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: 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 (e.g., MMR, total wins/losses) immediately after a match.

Preconditions

- The game has ended.
- The system has updated the player's stats.

Trigger: The player navigates to the post-game screen or selects "View Post Game Stats."

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 stats (MMR, record, rank).

Postconditions: The player sees their new stats reflecting the match result.

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?