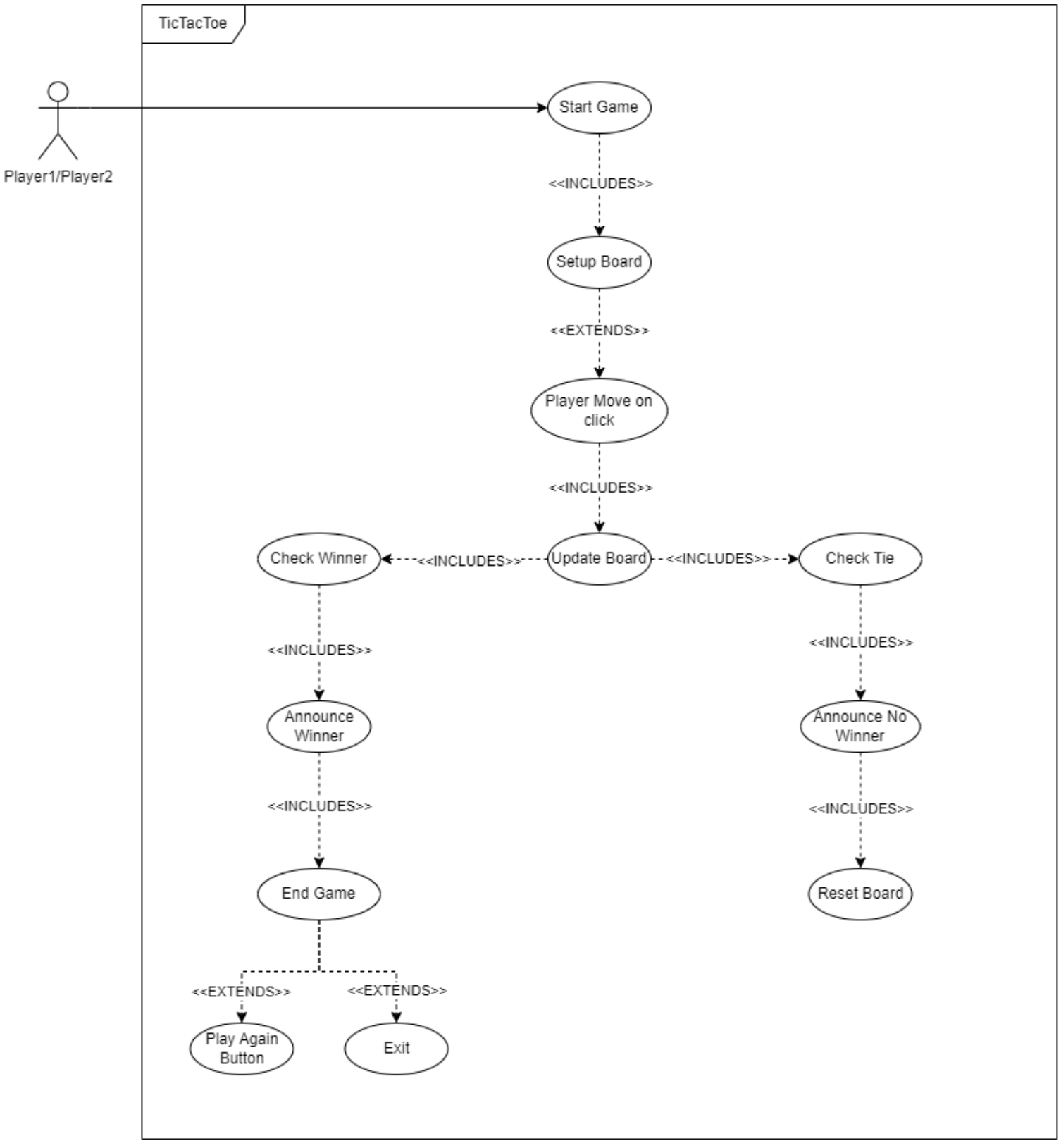

TicTacToe
Player1/Player2
Start Game
<<INCLUDES>>
Setup Board
<<EXTENDS>>
Player Move on click
<<INCLUDES>>
Update Board
<<INCLUDES>>
Check Winner
<<INCLUDES>>
Check Tie
<<INCLUDES>>
Announce Winner
<<INCLUDES>>
Announce No Winner
<<INCLUDES>>
End Game
<<INCLUDES>>
Reset Board
<<EXTENDS>>
Play Again Button
<<EXTENDS>>
Exit

Use Case Descriptions (Tic-Tac-Toe)

Use Case: Start Game

Iteration: 1st iteration

Primary actor: Player1, Player2

Goal in context: Begins a new Tic-Tac-Toe game session

Preconditions: The system is powered on and user has selected Tic-Tac-Toe as their game of choice

Trigger: Either Player1 or Player2 (depending on host) selects Tic-Tac-Toe as their game of choice with the “Tic-Tac-Toe” button)

Scenario:

1. Player selects Tic-Tac-Toe game option
2. The system will initialize a Tic-Tac-Toe session
3. The session will create any required information before setting up the board (visualization of the game).

