history allow undo/redo actions?<br>2. Should move validation be optimized for efficiency in larger chess variants? |

# Move Validation

Iteration: 2, last modification: March 07, 2025

| **Description** | **Move validation to primary actor.** |
|---|---|
| **Primary Actor** | Game engine. |
| **Goal in Context** | Ensure a player’s move follows all chess rules, including movement restrictions, check prevention, and special move conditions. |
| **Preconditions** | 1. A player has selected a piece and attempted a move.<br>2. The game is active.<br>3. The board is in a valid state. |
| **Trigger** | A move is attempted by the player. |
| **Scenario** | 1. The system receives a move request from the player.<br>2. The piece object checks if the target position is within its valid move set based on its movement rules.<br>3. The board verifies if the path to the target position is clear (except for knight moves).<br>4. The system ensures the move does not place or leave the player's king in check.<br>5. If the move is a special move (e.g., castling, en passant, pawn promotion), additional conditions are checked.<br>6. If the move is invalid, an error message is displayed, and the move is rejected.<br>7. If the move is valid, it proceeds to execution.<br><br>Iteration Details:<br>1. The system iterates through piece movement rules.<br>2. The system loops through board positions to check for obstructions and threats. |
| **Exceptions** | 3. **Blocked Path:** A piece attempts to move through another piece (except knights).<br>4. **Self-Check:** A move would leave the player's own king in check. |

| | |
|---|---|
| | 5. **Illegal Move:** The move does not conform to the piece's movement rules. |
| **Priority** | Critical for enforcing chess rules. |
| **Frequency of Use** | Every move attempt. |
| **Channel to Actor** | n/a |
| **Secondary Actors** | n/a |
| **Open Issues** | 1. Should there be an option to highlight all valid moves before selection |

# Capture Piece

Iteration: 2, last modification: March 07, 2025

| | |
|---|---|
| **Description** | **Allows a player to capture a piece.** |
| **Primary Actor** | Player |
| **Goal in Context** | Allow a player to capture an opponent's piece following chess rules and update the board accordingly. |
| **Preconditions** | 1. A valid move results in landing on a square occupied by an opponent's piece.<br>2. The game is active.<br>3. The move follows all legal movement rules. |
| **Trigger** | A player attempts to move a piece onto a square occupied by an opponent's piece. |
| **Scenario** | 1. he player selects a piece and a target square.<br>2. The system checks if the move is valid, and the target square contains an opponent's piece.<br>3. If valid, the opponent's piece is removed from the board.<br>4. The capturing piece moves to the target square.<br>5. The move is recorded in the game history.<br>6. The system checks if the move results in check, checkmate, or stalemate.<br>7. The turn switches to the opponent. |
| **Exceptions** | 1. Invalid Capture: The piece cannot capture due to movement restrictions<br>2. Self-Capture: A player attempts to capture their own piece; the move is rejected.<br>3. Illegal Move: The move does not comply with chess rules (e.g., capturing in a way not allowed for the piece). |
| **Priority** | Fundamental for game progression. |
| **Frequency of Use** | Occurs multiple times per game whenever a capture is possible. |
| **Channel to Actor** | Internal game logic. |
| **Secondary Actors** | Game Engine: Validates the move and ensures compliance with rules. |
| **Open Issues** | 1. when should en passant be handled within this use case? |

# Declare Winner

Iteration: 2, last modification: March 07, 2025

| **Description** | **Declare the winner of a game.** |
|---|---|
| **Primary Actor** | Game Engine |
| **Goal in Context** | Declare a winner when a player achieves checkmate or the opponent resigns, updating the game state and ending the match. |
| **Preconditions** | 1. The game is in progress.<br>2. A player has delivered checkmate to the opponent's king, or the opponent has resigned |
| **Trigger** | • A checkmate condition is detected.<br>• A player resigns. |
| **Scenario** | 1. The system detects a checkmate condition or receives a resignation action from a player.<br>2. The system verifies that the checkmate condition is valid (i.e., the opponent has no legal moves while in check).<br>3. The game engine determines the winner based on the current state.<br>4. The system updates the game state to "game over" and assigns victory to the winning player.<br>5. A message is displayed to both players declaring the winner.<br>6. The match result is recorded in the game history. |
| **Exceptions** | 1. **False Checkmate Detection:** If the system incorrectly detects checkmate due to an error in move validation, it must revert the decision.<br>2. **Resignation Error:** If a player accidentally resigns, there may be an option for confirmation before finalizing the result. |
| **Priority** | Required for completing a match. |
| **Frequency of Use** | Once per game, when a winning condition is met. |
| **Channel to Actor** | Internal game logic, displayed to players. |
| **Secondary Actors** | 1. Player: The losing player either resigns or is checkmated. |
| **Open Issues** | 1. Should the game allow a review of the final position after declaring a winner? |

# Declare Draw

Iteration: 2, last modification: March 07, 2025

| **Description** | **Declare Draw Primary Actor** |
|---|---|
| **Primary Actor** | Game Engine |
| **Goal in Context** | End the game as a draw when a stalemate, threefold repetition, fifty-move rule, or mutual agreement occurs. |
| **Preconditions** | 1. The game is in progress.<br>2. A draw condition is met (stalemate, insufficient material, threefold repetition, fifty- move rule, or both players agree). |
| **Trigger** | • A player offers a draw, and the opponent accepts.<br>• A stalemate is detected.<br>• A threefold repetition occurs.<br>• The fifty-move rule is met.<br>• Insufficient material is detected (e.g., only kings remain). |
| **Scenario** | 1. The system detects a draw condition or processes a draw offer.<br>2. If a draw offer is made, the opponent can accept or decline:<br>  • If accepted, the game ends in a draw.<br>  • If declined, the game continues.<br>3. If any of the following draw conditions are met, the system automatically declares a draw:<br>  • **Stalemate** (the current player has no legal moves, but their king is **not in check**).<br>  • **Threefold repetition** (the same board position occurs three times with the same possible moves).<br>  • **Fifty-move rule** (no pawn moves or captures for 50 consecutive moves).<br>  • **Insufficient material** (neither player has enough pieces to checkmate, e.g., king vs. king).<br>4. The game state is updated to "game over" with a draw result.<br>5. A message is displayed to both players declaring the draw.<br>6. The match result is recorded in the game history. |

| | |
|---|---|
| **Exceptions** | 1. **Incorrect Stalemate Detection:** If the system mistakenly identifies stalemate, the game should continue.<br>2. **Draw Offer Disagreement:** If a player offers a draw but the opponent declines, the game must proceed normally. |
| **Priority** | Required for fair gameplay. |
| **Frequency of Use** | May occur multiple times per game but finalizes once a valid draw condition is confirmed. |
| **Channel to Actor** | Internal game logic, displayed to players. |
| **Secondary Actors** | 1. Player: May initiate a draw offer or trigger a draw condition. |
| **Open Issues** | 1. Should players have an option to review the board after a draw is declared? |

# Select Piece

Iteration: 2, last modification: March 07, 2025

| | |
|---|---|
| **Description** | **Select Piece Primary Actor.** |
| **Primary Actor** | Player |
| **Goal in Context** | Allow a player to select a valid chess piece they control, initiating the move process. |
| **Preconditions** | 1. The game is active and in progress.<br>2. It is the current player's turn. |
| **Trigger** | A player clicks or selects a piece on the board. |
| **Scenario** | 1. The player selects a piece on the board.<br>2. The system verifies if the selected piece belongs to the current player.<br>3. The system highlights the selected piece.<br>4. The player either selects a valid move or cancels the selection.<br>**Iteration Details.**<br>• The system iterates through available pieces to determine legal selections. |
| **Exceptions** | 1. **Invalid Selection:** If the player selects an empty square or an opponent’s piece, an error message is displayed.<br>2. **Turn Violation:** If the player tries to select a piece when it is not their turn, the selection is blocked |
| **Priority** | Essential for gameplay interaction. |
| **Frequency of Use** | Every turn, once per move. |
| **Channel to Actor** | Internal game logic, visual feedback on the board. |
| **Secondary Actors** | 1. **Game Engine:** Validates piece selection. |
| **Open Issues** | 1. Should the system provide an option to deselect a piece without moving?<br>2. Should invalid selections be visually indicated? |

# Select Piece Validity

Iteration: 1, last modification: March 07, 2025

| **Description** | **Validity of primary actor selecting a piece.** |
|---|---|
| **Primary Actor** | Game Engine |
| **Goal in Context** | Ensure the player selects a valid piece that belongs to them and can legally move. |
| **Preconditions** | 1. The game is active.<br>2. A player attempts to select a piece. |
| **Trigger** | A piece is selected by the player. |
| **Scenario** | 1. The system receives the selection request.<br>2. The system verifies if the piece belongs to the player.<br>3. The system checks if the piece has any legal moves.<br>4. If invalid, the selection is rejected, and an error message is shown.<br>**Iteration Details:**<br>• The system checks each piece's available moves before confirming selection. |
| **Exceptions** | 1. **Invalid Selection:** If the selected piece belongs to the opponent or is ineligible to move, selection is denied.<br>2. **No Legal Moves:** If the selected piece has no legal moves, an error message is displayed, and selection is canceled. |
| **Priority** | High – Required for move validation. |
| **Frequency of Use** | Each time a piece is selected. |
| **Channel to Actor** | Internal game logic, visual feedback to the player. |
| **Secondary Actors** | 1. N/A |
| **Open Issues** | 1. Should the system automatically skip selection for pieces with no valid moves?<br>2. Should there be an option to display why a piece cannot move? |

