**Project Sprint #2**

The SOS game is described in CS449HomeworkOverview.docx. You should read the description very carefully.

Your submission must include the GitHub link to your project and you must ensure that the instructor has the proper access to your project. You will receive no points otherwise.

**GitHub link:** [**https://github.com/mohamede2022/SOS-game/tree/main/Sprint%202**](https://github.com/mohamede2022/SOS-game/tree/main/Sprint%202)

Implement the following features of the SOS game: (1) the basic components for the game options (board size and game mode) and initial game, and (2) S/O placement for human players ***without*** checking for the formation of SOS or determining the winner. The following is a sample interface. The implementation of a GUI is required. You should practice object-oriented programming, making your code easy to extend. It is required to separate the user interface code and the game logic code into different classes (refer to the TicTacToe example). xUnit tests are required.

|  |  |  |
| --- | --- | --- |
| 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 | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  |  | | O |  |  |  |  |  |  |  | |  |  | S | O | S |  |  |  | |  |  |  |  | S |  |  |  | |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | S | | Red player  Icon  Description automatically generated S  Icon  Description automatically generated O |
|  | Current turn: blue (or red) |  |

Figure 1. Sample GUI layout of the Sprint 2 program

**Deliverables:**

1. **Demonstration (8 points)**

Submit a link to a video of no more than three minutes, clearly demonstrating that you have implemented the required features and written some automated unit tests. In the video, you must explain what is being demonstrated. No points will be given without a video link.

**YouTube/Panopto link:** [**https://umsystem.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=8196bec0-222f-4894-98ba-b37e00267695**](https://umsystem.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=8196bec0-222f-4894-98ba-b37e00267695)

|  |  |
| --- | --- |
|  | **Feature** |
| 1 | Choose board size ✅ |
| 2 | Choose game mode |
| 3 | Start a new game of the chosen board size and game mode ✅ |
| 4 | “S” moves ✅ |
| 5 | “O” moves ✅ |
| 6 | Automated unit tests ✅ |

1. **Summary of Source Code (1 points)**

|  |  |  |
| --- | --- | --- |
| Source code file name | Production code or test code? | # lines of code |
| sosgame/board.py | Production | 20 |
| sosgame/game.py | Production | 27 |
| sosgame/controller.py | Production | 17 |
| sosgame/player.py | Production | 8 |
| sosgame/gui\_board.py | Production | 34 |
| sosgame/gui\_main.py | Production | 93 |
| tests/test\_board.py | Test | 23 |
| tests/test\_game.py | Test | 19 |
| tests/test\_gui\_board.py | Test | 11 |
| tests/test\_player.py | Test | 12 |
| tests/test\_controller.py | Test | 18 |
| Total | | 282 |

**You must submit all source code to get any credit for this assignment.**

1. **Production Code vs User stories/Acceptance Criteria (3 points)**

Update your user stories and acceptance criteria from the previous assignment and ensure they adequately capture the requirements. Summarize how each of the following user story/acceptance criteria is implemented in your production code (class name and method name etc.)

|  |  |
| --- | --- |
| **User Story ID** | **User Story Name** |
| 1 | Choose a board size |
| 2 | Choose the game mode of a chosen board |
| 3 | Start a new game of the chosen board size and game mode |
| 4 | Make a move in a simple game |
| 6 | Make a move in a general game |

| **User Story ID & Name** | **AC ID** | **Class Name(s)** | **Method Name(s)** | **Status** | **Notes** |
| --- | --- | --- | --- | --- | --- |
| 1. Choose a board size | 1.1 | GUIBoard (gui\_board.py), SOSApp (gui\_main.py) | build\_board(), start\_new\_game() | Complete | Board dynamically initializes based on the user input size. |
| 2. Choose the game mode | 2.1 | SOSApp (gui\_main.py), GameController (controller.py) | start\_new\_game(), set\_letter() | Complete | Toggles between Simple and General modes using radio buttons. |
| 3. Start a new game of chosen board size and mode | 3.1 | GameController (controller.py), Game (game.py), Board (board.py) | start\_new\_game(), new\_game(), reset() | Complete | Clears previous board and sets the new game parameters. |
| 4. Make a move in a simple game | 4.1 | Board (board.py), GUIBoard (gui\_board.py) | place\_letter(), handle\_click() | Complete | Places “S” or “O” in selected cell and updates GUI. |
| 6. Make a move in a general game | 6.1 | Game (game.py), Board (board.py), GameController (controller.py) | make\_move(), place\_letter(), handle\_move() | Complete | Handles player moves and toggles turn. |

