**Sudoku Puzzle Solver Application**

*Garrett, Jarrett, Matt, Carl*

***+4 points***

1. **Introduction**

Sudoku puzzles are a type of puzzle in which players insert the numbers one to nine into a grid consisting of nine squares subdivided into a further nine smaller squares in such a way that every number appears once in each horizontal line, vertical line, and square. The players of the puzzle are usually given some sort of starting state of the puzzle, and they must work to fill in the remaining squares of the puzzle, while still following the previous rules. These puzzles are commonly found in newspapers across the world, and are usually fairly simple to solve. However, what happens when they cannot solve these puzzles? This was the central idea behind the Sudoku Puzzle Solver.

The Sudoku Puzzle Solver is a software application built in Java. It was designed so that a user, who is having trouble solving a Sudoku puzzle, may insert their current puzzle into the program, and the Sudoku Puzzle Solver will help them solve it. The application is set up with a nine by nine grid, where the users may input their current state of the puzzle with a mouse and keyboard, or they may choose to insert the information from a file. Either way, once the information is inserted into the grid, the user may choose to have the application solve it entirely, outputting the finished puzzle, or to solve a single square of the puzzle, by clicking on the appropriate buttons. If the current state of the puzzle is unsolvable, the Sudoku Puzzle Solver will tell them so, allowing the user to correct their mistakes. In the end, the Sudoku Puzzle Solver will help the user in their quest to solve their Sudoku puzzle.

1. **Proposed System**

The proposed system will solve players unsolved Sudoku puzzles. The player will enter an unsolved puzzle into a GUI, and the program will solve the puzzle and output the solution to the GUI.

***2.1 Functional Requirements***

IN-FN-01 The system shall allow the user to enter the original unsolved puzzle through a GUI.

IN-FN-02 The system shall be able to receive input from mouse and keyboard.

IN-FN-03 The system shall be able to read in an unsolved puzzle from a file.

IN-FN-04 The system shall allow the user to select a solve action through a clickable button once the unsolved puzzle is entered.

OUT-FN-01 The system shall utilize a GUI to accept user input and display the completed grid after the solve process is run.

OUT-FN-02 The system shall be able to write the initial, unsolved sudoku puzzle to a file.

OUT-FN-03 The system shall display a blank 9x9 grid to accept input into the system.

OUT-FN-04 The user shall have the option to display the board as it's being solved.

OUT-FN-05 The system shall be able to write the final, solved sudoku puzzle, to a file.

OUT-FN-06 The system shall display a completed 9x9 grid after the solve button is pressed and the solve process is completed.

OUT-FN-07 The system shall output music during runtime.

PT-FN-01 The system shall check each row in accordance to the game rules during the solve process.

PT-FN-02 The system shall check each column in accordance to the game rules during the solve process.

PT-FN-03 The system shall check each 3x3 square in accordance to the game rules during the solve process.

PT-FN-04 The system shall store counts of each number in each row and each column.

PT-FN-06 The system shall use the count of numbers to determine probability.

PT-FN-07 The system shall remove numbers from the list of possible correct integers in each box after checking each row, column, and 3x3 box for the same number.

PT-FN-11 The system shall be able to solve sudoku puzzle for user.

PT-FN-12 The system shall be able to run the solve process step-by-step for user.

***2.2 Non-Functional Requirements***

OUT-NF-08 The system shall output the solved sudoku puzzle within five seconds of clicking the solve button.

PT-NF-08 The system shall run using Java 1.8 and higher.

PT-NF-09 The system shall work on Windows Vista, 7, 8, 10.

PT-NF-10 The system shall be unavailable no longer than 5 minutes everyday.

PT-NF-13 The system shall be able to run the program remotely through a connected server.

PT-NF-14 The system shall be compatible with android and iOS systems.

PT-NF-15 The system shall assume all puzzles are solvable.

GR-NF-01 The system shall only allow unique integers in every row between 1 and 9.

GR-NF-02 The system shall only allow unique integers in every column between 1 and 9.

GR-NF-03 The system shall require each column and row to contain numbers 1 through 9 without repetition or omission.