# Castling

Iteration: 1, last modification: March 07, 2025

| **Description** | **Castle a piece.** |
|---|---|
| **Primary Actor** | Player |
| **Goal in Context** | Allow a player to perform castling if all conditions are met, ensuring the king and rook move correctly and legally. |
| **Preconditions** | 1. The game is in progress.<br>2. It is the player's turn.<br>3. The king and the chosen rook have not moved previously.<br>4. There are no pieces between the king and the rook.<br>5. The king is not in check, and the squares it moves through or lands on are not under attack. |
| **Trigger** | A player selects the king and attempts to perform castling. |
| **Scenario** | 1. The player selects the king and attempts to move two squares toward a rook.<br>2. The system verifies that the selected move corresponds to a valid castling move.<br>3. The system checks if castling conditions are met:<br>• The king and rook have not moved before.<br>• The squares between the king and rook are empty.<br>• The king is not in check, and the move does not pass through or land on an attacked square.<br>4. If valid, the king moves two squares towards the rook, and the rook moves to the square next to the king.<br>5. The move is recorded in the game history.<br>6. The system checks for check, checkmate, or stalemate conditions.<br>7. The turn switches to the opponent.<br>**Iteration Details:**<br>• The system verifies all castling conditions before execution.<br>• The board updates both the king and rook's positions simultaneously. |
| **Exceptions** | 1. **Invalid Castling Attempt:** If any condition for castling is not met, the move is rejected, and an error message is displayed |

| | |
|---|---|
| | 2. **Check Restriction:** If the king is in check or would move through an attacked square, castling is blocked |
| **Priority** | Essential for correct chess rules implementation. |
| **Frequency of Use** | Rare, typically once per game per player. |
| **Channel to Actor** | Internal game logic, displayed on the board. |
| **Secondary Actors** | 1. **Game Engine:** Validates the castling move. |
| **Open Issues** | 1. Should an option to confirm castling be provided before executing the move? |

# Promote Pawn

Iteration: 1, last modification: March 7, 2025

| | |
|---|---|
| Description | Promote a pawn at the end of the board in chess |
| Primary Actor | Player |
| Goal in Context | Allow a player to promote a pawn when it reaches the eighth rank, replacing it with another piece (queen, rook, bishop, or knight). |
| Preconditions | The game is in progress.<br>A pawn reaches the last rank (rank 8 for White, rank 1 for Black). |
| Trigger | A player moves a pawn to the final rank. |
| Scenario | 1. The player moves a pawn to the last rank.<br>2. The system detects that the pawn must be promoted.<br>3. The system prompts the player to select a promotion piece (queen, rook, bishop, or knight).<br>4. The selected piece replaces the pawn on the board.<br>5. The move is recorded in the game history.<br>6. The system checks for check, checkmate, or stalemate conditions.<br>7. The turn switches to the opponent. |
| Exceptions | **Invalid Promotion Attempt:** If an issue occurs with selecting a promotion piece, a default promotion to a queen is applied. |
| Priority | High – Critical for proper chess gameplay. |
| Frequency of Use | Varies per game, occurs when a pawn reaches the last rank. |
| Channel to Actor | Internal game logic, displayed to the player. |
| Secondary Actors | **Game Engine:** Ensures promotion rules are followed. |
| Open Issues | 1. Should the system allow pre-selection of promotion choices (e.g., auto-queen option)?<br>2. Should the system visually highlight promotion opportunities before the move? |

# Tic Tac Toe – Placing a Mark

Iteration: 1, last modification: March 7, 2025

| | |
|---|---|
| Description | Manage the logic for placing marks on the Tic Tac Toe board, checking for win conditions or a draw, and managing player turns. |
| Primary Actor | Game Logic Controller |
| Goal in Context | Ensure the game logic for Tic Tac Toe processes mark placements efficiently and accurately, independent of user interface commands. |
| Preconditions | 1. The game is initialized with an empty board.<br>2. It is the current player's turn.<br>3. The cell selected for placing a mark is unoccupied. |
| Trigger | Command received to place a mark in a specified cell. |
| Scenario | 1. Receive cell selection information.<br>2. Confirm the cell is empty.<br>3. Update the game board with the new mark.<br>4. Check for three consecutive marks in a row, column, or diagonal.<br>5. If a winning condition is detected:<br>◦ Signal a win and update the game state accordingly.<br>6. If no win and cells are still available:<br>◦ Pass the turn to the next player.<br>7. If no cells are left:<br>◦ Signal a draw and update the game state. |
| Exceptions | 1. Invalid Cell Selection:<br>◦ Return an error or signal indicating the cell is already occupied.<br>2. Post-Game Action Attempted:<br>◦ Maintain game end status and ignore the move. |
| Priority | High – Essential for maintaining game integrity and logic. |
| Frequency of Use | Every turn until the game ends. |

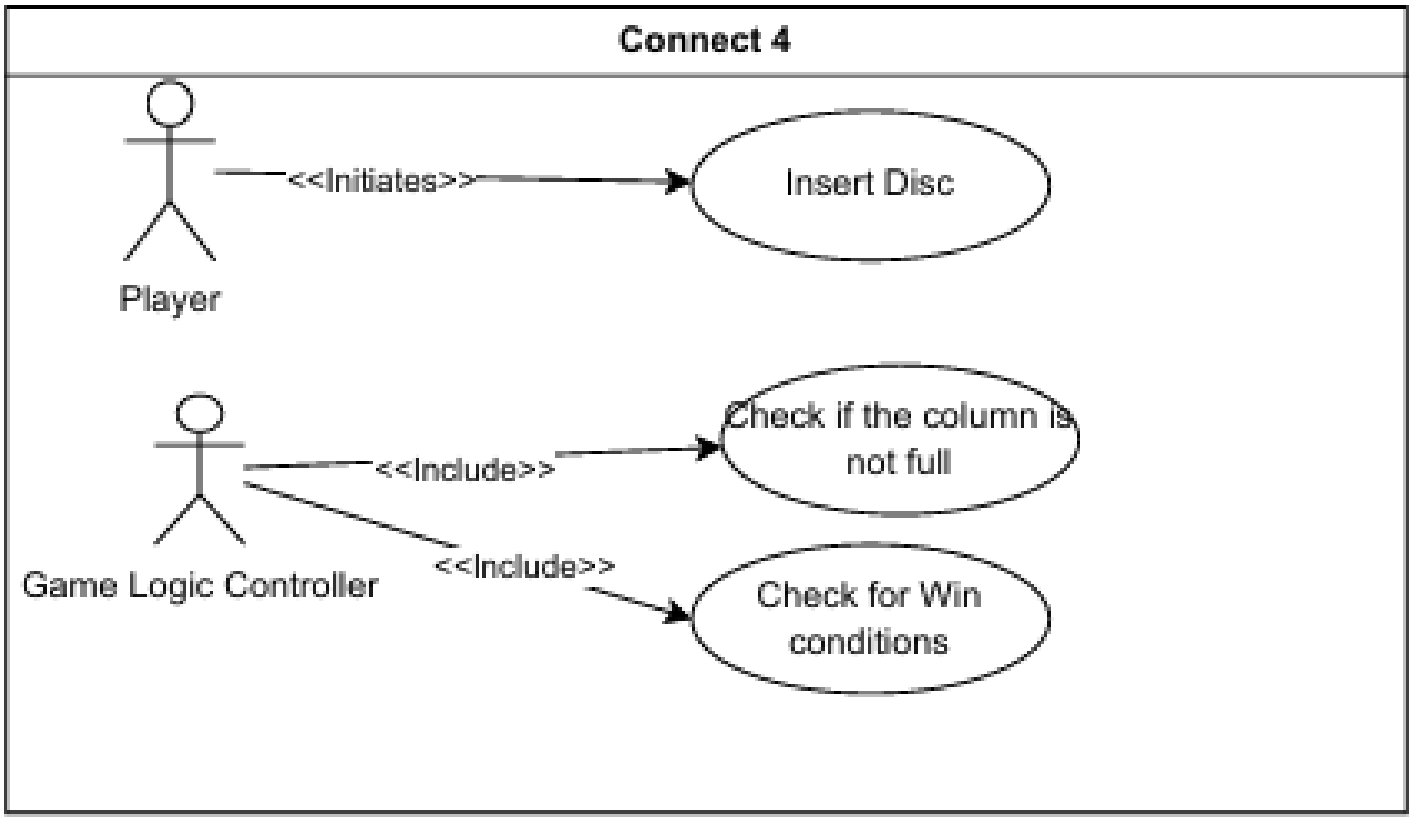
| | |
|---|---|
| Channel to Actor | Input from game session management or a networked event. |
| Secondary Actors | 1. Game State Manager (tracks and updates the game state).<br>2. Network Interface (handles communication with other players over the network). |
| Open Issues | 1. Methods for efficient state updates to support real-time multiplayer experiences.<br>2. Strategies for handling asynchronous gameplay in networked environments. |

