**Project Sprint #1**

GitHub link: **https://github.com/SalmanM1/CS449**

In this assignment, you aim to specify the requirements (i.e., user stories and acceptance criteria) of the target software that allows a human player to play a simple or general SOS game against a human opponent. These requirements will be fully implemented by the end of sprint 3. The minimum features include **choosing the board size,** **choosing the game mode (simple or general)**, **starting a new game**, **making a move (in a simple or general game)**, **determining if a simple or general game is over**. The following is a sample GUI layout.

|  |  |  |
| --- | --- | --- |
| SOS Icon  Description automatically generated Simple game Icon  Description automatically generated General game Board size  8 | | |
| Blue player  Icon  Description automatically generated S  Icon  Description automatically generated O | Chart, line chart  Description automatically generated | Red player  Icon  Description automatically generated S  Icon  Description automatically generated O |
|  | Current turn: blue (or red) | New Game |

Figure 1. Sample GUI layout of the first working program by the end of Sprint 3

Use the following tables to document your user stories and acceptance criteria.

You are required to use the free ChatGPT version to complete 2 user stories and their respective acceptance criteria. You also need to ensure that the generated user stories are correct and refine them if not. At the end of the submission, provide screenshots of your ChatGPT prompts and answers, along with errors the ChatGPT made and that you had to correct. You may also use LLMs hosted locally. Points will be deducted if no screenshots are provided.

1. **User Stories (3 points)**

* **User Story Template**: As a <role>, I want <goal> [so that <benefit>]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **User Story Name** | **User Story Description** | **Priority** | **Estimated effort (hours)** |
| 1 | Choose a board size | As a player, I want to choose the board size so that I can start a game with my preferred dimensions. | High | 4 |
| 2 | Choose the game mode of a chosen board | As a player, I want to choose between simple or general game modes so that I can play the version I prefer. | High | 3 |
| 3 | Start a new game of the chosen board size and game mode | As a player, I want to start a new game so that I can play on my selected board size and game mode. | High | 3 |
| 4 | Make a move in a simple game | As a player, I want to place an "S" or "O" on the board during my turn in the simple game mode. | Medium | 5 |
| 5 | A simple game is over | As a player, I want the game to determine when the simple game is over, so I know when someone has won or when it's a draw. | High | 6 |
| 6 | Make a move in a general game | As a player, I want to place an "S" or "O" on the board during my turn in the general game mode. | Medium | 5 |
| 7 | A general game is over | As a player, I want the game to determine when the general game is over, so I know who won or if it is a draw. | High | 6 |

1. **Acceptance Criteria (AC) (8 points): Add/delete rows as needed.**

|  |  |  |  |
| --- | --- | --- | --- |
| **User Story ID and Name** | **AC**  **ID** | **Description of Acceptance Criterion** | **Status (completed, toDo, inProgress)** |
| 1. Choose a board size | 1.1 | **AC 1.1** Selecting a valid board size:  **Given** a player at the game initialization screen, **When** the player selects a valid board size from a dropdown list (n×n, where n > 2), **Then** the selected board size is displayed, and the game initializes with that size. | toDo |
| 1.2 | **AC 1.2** Selecting an invalid board size:  **Given** a player at the game initialization screen, **When** the player selects a board size outside the acceptable range (n ≤ 2), **Then** an error message is displayed, and the game does not start. | toDo |
| 2. Choose the game mode of a chosen board | 2.1 | **AC 2.1** Choosing a game mode:  **Given** a player at the game initialization screen, **When** the player selects either the simple or general game mode, **Then** the selected game mode is displayed, and the game begins with that mode. | toDo |
| 3. Start a new game of the chosen board size | 3.1 | **AC 3.1** Starting a new game:  **Given** a player who has selected the board size and game mode, **When** the player clicks the "Start New Game" button, **Then** a new game should start with the selected settings (board size and game mode). | toDo |
| 4. Make a move in a simple game | 4.1 | **AC 4.1** Making a move in a simple game:  **Given** a player during a simple game mode, **When** it's their turn, **Then** they can place an "S" or "O" on an empty cell. | toDo |
| 5. A simple game is over | 5.1 | **AC 5.1** Simple game victory:  **Given** a player during a simple game, **When** a player creates an SOS sequence, **Then** the game declares that player as the winner, and the game ends. | toDo |
| 5.2 | **AC 5.2** Simple game draw:  **Given** the game is in simple mode, **When** the board is full and no SOS sequence has been created, **Then** the game declares the game a draw. | toDo |
| 6. Make a move in a general game | |  | | --- | | 6.1 |  |  | | --- | |  | | **AC 6.1** Making a move in a general game:  **Given** a player during a general game mode, **When** it's their turn, **Then** they can place an "S" or "O" on an empty cell. | toDo |
| |  | | --- | | 6.2 |  |  | | --- | |  | | **AC 6.2** Multiple turns after forming SOS in general game:  **Given** a player creates an SOS sequence during their turn in the general game mode, **When** they form an SOS, **Then** they take another turn until they can no longer form an SOS. | toDo |
| 7. A general game is over | 7.1 | **AC 7.1** General game victory:  **Given** the general game mode, **When** the board is full, **Then** the player with the most SOS sequences is declared the winner. | toDo |
| 7.2 | **AC 7.2** General game draw:  **Given** the general game mode, **When** both players have the same number of SOS sequences, **Then** the game declares a draw. | toDo |

