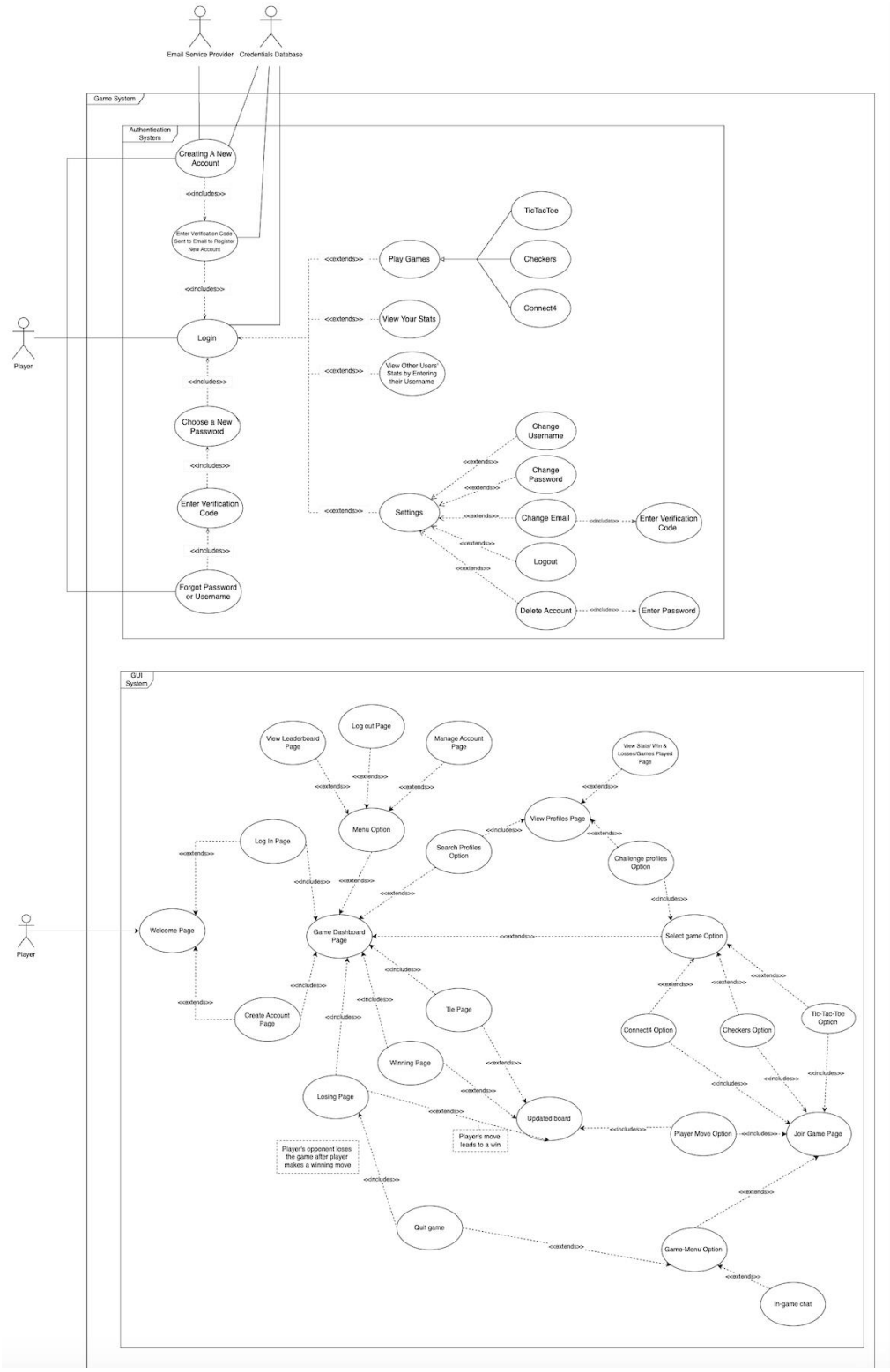
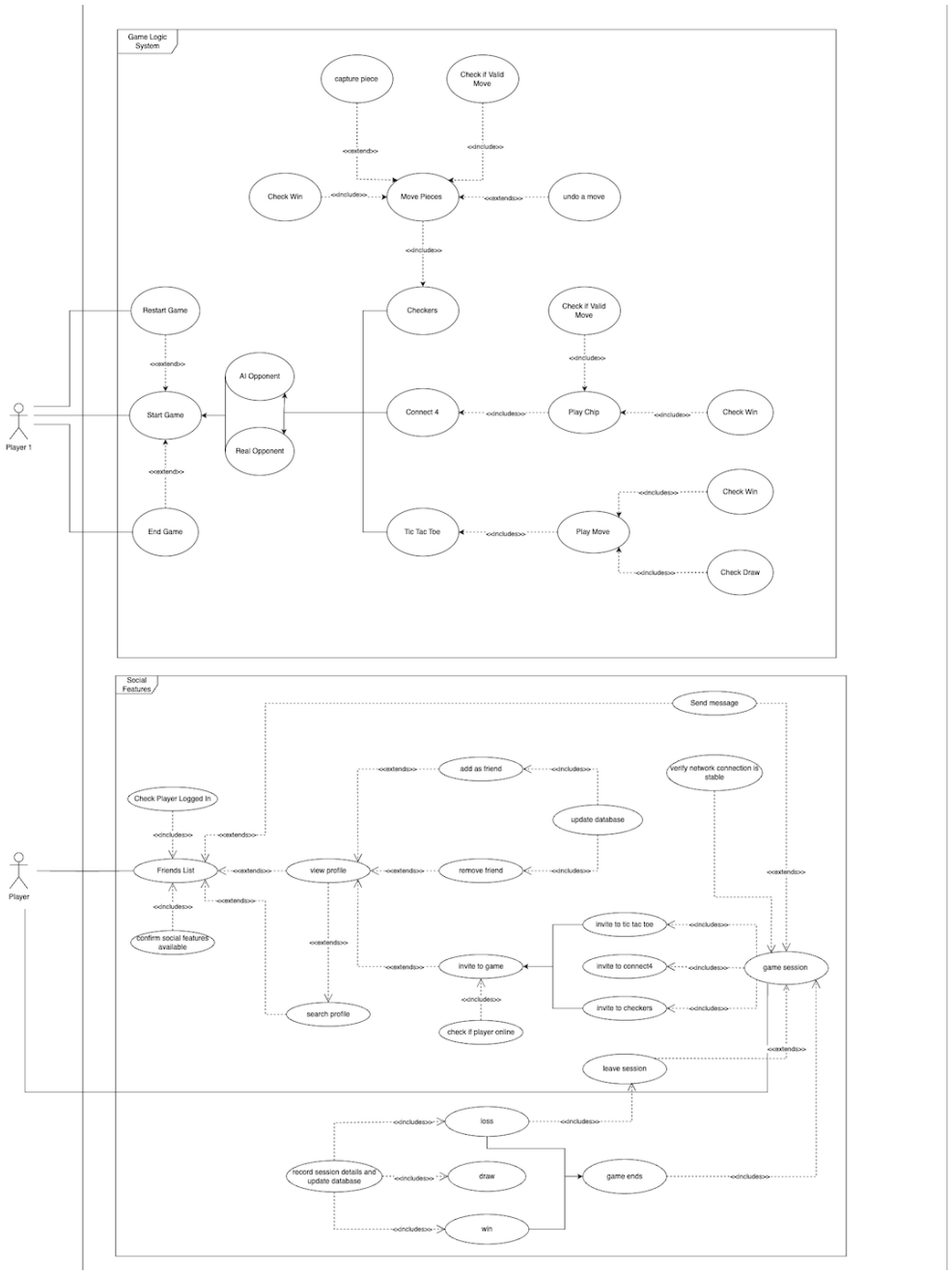
Use Case Diagram




Game Logic System
capture piece
Check if Valid Move
<<extend>>
<<include>>
Check Win
<<include>>
Move Pieces
<<extend>>
undo a move
<<include>>
Restart Game
Checkers
Check if Valid Move
<<extend>>
<<include>>
AI Opponent
Player 1
Start Game
Connect 4
<<includes>>
Play Chip
<<include>>
Check Win
Real Opponent
<<extend>>
<<includes>>
Check Win
End Game
Tic Tac Toe
<<includes>>
Play Move
<<includes>>
Check Draw
Social Features
Send message
<<extends>>
add as friend
<<includes>>
verify network connection is stable
Check Player Logged In
update database
<<includes>>
<<extends>>
Player
Friends List
<<extends>>
view profile
<<extends>>
remove friend
<<includes>>
<<extends>>
<<includes>>
invite to tic tac toe
<<includes>>
confirm social features available
<<extends>>
<<extends>>
invite to game
invite to connect4
<<includes>>
game session
search profile
<<includes>>
invite to checkers
<<includes>>
check if player online
<<extends>>
leave session
<<includes>>
loss
<<includes>>
record session details and update database
<<includes>>
draw
game ends
<<includes>>
<<includes>>
win