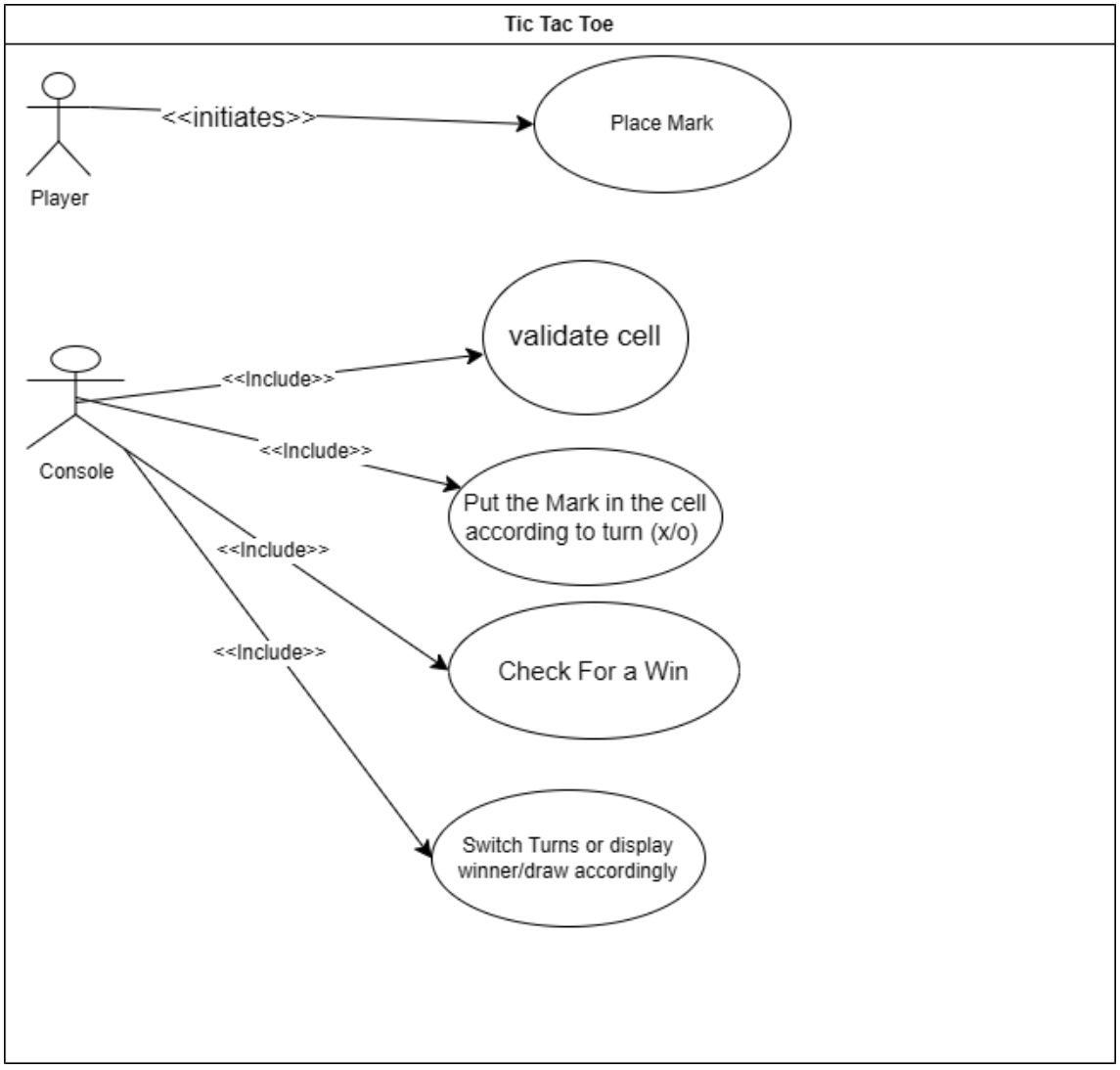
# Connect 4 – Placing a Disc

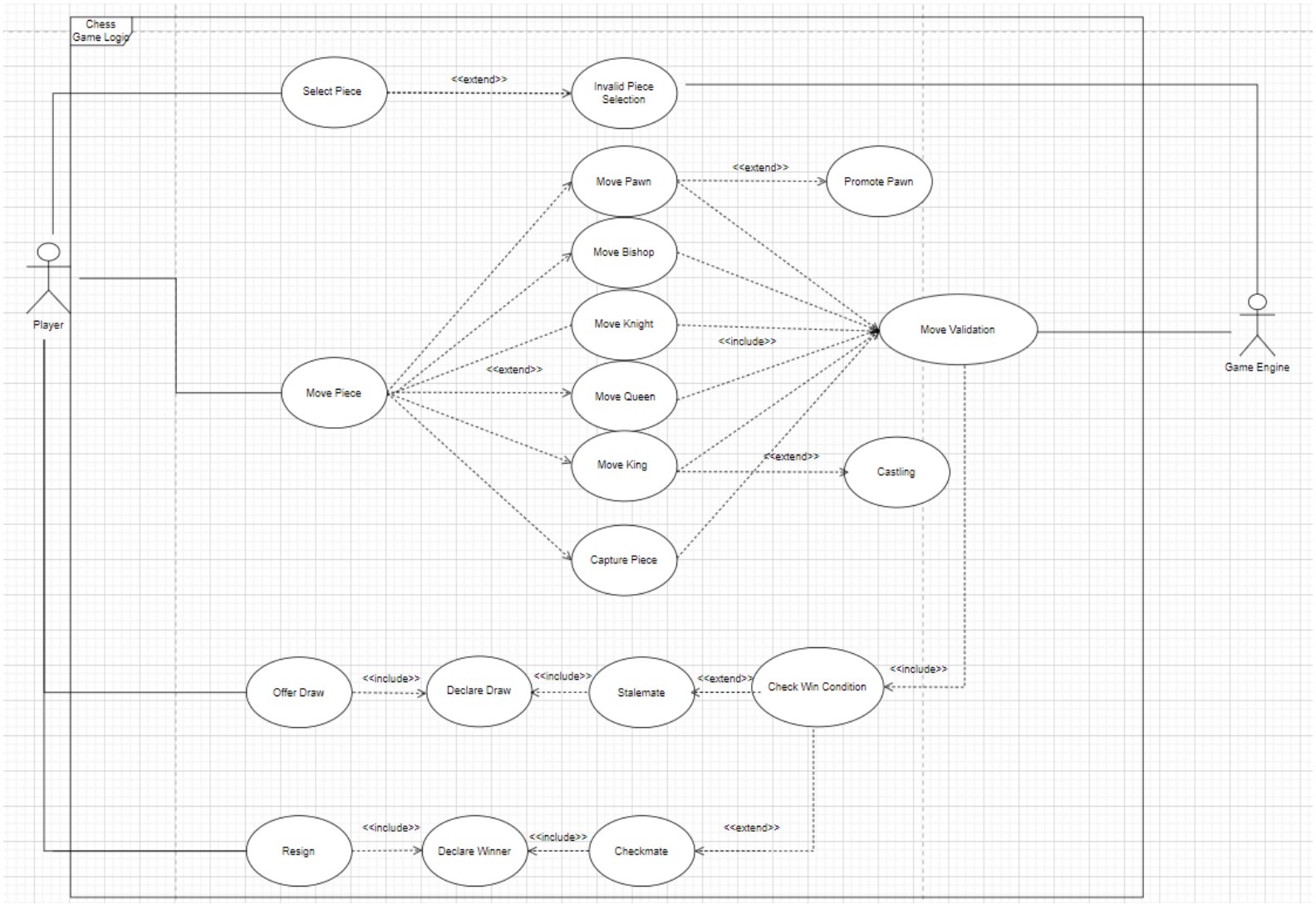
Iteration: 1, last modification: March 7, 2025

| | |
|---|---|
| Description | Handle the logic for inserting a disc into a column, evaluate whether this triggers a win condition or a draw, and manage turn transitions. |
| Primary Actor | Game Logic Controller |
| Goal in Context | Facilitate the game logic for Connect Four, ensuring the system checks for full columns, processes disc placements, and evaluates win conditions without direct interaction with the GUI. |
| Preconditions | The game state is initialized and ready for play.<br>It is currently the active player's turn.<br>The column selected for disc placement must not be full |
| Trigger | Receives a command to place a disc in a specified column through a console prompt. |
| Scenario | 1. Receive column selection information.<br>2. Verify the column is not full.<br>3. Update the game state by placing the disc in the lowest available slot in the column.<br>4. Check for any four consecutive discs aligning vertically, horizontally, or diagonally.<br>5. If a win condition is detected:<br>Update the game state to reflect the win and signal the end of the game.<br>6. If no win and the board has empty spaces:<br>Transition the turn to the next player.<br>7. If the board is full and no win is detected<br>Update the game state to reflect a draw. |
| Exceptions | 1. Full Column:<br>Return an error state or signal indicating the column is full.<br>2. Game Already Concluded:<br>Ignore moves and maintain game conclusion status. |
| Priority | High – Fundamental to game functionality. |
| Frequency of Use | Repeated each time a player makes a move. |

| | |
|---|---|
| Channel to Actor | Direct command from a game session manager or network event. |
| Secondary Actors | Game Session Manager (handles turn management and session state).<br><br>Network Service (receives and sends game state information across networked players). |
| Open Issues | How to effectively communicate game state changes to the GUI and networking teams?<br><br>Considerations for optimizing state checking for performance and reliability. |


Connect 4
Player
<<Initiates>>
Insert Disc
Game Logic Controller
<<Include>>
Check if the column is not full
<<Include>>
Check for Win conditions


Tic Tac Toe
Player
<<initiates>>
Place Mark
Console
<<Include>>
validate cell
<<Include>>
Put the Mark in the cell according to turn (x/o)
<<Include>>
Check For a Win
<<Include>>
Switch Turns or display winner/draw accordingly


Chess Game Logic
Select Piece
<<extend>>
Invalid Piece Selection
Move Pawn
<<extend>>
Promote Pawn
Move Bishop
Move Knight
Move Validation
<<include>>
Game Engine
Player
Move Piece
<<extend>>
Move Queen
Move King
<<extend>>
Castling
Capture Piece
Offer Draw
<<include>>
Declare Draw
<<include>>
Stalemate
<<extend>>
Check Win Condition
<<include>>
Resign
<<include>>
Declare Winner
<<include>>
Checkmate
<<extend>>