1. **Tests vs User stories/Acceptance Criteria (3 points)**

Summarize how each of the user story/acceptance criteria is tested by your test code (class name and method name) or manually performed tests.

|  |  |
| --- | --- |
| **User Story ID** | **User Story Name** |
| 1 | Choose a board size |
| 2 | Choose the game mode of a chosen board |
| 3 | Start a new game of the chosen board size and game mode |
| 4 | Make a move in a simple game |
| 6 | Make a move in a general game |

4.1 Automated tests directly corresponding to the acceptance criteria of the above user stories

You are required to use ChatGPT to create at least 2 unit tests. You also need to ensure that the generated user stories are correct, and refine them if not. At the end of the submission, provide the screenshots of your ChatGPT prompts and answers, along with errors ChatGPT made and you fixed. You may also use another LLM, including hosted locally. Points will be deducted if no screenshots are provided.

| **User Story ID** | **Acceptance Criterion ID** | **Class Name(s) of Test Code** | **Method Name(s) of Test Code** | **Description of Test Case (Input & Expected Output)** |
| --- | --- | --- | --- | --- |
| 1 | 1.1 | TestGame, TestBoard | test\_new\_game\_resets\_board(), test\_board\_initialization() | Input: new board size (e.g., 4 or 5) → Output: board grid updated to new size and all cells empty. |
| 2 | 2.1 | TestController | test\_new\_game\_flow() | Input: select mode “Simple” or “General” → Output: game mode updated correctly, board reset. |
| 3 | 3.1 | TestGame | test\_new\_game\_resets\_board() | Input: start new game Output: board cleared, current turn reset to blue player. |
| 4 | 4.1 | TestBoard, TestGuiBoard | test\_place\_letter(), test\_button\_grid\_creation() | Input: place letter “S” at (1,1) → Output: grid updated at (1,1) with “S”, GUI button exists. |
| 6 | 6.1 | TestGame | test\_turn\_toggle() | Input: make a move in general game → Output: turn toggled to next player. (modes implementation in sprint 3) |

4.2 Manual tests directly corresponding to the acceptance criteria of the above user stories

| **User Story ID** | **Acceptance Criterion ID** | **Test Case Input** | **Test Oracle (Expected Output)** | **Notes** |
| --- | --- | --- | --- | --- |
| 1 | 1.1 | User enters board size 5 | Board grid updates to 5x5, all cells empty | Verified via GUI and the console prints |
| 2 | 2.1 | User selects "General" mode | Game mode updated to "General" | Verified via GUI radio buttons |
| 3 | 3.1 | User clicks "New Game" | Board cleared, current turn resets to Blue | Verified via GUI and console prints |
| 4 | 4.1 | Player clicks cell (1,1) in Simple mode | Letter placed at (1,1), color corresponds to player | Verified via GUI button update |
| 6 | 6.1 | Player clicks cell (0,0) in General mode | Letter placed, turn toggled | Verified via GUI board prints |

4.3 Other automated or manual tests not corresponding to the acceptance criteria of the above user stories

| **Number** | **Test Input** | **Expected Result** | **Class Name of Test Code** | **Method Name of Test Code** |
| --- | --- | --- | --- | --- |
| 1 | Player chooses a letter | (current\_letter) updates | TestPlayer | test\_player\_choice() |
| 2 | Invalid move (cell already occupied) | Returns False, cell unchanged | TestBoard | test\_invalid\_move() |
| 3 | GUI button grid creation | Correct number of buttons (size x size) | TestGuiBoard | test\_button\_grid\_creation() |

A screenshot of a black and white screen

AI-generated content may be incorrect.

A computer screen shot of a black screen

AI-generated content may be incorrect.

**Errors:**

* ChatGPT initially suggested create\_board() and set\_board\_size() methods which do not exist in my code because I corrected them to build\_board() and start\_new\_game().
* Turn toggling tests required current\_turn comparisons. ChatGPT suggested comparing player names but my implementation uses player objects, so fixed accordingly.