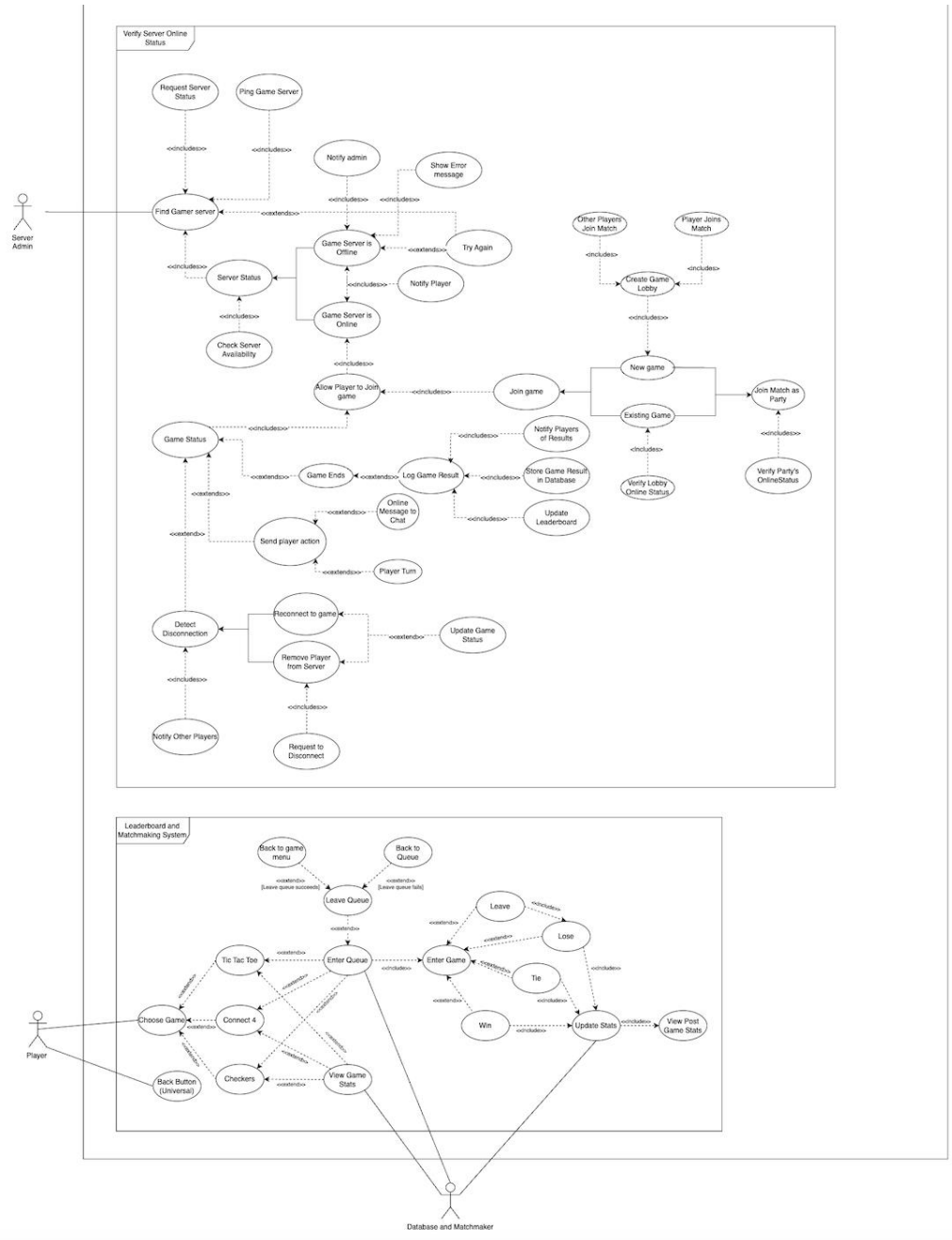


Figure 1: Use case diagram of the entire video game platform, divided into three pages. Different aspects are compartmentalized to help breakdown complexity and make it more readable.

Use Case Descriptions

Game Logic Team Use Case Descriptions:

Use Case: Load Game of Connect Four
Iteration: First
Primary Actor: Player, Player_2
Goal in Context: To load the connect four game so that a player or two players can play the game.
Preconditions:
- The player must already be on the Online Multiplayer Board Game Platform
- The player must have enough RAM to run the game
Trigger: A player chooses to play connect four on the platform
Scenario:
1. The player clicks the option to play connect four on the online platform
2. The player selects their opponent for the match, either single-player or multiplayer
3. The connect four game loads and begins
Postconditions:
- The game starts for the player who selected it
- The condition of the game (whether or not someone has won) is constantly checked
Exceptions:
- The server for connect four players is full
Priority: High. The Connect Four game is one of the initial games for the Online Multiplayer Board Game Platform, and must be up and running for the initial release
When Available: April 11, 2025
Frequency of Use: Dependent on the number of system users, but expected to be high based on being one of few initial games
Channel to Actor: Player interaction through the mouse and keyboard to select the opBon.
Secondary Actors: None
Channel to Secondary Actors: N/A
Open Issues:
- How should the system allow multiple Connect Four games to be run simultaneously?

Use Case: Choose Opponent for Connect Four
Iteration: First
Primary Actor: Player, Player_2
Goal in Context: To allow a player to select their opponent for the game
Preconditions:
- The player must already be on the Online Multiplayer Board Game Platform
- The player must have chosen to load connect four
Trigger: The player chooses to select their opponent for connect four
Scenario:
1. The player selects the option to choose their opponent in connect four

2. The player selects a player they would like to play against from a list of currently online players, chooses one of their friends on the database to request to play with, or chooses to play in single-player against an AI-bot
3. The player's opponent accepts, and the game begins
Postconditions:
- The player gains an opponent for the game
- The game can complete loading
Exceptions:
- There are no available players online to play connect four
- The players opponent leaves while the match is loading
When Available: April 11, 2025
Frequency of Use: As often as the connect four game is expected to be used, which is expected to be high
Channel to Actor: Player uses their mouse and keyboard to select the option to choose their opponent after they have chosen to load the game
Secondary Actors: None
Channel to Secondary Actors: N/A
Open Issues: - The server will need to have an opponent selection system where the player can request multiple opponents, since they may request a friend that is not currently online or a random online opponent that does not want to play with them, and ensure that no player is allowed into a match where they do not have an opponent by accident

Use Case: Restart on Connect Four
Iteration: First
Primary Actor: Player, Player_2
Goal in Context: To allow the player to restart the match they are in at any time
Preconditions:
- The player must be currently in a connect four match
Trigger: The player selects the restart button during the match
Scenario:
1. The player selects the option to restart the game during a match
2. The player is asked to confirm this choice and the opposing player is as well if the player is playing in multiplayer mode, and after doing so the match restarts so that the player is playing against the same opponent as before
Postconditions:
- The game has been reset with the same players involved
Exceptions:
- The opposing player leaves during the reset and as such the same players are not involved in the next match When Available: April 11, 2025
Frequency of Use: Minimal, as the matches are intended to last until a player wins however there will be some scenarios where it is logical to restart the match
Channel to Actor: Player uses their mouse and keyboard to select the restart button while the game is loaded Secondary Actors: None
Channel to Secondary Actors: N/A

Open Issues: - The ability to restart the match in the middle of the game could make some rounds of connect four unreasonably long, and as such a restart limit may be beneficial

Use Case: Exit Game For Connect Four
Iteration: First
Primary Actor: Player, Player_2
Goal in Context: To allow a player to exit the game during the match or when the match is complete
Preconditions:
- The player must already be on the Online Multiplayer Board Game Platform
- The player must have chosen to load a connect four game and picked an opponent in order to have a match to leave
Trigger: The player selects the exit button during the match
Scenario:
1. The player decides they want to leave the current connect four match, and subsequently exits the game using the exit button
2. The player is removed from the game, but is still in the online multiplayer platform
3. If the game is multiplayer and ongoing, the player who is left in the game will now continue the game against an AI bot
4. The database of user statistics is updated for the player who left the game. If the match has ended, both players have left the game and their statistics are updated accordingly
Postconditions: - The player is no longer in the connect four game they were currently playing in
Exceptions:
- The server is malfunctioning and cannot immediately remove the player
When Available: April 11, 2025
Frequency of Use: Fairly frequent, as it occurs at least once a match once the game has ended
Channel to Actor: Player uses their mouse and keyboard to select the exit button
Secondary Actors: None
Channel to Secondary Actors: N/A
Open Issues: - The transition from a human player acting in the game to an AI bot may be confusing for the opponent who has not left the game, they may benefit from a system that informs them of what has happened

Use Case: Play Chip For Connect Four
Iteration: First
Primary Actor: Player, Player_2
Goal in Context: To allow the player to play a chip (either blue or yellow) during their turn of the connect four game.
Preconditions: - The player must have chosen to load a connect four game and picked an opponent
- The match must currently be running and it must be the players turn
Trigger: The opposing player plays their turn or the match begins and the player starts the round.
Scenario:
1. The player is informed it is currently their turn, and that they may now play a chip

2. The player selects the column that they would like to play their chip in, and then a chip of their color is added to the lowest possible spot in that column, as though the chip fell until it landed on another
3. The system checks on whether the player has won with this turn, and then switches back to the opponents turn if the game is not won
Postconditions:
- The player has one more chip on the board than they had at the start of their turn
- The players pieces have been checked to ensure that if the player did win on their previous turn, the game has ended
Exceptions: - The game has ended abruptly so the player will no longer play on their turn
When Available: April 11, 2025
Frequency of Use: Incredibly often, this case will occur each timeme a chip is played
Channel to Actor: Player uses their mouse and keyboard to select the board to play their chip
Secondary Actors: None
Channel to Secondary Actors: N/A
Open Issues: - A time limit on how long a turn can take should be considered so that if one of the players is not participating, the game ends and the other player's statistics are not faulted

Use Case: Check Win on Connect Four
Iteration: First
Primary Actor: Player, Player_2
Goal in Context: Checks whether or not the game has been won in the last turn and acts accordingly
Preconditions:
- A Connect Four game must be currently running
- A player must have just taken their turn
Trigger: A player places their chip, prompting the game to check if they have won
Scenario:
1. A player drops a new chip into the board during their turn
2. The connect four game checks whether the player won the game with that turn
3. If the player won the game, then the player, or both players if the game is being played in multiplayer mode, exit the game and the statistics of the database are updated. Otherwise, the game continues running
Postconditions: - The game has either ended or the opposing player is allowed to play their turn
Exceptions:
- The game ended immediately after a player dropped their chip due to players exiting the game so the feature to check whether the game has been won is not utilized
When Available: April 11, 2025
Frequency of Use: Incredibly often, since it is included in Play Chip which will occur every time a chip is played Channel to Actor: The Play Chip use case includes the Check Win use case, which is how it is utilized to check whether a player has won each turn
Secondary Actors: None
Channel to Secondary Actors: N/A
Open Issues:
- How should the checking function for whether a player has won connect to the database to update a player's statistics?

Use Case: Choose Move for Connect Four
Iteration: First Primary Actor: Player_2 (only when Player_2 is an AI bot)
Goal in Context: To choose a move for the AI bot to play when a user is playing on single-player
Preconditions:
- Single-player mode is chosen so an individual is playing against an AI bot
- It is the AI bot's turn to play
Trigger: The human player completes their turn and does not win, prompting the AI bot to take it's turn.
Scenario:
1. The AI picks a row to drop its chip that is not full
2. Once the AI chooses a row, it plays the chip and the game checks whether or not it has won with that move Postconditions:
- The AI bot has added a chip to the board
- It is the human players turn
Exceptions:
- The game has ended abruptly due to the human player leaving so the AI bot is no longer required for gameplay
When Available: April 11, 2025
Frequency of Use: Will be used when single-player mode for the connect four game is implemented, which is expected to be quite frequently
Channel to Actor: The AI bot communicates directly with the computer
Secondary Actors: None
Channel to Secondary Actors: N/A
Open Issues: - Should the bot be optimized to choose the best move, or to give the users a beZer chance of winning should it simply pick a random move?

Use Case: Load Tic Tac Toe Game
Iteration: 1
Primary Actor: Player
Goal in Context: To load the tic tac toe game.
Preconditions:
● Player has accessed an online multiplayer game platform
● Tic tac toe is available to play
Trigger: Player selects the option to load the game.
Scenario:
1. The player selects "tic tac toe" to play.
2. The game system will load the game.
Post Conditions: The game mode selection screen appears allowing the player to choose the desired game mode they want to play.
Exceptions: The system fails to load the game.
Priority: High priority, this is a game that one of the online multiplayer games that must be available on the board game platform

When Available: Always available
Frequency of Use: Used once every game session, if a player decides they want to play tic tac toe
Channel to Actor: The user interface, using a mouse and/or a keyboard
Secondary Actors: None
Channel to Secondary Actors: N/A
Open Issues: How should the game deal with a player abandoning the game midway through? Should the game set a default mode if a game mode isn't selected? How should the system allow multiple games to be played at the same time? How should the system handle network issues when attempting to load into the game?

