Use Case 1: Challenge Players

Iteration: 1

Primary Actor: Registered Player

Goal in context: Allow a registered player to challenge another player to a match in an available board game

Preconditions:

1. The user is logged into their account
2. The user has an active internet connection
3. The opponent they wish to challenge must also be a registered user and online
4. The game server must be running
5. At least one game must be available for selection

Trigger: The player selects the option to challenge another player from the GUI

Scenario:

1. The player navigates to the "Challenge Players" section in the GUI
2. The system displays a list of available online players
3. The player selects an opponent from the list
4. The system prompts the player to select a game from the available options
5. The player selects a game and confirms the challenge
6. The system sends a challenge request to the selected opponent
7. The opponent receives a notification of the challenge
8. The opponent either accepts or declines the challenge
9. If accepted, the system initializes the game and directs both players to the game lobby
10. If declined, the system notifies the challenger of the declined request

Postconditions:

1. If the challenge is accepted, both players enter the game lobby, ready to start the match

2. If the challenge is declined, the challenger is returned to the "Challenge Players" screen

Exceptions:

1. The opponent goes offline before responding: the system must notify the challenger and cancel the challenge, sending the user back to the game lobby
2. The game server is unavailable or down: the system must inform the user and prevent challenges from being sent or game selection
3. The opponent is already in another match: the system must notify the challenger and direct them back to the screen of online players
4. The challenger cancels the request before the opponent responds: the system must withdraw the challenge and cancel the challenge notification to the opponent

Priority: High, this feature is essential for multiplayer games and direct player interaction and matchmaking

When available: This feature is always accessible to players in the GUI once they are logged in and connected to the game server

Frequency of use: Every time a player logs in to the server and wants to play a game with another user online

Channel to actor: The player interacts through the GUI by selecting from the options to challenge another player who is online

Secondary actors:

1. Opponent Player (receives the challenge and chooses whether to accept)
2. Game Server (handles matchmaking and game initialization)

Channel to secondary actors:

1. Opponent Player: receives an in-game notification
2. Game Server: Processes and maintains server status, game availability and matchmaking status for players

Open issues:

1. Multiple incoming challenges to the same player
2. Ability to block another user to avoid receiving challenge requests from them
3. Timeout mechanism for responding to challenges to prevent inactive or AFK players from slowing matchmaking times
4. Having a friends list of specific players you frequently play with

Use Case 2: Look up game

Iteration: 1.0, last modification: March 5th by Kemuel

Primary Actor: Registered player

Goal in context: Allows for a player to look through their list of games and read a description of its gameplay.

Preconditions:
1. Player is logged in.

Trigger:
1. When the customer starts the application.
2. When you return to your games list after playing a game.

Priority: High priority needed to interact with the rest of the program.

When is it available: Available anytime the application is open, and when a game has ended.

Scenario:
1. Player starts game suite application
2. Games on suite popup in a list
3. Player clicks on a game in the list and sees their descriptions

Exceptions:
1. Player Does Not Start application
2. Player is playing another game simultaneously
3. Player is not logged in

When available: Available as soon as the application starts.

Frequency of use: High Frequency. Every time a user wants to look through their games

Channel to actor: Player uses Keyboard/ mouse interactions to interact with the application

Secondary actors:

1. Game application client.
2. Game servers.

Open Issues:

1. Searching up games by text
2. Different ways to sort games. Example: Aplhabedicly, last played date, most played, self sort.

Use Case 3: Look up Players

Iteration: 1

Primary actor: Registered Player

Goal in context: To allow users to search for players using the GUI search feature, to access their profiles, and to examine stats before challenging them.

Preconditions:

1. The user is logged in.
2. The user has an internet connection.
3. The dashboard window is fully loaded, displaying a button called “Look up Player”

Trigger: The user selects the user “Look up Players” button with their mouse.

Scenario:

1. The user enters a player's name into the search bar and the list updates dynamically with similar names.
2. The user applies filters (e.g., level, games played) via dropdown menus.
3. The user sorts the list by name, level, or other attributes using sorting buttons.
4. The user enters a player's name into the search bar.
5. The user scrolls through the list using their mouse or the scroll bar.

Postconditions:

1. The selected player's profile is displayed in a new window.
2. Hovering over a name in the list triggers a pop-up showing the player's name and stats.

Exceptions:

1. If no matching players are found, the GUI displays a "No players found" message.
2. If the user loses connection while searching, the GUI displays a "Connection lost. Couldn't load list." message.

Priority: High. A multiplayer GUI has to provide a way to access player stats of all players, for users to find and interact with others.

When available: Third iteration.

Frequency of use: High. Players will frequently use this feature to find opponents.