Post Conditions: Game is setup and ready to print board (visual) to users

Exceptions:

- The button for Tic-Tac-Toe was unresponsive
- The system encounters an error while initializing
    - Errors
    - Missing information

Priority: High - the first initialization of the game and setups are important to the functionality of GUI and game logic.
When Available: Within 1 sprint (1st iteration)
Frequency of Use: Once per game session
Channel to actor: Interaction of a click of the Tic-Tac-Toe game option with their mouse
Secondary actor: N/A.
Channel to Secondary Actors: N/A
Open issues:

- The setup of the game (any information required) is finished before GUI implementation

Use Case: Setup Board

Iteration: 1st iteration

Primary actor: Player1, Player2

Goal in context: The game was set up and the board is ready to be displayed to the user. The game will call a function to print the board.

Preconditions: The game was properly setup as required with the GUI ready for response from printBoard function.

Trigger: The system has set up the game and will now call a function for printing the board.

Scenario:

1. Game information is setup and properly initialized
2. Game calls to a printBoard function to display to user

Post Conditions: board is set up (the visual) and ready for Player1 to make their move

Exceptions:

- The function call was unresponsive
- The system encounters an error while starting the visual
    - Possibly incorrect formatting

Priority: High - visual representation should be set up in-order for users to interact with the game.
When Available: Within 1 sprint (1st iteration)
Frequency of Use: Once per game session
Channel to actor: When they have selected Tic-Tac-Toe as their game of choice (is part of initialization)
Secondary actor: N/A.
Channel to Secondary Actors: N/A
Open issues:

- Board was incorrectly setup

Use Case: Player Move on click

Iteration: 1st iteration

Primary actor: Player1, Player2

Goal in context: Player1/player2 can make their move by clicking any of the tiles of tic-tac-toe to place their symbol (X or O) onto the board.

Preconditions: The game board was set up and is ready for user input.

Trigger: The user has clicked a tile to place their piece.

Scenario:

1. The board is ready for user to input their piece
2. User clicks the tile they desire to place their piece
3. The piece is placed in the desired spot

Post Conditions: the user has successfully placed their piece onto the board and will now prompt the other user to place their piece

Exceptions:

- The piece was not placed in the correct spot
    - The piece not placed at all
- The system encounters an error placing piece

Priority: High - Placing their pieces is the main functionality of the Tic-Tac-Toe game.
When Available: Within 1 sprint (1st iteration)
Frequency of Use: N times per player
Channel to actor: Interaction of a click on the tile they desire to place their symbol with their mouse
Secondary actor: N/A
Channel to Secondary Actors: N/A
Open issues: N/A

Use Case: Update board

Iteration: 1st iteration

Primary actor: Player1, Player2

Goal in context: After any user input, the board is updated and displayed to the user.

Preconditions: users have selected their move and the board needs to be updated.

Trigger: user has selected their move.

Scenario:

1. Player has made their move
2. GUI collects this information and passes to game logic
3. New updated game board is passed back to GUI to display

Post Conditions: The new updated board is returned to GUI to display to users

Exceptions:

- Board was not updated properly
- An error has occurred
    - No updates to board

Priority: High - In order to understand what is happening in the game, it is important to continually update the board after each player input.
When Available: Within 1 sprint (1st iteration)
Frequency of Use: N times per player
Channel to actor: player has clicked a tile to make their move and system calls to update the board
Secondary actor: N/A
Channel to Secondary Actors: N/A
Open issues: N/A

Use Case: Check Winner

Iteration: 2nd iteration

Primary actor: Player1, Player2

Goal in context: After each player move, the system should check if there is a winner every turn

Preconditions: the user has placed a symbol on the board and the board was updated properly

Trigger: the board was updated.

Scenario:

1. The board was updated based on player's move
2. The system will check if a winner was detected

Post Conditions: system successfully checks if there was a winner detected

Exceptions:

- Board was updated incorrectly and checking winner returns incorrect information
- Conditions were never checked

Priority: High - determines the game conditions in order to win and end the game
When Available: Within 2 sprint (2st iteration)
Frequency of Use: N times per player
Channel to actor: once the board has been updated, system calls to check if there is a winner based on move
Secondary actor: N/A
Channel to Secondary Actors: N/A
Open issues:

- Game logic is incorrect

Use Case: Check Tie

Iteration: 2nd iteration

Primary actor: Player1, Player2

Goal in context: After each player moves the system should check if there is a tie after every turn

Preconditions: the user has placed a symbol on the board and the board was updated properly

Trigger: the board was updated

1. **Scenario:** The board was updated based on player's move
2. The system will check if a tie was detected

Post Conditions: system successfully checks if there was a tie detected

Exceptions:

- Board was updated incorrectly and checking tie returns incorrect information
- Conditions were never checked