Use Case: Choose game mode for Tic Tac Toe
Iteration: 1
Primary Actor: Player
Goal in Context: To allow the player to choose between single player and multiplayer game mode.
Preconditions: The game mode selection screen is displayed
Trigger: The player loads into the game
Scenario:
1. The player selects single player mode or multiplayer mode.
2. The system prepares the tic tac toe game board.
3. If the single player mode is selected:
    a. The player and the bot are both assigned either an X or O.
4. If multiplayer mode is selected:
    a. The system allows the player to choose who they want to play against
    b. The player's are randomly assigned either an X or an O
5. The game display's who starts the game.
Post Conditions: The player takes turns making a move against the AI bot or another player
Exceptions:
● The system fails to load the game with the AI bot
● Fails to connect the AI bot
● Encounters some connection issues trying to find an online match.
● System fails to connect to the selected game mode
Priority: High priority
When Available: Always available during a game
Frequency of Use: Used once during a game session
Channel to Actor: The user interface, using a mouse and/or keyboard
Secondary Actors: None
Channel to Secondary Actors: N/A
Open Issues: How does the system determine the AI difficulty? What happens when the AI fails to make a move when it is their turn? How should the system handle network issues when the AI won't connect? If a player disconnects, can the player switch to an AI opponent? Should the game provide a communication feature? How should the system handle network issues when players won't connect or disconnect?

Use Case: Make a move in Tic Tac Toe
Iteration: 1
Primary Actor: Player
Goal in Context: To place the player's mark (X or O) in the selected cell in the game board.
Preconditions:
● The Game has started.
● It is the player's turn to a move
Trigger: There are still empty slots and the player clicks on an empty slot in the game board.
Scenario:
1. The player selects an empty slot on the game board.
2. The system places the player's mark on the selected slot.
3. The system updates the game board and switches turns to the other player.
Post Conditions:
● The selected slot is updated with the player's mark.
● The system is prepared for the next player's move.
Exceptions: The player selects a non-empty slot, and the system denies the move.
Priority: High priority
When Available: Always available during a game
Frequency of Use: Used multiple times a game
Channel to Actor: The user interface, using a mouse and/or keyboard
Secondary Actors: None
Channel to Secondary Actors: N/A
Open Issues: Should the game implement a time limit for making a move? Will the player be able to undo a move? What happens if a player chooses an invalid slot in the game board?

Use Case: Check if a player won the game for Tic Tac Toe
Iteration: 1
Primary Actor: Game system
Goal in Context: To check if a player has won the game by achieving three of the same mark in a row, column or diagonal.
Preconditions:
● The player has completed their turn
● There are at least 5 moves made on the game board.
Trigger: A player has completed their turn.
Scenario:
1. Player makes moves on the game board leaving their marks on the played slots (X or O)
2. A player has at three of the same mark in a row, column or diagonal
3. The game system verifies the win
Post Conditions: The game verifies a player has won and prepares to end the game
Exceptions: The system fails to evaluate a win (anything other then 3 of the same mark in a row, column or diagonal)
Priority: High priority as this determines the outcome of the game session

When Available: Always available during a game
Frequency of Use: Used after a player move, if valid
Channel to Actor: Automatic System Procedure
Secondary Actors: None
Channel to Secondary Actors: N/A
Open Issues: How should the game handle network issues or other issues if a win is not detected? How often should the game check for a win condition?

Use Case: Check if the game is a draw for Tic Tac Toe
Iteration: 1
Primary Actor: Game System
Goal in Context: To check if the game has ended in a draw, where all slots are filled with players' marks (X or O)
Preconditions:
● The player has completed their turn
● All slots are filled with players' marks (X or O)
Trigger: The player completes their turn, and no win condition is met
Scenario:
1. Player makes moves on the game board leaving their marks on the played slots (X or O)
2. The game board evaluates to see if all slots are filled
3. If all game slots are filled with no signs of a player's mark that is three in a row, column, or diagonal
4. If the board is not full, the next player makes a move
Post Conditions: The game verifies it ends with a draw and prepares to end the game
Exceptions: The system fails to evaluate a draw
Priority: medium as this determines the outcome of the game session if a player does not win
When Available: Always available during a game session
Frequency of Use: Used after a player move, if valid
Channel to Actor: Automatic System Procedure
Secondary Actors: None
Channel to Secondary Actors: N/A
Open Issues: Should the game suggest a new match or rematch option? Should the game reset automatically after the draw?

Use Case: End game of Tic Tac Tie
Iteration: 1
Primary Actor: System
Goal in Context: To conclude the game session and provide the player with end-game options
Preconditions: Players have completed all of their turns ending the game with a winner or a draw
Trigger: The game detects a win or a draw
Scenario:
1. The system declared the results of the game

2. The system offers the player with an option to play again(the system resets the game board) or exit the game(the systems navigates the player out of the game)
Post Conditions: The game board has reset or the player has exited the game
Exceptions: The game system fails to reset the game board or fails to exit
Priority: High priority as this determines the outcome of what to do next at the end of the game
When Available: Always available at the end of the game session
Frequency of Use: Used once at the end of every game session
Channel to Actor: Automatic System Procedure
Secondary Actors: None
Channel to Secondary Actors: N/A
Open Issues: Is it possible for a player to save results or statistics of the game? What should happen if a player disconnects during the ending of the game?

Use Case: Exit Game of Tic Tac Toe
Iteration: 1
Primary Actor: Player
Goal in Context: To exit the game.
Preconditions: Game is in progress or completed.
Trigger: The player selects the exit option.
Scenario:
1. The player chooses the exit game option .
2. The system closes the game interface.
Post Conditions: The player is returned to the main menu or exits the game interface.
Exceptions: The game system fails to close properly.
Priority: High priority as this allows the player to exit the interface whenever they desire.
When Available: Always available during the game
Frequency of Use: When the player wants to leave the game
Channel to Actor: Automatic System Procedure
Secondary Actors: None
Channel to Secondary Actors: N/A
Open Issues: How should the game handle unsaved progress if the player exits suddenly? Should the game prompt the player with a confirmation message to exit the game? Should the game provide an option to save statistics?

Use Case: Restart Game of Tic Tac Toe
Iteration: 1
Primary Actor: Player
Goal in Context: To restart the game after a win, draw.
Preconditions: When the current game session has ended.
Trigger: The player selects the restart option.
Scenario:
1. The player clicks on the restart option.

2. The system resets the game board and restarts the game session.
Post Conditions: The game will reset the gameboard to prepare for a new game.
Exceptions: The game system fails to reset the game board.
Priority: Medium priority as it is not necessary to complete the game, but gives the player options after the game ends.
When Available: Always available after the game ends
Frequency of Use: When a new game session is requested
Channel to Actor: Automatic System Procedure
Secondary Actors: None
Channel to Secondary Actors: N/A
Open Issues: Should the game offer multiple options when restarting, for instance, changing the game mode? Should the game provide an option to save statistics? Should the game prompt the player with a confirmation message to restart the game?

Use Case: Game of Checkers
Actors:
● Player 1: The first player, usually assigned either the darker pieces or light pieces .
● Player 2: The second player, usually assigned either the lighter pieces or dark pieces. This can be a real player or an AI (CPU).
● System: The game system, which manages moves, validates actions, and enforces rules.
Preconditions:
● A checkerboard is set up with all pieces in their initial positions.
● Each player is either physically present (local game) or connected online.
● If playing against an AI, the system assigns CPU logic for Player 2.
Triggers: ● A player initiates a new game, selecting an opponent (real player or AI).
Main Flow:
1. Game Initialization:
    ○ The system initializes the game board with pieces in their correct starting positions.
    ○ Player 1 starts the game.
2. Selecting Player 2:
    ○ Player 1 chooses whether to play against another real player or the AI.
3. Making Moves:
    ○ Players take turns making moves.
    ○ The system validates moves according to standard Checkers rules:
        ■ Regular pieces move diagonally forward.
        ■ If a jump is available, it must be taken.
        ■ A piece reaching the opposite end of the board is promoted to a king, which can move both forward and backward.
    ○ If Player 2 is AI, it takes input from Player 1 and generates a response move.
4. Undoing a Move:
    ○ A player can undo a move if needed but only if they didn't confirm it (if allowed by game settings).
5. Capturing Pieces:

○ If a player's move results in capturing an opponent's piece, the system removes the captured piece. 6. Ending the Game:

○ The game continues until one player captures all opposing pieces or blocks all available moves.

○ The system declares a winner and provides options for a rematch.

Alternative Flows:

● Invalid Move: If a player attempts an illegal move, the system alerts them and prevents execution.

● Resignation: A player can choose to exit or resign, granting victory to the opponent.

● Draw: If neither player can make a legal move, the game ends in a draw.

● Restarting the Game: Players can restart the game from the beginning.

Postconditions:

● The game result is recorded if applicable.

● Players can exit or start a new game.

Extensions (From the Diagram):

● Move Pieces extends to:

○ Undo Move: A player can reverse their last move.

○ Capture Piece: If a move results in capturing an opponent's piece, the system enforces the capture.

● Takes Player 1 Input extends to:

○ Random Move (For AI): If Player 2 is an AI, the system generates a random valid move.

● Start Game extends to:

○ Restart Game: Players have an option to restart after starting.

● Selecting Player 2: Determines whether Player 2 is a real player or the CPU.

● Exit Game: Ends the session.

Assumptions:

● Players have a basic understanding of Checkers rules.

● The system properly enforces game rules and AI logic (if applicable).

● The game interface functions without technical issues.

GUI Team Use Case Descriptions:

Use Case: Winning Page

Iteration: First Iteration

Primary Actor: Player

Goal in Context: To show the winning page after the player has won the game played.

Preconditions:

● The player is logged into the system.

● The game has been played with an opponent.

Trigger: The player has made a move that leads to victory in the game.

Scenario:

1. The player plays the last move, leading to the end of the game, or the player makes a move that results in victory.

2. The game page changes to display that the player has won.

Postconditions:

● The winning page returns to the main game library dashboard.

● Points have been added to the player's stats.
● The player's profile for games-won stats has been updated.
Exceptions:
● The system is powered off.
● The system fails to load the Welcome page; an error message will be displayed.
Priority: High
When Available: First Iteration
Frequency of Use: Every game, depending on the outcome.
Channel to Actor: Via GUI
Secondary Actors: N/A
Open Issues:
● Should there be an option for a rematch?
● Should there be an option to review the match?