# GUI Use Cases

## Chess GUI

Iteration: 1, last modification: February 28, 2025

| | |
|---|---|
| **Description** | Make a move in Chess GUI |
| **Primary Actor** | Player using the chess app. |
| **Goal in Context** | Allow the player to select and move chess pieces using a simple interface with visual feedback, following chess rules. |
| **Preconditions** | 1. The chess app is running and shows the game board<br>2. Chess pieces are correctly placed<br>3. The player's turn is shown (e.g., player name, turn indicator)<br>4. The board is properly aligned for the player. |
| **Trigger** | Player clicks, taps, or drags a chess piece on the board. |
| **Scenario** | 1. Player clicks/taps on a piece<br>2. The piece is highlighted<br>3. Valid move options are shown (green dots for empty squares, red dots for capturable pieces)<br>4. Player clicks a valid destination square or drags the piece<br>5. The piece moves smoothly to the new position<br>6. If capturing, the opponent's piece disappears<br>7. The board updates, showing the new position, the turn changes, and move history is updated<br>8. If the king is in check, it's highlighted and "Check!" is shown<br>9. For special moves (castling, en passant, promotion), appropriate animations or dialogs are shown. |
| **Exceptions** | 1. **Invalid Piece Selection**: - No highlight if the player clicks an opponent's piece or empty square. - A tooltip says "Not your turn"<br>2. **Invalid Move Attempt**: - Piece snaps back if dragged incorrectly. - A shake/pulse animation shows an invalid move. - A tooltip explains why it's invalid<br>3. **Promotion**: - A dialog appears for selecting a new piece (queen, rook, bishop, knight). - Cannot proceed without selection<br>4. **Technical Issues**: - A red indicator shows if the online game loses connection. - Auto-save indicator appears if the local game state is saved. |

| | |
|---|---|
| **Priority** | High priority – core gameplay interaction. |
| **Frequency of Use** | Very frequent (multiple times per minute during gameplay) |
| **Channel to Actor** | Mouse (click, drag & drop), touchscreen gestures, or keyboard |
| **Secondary Actors** | 1. Game State Manager (tracks board, move history, clock)<br>2. Network Service (online multiplayer). |
| **Open Issues** | 1. Should drag-and-drop and click-to-select both be supported?<br>2. How to show move hints and differentiate from move history?<br>3. What accessibility features should be added for visually impaired players?<br>4. Should animations be optional for performance? |

# Tic-Tac-Toe GUI

Iteration: 1, last modification: February 28, 2025

| | |
|---|---|
| **Description** | Place a mark in Tic-Tac-Toe GUI |
| **Primary Actor** | Player using the tic-tac-toe app |
| **Goal in Context** | Allow the player to place their mark (X or O) in an empty cell with clear feedback about the game state. |
| **Preconditions** | 1. The tic-tac-toe app is running and shows the 3x3 grid<br>2. All previous moves are shown<br>3. The turn indicator shows which player's turn it is (X or O)<br>4. There is at least one empty cell. |
| **Trigger** | Player clicks or taps on an empty cell in the grid |
| **Scenario** | 1. Player hovers over an empty cell and sees a preview of their mark<br>2. Player clicks/taps on the cell<br>3. The mark is animated (fade, grow, or draw effect)<br>4. A sound confirms the placement<br>5. The game checks for win or draw<br>6. If the game continues, the turn indicator updates<br>7. If a win happens, the winning line is animated and a victory message is shown<br>8. If a draw occurs, a message and "game over" animation appear<br>9. The "Play Again" button is enabled, and the score is updated. |
| **Exceptions** | 1. **Cell Already Marked**: - No action occurs if the player clicks a marked cell. - A shake/pulse animation indicates the invalid selection. - A tooltip says "Cell already marked"<br>2. **Input After Game End**: - Input is ignored after the game ends. - The "Play Again" button pulses for attention<br>3. **Technical Issues**: - An error message appears if the game state can't be saved. - A reload option is provided if rendering fails. |
| **Priority** | High priority – core gameplay interaction. |
| **Frequency of Use** | Frequent (up to 9 times per game, with multiple games in sequence) |
| **Channel to Actor** | Mouse clicks, touchscreen taps, or keyboard navigation |

| | |
|---|---|
| **Secondary Actors** | 1. Game Logic Controller (checks win conditions, manages turns)<br>2. Score Tracker (tracks win/loss/draw statistics) |
| **Open Issues** | 1. Should hovering over cells show a preview of the mark?<br>2. What animation style best shows the game state without distraction?<br>3. What sound effects are best for different events? |

# Connect 4 Game

Iteration: 1, last modification: March 07, 2025

| | |
|---|---|
| **Use Case** | Play Connect 4 |
| **Primary Actor** | Player using the game platform. |
| **Goal in Context** | Allow players to engage in a game of Connect 4 against an opponent, placing discs to connect four in a row. |
| **Preconditions** | 1. The player is logged into the game platform.<br>2. The player has selected Connect 4 from available game options.<br>3. The player has been matched with an opponent or invited a friend to play. |
| **Trigger** | Player enters the Connect 4 game session after matchmaking or accepting an invitation. |
| **Scenario** | 1. Game board displays a 7□6 grid □7 columns, 6 rows) initially empty.<br>2. Players are assigned colors (typically red and yellow).<br>3. The system randomly determines who goes first.<br>4. First player selects a column by clicking/tapping on it.<br>5. The game drops the player's disc to the lowest available position in the selected column.<br>6. Turn indicator switches to the opponent.<br>7. Second player selects a column for their disc.<br>8. Players alternate turns, with each trying to connect four discs horizontally, vertically, or diagonally.<br>9. Game continues until someone wins or the board fills up (draw). 10. Upon game completion, winner is highlighted and victory animation plays.<br>11. Game statistics are updated in players' profiles.<br>12. Players are offered options to play again, return to lobby, or view replay. |

| | |
|---|---|
| **Exceptions** | 1. **Column Full.** If a player selects a full column, selection is rejected with visual/audio feedback and player must choose another column.<br>2.<br>**Connection Lost.** Game is paused for up to 30 seconds. Reconnected players resume from last state. If no reconnection, remaining player wins by forfeit.<br>3.<br>**Player Inactivity.** Timer counts down 30 seconds per move). If time expires, an automatic random move is made or the game is forfeited |
| | after 3 consecutive timeouts.<br>4.<br>**Player Quits.** Opponent is notified and awarded victory by forfeit. |
| **Priority** | High priority |
| **Frequency of Use** | High – primary game offering on the platform. |
| **Channel to Actor** | Mouse/touch input for column selection, optional keyboard navigation (arrow keys and Enter). |
| **Secondary Actors** | 1. Matchmaking Service<br>2. Rating System (updates player rankings after each game).<br>3. Statistics Service (tracks wins, losses, draws, and gameplay metrics). |
| **Open Issues** | 1. Should special game modes be offered (e.g., timed matches, tournaments, custom board sizes)?<br>2. Should the platform offer spectator mode for popular or high ranked matches? |

# Challenge a User GUI

Iteration: 1, last modification: March 3, 2025

| | |
|---|---|
| **Description** | Challenge a User |
| **Primary Actor** | Player looking to challenge a user |
| **Goal In Context** | Allow the user to challenge another user to a game |
| **Preconditions** | 1. The OMG program is running<br>2. The player is signed in and online<br>3. The other user they wish to challenge is also signed in and online. |
| **Trigger** | Player goes to other user's profile and clicks the option to challenge that user |
| **Scenario** | 1. The player opens the program<br>2. They log into their account<br>3. The player searches up the user they wish to challenge<br>4. If the user is online, the player proceeds to click the option to challenge the user to a particular game |
| **Exceptions** | 1. Invalid login<br>2. The user being challenged is not online |
| **Priority** | High priority – essential to online multiplayer gaming |
| **Frequency Of Use** | High. This is one of the main functionalities of an online multiplayer game system |
| **Channel to Actor** | Mouse (through clicking), touchscreen(through physical interaction with the screen with fingers), and keyboard |
| **Secondary Actors** | 1. Network Service<br>2. Challenged User |
| **Open Issues** | 1. Does a user have to be friends with another user before they can challenge each other? |