Priority: high determines if a tie was detected and the game should end
When Available: Within 2 sprint (2st iteration)
Frequency of Use: N times per player
Channel to actor: once the board has been updated, system calls to check if there is a tie based on move
Secondary actor: N/A
Channel to Secondary Actors: N/A
Open issues:

- Game logic is incorrect

Use Case: Announce Winner

Iteration: 2nd iteration

Primary actor: Player1, Player2

Goal in context: A winner was found and a winner will be announced and displayed to GUI

Preconditions: a winner was found

Trigger: System has detected a winner

Scenario:

1. Winning conditions were found
2. A winner announcement is prompted to GUI

Post Conditions: A winner announcement is prompted to GUI

Exceptions:

- Announces incorrect prompt
    - Announces incorrect winner

Priority: Low - can be implemented at any stage of the program as long as winning conditions are correct
When Available: Within 2 sprint (2st iteration)
Frequency of Use: once per game
Channel to actor: winner was found and system announces winner to GUI
Secondary actor: N/A
Channel to Secondary Actors: N/A
Open issues: N/A

Use Case: Announce No winner

Iteration: 2nd iteration

Primary actor: Player1, Player2

Goal in context: No winner was found (board is full) and will be announced and displayed to GUI

Preconditions: no winner was found

Trigger: no winner was found

Scenario:

3. No winner was found
4. A no winner announcement is prompted to GUI

Post Conditions: A no winner announcement is prompted to GUI

Exceptions:

- Announces incorrect prompt

Priority: Low - can be implemented at any stage of the program as long as tie conditions are correct
When Available: Within 2 sprint (2st iteration)
Frequency of Use: once per game
Channel to actor: A tie game was found and system announces tie to GUI
Secondary actor: N/A
Channel to Secondary Actors: N/A
Open issues: N/A

Use Case: Reset Board

Iteration: 2nd iteration

Primary actor: Player1, Player2

Goal in context: create an empty board to reset the game

Preconditions: a previous game has ended

Trigger: "Restart" button

Scenario: 1. A previous game has ended
2.The user decides to start a new game

Post Conditions:
The game was properly setup as required with the GUI ready for response from printBoard function.

Trigger: The system has set up the game and will now call a function for printing the board.

Scenario:

3. Game information is setup and properly initialized
4. Game calls to a printBoard function to display to user

Post Conditions: board is set up (the visual) and ready for Player1 to make their move
Exceptions:

- The function call was unresponsive
- The system encounters an error while starting the visual
    - Possibly incorrect formatting

Priority: High - visual representation should be set up in-order for users to interact with the game.
When Available: Within 1 sprint (1st iteration)
Frequency of Use: Once per game session
Channel to actor: When they have selected Tic-Tac-Toe as their game of choice (is part of initialization)
Secondary actor: N/A.
Channel to Secondary Actors: N/A
Open issues:

- Board was incorrectly setup

Use Case: End Game

Iteration: 2nd iteration

Primary actor: Player 1, Play

Goal in context: end game

Preconditions: The game board is initialized.

- The players have taken turns placing their symbols (X or O) on the board.

Trigger: a tie or win has occurred

Scenario: players placed their marks on cells and a win or tie was achieved and the game is over

Post Conditions: restart or exit

Exceptions: If the board is in an invalid state (e.g., a player has already won but more moves are made), reset or display an error.

Priority: high- ends the game in finite time
When Available: 1st iteration
Frequency of Use: once each game
Channel to actor: on screen information about the previous game
Secondary actor: N/A
Channel to Secondary Actors: N/A
Open issues: win or tie was incorrectly detected

Use Case: Play Again Button

Iteration: 2nd iteration

Primary actor: Player1, Player2

Goal in context: reset board on button click

Preconditions: the game has ended

Trigger: user clicks “play again”

Scenario: the game ends the user is given options to “play again” or “exit”

Post Conditions: the board is re-initialized

Exceptions: the game ended unexpectedly

Priority: high- allows for continued play
When Available: 1st iteration
Frequency of Use: up to once per game
Channel to actor: button on screen when game ends
Secondary actor: N/A
Channel to Secondary Actors: N/A
Open issues: game ends unexpectedly because a win or tie was detected incorrectly