Use Case: Losing Page
Iteration: First Iteration
Primary Actor: Player
Goal in Context: To show the losing page after the player has lost the game played.
Preconditions:
● The player is logged into the system.
● The game has been played with an opponent.
Trigger: The opponent has made a move that leads to victory in the game, making the player lose the game. Scenario:
1. The opponent plays the last move, leading to the end of the game, or the opponent makes a move that results in their victory.
2. The player cannot make another move.
3. The game page changes to display that the player has lost.
Postconditions:
● The losing page returns to the main game library dashboard.
● Points have been added to the opponent's stats.
● The player's profile of games-lost stats has been updated.
Exceptions:
● The system is powered off.
● The system fails to load the Welcome page; an error message will be displayed.
Priority: High
When Available: First Iteration
Frequency of Use: Every game, depending on the outcome.
Channel to Actor: Via GUI
Secondary Actors: N/A
Open Issues:
● Should there be an option for a rematch?
● Should there be an option to review the match?
Use Case: Joining Game

Iteration: First Iteration
Primary Actor: Player
Goal in Context: Join a game from the game library.
Preconditions:
● The player is logged into the system.
● The player selects a game to play.
Trigger: The player has selected a game.
Scenario:
1. The player has logged into the game dashboard.
2. The player can hover and click on one of the clickable game Connect4, Checkers, or Tic-Tac-Toe.
3. The selected game opens for the player to play.
Postconditions:
● Clicking the selected game allows player to begin to play one of the games.
Exceptions:
● The system is powered off.
● The system fails to load the game options page; an error message will be displayed.
Priority: High
When Available: First Iteration
Frequency of Use: Often
Channel to Actor: Via GUI
Secondary Actors: N/A
Open Issues:
● Is there an option to view all players currently playing that game?
● Is there a level/difficulty option?

Use Case: Leaderboard Display
Iteration: First Iteration
Primary Actor: Player
Secondary Actors: N/A
Preconditions:
● The player must be logged into the system and in the Game Dashboard Page.
Trigger:
● The player navigates to the leaderboard interface.
Scenario:
1. The player selects the Leaderboard option from the main menu or interface.
2. The system retrieves the latest ranking data from the database.
3. The system displays the leaderboard with rankings based on MMR, including:
    - ○ Top-ranked players sorted by MMR.
    - ○ Rank tiers displayed for each player:
        - ■ BRONZE = 1
        - ■ SILVER = 2
        - ■ GOLD = 3
        - ■ PLATINUM = 4

        - DIAMOND = 5
        - MASTER = 6
        - GRANDMASTER = 7
    - Player statistics, such as wins, losses, and total score.
    - The user's own rank highlighted within the leaderboard.

4. The player can:
    - Search for specific players.
    - Filter leaderboard results
    - View a player's profile for more details.
5. The player can click an exit button and return to the previous screen.

Postconditions:

- The system successfully displays the ranked leaderboard with the latest data.
- If an error occurs, the player is informed through an appropriate message.

Exceptions:

- If the leaderboard data is unavailable, the system displays an error message.
- If the database is down, the system logs an error and provides a retry option.
- If the player's rank cannot be determined, a placeholder message is displayed.

Priority: High

When Available: First Iteration

Frequency of Use: Multiple times a day, every session

Channel to Actor: Graphical User Interface (GUI)

Channel to Secondary Actors: N/A

Open Issues:

- Should players be able to see historical rankings?
- Should rank progression trends be visualized?

Use Case: Selecting Games from Available Library

Iteration: First Iteration

Primary Actor: Player

Secondary Actors: N/A

Preconditions:

- The player must be logged into the system and in the Game Dashboard Page.

Trigger:

- The player navigates to the game dashboard.

Scenario:

1. The player is logged into their game account.
2. The player can view available games:
    - Tic-Tac-Toe
    - Checkers
    - Connect4
3. The player has the option to select one or multiple games.

Postconditions:

- The system successfully runs the selected game.

Exceptions:
● If the game does not run, the player has to refresh or choose another game.
● Players are limited to three different games.
Priority: High
When Available: First Iteration
Frequency of Use: Multiple times a day, every session
Channel to Actor: Graphical User Interface (GUI)
Open Issues:
● Should players be able to select the same game many times in one session?

Use Case: View Profiles
Iteration: First
Primary Actor: Player
Goal in Context: To allow a player to view another player's profile and check their game statistics, ranking, and match history.
Preconditions:
● The player must be logged into the system.
● The profile being viewed must exist in the database.
● The system must be connected to the game server or data storage.
Trigger: The player selects a profile from the leaderboard, match history, or search function.
Scenario:
1. The player navigates to a list of players (e.g., leaderboard, recent matches, or search results).
2. The player selects a player's profile from the list
3. The system retrieves the player's profile data from the database.
4. The system displays the profile, showing:
    a. Username and avatar
    b. Games played
    c. Win/Loss record
    d. Current ranking
    e. Recent match history
5. The player views the profile and may navigate back or challenge the player
Postconditions:
● The system successfully displays the requested profile.
● If an error occurs, the system informs the user through an appropriate message.
Exceptions:
● Profile data fails to load → The system displays a "Profile could not be loaded" message with a retry option. ● User profile does not exist → The system returns a "User not found" error.
● Database is unreachable → An error message is shown, and the system attempts to reconnect.
Priority: High
When Available: First iteration
Frequency of Use: Multiple times per session
Channel to Actor: Via GUI
Secondary Actors: N/A

Open Issues:
● Should users be able to send messages from the profile screen?
● Should there be privacy settings to limit profile visibility?

Use Case: Tie page
Iteration: First
Primary Actor: Player
Goal in Context: To display the tie page when a game ends with a draw, informing both players of the outcome
Preconditions:
● Game is in progress
● System detects that neither player has won, resulting in a tie
Trigger: The game reaches a state where no moves can lead to a win, leading to a tie
Scenario:
1. Players make their last move
2. The system evaluates the game board and determines that no winning conditions are met.
3. The system triggers the Tie Page.
4. The Tie Page displays a message stating the game ended in a tie.
5. Players have the option to:
    a. Return to the Game Dashboard.
    b. Start a rematch.
    c. Exit to the main menu.

Postconditions:
● System records the match as a tie
● Statistics are updated in the database
● Players are redirected to choose from dashboard, rematch, and main menu
Exceptions:
● Network disconnects before tie is registered → System fails to display tie page
● One player disconnects before the game is finalized → remaining player should still see the tie page
Priority: High
When Available: First iteration
Frequency of Use: Occasionally based on the game outcomes
Channel to Actor: Via GUI
Secondary Actors: N/A
Open Issues: ● Should the game automatically offer a rematch after a tie?

Use Case: Game Menu Option
Iteration: First
Primary Actor: Player
Goal in Context: Allow the user to access in-game options like quitting, resuming, and adjusting settings during an active game session
Preconditions:

● Player is in an active game session
● Game-menu option must be accessible during the gameplay
Trigger: The player selects menu button during an ongoing game
Scenario:
1. The player opens the game menu option by selecting the menu button.
2. The system pauses the game and displays the menu with the following options:
    a. Resume Game – Player returns to the game.
    b. Quit Game – Player can exit the game to the main dashboard.
    c. Game Rules – Displays the rules and objectives of the current game.
3. The player selects an option from the menu.
4. The system executes the chosen action:
    a. If Resume Game is selected → menu closes and resumes game.
    b. If Quit Game is selected → A confirmation pop-up appears to confirm the action.
    c. If Game Rules is selected → The rules are displayed.
Postconditions:
● The player successfully returns to the game or exits the match, based on their selection
● If the player quits, their rankings are updated accordingly
Exceptions:
● Network connection lost while accessing menu → system warns player and attempts to reconnect
Priority: High
When Available: First iteration
Frequency of Use: Occasionally based on the game outcomes
Channel to Actor: Via GUI
Secondary Actors: N/A
Open Issues:
● How should quitting affect the player's ranking?

Use Case: Challenge Profiles
Iteration: First
Primary Actor: Player
Goal in Context: To allow a player to challenge another player to a game from their profile
Preconditions:
● The player must be logged into the system
● The system must be connected to the game server or data storage, and matchmaking functionality has to be enabled.
● The player being challenged must be a registered user
Trigger: The player selects "Challenge" on another player's profile.
Scenario:
1. The player clicks on another player's profile.
2. The system shows a "Challenge" button.
3. The player selects "Challenge" and chooses a game (e.g., Chess, Go, Connect Four).
4. The system sends a match request to the opponent.
5. The opponent accepts or declines the challenge.

a. If accepted → The system initiates the game session.
b. If declined → The system notifies the challenger.

Postconditions:
● If the challenge is accepted, the game starts.
● If declined, the system returns to the profile view
● If opponent doesn't respond, request expires after time out

Exceptions:
● Opponent rejects the challenge → player is notified that challenge is rejected.
● Unstable network → An error message is shown, system alerts player.

Priority: High
When Available: First iteration
Frequency of Use: Multiple times per session
Channel to Actor: Via GUI
Secondary Actors: N/A

Open Issues:
● Should users be able to send requests if the opponent is already in a game?
● Should there be limits on how many challenges you can send in a short period of time?

Use case: Game Dashboard
Iteration: First
Primary Actor: Player
Goal in context: To let the player see the games available in the game dashboard

Preconditions:
1. The player is logged into the system
2. The system is running and showing the games in the game dashboard

Trigger: The player has access to the game board after logging in

Scenario:
1. The player logs into the system
2. The system directs the player to the game dashboard
3. The system shows all the available games to the player
4. The player views the games available in the game dashboard

Post conditions:
1. The player can successfully see the lists of games available in the dashboard.
2. The system is waiting and ready for further interaction, for example, selecting a game from the available games in the dashboard.

Exceptions:
1. The games are not available/showing up
2. System is down, fails to retrieve the games

Priority: High
When available: First iteration
Frequency of use: Multiple times a day, every session
Channel to actor: via GUI
Secondary actors: N/A

Channel to secondary actors: N/A
Open issues:
1. Should similar games be listed under categories?

Use case: In-game chat
Iteration: First
Primary Actor: Player
Goal in context: The player can successfully type and send messages in the in-game chat during the match
Preconditions:
1. The player is logged into the system
2. The player has joined a game
Trigger: The player selects the “In-game chat” from the Game-Menu
Scenario:
1. The player joins a game
2. The player clicks on the “Game-Menu option”
3. The player selects the “In-game Chat” option.
4. The player types a message
5. The player sends the message
Post conditions:
1. The player’s message is shown in the chat window
2. The opponent can see the message.
Exceptions:
1. The player is playing against a bot.
Priority: Medium
When available: Second iteration
Frequency of use: Multiple times a day, every session
Channel to actor: via GUI
Secondary actors: N/A
Channel to secondary actors: N/A
Open issues: 1. Should the system have a filter, in order to make sure no mean comments are made?

Use case: Game-Menu Option
Iteration: First
Primary Actor: Player
Goal in context: The player can successfully access the different options such as quit and in-game chat from the Game-menu option
Preconditions:
1. The player is logged into the system
2. The player has joined a game
Trigger: The player selects the “Game-Menu Option” in the game
Scenario:
1. The player joins a game

2. The player clicks on the “Game-Menu option”
Post conditions:
1. The player can see further options after clicking on “Game-Menu Option”
Exceptions:
1. The Game-menu fails to open
Priority: High
When available: First iteration
Frequency of use: Multiple times a day, every session
Channel to actor: via GUI
Secondary actors: N/A
Channel to secondary actors: N/A
Open issues:
1. Should the game be paused when the player clicks on the Game-menu option?