# Chat GUI

Iteration: 1, last modification: March 02, 2025

| | |
|---|---|
| **Description** | Use In-game Chat |
| **Primary Actor** | Player using the game platform. |
| **Goal in Context** | Allow players to communicate with their opponents during game sessions using a text-based chat interface. |
| **Preconditions** | 1. The player is logged into the game platform.<br>2. The player is in an active game session (chess, connect 4, or tic tac toe).<br>3. Chat functionality is enabled for the current game.<br>4. The player has chat permissions (not muted/banned). |
| **Trigger** | Player opens chat window or types in the chat input field. |
| **Scenario** | 1. Player clicks on the chat icon.<br>2. Chat window appears, showing recent messages if any.<br>3. Player types a message in the text input field.<br>4. Player presses Enter or clicks the send button.<br>5. The message appears in the chat window with the player's username and timestamp.<br>6. All other participants (opponent, spectators) see the message in their chat windows.<br>7. The chat notification icon shows a badge for players who have the chat window minimized.<br>8. Chat history is maintained throughout the game session.<br>9. Players can scroll through previous messages if the chat history is extensive. |

| | |
|---|---|
| **Exceptions** | 1. **Empty Message Attempt**: No message is sent if the input field is empty. Send button remains inactive until text is entered.<br>2. **Inappropriate Content**: Profanity filter replaces inappropriate words with asterisks. System warning appears for repeated violations. Persistent violations may result in temporary chat muting. 3. **Connection Issues**: Message shows "Sending..." status if connection is delayed. Failed messages are marked with a red exclamation icon. Retry option appears for failed messages. 4. **Chat Disabled**: Notification appears if opponent has disabled chat. Option to send predefined friendly messages only. |
| **Priority** | Medium priority – enhances player experience and social aspect. |
| **Frequency of Use** | Moderate to high (depends on player preference, typically several times per game session). |
| **Channel to Actor** | Text input via keyboard, touch keyboard, or voice-to-text functionality. |
| **Secondary Actors** | 1. Messaging Service (manages message delivery and history).<br>2. Moderation System (filters content and manages abuse reports).<br>3. Other players (recipients of messages). |
| **Open Issues** | 1. Should there be game-specific automated messages (e.g., "Good move!" or "Check!" for chess)?<br>2. How to implement effective moderation for real-time chat?<br>3. Should private messaging between friends be supported across different game sessions?<br>4. How to balance chat visibility with game focus (consider chat overlay vs. dedicated panel)?<br>5. Should emoji and GIF support be included? |

# Check Leaderboard GUI

Iteration 1, last modification: March 4, 2025

| | |
|---|---|
| **Description** | Looking at the leaderboard |
| **Primary Actor** | Player using the program |
| **Goal in Context** | Let the player see the leaderboard on the game they choose |
| **Preconditions** | 1. The OMG program is running<br>2. The player is signed in<br>3. The player is in the game select section of the program |
| **Trigger** | Player clicks around in the program |
| **Scenario** | 1. The player opens the program<br>2. They log into their account<br>3. The player is presented with the games they own<br>4. They click on a game<br>5. A leaderboard is shown on the top half of the screen for the game chosen<br>6. The user can scroll the leaderboard to view the top players of the game chosen |
| **Exceptions** | 1. Error getting leaderboard information |
| **Priority** | Medium priority – Is not needed to run games |
| **Frequency of Use** | Frequent (Whenever the player is viewing their games but not when a game is being played) |
| **Channel to Actor** | Mouse, touchscreen, monitor |
| **Secondary Actors** | 1. Network Service<br>2. Leaderboard information |
| **Open Issues** | 1. Should access to players profiles be allowed through the leaderboard |

# Edit Profile GUI

Iteration: 1, last modification: March 3, 2025

| | |
|---|---|
| **Description** | Editing a User's Profile Picture and Name |
| **Primary Actor** | Player |
| **Goal In Context** | Player can edit their profile picture and name to what they please |
| **Preconditions** | 1. The OMG program is running<br>2. The player is signed in<br>3. The player is on the Market place (general menu) of the game |
| **Trigger** | Player clicks on the profile picture icon on the Market place |
| **Scenario** | 1. The player clicks or taps on their profile picture and is taken to their profile page<br>2. If the player chooses to change their profile picture they proceed to tap on the current picture<br>3. A selection of possible profile pictures to pick from comes up on the screen<br>4. The player proceeds to select the picture they wish to use and selects the save option<br>5. The picture is updated to what was chosen<br>6. If the player chooses to edit their name, they proceed to tap on the edit name option on the screen<br>7. The player proceeds to enter the name they wish to use for their account<br>8. If the name is unique(has not been chosen by another user already in the game database), a valid name text shows up in green else an invalid name text is shown in red.<br>9. The player's name is updated to the valid name which was chosen. |

| | |
|---|---|
| **Exceptions** | 1. Invalid login<br>2. Error loading up the player's information |
| **Priority** | Medium priority – Profile picture or name is seldom changed |
| **Frequency of Use** | Not frequent (a player does not change their profile picture or name all the time) |
| **Channel to Actor** | Interaction using mouse, touchscreen or keyboard. |
| **Secondary Actors** | 1. Network service<br>2. Database of Players Information |
| **Open Issues** | 1. Should a player be allowed to have the option to upload a picture from their device?<br>2. If they are allowed to upload pictures from their device, how is it validated as appropriate?<br>3. What names is the player restricted from using? |

# Game Select GUI

Iteration 2, last modification: March 4, 2025

| | |
|---|---|
| **Description** | Selecting a game to play in the program |
| **Primary Actor** | Player using the program |
| **Goal in Context** | Let the player see the game options, view extra information on a game, and select a game to play |
| **Preconditions** | 1. The OMG program is running<br>2. The player is signed in<br>3. The player is in the game select section of the program |
| **Trigger** | Player clicks around in the program |
| **Scenario** | 1. The player opens the program<br>2. They log into their account<br>3. The player is presented with the games they own<br>4. They click on a game<br>5. A leaderboard is shown with the option to start the game |
| **Exceptions** | 1. Invalid Login<br>2. Error with opening game |
| **Priority** | High priority – is the main hub of the program |
| **Frequency of Use** | Frequent (Whenever the player is viewing their games but not when a game is being played) |
| **Channel to Actor** | Mouse, touchscreen, monitor |
| **Secondary Actors** | 1. Network Service (leaderboard)<br>2. Profile Authentication |
| **Open Issues** | 1. Should controller inputs be supported? |

# Create Account GUI

Iteration: 2, last modification: March 4

| **Description** | **Create an Account Using GUI** |
|---|---|
| **Primary Actor** | Unregistered player. |
| **Goal in Context** | Create an Account Using GUI. |
| **Preconditions** | The player has launched the OMG app. |
| **Trigger** | The player clicks the signup button on the login screen. |
| **Scenario** | 1. The player clicks the create account button at the bottom of the login screen.<br>2. The GUI display takes goes to the create account screen and in the center of it is 5 textboxes stacked on top of each other. From the top to bottom the textboxes are labeled “username”, “email”, “date of birth”, “password”, and “confirm password”. There is also a button labeled “submit” below these text boxes. There’s also a button labeled “return” on the top left corner and a button labeled “help” in the top right corner.<br>3. The player clicks on a textbox to begin typing in it.<br>4. The player enters their information into each textbox. The confirm password textbox replaces characters with *.<br>5. The player clicks the “submit” button once they have filled out their information.<br>6. Once the player clicks the submit button a new textbox will appear over most of the screen saying, “a confirmation email has been sent to <email address>”.<br>7. The player confirms their account through their email.<br>8. The new textbox appears and is promoted with a are you human test which they must complete.<br>9. The player is prompted that they have successfully created an account.<br>10. The player is automatically logged in to the account. |

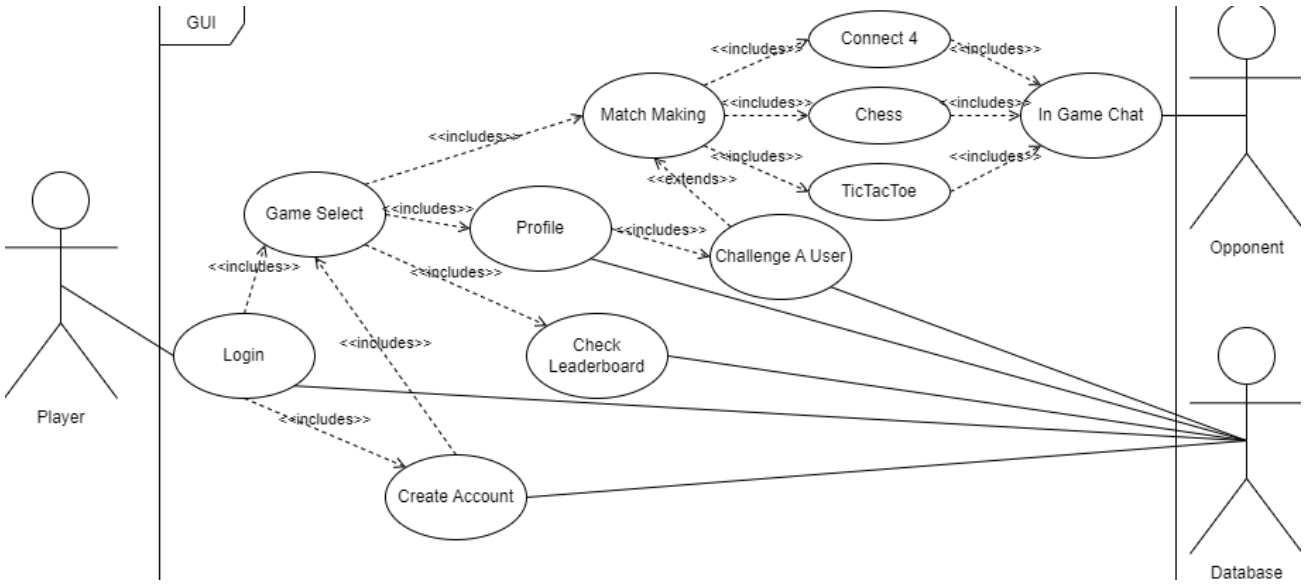
| | |
|---|---|
| **Exceptions** | 1. **Account Existence.**<br>- If the player clicks submit and the username is already a part of the system, the username text area is cleared and red text appears by the text area saying, “username already exists”.<br>- If the player clicks submit and the email is already a part of the system, the email text area is cleared and red text appears by the text area saying, “email already exists”.<br>2. **Invalid Password:**<br>- If the text in the password and confirm password text areas do not match when submit is clicked, the confirm password text area is cleared and red text appears next to it saying “password does not match”<br>3. **Empty Text Area**:<br>- If any of the text areas are empty when the player clicks submit red text will appear next to the empty text areas saying “please enter a <text area name>”.<br>4. **Abandon Action**:<br>- If a player clicks the close window button the GUI display closes.<br>- If a player clicks the return button they are sent back to the login screen. |
| **Priority** | High priority – players need an account to play |
| **Frequency of Use** | Infrequent – theoretically only once per player. |
| **Channel to Actor** | Player’s computer and computer input/output devices. |
| **Secondary Actors** | 1. GUI display.<br>2. Background authentication data. |
| **Open Issues** | 1. Is there password requirements e.g. length, numbers, etc.<br>2. What exactly is the are you human test. |