Use Case: Exit

Iteration: 2nd iteration

Primary actor: Player1, Player2

Goal in context: exit game on button click

Preconditions: the game has ended

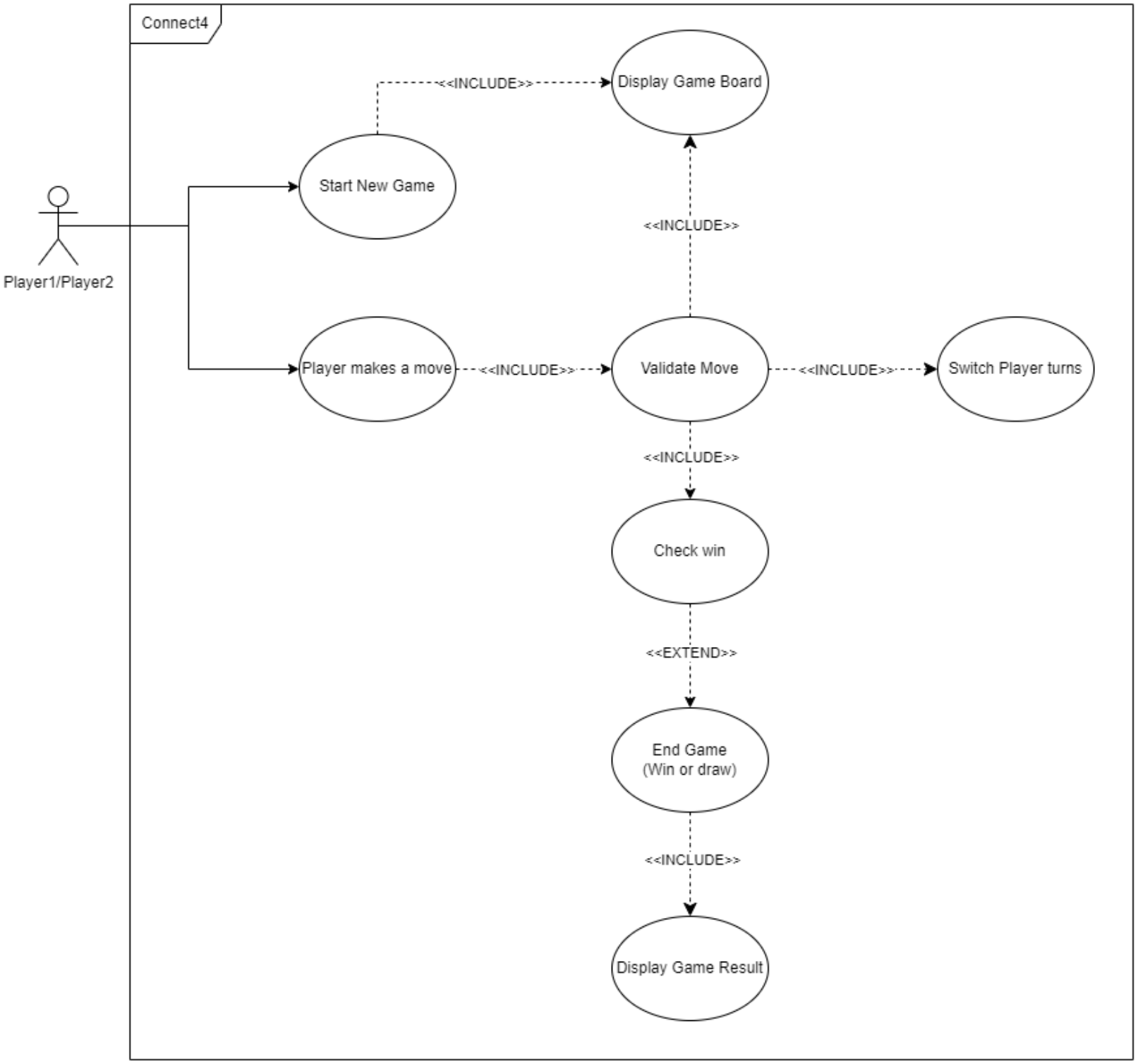
Trigger: user clicks “Exit”

Scenario: the game ends the user is given options to “play again” or “exit”

Post Conditions: the game is exited the user is given options to play a different game such as connect4 or checkers

Exceptions: the game ended unexpectedly

Priority: high- allows for continued play
When Available: 1st iteration
Frequency of Use: up to once per game
Channel to actor: button on screen when game ends
Secondary actor: N/A
Channel to Secondary Actors: N/A
Open issues: game ends unexpectedly because a win or tie was detected incorrectly


Connect4
Display Game Board
<<INCLUDE>>
Start New Game
Player1/Player2
<<INCLUDE>>
Player makes a move
<<INCLUDE>>
Validate Move
<<INCLUDE>>
Switch Player turns
<<INCLUDE>>
Check win
<<EXTEND>>
End Game
(Win or draw)
<<INCLUDE>>
Display Game Result

Use Case Descriptions (Connect4)

Use Case: Start New Game

Iteration: 1

Primary Actor: Player

Goal in context: To initiate a new game, resetting the game board and allowing players to begin a fresh round.

Preconditions: The application is active and ready for a new game.

Trigger: The player selects the option to start a new game.

Scenario:

1. The player chooses to start a new game.
2. The system clears the game board and prepares it for new moves.
3. The initial state of the game board is displayed.

Postconditions: The game board is reset and ready for the first move.

Exceptions: None.

Priority: High – Essential for game functionality.

When available: Within 1 sprint (first iteration).

Frequency of use: At the beginning of each new game.