Use case: Manage Account
Iteration: First
Primary Actor: Player
Goal in context: To allow the player to manage their account by updating their account details, such as notifications or privacy settings, language preferences, and changing their username or password
Preconditions:
1. The player is logged into the system
2. The player is in “Game Dashboard”
Trigger: The player clicks on “Manage Account” from the settings
Scenario:
1. The player is in the Game Dashboard
2. The player clicks on the “Settings Option”
3. The player selects the “Manage Account” option
4. The player selects an option and does the appropriate details required by the option they selected.
Post conditions:
1. The player’s account gets updated from the changes made.
Exceptions:
1. The new information entered by the player is not valid
2. Fail to save changes due to a server error
Priority: Medium
When available: First iteration
Frequency of use: Multiple times a day, every session
Channel to actor: via GUI
Secondary actors: N/A
Channel to secondary actors: N/A
Open issues: N/A

Use case: Log out

Iteration: First
Primary Actor: Player
Goal in context: To allow the player to log out of the game platform
Preconditions:
1. The player is logged into the system
2. The player is in “Game Dashboard”
Trigger: The player selects the “Log out” option from the settings
Scenario:
1. The player is in the Game Dashboard
2. The player clicks on the “Settings Option”
3. The player selects the “Log out” option
Post conditions:
1. The player is successfully logged out of the game platform
2. The session is closed
Exceptions:
1. The player fails to log out due to a server error
Priority: High
When available: First iteration
Frequency of use: Multiple times a day, every session
Channel to actor: via GUI
Secondary actors: N/A
Channel to secondary actors: N/A
Open issues: N/A

Use case: Welcome Page
Iteration: First
Primary Actor: Player
Goal in context: To initially welcome the player to the gaming dashboard and encourage them to interact with the system.
Preconditions:
1. System must be powered on
2. The system must be in initial state and connected to the Gaming Dashboard
3. Game Dashboard is functional
Trigger: Connected to the Gaming Dashboard: Once connected the system will present the initial welcome screen,
Scenario:
1. Player connects to the gaming dashboard via network
2. System loads Gaming dashboard server, where the welcome screen is initially presented.
Post conditions:
1. The customer will be prompted to input their password and user.
Exceptions:
1. System is powered off
2. Unable to connect to the gaming dashboard server.

3. System fails to load the Welcome page, therefore an error message will be displayed,
Priority: low
When available: second iteration
Frequency of use: Once per session
Channel to actor: Through Graphical User Interface, the Welcome screen can be presented to the player.
Secondary actors: N/A
Channel to secondary actors: N/A
Open issues:
1. Should there be an option to proceed as a guest?
2. Will there be accommodations for people with disabilities (i.e. Color blind, blind)

Use case: Quit Option
Iteration: First
Primary Actor: Player
Goal in context: To allow the player to quit the game whenever the player wishes.
Preconditions:
1. There must be an active game.
2. The player must be able to access the quit option.
Trigger: The player presses the quit option when they choose to quit the game.
Scenario:
1. Player wishes to confirm to quit.
    a. Player is currently in an active game and wishes to quite
    b. Player chooses the quit option on the screen.
    c. Player confirms their selection when prompted with "Are you sure?"
    d. Player loses the game and successfully quits.
2. Player does not confirm to quit
    a. Player is currently in an active game and wishes to quite
    b. Player chooses the quit option on the screen.
    c. Player changes their mind and selects "no" when prompted with "Are you sure?"
    d. Player continues the game normally.
Post conditions:
    1. Players must confirm their selection when prompted with "Are you sure?"
    2. If the player declines the game continues normally.
Exceptions:
    1. If the game crashes during the game session.
    2. Player loses network connection during the game session.
    3. The system fails to register teh quit function.
Priority: High
When available: first iteration
Frequency of use: once per session
Channel to actor: Through Graphical User Interface, the quit option will be a button for the player to click. Secondary actors: System
Channel to secondary actors: When connected to a game in the gaming dashboard.

Open issues:
1. How are we going to deal with the opponent's game becoming disrupted?
2. Should we allow the player to quit when the game is about to end?
3. Should the player be banned from quitting after a certain amount of quits?

Use case: Create account
Iteration: First
Primary Actor: Player
Goal in context: Option to allow players to create an account in the game dashboard.
Preconditions:
1. System is connected to the internet.
2. Player is currently on the welcome page.
Trigger: Clicking the create account option on the welcome page.
Scenario:
1. Player connects with the Game Dashboard.
2. Player wants to make an account
3. Player selects "Create Account".
Post conditions:
1. Player must input correct information
2. Players must not already have an account with the same information.
Exceptions:
1. If the network is unstable.
2. Player is on the Game Dashboard.
3. Player already has an account with the same account information.
Priority: High
When available: first iteration
Frequency of use: once
Channel to actor: Option to select from on the graphical user interface.
Secondary actors: N/A
Channel to secondary actors: N/A
Open issues:
1. Should one player be allowed to make multiple accounts
2. Should there be an email verification?
3. What if someone who a is not fluent in English cannot read the requirements of "Create Account"

Use case: Login
Iteration: First
Primary Actor: Player
Goal in context: Allows the user to access an option to log in.
Preconditions:
1. Players must be on the welcome page.
2. Player must be connected to the network.

Trigger: Clicking the login account option on the welcome page.
Scenario:
1. Player connects with the Game Dashboard.
2. Player wants to Login to their account
3. Player selects "Login".
Post conditions:
1. Player must input correct username and password
2. Players must not already have an account with the same information.
Exceptions:
1. If the network is unstable.
2. Player is on the Game Dashboard.
3. Field(s) are left empty by the player.
Priority: High
When available: first iteration
Frequency of use: once per session
Channel to actor: Option to select login from on the graphical user interface.
Secondary actors: N/A
Channel to secondary actors: N/A
Open issues:
1. Do we block the player from entering the wrong username password?
2. Do we have an option for forgetting passwords?

Use case: Placing/Moving Pieces
Iteration: First
Primary Actor: Player
Goal in context: Allows players to place or move pieces around in order for them to play the game. It gives functionality to Tic-Tac-Toe, Checkers, and Connect 4.
Preconditions:
1. There must be an active game (i.e Tic-Tac-Toe, Checkers, and Connect 4)
2. Moving or placing of the piece must be a valid play:
    a. Player move must be on the game grid.
    b. Player move must obey game rules (how the pieces can be moved).
    c. It is the players turn to move the piece.
3. Network is connected for the game to register the player move.
Trigger: The player selects the spot they would like to move the piece to.
Scenario:
1. Player makes a move:
    a. Player is playing Tic-Tac-Toe:
        i. Players turn to place their piece.
        ii. Player selects an empty space on the grid.
        iii. The system places "X" or "O" to register the player's choice.
    b. Player is playing Connect 4:
        i. Players turn to place their piece.

ii. Player selects a column that has empty spaces.
iii. The Player's piece (either blue or red) gets placed into the lowest available row in that column.

c. Player is playing Checkers:
i. It is the player's turn to move a piece.
ii. Player selects one of their pieces and selects a valid diagonal move.
iii. If the player jumps over the opponent's piece(s), their pieces are removed.
iv. If the piece is moved to the opponent's end of the grid, their piece gets promoted to "King".

2. If the move is valid the board gets updated, otherwise the player is prompted to make a valid move.
3. Once the player has made their move, the system checks if the player won or drawn the game, otherwise control switches to the opponent.

Post conditions:
1. Game must update after the player has successfully made a move.
2. If the game has been won or drawn, the system should display the results
3. Control should shift to the opponent if the game has not ended after the players move.

Exceptions:
1. Invalid moves should not be accepted by the system
a. Placing outside the grid
b. Placement does not align with game rules
c. Placing in a occupied space
2. If the player has quit upon their turn, they should not be eligible to make a move.
3. Network connection is unstable to register the move.

Priority: High
When available: first iteration
Frequency of use: many times per session
Channel to actor: Through Graphical User Interface, the player can select where they would like to make their move by selecting the piece and the spot, or by simply selecting the spot.
Secondary actors: N/A
Channel to secondary actors: N/A
Open issues:
1. Should there be a penalty if the player selects too many invalid moves?
2. Should there be a time limit on how long a player can take to make their move?
3. Should there be hints/recommendations given to the player on where they should make their move?

Networking Team Use Case Descriptions:

Use Case: Immersion
Iteration: 1
Primary Actor: User
Goal in context: Keep players engaged with environment so that they are not constantly aware of the fact that they are engaging in a simulated environment, and can participate in the games to their utmost capacity. Preconditions:
● Player is logged in

● Game environment is fully functional
● Online status has been verified and made fully operational
Trigger: Players will want to play on the platform for longer periods of time, and have minimal complaints about the platform. This will be beneficial for the culture surrounding the online interactions, as well as the mental wellbeing of the developers
Scenario:
1. Player selects a game, engaging with responsive UI
2. Player has options displayed to play against AI or another person
3. Online chat features create social incentive to keep playing
4. Competitive rankings push players to perform more intelligently/regularly
5. Misbehaving players will be reported and dealt with
Post conditions:
● Player desires to keep playing on a regular basis
● Moderation staff have minimal issues with online misbehavior
● Development staff can implement patches without backlash
Exceptions:
● User cannot connect to Internet for competitive play
● User is overly entitled and contributes to a negative atmosphere
● Servers cannot sustain proper gameplay performance
● Support for the online features and bug fixes slows to a crawl
Priority: Required for platform's longevity - paramount
When available: Always
Frequency of use: Whenever players are on the game platform
Channel to actor: GUI
Secondary actors: N/A
Channel to secondary actors: N/A
Open issues:
● Identifying the gameplay loop which is the most "fun"
● Moderating game environment without being overly paternalistic
● Responding to player feedback in terms of what works/doesn't work

Authentication Profile Team Use Case Descriptions:

Use Case: Creating a New Account
Iteration: 1
Primary Actor: Player
Goal in Context: The player wants to create a new account with an email, username and password which would allow them to play available games.
Preconditions:
- The system is online and operational.
- The login interface is accessible.
- The player has a valid email address.
Trigger: The player clicks the "Create A New Account" button on the login interface.
Scenario:

1. The player navigates to the login interface of the multiplayer game platform.
2. The player selects “Create A New Account”.
3. The player enters a valid email address, chooses a unique username and a strong password.
4. The system validates the format of the email address.
5. The system validates if the username is available.
6. The system validates if the password meets security requirements, if there are any in place.
7. If any of the input is invalid, the player is prompted to re-enter a valid one.
8. After entering all valid inputs, the system sends a verification code to the email address provided.
9. The system prompts the player to check their email address and is taken to the next interface to enter the verification code.

Postconditions:
- A verification code is sent to the email address provided.
- The player is taken to a new interface where they can enter the verification code.

Exceptions:
- The system displays an error message if the player enters a wrong email format, or a username that has already been taken by an existing user, or a weak password, and is prompted to re-enter the details.
- The email address does not exist.

Priority: Highest
When Available: Mar. 7, 2025
Channel to Actor: GUI - Login Interface
Secondary Actors: Email service provider
Channel to Secondary Actors: Email API
Open Issues:
- Should there be any password requirements?
- If the user enters an existing username, should the system suggest a similar yet unique username?

