Use Case: Join Match As A Party

Primary Actor: Party Leader (The player who initiates the game for the group)

Secondary Actors: Party/Group Members and the Matchmaking System

Goal in Context: To allow a group of players to be able to join a multiplayer match together using the matchmaking system.

Preconditions:

- Party leaders and members must be signed in
- Party must be formed before joining
- Party members must be available and not in another match

Trigger: Party leader initiates the "Join Match as Party" action.

Scenario:

1. Party Leader selects "Join Match as Party" option.
2. The system then verifies that all party members are both online and available.
3. The system then sends a request to the matchmaking service and system.
4. The matchmaking service then finds an appropriate match based on skill level, rank, and other factors.
5. The system then notifies party members that a match has been found.
6. The party members then ready up
7. The system then transitions all players into the match session.

Postconditions:

- The party members are placed in a multiplayer match.
- The match session is initialized and every player is ready to start.

Exceptions:

- If a party member isn't online an error message is displayed
- If a suitable match is found, matchmaking is cancelled
- If a party member declines the match the party leader has to reattempt matchmaking.

Priority:

High, as it is a core multiplater feature which enables social gameplay and connectivity

Channel to Actor:

GUI interaction through game client.

Secondary Actors:

- Matchmaking System

- Game Session Manager (party leader)

Open Issues:

- Should party members be able to leave after matchmaking starts?
- What happens if a player disconnects during matchmaking?

Use Case: Log Game Result

Primary Actor: Game Session Manager

Secondary Actors: The game players, the game's leaderboard system, and database.

Goal in Context: To allow the system to record the outcome of a finished match, while updating player stats and adjusting leaderboards accordingly.

Preconditions:

- The game has to have been played and complete with a result.
- The system has to have access to the relevant game data, so players, scores, and outcomes
- Players must be logged in to store and ensure result tracking.

Trigger: The session ends, and system is prompted to load and store results

Scenario:

1. The game completes through a win, draw, or forfeit.
2. The game session leader/manager then initializes game data
3. The system then formats the data, and validates accordingly based on formatting The
4. The leaderboard system then processes the final game result to determine any rankings
5. The leaderboard then adjusts player rankings and statistics based on stat conditions
6. The final result is stored in the game database
7. The players then receive a notification confirming their match result has been recorded

Postconditions:

- The game result is recorded in the system.
- Players stats and leaderboards are adjusted
- The system is then able to retrieve match data when needed.

Exceptions:

- If the system fails to log the result due to a server issue, data should be retried or temporarily stored
- If data is missing or corrupted, an error is logged, and then data may need manual evaluation
- If a player disconnects, the system should assign a forfeit to the player

Priority:

High as this is vital for maintaining game stats , and player performance history.

Channel to Actor: Server-side process

Open Issues:

- Should unranked games be logged separately from ranked matches?
- How long should data be stored before archiving?
- Should players be able to review past match results in their profiles?