Channel to actor: Visual confirmation of a new game board.

Secondary actors: N/A

Channel to secondary actors: N/A

Open issues: N/A

Use Case: Display Game Board

Iteration: 1

Primary Actor: Player

Goal in context: To provide a visual display of the current game board, enabling players to understand the game state.

Preconditions: The game has started, and at least one move has been made.

Trigger: The game begins or a move is completed.

Scenario:

1. The system generates and displays a visual representation of the board.
2. Column numbers and current piece placements are shown.
3. The system updates the display after each move.

Postconditions: The board is continuously visible, reflecting the latest moves.

Exceptions: Display errors that prevent the board from being shown accurately.

Priority: High – Vital for game clarity.

When available: Within 1 sprint (first iteration).

Frequency of use: Continuously, after each move and at game start.

Channel to actor: Visual display of the board.

Secondary actors: N/A

Channel to secondary actors: N/A

Open issues: N/A

Use Case: Player Makes a Move

Iteration: 1

Primary Actor: Player

Goal in context: To drop a piece in a chosen column, advancing the game.

Preconditions: The game is in progress, and it is the player's turn.

Trigger: The player selects a column for their move.

Scenario:

1. The player selects a column to place their piece.
2. The system validates the move for legality.
3. If valid, the piece is added to the board in the selected column.

Postconditions: The move is recorded, and the board reflects the new state.

Exceptions: If the column is full, prompt the player to select another column.

Priority: High – Core game mechanic.

When available: Within 1 sprint (first iteration).

Frequency of use: Repeatedly, throughout the game.

Channel to actor: Confirmation of the move.

Secondary actors: N/A

Channel to secondary actors: N/A

Open issues: None

Use Case: Validate Move

Iteration: 1

Primary Actor: Player

Goal in context: To ensure the move chosen is valid and can be executed on the board.

Preconditions: The player has selected a column for their move.

Trigger: The player attempts a move.

Scenario:

1. The player's move triggers a check if the selected column has space for a new piece.
2. If the move is valid, the system allows the piece to be placed.
3. If not, the system prompts the player to select another column.

Postconditions: Only valid moves are executed, preventing errors.

Exceptions: Full columns prevent the move; the player is prompted to try again.

Priority: High – Ensures fair gameplay.

When available: Within 1 sprint (first iteration).

Frequency of use: Every time a player makes a move.

Channel to actor: Visual prompt if the move is invalid.

Secondary actors: N/A

Channel to secondary actors: N/A

Open issues: None.

Use Case: Check Win

Iteration: 1

Primary Actor: Player

Goal in context: To determine if a winning condition has been met after making a move.

Preconditions: A valid move has been made.

Trigger: Completion of the player's move.

Scenario:

1. The player's move triggers an analysis of the board for four consecutive pieces by the same player.
2. If a winning condition is met, it triggers the end game.
3. If no winning condition is met, the game continues.

Postconditions: The game state is updated based on the win check.

Exceptions: None.

Priority: High – Critical for game outcome.

When available: Within 1 sprint (first iteration).

Frequency of use: After every move.

Channel to actor: N/A (system-initiated).

Secondary actors: N/A

Channel to secondary actors: N/A

Open issues: None.

Use Case: End Game (Win or Draw)

Iteration: 1

Primary Actor: Player

Goal in context: To conclude the game if a player has won or if the board is full, resulting in a draw.

Preconditions: A win condition is met, or the board is full.

Trigger: A win is detected or no moves are left.

Scenario:

1. The player's move or full board triggers the end game.
2. The system halts further moves.
3. It displays the game result to the players.

Postconditions: The game concludes, with results visible to players.

Exceptions: None.

Priority: High – Essential for completing each game round.

When available: Within 1 sprint (first iteration).

Frequency of use: Once per game.

Channel to actor: Game result display.

Secondary actors: N/A

Channel to secondary actors: N/A

Open issues: None.

Use Case: Display Game Result

Iteration: 1

Primary Actor: Player

Goal in context: To show the outcome of the game, indicating a winner or a draw.

Preconditions: The game has ended.

Trigger: The game reaches a conclusion (win or draw).

Scenario:

1. The system shows the final game result.
2. It indicates the winning player or a draw.
3. Optionally, players are given the option to start a new game.

Postconditions: Players are informed of the game's outcome.

Exceptions: None.

Priority: Medium – Important for player feedback.

When available: Within 1 sprint (first iteration).

Frequency of use: Once per game.

Channel to actor: Visual display of the result.

Secondary actors: N/A

Channel to secondary actors: N/A

Open issues: None.

Use Case: Switch Player Turns

Iteration: 1

Primary Actor: Player

Goal in context: To alternate between players after each valid move, ensuring fair gameplay.

Preconditions: A player has successfully completed a valid move.

Trigger: A valid move is executed by the current player.

Scenario:

1. The current player completes a valid move.
2. The system identifies the next player in turn.
3. The system updates the active player indicator, signalling whose turn it is.
4. The new active player is allowed to make their move.