Use Case: Enter Verification Code to Register the Account
Iteration: 1
Primary Actor: Player
Goal in Context: The player wants to enter the verification code sent to their email to verify it to the system and activate their account.
Preconditions:
- The player is in process of registering a new account
- A verification code has been sent to the email address provided

Trigger: The player has clicked “Register” after entering valid email, username and password in the process of creating a new account.
Scenario:
1. The system has sent a verification code to the email address provided.
2. The player is taken to the verification interface.
3. The player checks their email and enters the verification code in the verification page.
4. The system checks if the code is valid and unexpired.
5. If valid, the system successfully registers the player’s account with the email address, and username and password chosen before, in the database.

6. The player is redirected to the login page.
Postconditions:
- The account has been activated and registered in the database.
- The user can login with the email/username and password.
Exceptions:
- The system prompts an error message if the verification code is incorrect.
- The code entered is expired.
Priority: Highest
When Available: Mar. 7, 2025
Channel to Actor: GUI - Verification interface extending the registration and login interface
Secondary Actors: Database
Channel to Secondary Actors: Database API
Open Issues:
- Should the player be allowed to re-enter the verification code if they enter an incorrect one?
- Should players be allowed to request a new code or do they have to go through registration again?
- What is the expiration time limit for the code?

Use Case: Delete Your Account
Iteration: 1
Primary Actor: Player
Goal in Context: The player wants to permanently delete their multiplayer game platform account, deleting all the data, personal information and progress associated with that account.
Preconditions:
- The player must have a registered account.
- The player must be logged into their account.
- The player must remember the password of their account.
Trigger: The player selects “Delete Your Account” in the settings page.
Scenario:
1. The player goes to the settings page.
2. The player selects “Delete Your Account”.
3. The system prompts the player to enter the password of their account for verification purposes.
4. If the password is entered correctly, the system prompts the player with confirmation warning of the loss of associated data and progress.
5. Player confirms the warning.
6. The system permanently deletes the account from the database and any data associated with the account. 7. The player is redirected to the login page.
Postconditions:
- The account is permanently deleted and the player cannot log in using the deleted account.
- The player is taken back to the login page.
Exceptions:
- The player enters an incorrect password for verification.
- The player does not proceed with the account deletion after receiving the confirmation warning.
Priority: Medium

When Available: Mar. 7, 2025
Channel to Actor: GUI Interface for Settings
Secondary Actors: Database
Channel to Secondary Actors: Database API
Open Issues - Should there be a limit on the number of password attempts during verification?

Use Case: Settings
Iteration: 1
Primary Actor: Player
Goal in Context: Player goes to their account settings on the platform
Preconditions: Player is on the multiplayer board game platform and has logged into their account
Trigger: Player clicks settings button on the platform
Scenario:
1. Player logs into their account on the game platform
2. Home screen is displayed with different options
3. Player selects settings button
4. Screen shows the different options available in the settings section
5. Player selects which section they would like to go to
Postconditions: Player is taken to the section they chose
Exceptions:
1. GUI malfunctions
Priority: High - must be implemented
When Available: Mar. 7, 2025
Channel to Actor: GUI
Secondary Actors: N/A
Channel to Secondary Actors: N/A
Open Issues: None

Use Case: Change Username
Iteration: 1
Primary Actor: Player
Goal in Context: Player can change username associated with their account
Preconditions: Player is on the multiplayer board game platform and has logged into their account. Player knows their previous username
Trigger: Player clicks change username button in the settings
Scenario:
1. Player selects change username button available in the settings
2. Screen shows page where input is required for two questions
    a. Q1: Old Username
    b. Q2: New Username
3. Player fills in all two slots and clicks submit button at the end
Postconditions: Player's username is changed

Exceptions:
1. Username already being used by someone else
Priority: High - must be implemented
When Available: Mar. 7, 2025
Channel to Actor: GUI Interface
Secondary Actors: N/A
Channel to Secondary Actors: N/A
Open Issues:
1. Are there going to be restrictions on the username (inaaproapriate words, number of characters, etc.)?

Use Case: Change Password
Iteration: 1
Primary Actor: Player
Goal in Context: Player can change their account's password
Preconditions: Player is on the multiplayer board game platform and has logged into their account. Player knows their previous password
Trigger: Player clicks change password button in the settings
Scenario:
1. Player selects change password button available in the settings
2. Screen shows page where input is required for three questions
    a. Q1: Old Password
    b. Q2: New Password
    c. Confirm New Password
3. Player fills in all three slots and clicks submit button at the end
Postconditions: Player's password is changed
Exceptions:
1. Player forgets old password and therefore cannot change it
2. Password is already being used by someone else
Priority: High - must be implemented
When Available: Mar. 7, 2025
Channel to Actor: GUI Interface
Secondary Actors: N/A
Channel to Secondary Actors: N/A
Open Issues:
1. Is the password going to have restrictions (number of characters, numbers, special characters, etc.)?

Use Case: Change Email
Iteration: 1
Primary Actor: Player
Goal in Context: Player can change email associated with their account
Preconditions: Player is on the multiplayer board game platform and has logged into their account. Player has access to the previous and new email.

Trigger: Player clicks change email button in the settings
Scenario:
1. Player selects change email button available in the settings
2. Screen shows page where input is required for three questions
    a. Q1: Old Email
    b. Q2: New Email
    c. Q3: Confirm New Email
3. A verification code is sent to the previous email to confirm this change
4. Player inputs verification code into verification slot that shows up on the screen
5. If the code matches, player can click submit button at the end
Postconditions: Player's email associated with their account is changed
Exceptions:
1. Player doesn't have access to previous email and cannot confirm verification code
2. New email is already being used for a different account
Priority: High - must be implemented
When Available: Mar. 7, 2025
Channel to Actor: GUI Interface
Secondary Actors: N/A
Channel to Secondary Actors: N/A
Open Issues:
1. Does the verification code expire after a certain period of time?

Use Case: Logout
Iteration: 1
Primary Actor: Player
Goal in Context: Player can logout of their account on the platform
Preconditions: Player is on the multiplayer board game platform and has logged into their account
Trigger: Player clicks logout button in settings
Scenario:
1. Player clicks logout button at the bottom of the settings page
2. Platform logs out the player's account
3. Screen is redirected to login page
Postconditions: Player is logged out of their account
Exceptions: None
Priority: High - must be implemented
When Available: Mar. 7, 2025
Channel to Actor: GUI Interface
Secondary Actors: N/A
Channel to Secondary Actors: N/A
Open Issues: None

Use Case: Login

Iteration: First
Primary Actor: Player
Goal in Context: The player wants to access their account by entering valid email or username and password. Preconditions:
- The system is online and operational.
- The login interface is accessible.
- The player must have a registered account.
Trigger: The player attempts to log in by entering their username/email and password in the provided fields. Scenario:
1. The player navigates to the login screen.
2. The player enters their username or email and password.
3. The player clicks the "Login" button.
4. The system verifies the username/email and password.
5. If the credentials are valid, the system grants access to the player's account and redirects them to their profile homepage, where they can play games, view their stats, and see other users' stats.
6. If the credentials are invalid, the system displays an error message on the login page.
Postconditions: The player is authenticated and granted access to their account.
Exceptions:
- The system displays an error message and prompts the player to re-enter login details.
- If the player enters incorrect credentials multiple times, the account may be temporarily locked.
- If the player cannot remember their password, they can request a password reset.
Priority: High
When Available: March 7, 2025
Channel to Actor: GUI - Login Interface
Secondary Actors: User Credentials Database
Channel to Secondary Actors: User Credentials Database API
Open Issues:
- Should multi-factor authentication be required?
- How many failed attempts before account lockout?

Use Case: View Your Stats
Iteration: First
Primary Actor: Player
Goal in Context: Allow the player to view their win/loss records, a history of games played, and their stats. Preconditions:
- The player must have an active account.
- The player must be logged into the account.
- The system must have stored stats related to the player's performance in games.
Trigger: The player selects the "View Your Stats" option on the player's profile homepage.
Scenario:
1. The player navigates to their profile's homepage.
2. The player selects the "View Your Stats" option.
3. The system retrieves the player's statistics from the database.

4. The system displays the statistics, including the history of games played, win/loss record, and player rankings.
5. Once the player is done reviewing their stats, they can use the “Back” button to return to their profile's homepage.
Postconditions: The player successfully views their stats.
Exceptions: If the player has not played any games, there will be no game history or recorded stats for their profile. In this situation, the system can display a message saying, 'You haven't played any games yet, so there is no game history or recorded stats for your profile.
Priority: High as this use case is essential for player engagement and tracking progress.
When Available: March 7, 2025
Channel to Actor: GUI - "View Your Stats" interface extending the player's profile homepage.
Secondary Actors: Database
Channel to Secondary Actors: Database API
Open Issues: Should players be able to filter or sort their stats by time period or by the type of game they played?

Use Case: Forgot Password or Username
Iteration: 1
Primary Actor: Player
Goal in Context: Player restores their forgotten password or username
Preconditions: Player already has an existing account
Trigger: Player selects Forgot Password or Username button on the login screen
Scenario:
1- Player clicks on the Forgot Password or Username button
2- System asks for the email used to create the account
3- Player enters the email
4- System sends a verification code
5- After entering the code the system allows the password reset or show the username
Postconditions: Player can log in again
Exceptions:
1- Entering wrong email
Priority: High
When Available: March 7
Channel to Actor: Gui interface
Secondary Actors: N/A
Channel to Secondary Actors: N/A
Open Issues:
1- How long should the verification code stay valid?
2- Should there be a limit of attempts when the verification code is wrong?

Use Case: Play Games
Iteration: 1

Primary Actor: Player
Goal in Context: Player selects and plays a game
Preconditions: Player is logged into their account
Trigger: Player selects the Play Games button
Scenario:
1- Player clicks on Play Games button
2- System displays the available games
3- Player chooses a game
4- Player starts playing
Postconditions: Player played the selected game
Exceptions:
1- Game fails to load
Priority: High
When Available: March 7
Channel to Actor: GUI Interface
Secondary Actors: N/A
Channel to Secondary Actors: N/A
Open Issues:
1- Could the player play multiple games at once?
2- How would the system handle networking issues during the game?

Use Case: Connect4
Iteration: 1
Primary Actor: Player
Goal in Context: Player selects and plays Connect4
Preconditions: Player is logged in and selected Play games, Connect4 buttons
Trigger: Player chooses Connect4 from the games option
Scenario:
1- Player selects Connect4
2- System loads Connect4
3- Player starts playing Connect4
Postconditions: Player is playing Connect4
Exceptions:
1- Connect4 fails to load
2- Game crashes
Priority: High
When Available: March 7
Channel to Actor: Gui Interface
Secondary Actors: N/A
Channel to Secondary Actors: N/A
Open Issues:
1- How would the player track his score?