# Login GUI

Iteration: 2, last modification: March 4

| **Description** | **Create an Account Using GUI** |
|---|---|
| **Primary Actor** | Registered player. |
| **Goal in Context** | Login to an account of a registered player. |
| **Preconditions** | The player has their computer running and the OMG app downloaded. |
| **Trigger** | The player launches the OMG app. |
| **Scenario** | 1. The GUI display immediately shows the login screen upon app. In the center of the screen there is 2 textboxes stacked on top of each other. From the top to bottom the text areas are labeled “username”, “password”. Below the text areas there’s a button labeled “Log In”. Below that is two buttons side by side one saying, “create account” and the other saying “forgot password”. To the right of the “Log In” button there is a checkmark box labeled stay logged in. There is also a button labeled “help” in the top right corner.<br>2. The player clicks on a textbox to begin typing in it.<br>3. The player enters their username and password into the text boxes.<br>4. The player clicks the “Log In” button.<br>5. The player is brought to the main menu if valid. |
| **Exceptions** | 1. **Information Validity.**<br>- If the player clicks submit and the does not exist red text will appear next to the text area saying, “username does not exist”.<br>- If the player clicks submit and the password does not match the players password red text will appear next to the text area saying, “incorrect password”.<br>2. **Abandon Action**:<br>- If a player clicks the close window button the GUI display closes.<br>3. **Empty Text Area**:<br>- If any of the text areas are empty when the player clicks log in red text will appear next to the empty text areas saying, “please enter a <text area name>”. |
| **Priority** | High priority – players need to be logged into an account to play |
| **Frequency of Use** | Frequent – every time the player wishes to access the system. |

| | |
|---|---|
| **Channel to Actor** | Player’s computer and computer input/output devices. |
| **Secondary Actors** | 3. GUI display.<br>4. Background authentication data. |
| **Open Issues** | 1. N/A |


GUI
Connect 4
<<includes>>
<<includes>>
Match Making
<<includes>>
Chess
<<includes>>
In Game Chat
<<includes>>
<<includes>>
<<includes>>
<<extends>>
TicTacToe
<<includes>>
Game Select
<<includes>>
Profile
<<includes>>
Challenge A User
<<includes>>
<<includes>>
Login
<<includes>>
Check
Leaderboard
<<includes>>
Create Account
Player
Opponent
Database

# Leaderboard & Matchmaking Use Cases

## See Personal Stats

Iteration: 1, last modification: March 7, 2025

| | |
|---|---|
| **Description** | Passing the player's data to profile for user to view the personal statistics |
| **Primary Actor** | Game Statistics Class |
| **Goal in Context** | To pass along data from the database to profile and UI to display the person's data |
| **Preconditions** | 1. The player must be logged in and the user ID exists in the database<br>2. There must be a request from the player via the UI<br>3. The data must be converted from SQL to Java |
| **Trigger** | The person's data will be converted from Queries to Java then profiles which will be displayed on the UI |
| **Scenario** | 1. The player clicks on the profile button<br>2. It will send a request to Game Statistics Class which will get data from the database and converting it to Java<br>3. It will send the data to profile<br>4. Profile will display the data on UI |
| **Exceptions** | 1. Data not found<br>- Personal data was not found in the database |
| **Priority** | High. This is a core system requirement |
| **When available** | Third Iteration |
| **Frequency of Use** | Once a day |
| **Channel to Actor** | Interaction with the user profile menu UI |
| **Secondary Actors** | Database |
| **Channel to Secondary Actors** | Queries to the database |
| **Open Issues** | 1. How to get data from the database team, how will we communicate with the database team<br>2. What type of data should we pass to user profile |

# Match Making

Iteration: 1, last modification: March 7, 2025

| | |
|---|---|
| **Description** | To put the player into a queue to find a game by changing the player's status as interested in the database |
| **Primary Actor** | Player |
| **Goal in Context** | To put the player in a match's queue and change their status in the database as interested |

| | |
|---|---|
| **Preconditions** | 1. The player must be logged in with a valid ID<br>2. Internet connection to the database must be made<br>3. The player must select a game that they are interested in playing |
| **Trigger** | Update database and change the player status in a game to be interested |
| **Scenario** | 1. The player clicks on a game they want to play<br>2. The player clicks on the find a match button<br>3. That will send an update to the match making database class<br>4. That will send a Query to the database |
| **Exceptions** | 1. Player ID not found<br>- The player ID does not match the player ID<br>2. No game selected<br>- The player did not select a game<br>3. Suitable opponent is not found<br>- No player with similar elo |
| **Priority** | High. This is an important function since this would be the easiest way to enter a match |
| **When available** | Third Iteration |
| **Frequency of Use** | Several times a day |
| **Channel to Actors** | Interaction with game selection menu UI |
| **Secondary Actors** | Database |
| **Channel to Secondary Actors** | Queries to the database |
| **Open Issues** | How will we update and find a suitable match using the database |

# Challenge A User

Iteration: 1, last modification: March 7, 2025

| | |
|---|---|
| **Description** | Interaction with a friend's invitation to start a game |
| **Primary Actor** | Player |
| **Goal in Context** | To accept or deny a request for a game invite |
| **Preconditions** | 1. The player must be logged in with a valid ID<br>2. A request must be sent to the player |
| **Trigger** | Either abort the Match creation or create a Match variable that will be passed to game logic server code |
| **Scenario** | 1. The player sends out an invitation to their friend<br>2. The player's friends will either accept or deny the request through the UI<br>3. .That will either trigger the creation of Match in the server or Not. |

| | |
|---|---|
| **Exceptions** | 1. Player ID not found<br>- Either the player or player's friend ID is not found<br>2. Match no longer exist<br>- If the match invites times out |
| **Priority** | Medium. This is an important function but automatically finding a match is more important |
| **When available** | Third iteration |
| **Frequency of Use** | Several times a day |
| **Channel to Actors** | Interaction with game selection menu |
| **Secondary Actors** | Player's friends |
| **Channel to Secondary Actors** | Via networking to send the invite and receive the answer |
| **Open Issues** | 1. How would we go about sending the invites over the network?<br>2. How long should the invite last before canceling the invite<br>3. How would we put both people into a queue so the system knows when to start the game with those two people |

# Start a Party Match

Iteration: 2, last modification: March 7, 2025

| | |
|---|---|
| **Description** | Starting a match for all players in the party |
| **Primary Actor** | Players in the party |
| **Goal in Context** | To create a Match with the players in the party as its variable |
| **Preconditions** | 1. Both players must be logged in with a valid player's ID<br>2. Both players must be in the same party and the party size must be 2<br>3. A game must be selected prior to starting a match |
| **Trigger** | A creation of Match in MatchMaking Class with both player's ID as its parameter |
| **Scenario** | 1. Both players are in the same party<br>2. The party's leader will select a game to start<br>3. The party's leader the click the start button which will get both players into a game |

| | |
|---|---|
| **Exceptions** | 1. Invalid player's ID<br>- If one or both players have a non- matching ID with the database<br>2. Invalid party's size<br>- If the party size is not 2<br>3. Game selection missing<br>- If a game is not selected |
| **Priority** | Medium. This is an important function but automatically finding a match is more important |
| **When available** | Third Iteration |
| **Frequency of Use** | Several times a day |
| **Channel to Actors** | Interaction with game selection menu |
| **Secondary Actors** | None |
| **Channel to Secondary Actors** | N/A |
| **Open Issues** | 1. We are not too sure on how we will check the party size system |

# View Personal Leaderboard

Iteration: 3, last modification: March 7, 2025

| | |
|---|---|
| **Description** | Make a list of 50 players that are above and below the players current rank and pass<br>the list to GUI |
| **Primary Actor** | Player |
| **Goal in Context** | A player wants to view where they are in<br>relation to player at the same skill level |
| **Preconditions** | 1. The player is logged in with a valid player's ID<br>2. The player navigated to their profile |
| **Trigger** | Make a sorted list of 50 players that are<br>directly ranked above and below them |
| **Scenario** | 1. The player logs in<br>2. The player navigated to their profile<br>3. The player wants to check their standing with players of similar elo |
| **Exceptions** | 1. Invalid player's ID<br>- If the player's ID does not match the database<br>2. Sever Interruption<br>- If the connection with the database<br>is interrupted |

| | |
|---|---|
| **Priority** | Low. This is a quality-of-life feature that will<br>encourage players, and help them to be more engaging with the game |
| **When available** | Iteration 3 |
| **Frequency of Use** | However often the players want to know<br>their rank |
| **Channel to Actors** | Interaction with game UI |
| **Secondary Actors** | Database |
| **Channel to Secondary Actors** | Database's queries to get player's data |
| **Open Issues** | 1. Have not picked a data type to pass<br>to GUI |

