CONNECT FOUR

Use Case: Player joins matchmaking queue

Iteration: 1

Primary Actor: Player

Goal in Context: The player wants to quickly find an opponent by entering the Connect Four matchmaking system.

Preconditions:

- The player is logged in.
- The player has a valid account and an assigned MMR/rank for Connect Four.

Trigger: The player selects the “Join Queue” option from the Connect Four game interface.

Scenario:

1. The player clicks the “Join Queue” button.
2. The system verifies the player’s current Connect Four rank and MMR.
3. The system randomly assigns the player to one of the two queue pairs for their rank.
4. The player is added to the selected Connect Four matchmaking queue.

Postconditions: The player is now waiting in a Connect Four matchmaking queue for pairing.

Exceptions:

- Player is already in an active queue.
- Network/server issues prevent the queue join.

Priority: High

When Available: Always

Frequency of Use: High (each game session)

Channel to Actor: Connect Four client interface

Secondary Actors: Connect Four matchmaking service

Channel to Secondary Actors: N/A

Open Issues:

- Determining how to handle simultaneous queue join requests.

Use Case: MMR changes after a game completes

Iteration: 1

Primary Actor: Game/Player

Goal in Context: Update the player's MMR based on the outcome of a completed Connect Four game.

Preconditions:

- The Connect Four game has just finished.
- The game outcome (win, loss, or tie) is recorded.

Trigger: The Connect Four game has ended.

Scenario:

1. The game computes the result of the Connect Four game.
2. The system invokes the corresponding method on the player's Connect Four stats object.
3. The player's MMR is recalculated.
4. Updated MMR is stored and reflected in the player's profile and leaderboard.

Postconditions: The player's Connect Four MMR accurately reflects their recent game performance.

Exceptions:

- Errors in MMR calculation.

Priority: High

When Available: Always

Frequency of Use: Every Connect Four game completion.

Channel to Actor: Internal game processing.

Secondary Actors: Stats and leaderboard

Channel to Secondary Actors: N/A

Open Issues:

- Refine the formula used to update player ratings in Connect Four so that changes are smooth and fairly reflect performance differences.

Use Case: Display Leaderboard

Iteration: 1

Primary Actor: Game/Player

Goal in Context: The player wants to view a ranked list of Connect Four players based on metrics such as MMR or wins.

Preconditions:

- The Connect Four leaderboard data is available.
- The game is responsive.

Trigger: The player selects the “View Leaderboard” option from the Connect Four game menu.

Scenario:

1. The player clicks on the “Leaderboard” tab.
2. The system retrieves Connect Four leaderboard data.
3. The sorted leaderboard is displayed on the player's screen.

Postconditions: The player is presented with an updated and ranked leaderboard for Connect Four.

Exceptions:

- Missing leaderboard stats.

Priority: High

When Available: Always

Frequency of Use: Moderate

Channel to Actor: Connect Four game client interface

Secondary Actors: Leaderboard system

Channel to Secondary Actors: N/A

Open Issues:

- How will we ensure the stats are calculated effeciently and that the display of the leaderboard is consistent even with large data sets?

Use Case: Player Leaves Matchmaking Queue

Iteration: 1

Primary Actor: Player

Goal in Context: The player wants to exit the matchmaking queue before being matched with an opponent.

Preconditions:

- The player is currently in the matchmaking queue.
- The matchmaking system is responsive.

Trigger: The player selects the "Leave Queue" option from the matchmaking screen.
Scenario:

1. The player clicks on the “Leave Queue” button.
2. The system removes the player from the matchmaking queue.
3. The system confirms the player has successfully left the queue.

Postconditions: The player is no longer in the matchmaking queue and can take other actions.

Exceptions:

- The player is matched with an opponent at the same time they attempt to leave, making the action invalid.
- Server issues prevent immediate removal from the queue.

Priority: Medium

When Available: Always, as long as the player is in the queue.

Frequency of Use: Occasional

Channel to Actor: Connect Four game client interface

Secondary Actors: Matchmaking system

Channel to Secondary Actors: N/A

Open Issues:

- Should there be a cooldown or penalty for frequently leaving the queue?
- How will we handle edge cases where a match is found at the same time the player tries to leave?

Use Case: Player Gets Promoted a Rank

Iteration: 1

Primary Actor: Player

Goal in Context: The player wants to be promoted to a higher rank after achieving the required performance or MMR.

Preconditions:

- The player has completed a game with enough points or performance to qualify for a promotion.
- The player's MMR or rank is updated post-game.

Trigger: The player reaches the required MMR or wins after completing a match.

Scenario:

1. The player finishes a game, and the system evaluates their performance (MMR, wins, etc.).
2. The system checks if the player qualifies for a promotion.
3. If the player qualifies, the system updates the player's rank to the next level.
4. The system displays a promotion notification to the player.

Postconditions: The player's rank is updated to the next level, and the promotion is confirmed with a notification.

Exceptions:

- The player's MMR does not meet the promotion requirement, and no promotion is awarded.
- Server issues prevent rank updates or notifications.

Priority: High

When Available: After each game or period of time when rank is recalculated.

Frequency of Use: Occasionally, depending on the player's performance.

Channel to Actor: Connect Four game client interface (promotion notification)

Secondary Actors: Ranking system

Channel to Secondary Actors: N/A

Open Issues:

- How will we handle promotions if the player's performance fluctuates around the threshold?
- Should there be a delay between the game's end and the promotion notification to prevent spam or errors?

Use Case: Player Gets Demoted a Rank

Iteration: 1

Primary Actor: Player

Goal in Context: The player gets demoted to a lower rank after a series of poor performance, losses, or a drop in MMR.

Preconditions:

- The player has been performing poorly or has lost enough games to drop below the required MMR threshold for their current rank.
- The player's MMR or rank is updated post-game.

Trigger: The player's MMR or performance falls below the threshold required for their current rank.

Scenario:

1. The player finishes a game, and the system evaluates their performance (MMR, losses, etc.).
2. The system checks if the player's MMR has dropped below the threshold for their current rank.
3. If the player qualifies for demotion, the system updates the player's rank to the next lower level.
4. The system displays a demotion notification to the player.

Postconditions: The player’s rank is updated to the next lower level, and the demotion is confirmed with a notification.

Exceptions:

- The player's MMR does not drop enough to warrant demotion.
- Server issues prevent rank updates or notifications.

Priority: Medium

When Available: After each game or period of time when rank is recalculated.

Frequency of Use: Occasionally, depending on the player’s performance.

Channel to Actor: Connect Four game client interface (demotion notification)

Secondary Actors: Ranking system

Channel to Secondary Actors: N/A

Open Issues:

- How do we handle cases where a player’s performance fluctuates, making demotion unclear?
- Should there be a grace period or warning system to notify players before they are demoted?