**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/kimbrow-slice/SOSGame**

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:   
Game Interface:** [**https://umsystem.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=a854ad3a-3e83-4fdd-bd42-b299005f303f**](https://umsystem.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=a854ad3a-3e83-4fdd-bd42-b299005f303f) **Tests:** [**https://umsystem.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=716e533d-74cd-4ba7-a6fd-b2990060375b**](https://umsystem.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=716e533d-74cd-4ba7-a6fd-b2990060375b)

|  |  |
| --- | --- |
|  | **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 |
| GridSystem.cs | Production | 131 |
| MainWindow.axaml | Production | 233 |
| MainWindow.axaml.cs | Production | 144 |
| GameController.cs | Production | 45 |
| GameLogic.cs | Production | 91 |
| GameControllerTests.cs | Test | 34 |
| GameLogicTests.cs | Test | 83 |
| GridSystemUnitTest.cs | Test | 74 |
| Total | | 835 |

**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 and Name** | **AC ID** | **Class Name(s)** | **Method Name(s)** | **Status (complete or not)** | **Notes (optional)** |
| 1 Choose Board Game Size | 1.1 | GridSystem | SetGridSize | Completed | Initalizes grid size within the valid input range (3-12) |
|  | 1.2 | GridSystem | SetGridSize | Completed | Throw an exception for invalid size |
| 2 Choose the game mode | 2.1 | GameController | SetGameMode | Completed | Select a specific game mode for session |
| 3 Start a new game | 3.1 | Game Controller | StartNewGame | Completed | Game initiates correctly |
| 4 Make a move in a simple game | 4.1 | GameLogic | MakeMove | Completed | Valid moves in simple games will update the grid. |
|  | 4.2 | GameLogic | ValidateMove | Completed | Block invalid inputs and occupied cells |
| 5 Detect end of simple game | 5.1 | GameLogic | CheckGameEnd | In Progress | Detect game end based on the board status (single SOS sequence) |
| 6 Make a move in a general game | 6.1 | GameLogic | MakeMove | Completed | Valid moves in general games will update the grid. |

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.

**Summary of Production Code and Tests**

The automated and manual tests below fully cover the identified user stories. Each story clearly links acceptance criteria to the final implementation of methods and tests. This approach has allowed me to ensure that there is verifiable functionality within the program. Automated tests specifically validate grid sizing, move validation, turn handling, and boundary conditions. The manual tests verify the users interactions and visual feedback within the user interface.

|  |  |
| --- | --- |
| **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 and Name** | **Acceptance Criterion ID** | **Class Name (s) of the Test Code** | **Method Name(s) of the Test Code** | **Description of the Test Case (input & expected output)** |
| 1 Choose Board Game Size | 1.1 | GridSystemUnitTests | Grid\_Updates\_Size\_When\_SetGridSize\_Is\_Called\_With\_Valid\_Size | Input: Set grid size =5; Expected Output: Grid Size is updated to 5 |
|  | 1.2 | GridSystemUnitTests | SetGridSize\_Should\_Throw\_Exception\_When\_Invalid\_Size\_Is\_Given | Input: Grid Size=2; Expected Output: ArgumentException Thrown |
| 2 Choose the game mode | 2.1 | GameControllerTests | TestTurnSwitching | Input: Make Move (0,0,’S’); Expected Output: Turn switches successfully |
| 4 Make a move in a simple game | 4.1 | GameLogicTests | MakeMove\_Update\_Grid | Input: Move at (2,2) with ‘S’; Expected Output: Move is successful and grid updated correctly |
|  | 4.2 | GameLogicTests | MakeMove\_ReturnsFalse\_WhenCellOccupied | Input: Move at an occupied cell; Expected Output: Move is blocked |
|  | 4.3 | GameLogicTests | MakeMove\_ReturnsFalse\_WhenInvalidLetterProvided | Input: Invalid letters 'X', 'A', '1'; Expected Output: Move rejected |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User Story ID and Name** | **Acceptance Criterion ID** | **Test Case Input** | **Test Oracle (Expected Output)** | **Notes** |
| 2 Choose the game mode | 2.1 | Manually select game mode (Simple/General) | Game mode is correctly applied |  |
| 3 Start a new game | 3.1 | Manually start new game with selected settings | New game initiates successfully | There is additional input validation to ensure all inputs required are made |
| 5 Detect end of simple game | 5.1 | Play until no moves left | Game detectes end conditions for simple game based on SOS the first SOS sequence formed | NO END GAME IMPLEMENTED |
| 2 | 2.1 |  |  |  |
|  | … |  |  |  |

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 the Test Code** | **Method Name of the Test Code** |
| 1 | Negative grid coordinates (-1,-1) | Invalid move isn’t placed | GameLogicTests | MakeMove\_ReturnsFalse\_WhenCoordinatesNegative |
| 2 | Coordinates outside of grid boundaries | Invalid move isn’t placed as it is out of bounds | GameLogicTests | MakeMove\_ReturnsFalse\_WhenCoordinatesOutsideGrid |
| 3 | Verify initial grid creation | Grid initialize with 3x3 | GridSystemUnitTests | Grid\_Initializes\_With\_Default\_Size |

5 ChatGPT Automated Unit Test

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Class Name (s) of the Test Code** | **Method Name(s) of the Test Code** | **Description of the Test Case (input & expected output)** |
| 1 | GameControllerTests | Constructor\_InvalidGridSize\_ThrowsArgumentException | Input: Initialize an invalid grid size =2; Expect Output: Throw ArgumentException and do no update state |
| 2 | GameControllerTest | TestTurnSwitching | Test switching turns moveMade = (0,0, ‘S’); Expected Output: Switch turn to player 2 |

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.