# View Friend's List

Iteration: 3, last modification: March 7, 2025

| | |
|---|---|
| **Description** | Returns a sorted list of all the other players that have been friended and sorted them by elo. |
| **Primary Actor** | Player |
| **Goal in Context** | To view the player elo in their friend's list |
| **Preconditions** | 1. The player opens the game<br>2. The game is connection to the internet<br>3. The player is logged in<br>4. The game is connected to the internet<br>5. The player navigates to the friend's list |
| **Trigger** | Gives an up-to-date list of all their friends sorted by their elo |
| **Scenario** | 1. The player logs in<br>2. The player navigates to the friend's list<br>3. The player wants to check where they stand in elo ranking to their friends |
| **Exceptions** | 1. Invalid player's ID<br>- If the player's ID does not match the database<br>2. Sever Interruption<br>- If the connection with the database is interrupted |

| | |
|---|---|
| **Priority** | Low. This is a quality-of-life feature that boost the competitiveness of the player and encourage interactions with the game |
| **When available** | Iteration 3 |
| **Frequency of Use** | Dependent on how serious the player is about the game, ranging from after every match to once per day. |
| **Channel to Actors** | Interaction with game UI. |
| **Secondary Actors** | None |
| **Channel to Secondary Actors** | N/A |
| **Open Issues** | Coordinating with GUI |

# View Top Players Leaderboard

Iteration: 2, last modification: March 7, 2025

| | |
|---|---|
| **Description** | Provides a list of global player base sorted by elo |
| **Primary Actor** | Player |
| **Goal in Context** | To have a list of the top 500 players sorted by elo |
| **Preconditions** | 1. The player opens the game<br>2. The game is connection to the internet<br>3. The player logs in<br>5. The player navigate to the leaderboard |
| **Trigger** | Show the leaderboard with the top 500 players |
| **Scenario** | 1. The player logs in<br>2. The player selects the leaderboard button<br>3. The game will show a list of the top 500 players that the player can scroll through |
| **Exceptions** | 1. Invalid player's ID<br>- If the player's ID does not match the database<br>2. Sever Interruption<br>- If the connection with the database is interrupted |
| **Priority** | Low. This is a quality-of-life feature that can be implemented later |
| **When available** | Iteration 3 |
| **Frequency of Use** | Low, once per day |

| | |
|---|---|
| **Channel to Actors** | Interaction with game UI. |
| **Secondary Actors** | None |
| **Channel to Secondary Actors** | N/A |
| **Open Issues** | Coordinating with GUI |

# Networking Use Cases

## Session Timeout

Iteration: 1, last modification: March 07, 2025

| | |
|---|---|
| **Description** | Start a timer that count down the player's inactive time and automatically |
| **Primary Actor** | Player |
| **Goal in Context** | Ensure inactive players are automatically disconnected to free up server resources and maintain a smooth multiplayer experience. |
| **Preconditions** | 1. The player is connected to the server and in an active game session.<br>2. A session timeout policy is defined (e.g., disconnect after 5 minutes of inactivity). |
| **Trigger** | The player has been inactive for a predefined period. |
| **Scenario** | 1. The player stops interacting with the game (no moves, chat, or other actions).<br>2. The server starts a countdown for the session timeout.<br>3. The server sends a warning message to the player about an impending connection.<br>4. If no response is received, the server disconnects the player from the game.<br>5. The server updates the game state and notifies the remaining players. |
| **Exceptions** | 1. If the player resumes activity before the timeout, the countdown is reset.<br>2. If the disconnection was due to network issues, an automatic reconnection attempt is made. |
| **Priority** | High (ensures smooth server performance and fair gameplay). |
| **Frequency of Use** | Occasionally (depends on player inactivity) |
| **Channel to Actor** | 1. : Manages the session timeout and disconnection process.<br>Other Players: Receive a notification that the player has been removed |

| | |
|---|---|
| **Secondary Actors** | GameServerManager, Other players |
| **Open Issues** | 1. Should disconnected players be allowed to rejoin the same session?<br>1. How long should the server wait before removing inactive players? |

# Network Connection

Iteration: 1, last modification: March 7, 2025

| **Description** | |
|---|---|
| **Primary Actor** | Player |
| **Goal in Context** | Ensure the player can establish and maintain a stable connection with the game server for real-time interactions. |
| **Preconditions** | • The player has a stable internet connection.<br>• The game client is launched and ready to connect |
| **Trigger** | The player starts the game and attempts to connect to the server. |
| **Scenario** | The player opens the game and selects "Connect to Server." The game client sends a connection request to the server.<br>The server authenticates the request and establishes a session. If successful, the server sends a confirmation message.<br>The player is now connected and can proceed to matchmaking or gameplay. |
| **Exceptions** | • If the connection request fails:<br>  ◦ The player receives an error message (e.g., “Server Unreachable”)<br>  ◦ The system may retry the connection automatically<br>• If the player loses connection mid-game:<br>  ◦ The system attempts reconnection within a set time.<br>  ◦ If unsuccessful, the player is marked as disconnected in the game session. |
| **Priority** | Critical (network stability is essential for multiplayer functionality). |
| **Frequency of Use** | Every time a player starts a session. |
| **Channel to Actor** | In-game UI notifications (connection status, error messages). |
| **Secondary Actors** | GameServerManager, Authentication System. |
| **Open Issues** | • How many reconnection attempts should be allowed before disconnecting the player? |

| | |
|---|---|
| | • Should the game allow offline mode if the server is unreachable? |

# Update States

Iteration: 1, last modification: March 7, 2025

| **Description** | |
|---|---|
| **Primary Actor** | Player |
| **Goal in Context** | 6. Update the states of the game synchronously<br>7. Enhance the user experience with seamless transitions/ changes<br>8. Handle delays accordingly and minimize delays |
| **Preconditions** | 1. The game is running and the network is established<br>2. The player has a stable internet connection |
| **Trigger** | 1. The player makes a move in the game<br>2. The players interact with each other in the game |
| **Scenario 1** | 1. The player makes a valid move(change) in the game.<br>2. This triggers the “Update states” use case.<br>3. The game server updates the changes in the interface of the other players. |
| **Scenario 2** | 1. The players send messages to each other through in-game chat.<br>2. This triggers the “Update states” use case.<br>3. The game server updates the messages in the interface of all players through web socket servers. |
| **Exceptions** | 1. If the connection of a player fails, it will cause some delays in updating the states, and the player will potentially leave the server temporarily<br>2. If the server is packed, the server may face some issues with updating the states |
| **Priority** | Critical. This feature is the most fundamental aspect of a multiplayer game |
| **Frequency of Use** | Multiple Times |

| | |
|---|---|
| **Channel to Actor** | GUI, Game server |
| **Secondary Actors** | Game server system |
| **Open Issues** | 1. How should the system handle the connection problems so that the delays in the game are minimized?<br>2. Should the system save the chat history? |

# Retrieve Data

Iteration: 1, last modification: March 7, 2025

| **Description** | |
|---|---|
| **Primary Actor** | Player |
| **Goal in Context** | 1. Enable authentication of user information and retrieving the data for leaderboard and matchmaking processes<br>2. Save the current state of the game to continue from the saved state<br>3. Update miscellaneous information and store properly |
| **Preconditions** | 1. The game is running and the network is established<br>2. The database is running properly<br>3. The player has a stable internet connection<br>4. The request is valid |
| **Trigger** | 1. The player log into the game<br>2. The player saves the current state of the game<br>3. The player looks at the real-time leaderboard<br>4. The player updates user information<br>5. The player looks for the players in the network to play with |
| **Scenario 1** | 1. The player enters the game<br>2. The player log into the account by providing the username and the password<br>3. The player clicks the button to authenticate<br>4. The system retrieves data from the database and check if the information matches or not<br>5. The system approves the player if the information matches |
| **Scenario 2** | 1. The player wants to save the current state of the game<br>2. The player clicks “Save” button<br>3. The system accesses the database and updates the information |

| | |
|---|---|
| | 4. The data is updated and ready to be retrieve to resume the saved game |
| **Scenario 3** | 1. The player clicks the button to look at the leaderboard<br>2. The system accesses the database to retrieve the corresponding data.<br>3. The leaderboard is shown to the player through GUI |
| **Scenario 4** | 1. The player wants to update the user information<br>2. The player makes changes to the information and clicks ‘Update’<br>3. The system accesses the database and updates the information |
| **Scenario 5** | 1. The player starts a game and looks for the players to play with<br>2. The system accesses the database to find the players who are active now and have the same ranking as the player<br>3. The system adds those players to the lobby and the game is ready to begin |
| **Exceptions** | 1. Returns an error message if the data requested is not in the database.<br>2. If connection to the database is not successful, accessing the database fails.<br>3. Wrong data may be retrieved if the database is corrupted. |
| **Priority** | Critical. This feature is the most fundamental aspect of a multiplayer game |
| **Frequency of Use** | Multiple times |
| **Channel to Actor** | GUI, Game server |
| **Secondary Actors** | WebSocket server |
| **Open Issues** | 1. What security measures should be added to the system to ensure the integrity of the database?<br>2. What other information should be allowed to store in the database to enhance the experience of the user? |