Channel to actor: The GUI

Secondary actors:

1. The player database.
2. The backend server.

Channels to secondary actors:

1. Internet

Open issues: None

Use Case 4: Join a game from the library

Iteration: 1.0

Primary actor: registered player

Goal in context: After looking up for games read descriptions of gameplay or rules, allowing a registered player to participate in the game by clicking and choosing the gameplay.

Pre-conditions:

1. The player is successfully logged into their account.
2. The player is directed to the game library page.
3. The game library is not empty and is available for selection

Trigger: The registered players join a game by choosing and clicking on the game icon or "start" button from the game library

Priority: High-needed, the players should join in the game first, either want to play solo or join a team in multi-player games later.

When available: Available any time if the players have good internet connection and have logged into the game as a registered player to see the game library and pick one from it.

Scenario:

1. Players log into the game application
2. Players navigate to the game library
3. Players read the descriptions and rules of the game
4. Players engage in a game by clicking game icon to start game session

Exceptions:

1. **Internet corruption**
    i. The Wi-fi is so unstable that the game cannot continue
    ii. The system prompts a message on refreshing the Wi-fi or reloading the game.
2. **Requirement for update**
    i. The game has updated new features

    ii. The system prompts a message on updating the game that players chose.

3. **Unavailable participation due to full session**
    i. The game is full as the number of players for that game reaches the capacity
    ii. The system prompts a message for matchmaking the player with the AI bot; or suggests estimated waiting time until the new session is available.

Frequency of use: High frequency, the players engage in the game by first logging in and joining in a game

Channel to actor: the GUI

Secondary actors: the game server

Channel to the secondary actors: Game server: maintain the server to run 24/7

Open issues: N/A

Use case 5: In-Game Chatting

Iteration: 1

Primary Actor: Player (active in game session)

Goal in context: Enable real-time texting between players during gameplay without interrupting the gaming experience, allowing for social interaction, game strategy, and sportsmanship.

Preconditions:
1. User is logged in to the system
2. User is currently participating in an active game session
3. At least one opponent or teammate is connected to the same game session
4. User has not been muted or restricted from the chatting feature

Trigger:

1. User activates chat interface by clicking chat icon or using keyboard shortcut   User receives message from another player
2. System event occurs requiring player notification (game state changes, turn alerts)

Scenario:

1. The player initiates the chat interface during an active game
2. System displays chat window in a non-intrusive position relative to the game board
3. System loads and displays the most recent message history (last 20 messages or configurable amount).
4. Player types the message in the text input box.
5. Player sends a message by pressing Enter key or clicking Send button
6. System validates messages for length and content restrictions.
7. System transmits message to messaging service with metadata (sender, timestamp, game session)
8. System displays the message in the local chat history with appropriate formatting
9. Recipients receive and view the message in their chat window
10. System sends a subtle notification sound (if enabled in user preferences)
11. Chat window can be minimized or relocated by user if desired
12. User can scroll through message history while game continues
13. If chat is inactive for 30 seconds, it automatically minimizes (optional tweek)

Postconditions:

1. Messages are left in the chat history throughout the game session
2. All active participants can view the message
3. Game state and performance remain unaffected
4. Chat history is saved with game record for potential review

Exceptions:

1. Message exceeds length limit - System displays character count warning and prevents submission
2. Inappropriate content detection - System filters content according to platform policy, or blocks message and warns user
3. Rapid message flooding - System implements cooldown and notifies user to prevent spam
4. Connectivity issue - System queues message for retry and displays sending status indicator
5. Recipient has blocked sender - Message is sent but not delivered, with no notification to sender
6. User has enabled "focus mode" - Incoming messages are queued but notifications suppressed
7. Game is in critical state (time pressure, final moves) - System may delay non-urgent notifications

Priority: Medium-High

When available: Iteration 2

Frequency of use: Very frequent - multiple times per game session

Channel to actor:
1. Visual chat interface within game UI
2. Notification system (visual and/or audio)
3. Optional keyboard shortcuts

Secondary actors:
1. Other players in the game (Opponents/Opponent)
2. The system that handles messages
3. Basic content filtering

Channel to secondary actors:
1. Basic communication between game components
2. Real-time connection for instant messaging
3. Connection to the main game session

Open issues:

1. Should private messaging between specific groups (e.g., team members) be allowed in team-based game modes?
2. How should chat history be saved between game sessions for regular opponents?
3. Should chat have rich text formatting or emoji reactions?
4. How to balance chat visibility with space constraints in small screens
5. Should voice chat be considered a future enhancement?
6. How to manage international player translations?
7. What level of administrative control is needed for chat moderation?
8. Should chat support auto-send game event notifications?
9. How will the "chat" feature scale between different game types (e.g., turn-based vs. real-time)?