Postconditions: The turn alternates to the next player, allowing them to proceed with their move.

Exceptions: If a turn switch error occurs, the system reverts to the previous player for correction.

Priority: High – Ensures balanced and structured gameplay.

When available: Within 1 sprint (first iteration).

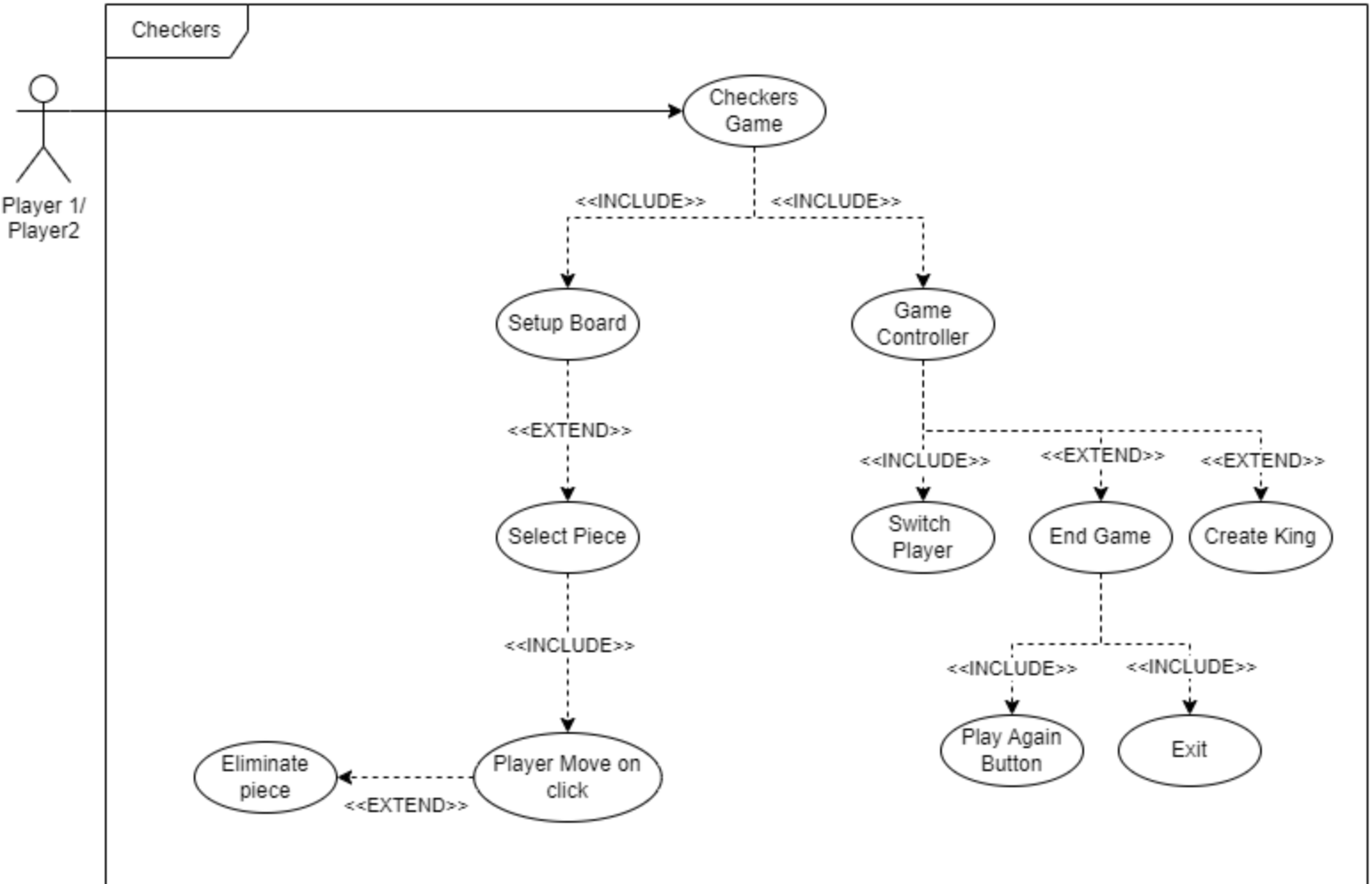
Frequency of use: Every turn, throughout the game.

Channel to actor: Visual or indicator prompt showing the current player.

Secondary actors: N/A

Channel to secondary actors: N/A

Open issues: None.