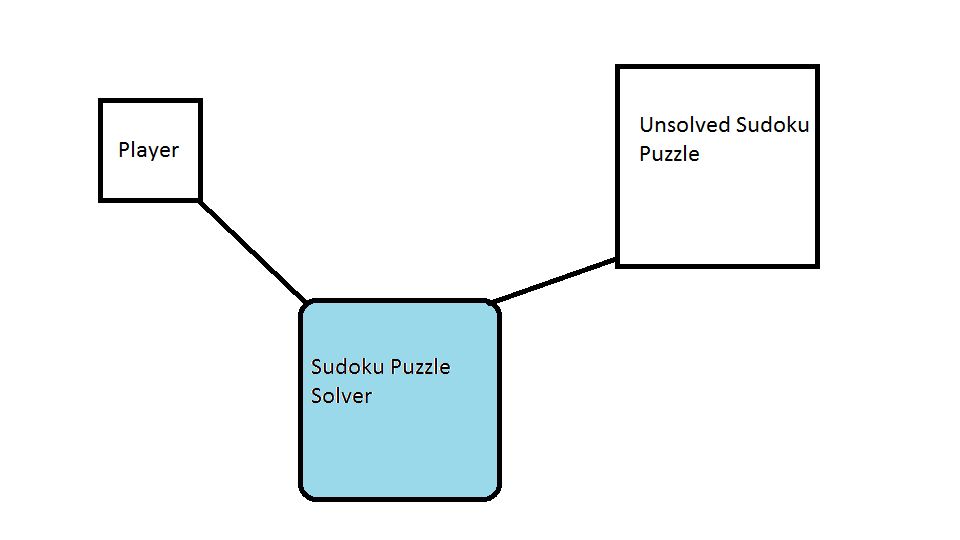
GR-NF-04 The system shall only permit unique integers in every one of the 9 3x3 boxes.

GR-NF-05 The system shall only permit one unique solution to each puzzle.

***2.3 System Models***

The models in this section include a context diagram, showing the external forces acting on the Sudoku Puzzle Solver, a single scenario, describing the features of the system, and use cases and use case diagrams, which describe the processes of the program in detail.

2.3.1 Context Diagram



2.3.2 Scenarios

|  |  |
| --- | --- |
| Scenario Name | Enter unsolved puzzle |
| Participating Actors instances | Bob: Player |
| Flow Of Events | 1. Bob can't solve his sudoku puzzles. 2. Bob opens Sudoku Puzzle Solver application to solve his sudoku puzzle. 3. Bob enters the unsolved puzzle into the GUI's nine by nine table with his keyboard 4. Bob presses the solve table button    1. The system solves the unsolved portion of the puzzle and outputs the result to the nine by nine table. 5. Bob presses the revert table button.    1. The system generated results are removed from the table, leaving the numbers from Bob's original input. 6. Bob presses the reset table button.    1. The nine by nine table is cleared of all input 7. Bob presses the choose file button.    1. The file explorer opens 8. Bob opens a .txt file that holds an unsolved Sudoku puzzle.    1. The 9x9 grid is filled with the .txt file data. 9. Bob checks the Save to File check box so that the next time he solves a puzzle it is written to a .txt file. 10. Bob presses the Solve Step button.     1. The system solves one cell of the 9x9 grid. 11. Bob presses the Solve Table button.     1. The system fills the empty cells of the 9x9 grid with the rest of the puzzle data.     2. The system writes the solved puzzle data to a .txt file. 12. Bob closes the window.     1. System terminates. |

2.3.3 Use Cases

|  |  |
| --- | --- |
| Use Case Name | Solve table |
| Participating Actors | Player |
| Flow Of Events | 1. Player opens the Sudoku Solver application.    1. Sudoku Solver prompts player to specify size and numbers of sudoku puzzle. 2. Player enters the unsolved sudoku puzzle into the GUI and submits it.    1. Sudoku Solver computes solved sudoku puzzle.    2. Sudoku Solver outputs solved puzzle to the GUI. |
| Entry Condition | * Player has Sudoku Solver on computer. |
| Exit Conditions | * Sudoku Solver outputs solved puzzle. * Sudoku Solver outputs that the puzzle can not be solved. |
| Quality Requirements | * Player should receive output five seconds after submission. * Sudoku Solver should handle unsolvable exceptions and prompt player. |

