1. Start New Game

Primary Actor: Game Player ← may want to include a second player since one is referenced
Goal: Initiate a new Connect 4 game session within the application.

Pre-Conditions:
• The game application is running locally.
• No active game session is in progress.
• Networking is available.
• The player is logged into the game client.

Trigger:
• The player selects the "New Game" option from the game menu.

Scenario:

1. The player launches the game client.
2. The player selects "New Game" from the main menu.
3. The system initializes an empty custom-size board.
4. The system determines which player goes first (either randomly or by a predefined rule).
5. The initialized board and the turn indicator are displayed on the local interface.

Post-Conditions:
• A new game session is started with a fresh board.
• The active player is prompted to make the first move.

Exceptions:
• System or application error during game initialization.
• Network issues in multiplayer mode.

Priority: High (Essential for starting gameplay)
When Available: At the beginning of every game session.
Frequency of Use: Every time a player wants to start a game.
Channel to Actor: Local game interface
Secondary Actors: System
Channels to Secondary Actors: Internal application processes
Open Issues: Handling unexpected shutdowns or reconnections during game startup.

2. Making a Move

Primary Actor: Game Player
Goal: Allow the active player to drop a disc into a chosen column.

Pre-Conditions:
• A game session is active with an initialized board.
• It is the active player's turn.
• The current board state is displayed on the game client.

Trigger:
• The player selects a column within the allowed range using the local input device (e.g., keyboard).

Scenario:

1. The active player selects a column via the game client interface.
2. The system verifies that the chosen column number is within the valid range.
3. The system checks that the selected column is not already full.
4. The disc is placed in the lowest available row of the selected column.
5. The board is updated immediately on the local interface to reflect the new move.

Post-Conditions:
• The game board displays the new move.
• The turn passes to the opposing player unless a win/draw condition is met.

Exceptions:
• Column selection is out-of-range.
• The chosen column is already full.
• An error occurs during the board update process.

Priority: High (Core gameplay functionality)
When Available: Each turn during an active game session.
Frequency of Use: Multiple moves per game session.
Channel to Actor: Local game UI (input devices such as keyboard or mouse)
Secondary Actors: System
Channels to Secondary Actors: Internal processing and UI rendering
Open Issues: Resolving simultaneous move attempts in multiplayer mode and providing visual feedback (animations, sounds) for move validation.

3. Checking for Win/Draw

Primary Actor: System
Goal: Automatically determine if the latest move results in a win or if the game concludes in a draw.

Pre-Conditions:
• A move has just been made and the board state has been updated.
• The game session is active.

Trigger:
• Completion of a player's move.

Scenario:

1. The game engine scans the board after each move.
2. It checks for four consecutive discs in a row horizontally, vertically, or diagonally.
3. If a win is detected:
    - The system declares the active player as the winner.
    - The game session is ended.
4. If no win is detected and the board is full:
    - The system declares a draw.
    - The game session is ended.
5. If neither condition is met:
    - The turn is passed to the opposing player.

Post-Conditions:
• The game session either continues with the next turn or concludes with a win/draw.

Exceptions:
• System error during evaluation.
• Inconsistencies in the board state.

Priority: High (Critical for game resolution)
When Available: After every move.
Frequency of Use: Every turn during the game.
Channel to Actor: Internal processing with results displayed on the game client
Secondary Actors: System

Channels to Secondary Actors: Backend evaluation
Open Issues: Optimizing win/draw detection, especially for customizable board sizes.

4. Handling Invalid Moves

Primary Actor: Game Player
Goal: Prevent and handle invalid move attempts gracefully during gameplay.

Pre-Conditions:
• A game session is active.
• It is the player's turn.

Trigger:
• The player attempts to drop a disc into an invalid column (either out-of-range or full).

Scenario:

1. The player selects an invalid column.
2. The system detects that the column is either out-of-range or full.
3. An error message is immediately displayed on the local interface, indicating the nature of the invalid move.
4. The system maintains the turn for the same player until a valid move is made.

Post-Conditions:
• The board remains unchanged.
• The player is prompted to select a valid move.

Exceptions:
• Multiple invalid attempts might trigger additional warnings or a temporary move lock-out (if configured).

Priority: High (Essential for maintaining game integrity)
When Available: Every time an invalid move is attempted.
Frequency of Use: As frequently as players may make input errors.
Channel to Actor: Local game UI using available input devices
Secondary Actors: System
Channels to Secondary Actors: Internal input validation and feedback mechanisms
Open Issues: Defining thresholds for temporary lock-outs after repeated invalid moves.

5. Ending the Game

Primary Actor: System
Goal: Conclude the game session when a win, draw, or game end by player condition is met. ← what if a player leaves the game partway through?

Pre-Conditions:
• A winning move, draw, or game end by player condition has been detected.
• A game session is active.

Trigger:
• A move results in a win or draw, or the player issues a game end by player command.

Scenario:

1. The game engine detects a win, a draw, or receives a game end by player signal.
2. The system terminates the active game session.
3. The final board state is displayed on the game client.
4. The system announces the game result (win, draw, or game end by player) on the interface.
5. Optionally, the player is offered the option to start a new game.

Post-Conditions:
• The game session is concluded.
• The result is clearly communicated to the player.
• The game client returns to the main menu or game summary screen.

Exceptions:
• System error during game termination.

Priority: High (Necessary for clear and proper game closing)
When Available: At the end of every game session.
Frequency of Use: Once per game session.
Channel to Actor: Local game client displaying alerts and final game summary
Secondary Actors: System
Channels to Secondary Actors: Internal processes and UI notifications
Open Issues: Offering a detailed summary screen with game statistics and a rematch option.

this may need to connect to a network actor to operate properly

6. Writing Chat Message

Primary Actor: Game Player
Goal: Allow the active player to send a chat message to their opponent or other players during a multiplayer game session.

Pre-Conditions:
• A game session is active.
• Players are in a multiplayer environment (local or online).
• The chat feature is available and enabled in the game client.

Trigger:
• The player selects the "Chat" option from the game interface and types a message.

Scenario:

1. The player accesses the chat window from the game interface.
2. The player types a message and hits the "Send" button.
3. The system validates the message (e.g., checking for inappropriate content or enforcing length restrictions).
4. The system displays the message on the screen for the opposing player(s) to view in real time.
5. Optionally, the system plays a notification sound to inform the opponent(s) of the new message.

Post-Conditions:
• The message is successfully sent and displayed to the other players.
• The game continues without interruption.

Exceptions:
• The message contains invalid characters or inappropriate language.
• The message fails to send due to network issues.

Priority: Medium (Enhances player interaction but is not critical for gameplay)
When Available: Throughout the game, during both player turns and intermissions.
Frequency of Use: Occasional, based on player communication needs.
Channel to Actor: Game client interface (via keyboard or chat UI)
Secondary Actors: System
Channels to Secondary Actors: Internal message validation and display

Open Issues: Defining chat message filters for inappropriate content and ensuring smooth performance in multiplayer scenarios.