Checkers
Player 1/
Player2
Checkers
Game
<<INCLUDE>>
<<INCLUDE>>
Setup Board
Game
Controller
<<EXTEND>>
<<INCLUDE>>
<<EXTEND>>
<<EXTEND>>
Select Piece
Switch
Player
End Game
Create King
<<INCLUDE>>
<<INCLUDE>>
<<INCLUDE>>
Play Again
Button
Exit
Eliminate
piece
Player Move on
click
<<EXTEND>>

Use Case Descriptions Checkers

Use Case: Checkers Game

- Iteration: 1
- Primary Actor: Player 1, Player 2
- Goal in Context: Start and manage a complete game of Checkers between two players.
- Preconditions: None.
- Trigger: A player initiates the game.
- Scenario:
    - The system initializes the game environment and displays the board.
    - The system prepares each player's pieces in starting positions.
    - The game is ready for players to begin taking turns.
- Post Conditions: The game is ready to proceed with players taking turns and making moves.
- Exceptions:
    - If the system fails to initialize, an error message is shown, and the game cannot start.
- Priority: High, as it is essential for gameplay initiation.
- When Available: Within its 1st iteration.
- Frequency of Use: Once per game session.
- Channel to Actor: Player interacts through the game’s user interface.
- Secondary Actors: N/A
- Channel to Secondary Actors: N/A
- Open Issues: None

Use Case: Setup Board

- Iteration: 1
- Primary Actor: System
- Goal in Context: Arrange all pieces on the board for the beginning of the game.
- Preconditions: The Checkers Game has been initiated.
- Trigger: The system initiates board setup as part of game start.
- Scenario:
    - The system places each player's pieces on the designated starting positions.
    - The board is visually updated to show the initial setup.
- Post Conditions: All pieces are correctly positioned on the board for the game to begin.
- Exceptions:
    - If the board setup fails, an error message is displayed, and the game does not start.
- Priority: High, as it is crucial for starting the game.
- When Available: Within its 1st iteration.
- Frequency of Use: Once per game session.
- Channel to Actor: System controls the setup directly.
- Secondary Actors: N/A
- Channel to Secondary Actors: N/A
- Open Issues: None

Use Case: Game Controller

- Iteration: 1
- Primary Actor: System
- Goal in Context: Manage turn-taking and special game events.
- Preconditions: The game is in progress.
- Trigger: A player ends their turn.
- Scenario:
    - The system ends the current turn.
    - The system enforces any additional game rules (e.g., king creation, elimination).
- Post Conditions: The game continues in accordance with the rules.
- Exceptions:
    - System error in handling turn or rule enforcement.
- Priority: High, essential for game management.
- When Available: Within its 1st iteration.
- Frequency of Use: Continuously throughout the game.
- Channel to Actor: System-managed.
- Secondary Actors: N/A
- Channel to Secondary Actors: N/A
- Open Issues: None

Use Case: Select Piece

Iteration: 1

Primary Actor: Player

Goal in Context: Allows the player to select a piece to move on their turn.

Preconditions: The game must be in progress, and it must be the player's turn.

Trigger: The player clicks on or selects a piece on the board.

Scenario:

1. The player selects a piece by clicking or entering its coordinates.
2. The system highlights or marks the selected piece.
3. The system verifies that the selected piece belongs to the current player.

Post Conditions: The piece is highlighted, and the player can now select a destination.

Exceptions:

- The selected piece does not belong to the player, and an error message is shown.

Priority: High priority, as it is essential for gameplay.

When Available: Within its 1st iteration.

Frequency of Use: Multiple times per player turn.

Channel to Actor: Player interacts directly with the board.

Secondary Actors: N/A

Channel to Secondary Actors: N/A

Open Issues: None

Use Case: Move Piece on click

Iteration: 1
Primary Actor: Player

Goal in Context: Allows the player to move a selected piece to a new position on the board.

Preconditions: A piece has been selected by the player, and it is a valid piece to move.

Trigger: The player clicks on or selects a destination square on the board.

Scenario:

1. The player selects a destination for the selected piece.
2. The system checks if the move is valid according to the rules of Checkers. ○ If valid, the piece is moved to the new position.
3. The board is updated to show the new position of the piece.

Post Conditions: The piece has been successfully moved to the new position.

Exceptions:

- The move is invalid, and an error message is displayed.

Priority: High priority, as it is essential for gameplay.
When Available: Within its 1st iteration.
Frequency of Use: Multiple times per game, depending on player moves.
Channel to Actor: Player interacts directly with the board.
Secondary Actors: N/A
Channel to Secondary Actors: N/A
Open Issues: None

Use Case: Eliminate Piece

- **Iteration**: 1
- **Primary Actor**: Player
- **Goal in Context**: Allows the system to remove an opponent’s piece when a capture move is made.
- **Preconditions**: The current player has made a valid capture move.
- **Trigger**: The player completes a capture move.
- **Scenario**:
    - The system identifies the opponent’s piece that is being captured.
    - The captured piece is removed from the board.
    - The board state is updated and displayed to both players.
- **Post Conditions**: The opponent's captured piece is removed from the board.
- **Exceptions**:
    - The system fails to remove the piece due to an error.
- **Priority**: High priority, as it impacts the game's progress and rules.
- **When Available**: Within its 1st iteration.
- **Frequency of Use**: Multiple times per game, depending on capture moves.
- **Channel to Actor**: Game Controller updates the board.
- **Secondary Actors**: System
- **Channel to Secondary Actors**: N/A
- **Open Issues**: None