|  |  |
| --- | --- |
| Use Case Name | Solve step |
| Participating Actors | Player |
| Flow Of Events | 1. Player opens the Sudoku Solver application. 2. Player enters the unsolved sudoku puzzle into the GUI. 3. Player clicks solve step.    1. Sudoku Solver one cell of 9x9 grid. |
| Entry Condition | * Player has Sudoku Solver on computer. * There is an incomplete puzzle entered into grid. |
| Exit Conditions | * Sudoku Solver outputs 1 step more completed puzzle. * Sudoku Solver outputs that the puzzle can not be solved. |
| Quality Requirements | * Player should receive output five seconds after submission. * Sudoku Solver should handle unsolvable exceptions and prompt player. |

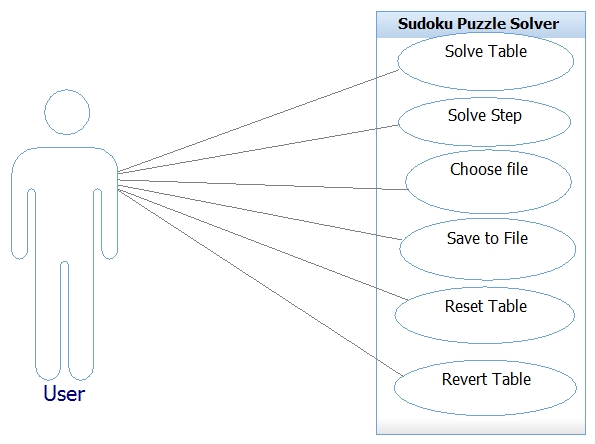
|  |  |
| --- | --- |
| Use Case Name | Choose File |
| Participating Actors | Player |
| Flow Of Events | 1. Player clicks choose file.    1. Open file window appears. 2. Player chooses unsolved .txt file.    1. File is loaded into UI. |
| Entry Condition | * Player has Sudoku Solver on computer. * Player has unsolved puzzle in .txt format. |
| Exit Conditions | * Sudoku Solver fills UI based on file. |
| Quality Requirements | * File should load in under 1 second. * File explorer should only allow correct file format to be selected. |

|  |  |
| --- | --- |
| Use Case Name | Reset table |
| Participating Actors | Player |
| Flow Of Events | 1. Player clicks reset table.    1. UI is reset to default properties. |
| Entry Condition | * Player has Sudoku Solver on computer. |
| Exit Conditions | * Sudoku Solver clears UI. |
| Quality Requirements | * UI should clear in under 1 second. |

|  |  |
| --- | --- |
| Use Case Name | Revert table |
| Participating Actors | Player |
| Flow Of Events | 1. Player clicks revert table.    1. UI reloads file or input before the solve button was pressed. |
| Entry Condition | * Player has Sudoku Solver on computer. |
| Exit Conditions | * Sudoku Solver reloads UI based on file, if file exists. * Sudoku Solver reloads UI based on first grid data, if file does not exist. |
| Quality Requirements | * File should load in under 1 second. |

|  |  |
| --- | --- |
| Use Case Name | Save To File |
| Participating Actors | Player |
| Flow Of Events | 1. Player checks save to file box. 2. Player clicks solve table or solve step reaches last step.    1. System saves UI grid to .txt file. |
| Entry Condition | * Player has Sudoku Solver on computer. * Player has solved sudoku file in table grid. |
| Exit Conditions | * Sudoku Solver saves file based on table grid. |
| Quality Requirements | * File should save in under 1 second. * File should be saved as .txt. |

2.3.4 Use Case Diagrams



1. **Glossary**

|  |  |
| --- | --- |
| Abbreviation | Description |
| GR-NF | Functional game rules requirement |
| GUI | Graphic User Interface |
| IN-FN | Functional input requirement |
| IN-NF | Non-functional input requirement |
| iOS | Apple iPhone Operating System |
| OUT-FN | Functional output requirement |
| OUT-NF | Non-functional output requirement |
| PT-FN | Functional process or task requirement |
| PT-NF | Non-functional process or task requirement |

1. **Reference**

*None*