# Verify Data

Iteration: 1, last modification: March 7, 2025

| Description | |
|---|---|
| **Primary Actor** | Player |
| **Goal in Context** | Authorize the log-in process to make sure that only the intended user can acce |
| **Preconditions** | 1. The game is running and the network is established<br>2. The database is running properly<br>3. The player has a stable internet connection |
| **Trigger** | 1. The player clicks ‘Log-in’ button to access the account |
| **Scenario** | 1. The player tries to log into the account by providing necessary personal<br>2. The system retrieves the data from the database corresponding to the p<br>3. The system compares the two data and decides to authorize the player |
| **Exceptions** | 1. The log-in attempt is rejected if the information does not match<br>2. If there is no existing data regarding the player, the player can sign up fo<br>3. The log-in process will fail if the database is corrupted |
| **Priority** | Critical. This feature is the most fundamental aspect of a multiplayer game |
| **Frequency of Use** | Multiple times |
| **Channel to Actor** | GUI, Game server |
| **Secondary Actors** | WebSocket server |
| **Open Issues** | 1. What security measures should be added to the system to ensure the in<br>2. What action should be taken if the player fails 5 times to log in?<br>3. What account recovery methods should be in the system? |

# Update Data

Iteration: 1, last modification: March 7, 2025

| **Description** | |
|---|---|
| **Primary Actor** | Player |
| **Goal in Context** | Update the data in the database |
| **Preconditions** | 1. The game is running and the network is established<br>2. The database is running properly<br>3. The player has a stable internet connection<br>4. The player is already logged into the account successfully |
| **Trigger** | 1. The player clicks ‘Update’ button to update the user information<br>2. The player clicks ‘Save’ button to save the current states of the game<br>3. |
| **Scenario 1** | 1. The player makes some changes to the user information<br>2. The player clicks ‘Update’ button to update the information<br>3. The system updates the user information in the database |
| **Scenario 2** | 1. The player wants to save the current state of the game to resume at where it left off<br>2. The player clicks ‘Save’ button to save the states<br>3. The system updates the saved states in the database |
| **Exceptions** | 1. No update is done if the player’s information has no changes.<br>2. Update is rejected if the information is in an unacceptable format<br>3. If the database fails to connect, update cannot be done |
| **Priority** | Critical. This feature is the most fundamental aspect of a multiplayer game |
| **Frequency of Use** | Multiple times |
| **Channel to Actor** | GUI, Game server |
| **Secondary Actors** | WebSocket server |

| | |
|---|---|
| **Open Issues** | 1. What happens if an update fails midway?<br>2. Should changes be logged for tracking and compliance purposes? |

# Retrieve Data

Iteration: 1, last modification: March 7, 2025

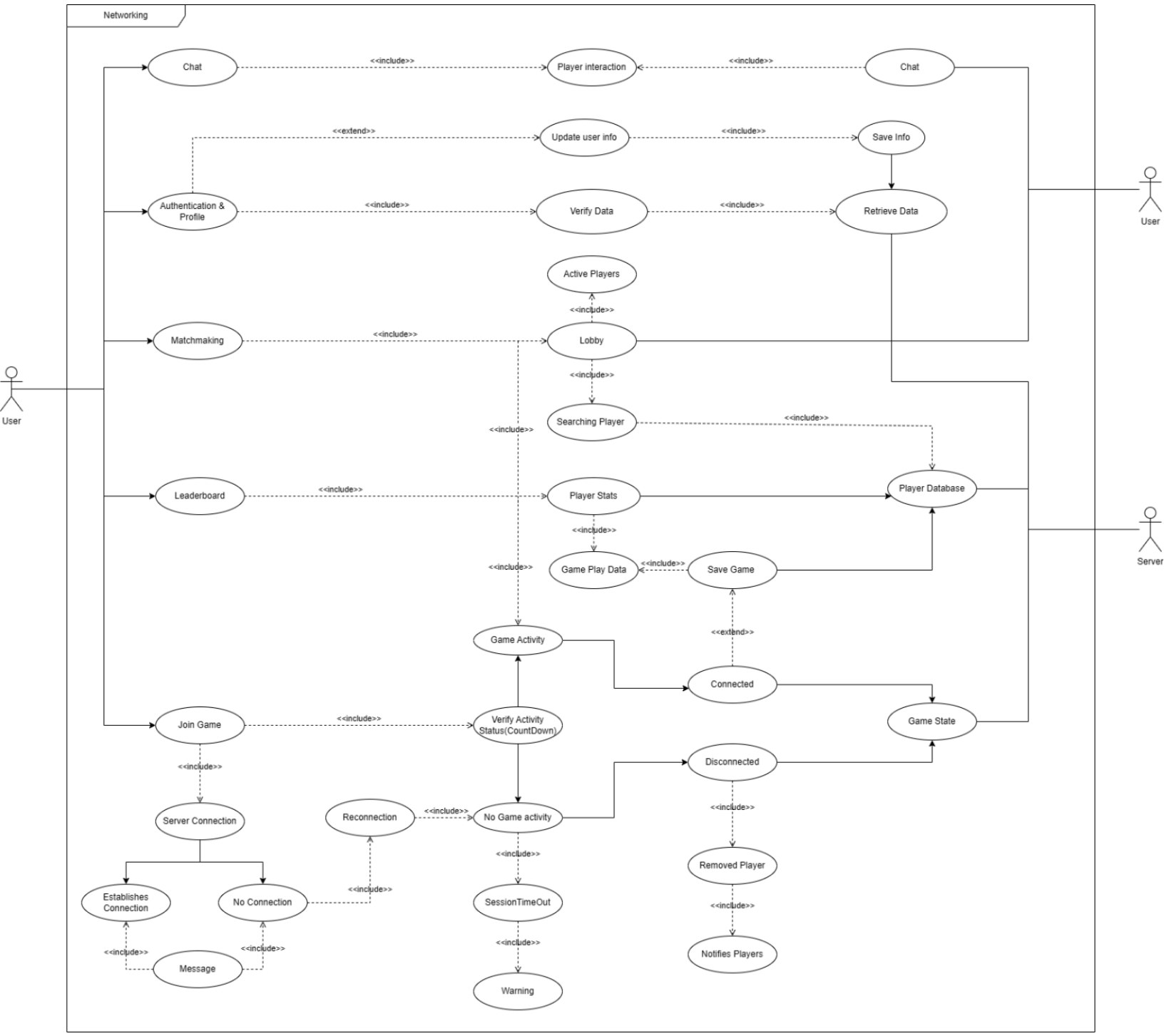
| **Description** | |
|---|---|
| **Primary Actor** | Player |
| **Goal in Context** | Retrieve the data from the database for various purposes |
| **Preconditions** | 1. The game is running and the network is established<br>2. The database is running properly<br>3. The player has a stable internet connection<br>1. The player is already logged into the account successfully |
| **Trigger** | The player make actions which requires the system to retrieve data from the database |
| **Scenario 1** | 1. The player logs into the account<br>2. The player provides the log-in credentials<br>3. The system retrieves data from the database to verify |
| **Scenario 2** | 1. The player opens the leaderboard<br>2. The system retrieves data regarding the leaderboard from the database |
| **Scenario 3** | 1. The player starts a game and wait for other players to join<br>2. The system accesses the database to find the players whose ranks match with the player who is in the game lobby |
| **Exceptions** | 1. The requested data does not exist in the database<br>2. The user lacks the necessary permissions to retrieve the data<br>3. The request contains incorrect or malformed filters |
| **Priority** | Critical. This feature is the most fundamental aspect of a multiplayer game |
| **Frequency of Use** | Multiple times |
| **Channel to Actor** | GUI, Game server |
| **Secondary Actors** | WebSocket server |

| | |
|---|---|
| **Open Issues** | 1. Should frequently accessed data be cached to improve performance?<br>2. Should data retrieval be logged or monitored for unauthorized access? |

# In-Game Chat System

Iteration: 1, last modification: March 7, 2025

| **Description** | |
|---|---|
| **Primary Actor** | Player |
| **Goal in Context** | Enable real-time communication between players in a game session. Ensure chat messages are delivered efficiently and securely. |
| **Preconditions** | 1. Players are connected to the game server<br>2. The game session is active<br>3. The chat system is enabled |
| **Trigger** | A player types and sends a message in the in-game chat |
| **Scenario** | 1. The player enters a message in the chat box<br>2. The client sends the message to the GameServerManager using WebSockets.<br>3. The GameServerManager processes and filters the message (e.g., checks for inappropriate content).<br>4. The server broadcasts the validated message to all players in the session.<br>5. Each player's client receives the message and displays it in the chat window. 6. |
| **Exceptions** | 1. Network failure: The message is delayed or not sent.<br>2. Server overload: Chat messages experience a delay.<br>2. Inappropriate content detected: The message is blocked or censored. |
| **Priority** | High (important for player communication and engagement) |
| **Frequency of Use** | Multiple times per game session. |
| **Channel to Actor** | In-game chat UI. |
| **Secondary Actors** | GameServerManager, Other players |
| **Open Issues** | 1. Should chat history be stored after the game ends?<br>2. Should players be able to mute other players?<br>3. How should spam messages be handled? |


Networking
Chat
<<include>>
Player interaction
<<include>>
Chat
<<extend>>
Update user info
<<include>>
Save Info
Authentication & Profile
<<include>>
Verify Data
<<include>>
Retrieve Data
User
Active Players
<<include>>
Matchmaking
<<include>>
Lobby
User
<<include>>
<<include>>
Searching Player
<<include>>
Leaderboard
<<include>>
Player Stats
Player Database
<<include>>
<<include>>
Game Play Data
<<include>>
Save Game
Server
<<include>>
Game Activity
<<extend>>
Connected
Join Game
<<include>>
Verify Activity Status(CountDown)
Game State
<<include>>
Disconnected
Server Connection
Reconnection
<<include>>
No Game activity
<<include>>
<<include>>
Removed Player
Establishes Connection
No Connection
<<include>>
SessionTimeOut
<<include>>
<<include>>
<<include>>
Message
<<include>>
Warning
<<include>>
Notifies Players