1. **Data flow diagram (DFD) (4 points):**

Consider a web-based SOS game that allows players from all over the world to play SOS games against each other (similar to chess.com for chess). Using data flow diagram examples presented in class, draw the data flow diagram for your global SOS game. You may use the tool of your choice, but it is a good idea to get familiar with draw.io (<https://app>.diagrams.net/) or other similar tools.

A diagram of a diagram

Description automatically generated

**Context and Level 0:**

At the top level, the diagram shows **Player 1** and **Player 2** as external entities, each interacting with the **Game Server**. The players send their moves to the server, which processes these moves and updates the game board. The server, in turn, sends the updated board back to both players, allowing them to see the current state of the game.

The **Game Server** interacts with two data stores: **Game State Database**: This stores the current game state, including the board layout and any moves made by the players. **Player Database**: This stores player-specific information such as player statistics, results from completed games, and other relevant data. Data flows from the players to the game server, where it is processed and stored. The server also retrieves information from the databases as needed, such as retrieving game states or updating player results after a game concludes.

**Level 1 (Decomposition of Game Server):**

I also decomposed the **Game Server** into three distinct processes for clarity: **Move Validation**: This process checks the validity of the moves sent by each player. Once validated, the valid move is passed on to the next process. **Game State Update**: After the move is validated, this process updates the current game state, storing the new board layout in the **Game State Database**. **Game Over Check**: This process checks whether the game has reached a conclusion (either a win or a draw). If the game is over, the player results are stored in the **Player Database**, and a notification is sent to both players. If the game is still ongoing, the updated game board is sent back to the players.

The data flows between these processes are straightforward:

Moves are validated and either rejected or passed on to the next stage.

The game state is updated and stored after each valid move.

The system checks for game over conditions and stores results if the game has ended.

**ChatGPT Free Version (auto) Outputs:**

**Initial Prompt:**

Complete the following user stories and their acceptance criteria:

1 Choose a board size

2 Choose the game mode of a chosen board

A screenshot of a black and white screen

Description automatically generated

A screenshot of a black screen

Description automatically generated

A screenshot of a computer program

Description automatically generated

While the generated outputs were mostly correct, there were a few refinements I needed to make:

I had to adjust the formatting of the acceptance criteria to match the format provided in the assignment template (i.e., Given-When-Then). ChatGPT provided the criteria in bullet points, which I then adapted into the required format. For the Choose a Board Size user story, I ensured that edge cases (like invalid inputs) were explicitly covered in the acceptance criteria. ChatGPT suggested validation for invalid board sizes, which I incorporated and refined to ensure clarity. I cross-checked the user stories to ensure they aligned with the project scope and matched the functionality described in the assignment. The generated user stories were accurate, so no major changes were required beyond some wording adjustments for clarity. Overall, ChatGPT’s output was highly useful in generating user stories and acceptance criteria. However, I needed to make some minor corrections and refinements to ensure the output was formatted correctly and covered all scenarios. I’ve included screenshots of the ChatGPT prompts, the initial outputs, and the corrections made for reference above.