Use Case: TicTacToe
Iteration: 1
Primary Actor: Player
Goal in Context: Player selects and plays TicTacToe
Preconditions: Player is logged in and selected Play games, TicTacToe buttons
Trigger: Player chooses TicTacToe from the games option
Scenario:
1- Player selects TickTacToe
2- System loads TicTacToe
3- Player starts playing TicTcToe
Postconditions:
Player is playing TicTcToe
Exceptions:
1- TicTcToe fails to load
2- Game crashes
Priority: High
When Available: March 7, 2025
Channel to Actor: Gui Interface
Secondary Actors: N/A
Channel to Secondary Actors: N/A
Open Issues:
1- How would the player track his score?
2- How would the system handle multiplayer cases?

Use Case: Checkers
Iteration: 1
Primary Actor: Player
Goal in Context: Player selects and plays Checkers
Preconditions: Player is logged in and selected Play games, Checkers buttons
Trigger: Player chooses Checkers from the games option
Scenario:
1- Player selects the game Checkers
2- System loads Checkers
3- Player starts playing Checkers
Postconditions: Player is playing the game Checkers
Exceptions:
1- The game Checkers fails to load
2- Game crashes
Priority: High
When Available: March 7, 2025
Channel to Actor: Gui Interface
Secondary Actors: N/A

Channel to Secondary Actors: N/A
Open Issues:
1- How would the player track his score?
2- Can players invite friends to play?

Leaderboard Matchmaking Use Case Descriptions:

Use Case: Choose Game
Iteration: 1
Primary Actor: Player
Goal in Context: The player wants to select which games to play (Tic Tac Toe, Connect Four, or Checkers) Preconditions

- The player is logged in.
- The main menu is displayed with available game options.

Trigger: The player navigates to the main game selection screen.
Scenario:
1. The system shows a list of available games (Tic Tac Toe, Connect Four, Checkers).
2. The player selects one of the games.
3. The system transitions to that game's interface.
Postconditions: The chosen game is now active and the player an proceed to join a queue.
Exceptions:

- Game list fails to load due to system error.
- Chosen game is temporarily unavailable.

Priority: Medium
When Available: Always
Frequency of Use: High (everytime a player wants to switch or start a new game)
Channel to Actor: Game interface
Secondary Actors: Database
Channel to Secondary Actors: N/A
Open Issues

- How do we handle showing the GUI when a game is under maintenance?

Use Case: Back Button (Universal)
Iteration: 1
Primary Actor: Player
Goal in Context: Provide a universal navigation option allowing the user to return to the previous screen.
Preconditions

- The user is in a state or screen that supports a "back" action.

Trigger: The player taps the universal "Back" button in the GUI.
Scenario:
1. The user presses the "Back" button.
2. The system navigates to the previous screen or state.

3. Any unsaved data or ongoing process is handled or the user is prompted.
Postconditions: The user sees the screen they were on prior to the current one.
Exceptions:

- The user is already at the main menu (no previous screen).
- The system doesn't allow going back if the action would disrupt an active match.

Priority: Low
When Available: Always
Frequency of Use: Frequent (basic navigation)
Channel to Actor: Game interface
Secondary Actors: N/A
Channel to Secondary Actors: N/A
Open Issues

- N/A

Use Case: Choose Game (Tic Tac Toe)
Iteration: 1
Primary Actor: Player
Goal in Context: The player wants to select which game to play (Tic Tac Toe, Connect Four, or Checkers). Preconditions

- The player is logged in.
- The main menu is displayed with available game options.

Trigger: The player navigates to the main game selection screen.
Scenario:
1. The system shows a list of available games (Tic Tac Toe, Connect Four, Checkers).
2. The player selects "Tic Tac Toe."
3. The system transitions to the Tic Tac Toe game interface.
Postconditions: The chosen game (Tic Tac Toe) is now active, and the player can proceed to join a queue.
Exceptions:

- Game list fails to load due to system error.
- Chosen game is temporarily unavailable.

Priority: Medium
When Available: Always
Frequency of Use: High (everytime a player wants to switch or start a new game)
Channel to Actor: Game interface
Secondary Actors: Database
Channel to Secondary Actors: N/A
Open Issues

- How do we handle showing the GUI when a game is under maintenance?

Use Case: Choose Game (Connect Four)
Iteration: 1
Primary Actor: Player

Goal in Context: The player wants to select which game to play (Tic Tac Toe, Connect Four, or Checkers). Preconditions

- The player is logged in.
- The main menu is displayed with available game options.

Trigger: The player navigates to the main game selection screen.
Scenario:
1. The system shows a list of available games (Tic Tac Toe, Connect Four, Checkers).
2. The player selects "Connect Four."
3. The system transitions to the Connect Four game interface.
Postconditions: The chosen game (Connect Four) is now active, and the player can proceed to join a queue. Exceptions:

- Game list fails to load due to system error.
- Chosen game is temporarily unavailable.

Priority: Medium
When Available: Always
Frequency of Use: High (everytime a player wants to switch or start a new game)
Channel to Actor: Game interface
Secondary Actors: Database
Channel to Secondary Actors: N/A
Open Issues

- How do we handle showing the GUI when a game is under maintenance?

Use Case: Choose Game (Checkers)
Iteration: 1
Primary Actor: Player
Goal in Context: The player wants to select which game to play (Tic Tac Toe, Connect Four, or Checkers). Preconditions

- The player is logged in.
- The main menu is displayed with available game options.

Trigger: The player navigates to the main game selection screen.
Scenario:
1. The system shows a list of available games (Tic Tac Toe, Connect Four, Checkers).
2. The player selects "Checkers."
3. The system transitions to the Checkers game interface.
Postconditions: The chosen game (Checkers) is now active, and the player can proceed to join a queue.
Exceptions:

- Game list fails to load due to system error.
- Chosen game is temporarily unavailable.

Priority: Medium
When Available: Always
Frequency of Use: High (everytime a player wants to switch or start a new game)
Channel to Actor: Game interface
Secondary Actors: Database

Channel to Secondary Actors: N/A
Open Issues

- How do we handle showing the GUI when a game is under maintenance?

Use Case: View Game Stats Iteration: 1
Primary Actor: Database/System
Goal in Context: The player wants to see their updated stats (e.g., MMR, total wins/losses).
Preconditions

- The player is logged in.
- The player clicks the “View Game Stats” button.

Trigger: The player selects “View Game Stats.”
Scenario:
1. The system player selects the “View Game Stats”
2. The system displays the updated stats (MMR, record, rank).
Postconditions: The player sees their stats reflecting the matches result.
Exceptions:

- Stats update not completed yet.

Priority: Medium
When Available: Always
Frequency of Use: Occasional (everytime a player wants to view their stats)
Channel to Actor: Game interface
Secondary Actors: Stats/Leaderboard system
Channel to Secondary Actors: Database queries
Open Issues

- How do we handle delays if stats are still processing?

Use Case: Enter Queue
Iteration: 1
Primary Actor: Player
Goal in Context: The player wants to find an opponent by joining the matchmaking queue for the chosen game. Preconditions

- The player has selected a game (Tic Tac Toe, Connect Four, or Checkers)
- The player has a valid MMR rank.

Trigger: The player clicks the “Enter Queue” or “Join Queue” button.
Scenario:
1. The system checks the player’s rank and MMR for the chosen game.
2. The system places the player in the corresponding matchmaking queue (or queue pair).
3. The player sees a confirmation that they have joined the queue.
Postconditions: The player is now in the matchmaking queue, awaiting an opponent.
Exceptions:

- The player is already in a queue.

Priority: High When Available: Always

Frequency of Use: High (everytime a player wants to start a match)
Channel to Actor: Game interface
Secondary Actors: Matchmaking Service
Channel to Secondary Actors: N/A
Open Issues

- How do we handle a large number of concurrent queue requests?

Use Case: Leave Queue
Iteration: 1
Primary Actor:
Player Goal in Context: The player wants to exit the matchmaking queue before being matched.
Preconditions

- The player is currently in a matchmaking queue.

Trigger: The player selects the “Leave Queue” option.
Scenario:
1. The system locates the player in the queue.
2. The system removes the player from the queue.
3. The player is notified that they have left the queue.
Postconditions: The player is no longer waiting in the matchmaking queue.
Exceptions:

- The player is not found in a queue.
- The player is over the buffer period to leave the queue and has to join a game.

Priority: Medium
When Available: Always
Frequency of Use: Occasional (if the player changes their mind)
Channel to Actor: Game interface
Secondary Actors: Matchmaking Service
Channel to Secondary Actors: N/A
Open Issues

- How long is the buffer period?
- What if a match is found at the exact moment the player tries to leave?

Use Case: Enter Game Iteration: 1
Primary Actor: Player
Goal in Context: The player begins a match after being matched with an opponent through the matchmaking system.
Preconditions

- The player is matched with an opponent.
- The system has a valid game session ready to start.

Trigger: The player starts the game.
Scenario:
1. The system informs the player that a game is ready.

2. The player clicks “Enter Game”.
3. The system loads the game board/environment.
Postconditions: The player is now in an active game session.
Exceptions:

- The opponent leaves or disconnects from the game.
- A system issue preventing from loading the game.

Priority: High When Available: Always
Frequency of Use: High (each time a new match starts)
Channel to Actor: Game interface
Secondary Actors: Game server
Channel to Secondary Actors: N/A
Open Issues

- How is the multiple potential opponents case handled?

Use Case: Win Iteration: 1
Primary Actor: Player
Goal in Context: The player is declared the winner at the end of a match.
Preconditions

- A game is in progress.
- A winning condition/ final board state is met.

Trigger: The game logic detects a winning board or condition met by the player.
Scenario:
1. The system identifies the winning condition for the player.
2. The system announces the win to both players.
3. The system transitions to the post-game screen.
4. The system updates the winner’s stats (e.g., wins count, MMR).
Postconditions: The player’s record shows a win, and the post-game screen is displayed.
Exceptions:

- Incorrect board evaluation.
- Stats update failure.

Priority: High When Available: Always
Frequency of Use: Whenever a player wins a match.
Channel to Actor: Game interface
Secondary Actors: Stats/Leaderboard system
Channel to Secondary Actors: Database update
Open Issues

- Disputes over the final board state.

Use Case: Lose
Iteration: 1
Primary Actor: Player
Goal in Context: The player is declared the loser at the end of a match.

Preconditions

- A game is in progress.
- A losing condition is met.

Trigger: The game logic detects that the opponent has a winning board or condition.
Scenario:
1. The system identifies the losing condition for the player.
2. The system announces the loss to both players.
3. The system transitions to the post-game screen.
4. The system updates the player's stats (loss count, MMR).

Postconditions: The player's record shows a loss, and the post-game screen is displayed.
Exceptions:

- Incorrect board evaluation.
- Stats update failure.

Priority: High
When Available: Always
Frequency of Use: Whenever a player loses a match.
Channel to Actor: Game interface
Secondary Actors: Stats/Leaderboard system
Channel to Secondary Actors: Database update
Open Issues

- How do we handle forced losses if a player disconnects or quits?

Use Case: Tie Iteration: 1
Primary Actor: Player
Goal in Context: The game concludes with neither player winning (e.g., a full board in Tic Tac Toe with no winner). Preconditions

- A game is in progress.
- The final board state or condition indicates no winner can be declared.