Use Case: Switch Player

- **Iteration**: 1
- **Primary Actor**: Player
- **Goal in Context**: Allows the system to alternate turns between players.
- **Preconditions**: The current player has completed their turn.
- **Trigger**: The player's turn is over (either after a move or capture).
- **Scenario**:
    - ○ The system verifies that the player's turn is complete.
    - ○ The active player is switched to the other player.
    - ○ The system updates the game state to reflect the new player's turn. ○ The new player is notified that it is now their turn.
- **Post Conditions**: The turn is switched to the other player.
- **Exceptions**:
    - ○ The system fails to switch the player due to an error.
- **Priority**: High priority, as it is critical for turn-based gameplay.
- **When Available**: Within its 1st iteration.
- **Frequency of Use**: Multiple times per game as turns alternate.
- **Channel to Actor**: Game Controller handles turn management.
- **Secondary Actors**: System
- **Channel to Secondary Actors**: N/A
- **Open Issues**: None

Use Case: End Game

- **Iteration**: 1
- **Primary Actor**: Player
- **Goal in Context**: Ends the game when a win/loss condition is met or if a player forfeits.
- **Preconditions**: One player has no remaining pieces or legal moves, or a player chooses to forfeit.
- **Trigger**: A win/loss condition is met or a player forfeits.
- **Scenario**:
    - ○ The system checks the board state and verifies if a win/loss condition is met. ○ The game state is updated to GAME_OVER.
    - ○ The system determines and displays the winner.
    - ○ The end-game screen is displayed with options to "Play Again" or "Exit."
- **Post Conditions**: The game ends, and the result is displayed.
- **Exceptions**:
    - ○ The system fails to end the game due to an error.
- **Priority**: High priority, as it is essential for game completion.
- **When Available**: Within its 1st iteration.
- **Frequency of Use**: Once per game.
- **Channel to Actor**: Game Controller manages end-game logic.
- **Secondary Actors**: System
- **Channel to Secondary Actors**: N/A
- **Open Issues**: None

Use Case: Create King

- **Iteration**: 1
- **Primary Actor**: Player
- **Goal in Context**: Promotes a piece to a king when it reaches the opposite end of the board.
- **Preconditions**: A piece reaches the last row on the opponent's side during a move.
- **Trigger**: The player moves a piece to the last row.
- **Scenario**:
    - ○ The system identifies that the piece has reached the opponent's last row. ○ The piece is promoted to a king, with a visual indication.
    - ○ The board state is updated to reflect the promotion.
- **Post Conditions**: The selected piece is promoted to a king with enhanced movement.
- **Exceptions**:
    - ○ The system fails to promote the piece due to an error.
- **Priority**: Medium priority, as it enhances gameplay but is not essential.
- **When Available**: Within its 1st iteration.
- **Frequency of Use**: As needed during gameplay.
- **Channel to Actor**: Game Controller handles promotion.
- **Secondary Actors**: System
- **Channel to Secondary Actors**: N/A
- **Open Issues**: None

Use Case: Play Again Button

- **Iteration**: 1
- **Primary Actor**: Player
- **Goal in Context**: Allows the player to restart the game with a fresh board after the game has ended.
- **Preconditions**: The game is over, and the "Play Again" button is available on the end-game screen.
- **Trigger**: The player clicks the "Play Again" button.
- **Scenario**:
    - The player selects the "Play Again" button.
    - The system resets the game state and initializes a new board.
    - The board is reinitialized, and the players are notified of the new game start.

Post Conditions: A new game begins with the board reset.

- **Exceptions**:
    - The system fails to reset the board due to an error.
- **Priority**: Medium priority, as it enhances user experience.
- **When Available**: Within its 1st iteration.
- **Frequency of Use**: Once per session, if the player wants to replay.
- **Channel to Actor**: User interface for player interaction.
- **Secondary Actors**: N/A
- **Channel to Secondary Actors**: N/A
- **Open Issues**: None

Use Case: Exit

- **Iteration**: 1
- **Primary Actor**: Player
- **Goal in Context**: Allows the player to exit the game after it has ended.
- **Preconditions**: The game is over, and the "Exit" button is available.
- **Trigger**: The player clicks the "Exit" button.
- **Scenario**:
    - The player selects the "Exit" button.
    - The application closes or returns to the main menu.
- **Post Conditions**: The game session is terminated.
- **Exceptions**:
    - The application fails to exit properly.
- **Priority**: Low priority, as it provides a way to end the session.
- **When Available**: Within its 1st iteration.
- **Frequency of Use**: Once per session, if the player wants to end the game.
- **Channel to Actor**: User interface for player interaction.
- **Secondary Actors**: N/A
- **Channel to Secondary Actors**: N/A
- **Open Issues**: None