Trigger: The game logic detects a tie scenario.
Scenario:
1. The system identifies the tie condition.
2. The system notifies both players of the tie.
3. The system transitions to the post-game screen.
4. The system updates both players' stats (ties count, MMR adjustments if any).

Postconditions: Both players' records reflect a tie, and the post-game screen is shown.
Exceptions:

- Incorrect tie detection.
- Stats update failure.

Priority: Medium
When Available: Always
Frequency of Use: Occasional (whenever players tie a match)
Channel to Actor: Game interface
Secondary Actors: Stats/Leaderboard system

Channel to Secondary Actors: Database update
Open Issues

- Do we handle tie-breaker rules in certain games?

Use Case: Update Stats Iteration: 1
Primary Actor: Database/System
Goal in Context: Record the game outcome (win, loss, tie) and adjust player stats (MMR, total wins, etc.).
Preconditions

- A match has ended with a known result.
- The system has the correct player data.

Trigger: The game engine signals that the match is over.
Scenario:
1. The system retrieves the outcome (win/loss/tie) for each player.
2. The system applies game-specific MMR updates and increments win/loss/tie counts.
3. The updated stats are stored in the database/CSV .
Postconditions: Player stats are accurately reflected in the leaderboard and personal records.
Exceptions:

- Incorrect outcome data leads to wrong updates.

Priority: High
When Available: Always
Frequency of Use: After every completed match.
Channel to Actor: Server-side process
Secondary Actors: Stats/Leaderboard system, Players
Channel to Secondary Actors: Database update
Open Issues

- How do we handle concurrency if many matches end simultaneously?

Use Case: View Post Game Stats
Iteration: 1
Primary Actor: Database/System
Goal in Context: The player wants to see their updated stats relevant to the specific game (e.g., Tic Tac Toe, Connect Four, or Checkers) immediately after a match.
Preconditions

- The game has ended.
- The system has updated the player's stats for the specific game (e.g., Tic Tac Toe, Connect Four, or Checkers).

Trigger: The player navigates to the post-game screen or selects "View Post Game Stats" after completing the match.
Scenario:
1. The system finishes updating the stats for both players.
2. The player opens the post-game stats screen.

3. The system displays the updated game-specific stats (e.g., number of wins, losses, and MMR for Tic Tac Toe or Connect Four),
Postconditions: The player sees their updated stats reflecting the match result, specific to the chosen game. Exceptions:

- Stats update not completed yet.

Priority: Medium
When Available: Always
Frequency of Use: After each match conclusion
Channel to Actor: Game interface
Secondary Actors: Stats/Leaderboard system
Channel to Secondary Actors: Database queries
Open Issues

- How do we handle delays if stats are still processing?

Use Case: Back to Queue
Iteration: 1
Primary Actor: Player
Goal in Context: The player attempts to quit the queue, but the buffer period has already expired, so they are forced to remain in (or return to) the matchmaking queue.
Preconditions

- The player is currently in a matchmaking queue.
- The buffer period to exit the queue without penalty has passed.

Trigger: The player selects an option to leave or tries to navigate away from the queue after the buffer period. Scenario:
1. The player attempts to leave the queue (e.g., clicks “Leave Queue”).
2. The system checks if the buffer period has expired.
3. Since the buffer period is over, the system prevents the player from quitting and redirects them back to the matchmaking queue interface.
4. A message is displayed indicating that it is too late to leave the queue.
Postconditions: The player remains in the matchmaking queue, awaiting an opponent.
Exceptions:

- The system incorrectly calculates the buffer period, letting the player leave prematurely.

Priority: Medium
When Available: Always
Frequency of Use: Occasional (only after the buffer period ends)
Channel to Actor: Game interface
Secondary Actors: Matchmaking service
Channel to Secondary Actors: N/A
Open Issues

- Determining the exact duration of the buffer period.
- Communicating clearly to the player when they are locked into the queue.

Use Case: Back to Game Menu
Iteration: 1
Primary Actor: Player
Goal in Context: The player successfully exits the matchmaking queue before the buffer period ends, returning to the main game menu.
Preconditions

- The player is currently in a matchmaking queue.
- The buffer period to leave without penalty has not yet expired.

Trigger: The player clicks “Back to Game Menu” (or a similar option) during the valid buffer period.
Scenario:
1. The player selects “Back to Game Menu” while still within the allowed buffer period.
2. The system verifies that the buffer period has not expired.
3. The player is removed from the queue or game session without penalty.
4. The system navigates the player to the main game menu.
Postconditions: The player is no longer in the queue or game session and is free to select another activity.
Exceptions:

- The system incorrectly marks the buffer period as expired and forces the player to stay.
- A match is found at the exact moment the player tries to leave, leading to a potential conflict.

Priority: Medium
When Available: Always
Frequency of Use: Occasional (if the player changes their mind during the buffer window)
Channel to Actor: Game interface Secondary Actors: Matchmaking service
Channel to Secondary Actors: N/A
Open Issues

- Handling edge cases where the buffer period is nearly expired at the moment of the player’s action.

Integration Use Case Descriptions:
Use Case: External Database Interaction with the System
Iteration: Iteration 1
Primary Actor: External Database
Goal in Context: Ensure synchronization as actions are made and storage of data within the Online Multiplayer Board Game (OMG) platform.
Preconditions:
1. The system is connected to the database.
2. An active account and database exists and interactions occur within the platform.
Trigger:
An action is performed that requires data retrieval or storage, such as logging in, updating settings, searching profiles, joining a queue, or participating in a game.
Scenario:
1. Login is initiated.
2. The database verifies credentials and retrieves profile data.
3. The system updates the last login timestamp.

4. Profile settings are updated.
5. The new settings are validated and stored in the database.
6. A profile search is performed.
7. The database retrieves the requested profile and includes stats, rank, and match history.
8. The matchmaking queue is created.
9. The system queries the database to find potential opponents based on skill level and availability.
10. A match is found, and the game begins.
11. The database updates player statuses to "in-game".
12. The game progresses with turn-based actions.
13. The database logs every move in real-time for match history tracking.
14. The game ends, and results are recorded.
15. The database updates player stats, including wins, losses, and leaderboard rankings.
16. Match history is reviewed.
17. The system fetches and displays past game records from the database.
18. The leaderboard is updated.
19. The database recalculates rankings based on recent match results.
20. A friend is added to the friends list.
21. The system updates the friends list in the database.
22. Logout occurs.
23. The system updates session data and stores the logout timestamp.

Post Conditions:

1. The database reflects all changes made during interactions.
2. Player stats, matchmaking data, and leaderboard rankings are up to date.
3. Match history is accessible for review.
4. Friends list updates are stored.

Exceptions:

1. Database Connection Failure:
2. The system notifies and tries to.

Invalid Login Credentials:

1. Access is denied with an error message.

Priority: High (Critical for platform functionality).
When Available: Always online, with real-time updates.
Frequency of Use: Constant during platform interactions.
Channel to Actor: API calls between the system and the database.
Secondary Actors: Matchmaking system, game session manager.
Channel to Secondary Actors: Matchmaking logic and game engine API.
Open Issues:
-Handling of simultaneous updates to player stats.
-Ensuring database integrity under heavy load conditions.
-Managing real-time updates of the friends list and profile searches.

Use Case: Game Synchronization
Iteration: 1

Primary Actor: Game Client
Goal in Context:
Make sure that all players experience a synchronized game state in real-time, preventing desynchronization issues and maintaining a smooth multiplayer experience across devices and platforms.
Preconditions:
1. The player is connected to the internet.
2. The game client has established a connection with the game server.
3. The player's session is active, and game data is available for synchronization.
Possible Triggers:
1. A player starts or joins a game session.
2. A player's action (e.g., movement, interaction, or event) requires synchronization.
3. A game state change (e.g., moving pieces) needs to be shown to all participants.
Scenario:
1. The player starts or joins a game session.
2. The game client establishes a connection with the game server.
3. The game client sends player actions and game state updates to the server.
4. The server processes the incoming data and updates the global game state.
5. The server distributes updated game state information to all connected clients.
6. Each game client applies the synchronized game state locally.
7. The process continues in real-time to maintain synchronization.
Postconditions:
1. The game state is synchronized across all players.
2. Player actions are consistently reflected in the game world.
3. Statistics are updated according to the game state.
Exceptions:
1. Network Interruption: The client attempts to reconnect and retrieve the latest game state.
2. Desynchronization: If inconsistencies are detected, the game client requests a state reset from the server.
3. High Latency: Lag can cause delays.
4. Server Downtime: Players receive a notification, and the game may switch to offline mode if supported.
5. Data Corruption: The server restores the last known good state.
Priority: High – Needed for a smooth multiplayer experience.
Channel to Actor:
- Game Client → Game Server
- Game Server → Game Client
Secondary Actors:
- Game Server
- Third-Party Multiplayer Services (e.g., cloud save services, matchmaking systems)
Channel to Secondary Actors:
- Game Server → Third-Party Services
- Third-Party Services → Game Server
Open Issues:
1. How to handle synchronization for low-bandwidth players effectively?

2. What is the ideal threshold for detecting and correcting desynchronization?
3. How to minimize server costs while ensuring low-latency synchronization?

Use Case: Player Interacting with social features
Iteration: 1 Primary Actor: Player
Goal in Context: player is able to interact with social features, such as adding friends, removing friends, and inviting friends to game.
Preconditions:
1. The player is logged into the system.
2. The player has stable network connection.
3. social features such as friends list, game invitations, messages are available.
4. player is online (pre-condition only for invite to game)
Trigger: The player chooses friends list and interacts with a social feature such as searching and adding a friend, inviting another player to a game, or sending a message.
Scenario:
1. Player selects friends list from home page of app or during game.
2. System displays the player's friends list and other social features such as send message or add friend.
3. Player then selects an action to perform:
- Add friend: player searches for another player by username and sends a friend request.
- Accept or reject friend request: player reviews pending requests and accepts or decline them.
- Send game invite: player selects a friend from friend list and invites them to a session.
- Send message: player selects a friend and sends a message.
4. System processes and responds to selected action:
- Friend requests: other player receives a notification.
- Game invites: other player receives request to join a session.
- chat messages: system delivers message.
5. The other player responds to the action by:
- Accepting or declining a friend request.
- Joining or declining game invitation.
- Replying to a message.
6. If friend request is accepted system updates both players friend lists.
7. If a game invitation is accepted system moves players into a game session.
8. If a message is received system displays message in a chat box of some sort.
9. Player uses more social features or quits social menu.
Postconditions:
1. system updates friend list, messages, or game session status based on interactions.
2. system updates profile details such as wins, loses, and more after multiplayer sessions.
Priority: High
When Available: Whenever player has stable network connection and is in app on homepage. Messages however can be accessed in game.
Frequency of Use: Varying (depending on how many social interactions take place).
Channel to Actor: friends list (friend requests, chat menu, game invites) within the app.
Secondary Actors: Other players, game server, database.

Channel to Secondary Actors: Network connection (for communication with server and database).
Open issues:
- What action should be taken by system if during a multiplayer session one player's game were to crash?
- To what degree should messages be censored?