Sudoku Solvers Requirement Document

1. Overview

1.1 Scope

Sudoku solvers is software that serves to promote a fun and educational experience for users. To accomplish this goal the application will create a functional sudoku puzzle that users can complete. The board needs to integrate a difficulty system that allows users to change the sudoku puzzles' difficulty level. The software will have an easily accessible section that describes the rules, tips, and strategies that pertain to solving a sudoku puzzle. The software will allow users to ask the software to give hints to any sudoku puzzle the user is currently solving. Lastly, the software will provide different variations of sudoku puzzles to ensure the user can continue using and adapting techniques taught to improve their skills.

2. Specific Requirements

2.1 External Interfaces

2.1.1 User Interface

The user interface will be designed to both provide an intuitive and engaging experience for the user while allowing them to learn and expand their knowledge of sudoku. The user interface will provide a clean and easily understandable UI layout that aims to provide the user immediate access to the game and simple directions the user can take to adjust the board. The user will also be able to navigate to other sections of the page that will educate the user about sudoku. Sudoku puzzles will be prioritized being displayed to the user with smaller buttons such as difficulty and puzzle generation being above the board while buttons that affect the current puzzle such as hints and solving the puzzle being displayed below the board.

The system will have information pertaining to sudoku accessible on the same page, but outside of the initial view of the user. The user interface enables the user to play a game of sudoku without information cluttering the user's view, but allows the user to easily access this information. The user interface will be done using PyQt for a multitude of reasons such as GUI, familiarity, and tools. PyQt is a good tool that is easy to use and has many features that the application needs such as grids, drop-down menus, buttons, and message handlers. Additionally, PyQt has a low entry barrier to begin working with, enabling a newer team to take less time learning how to use the software.

2.1.2 Hardware Interface

The application is intended for desktop use with interactions with the board and user happening through mouse and keyboard. The mouse will be able to click individual cells and enter numbers by directly pressing the associated number directly on the keyboard. It should also allow the function of the arrow keys to move between cells at any given point. There will be no need of hardware past mouse and keyboard

2.1.3 Software Interface

The application will operate with local storage for saving puzzle progress where a security measure will take place to ensure smooth and safe connections. Additionally, creating an interactive sudoku game is a goal of the application, but having it provide educational value to the user is another purpose of this application. By allowing the user to view the logic behind the generation of puzzles the user may learn about the software, algorithms, and data values and how they function together.

2.1.4 Communications Interface

The application uses local storage with little external dependencies; however, the application may in the future include events where users can compete with one another. If in the case the application allows for users to interact with each other then future iterations of the software will include a database to store times users can get while playing sudoku. This time can then be saved and compared to other time values to create an user to user interaction. This would become a feature. A current idea would be to have the application external connect to retrieve educational material that users can use if the team chooses to use others' sudoku research.

2.2 System Features

2.2.1 Puzzle generation

2.2.1.1 Introduction

The system will generate a puzzle with conditions specified by the user. The condition the system uses is the difficulty level the user sets before generating the puzzle. The user will be able to adjust puzzles by selecting a difficulty level from easy, medium, and hard with each level decreasing the number of cells the user will be given at the beginning of each sudoku game. The sudoku puzzle adheres to the rules of sudoku ensuring that each board corresponds to a 9x9 size limit with numbers being unique across rows, columns, and boxes. By reducing numbers present on the board at the start this will make the puzzle more difficult for the user to solve. The puzzle will be displayed within 3 seconds from when the user asks the system to generate a puzzle. The puzzle will be displayed on a clear board and be intractable with simple buttons around the board to input numbers, ask for hints, or generate a new puzzle. Once the user completes the puzzle the system will congratulate the user and will ask the user if they want to start a new puzzle.

2.2.1.2 Stimulus

- User clicks difficulty button

- User selects (easy, medium, hard)
- User clicks “Generate Puzzle” button

2.2.1.3 Response

- System generates a board with number of cells reflecting difficulty chosen
- System sends user message when solving generated board

2.2.1.4 Functional Requirements

- The system shall have a difficulty button
- The system shall have 3 different difficulty levels
- The system shall use the correct difficulty level to determine pre filled cells before generating
- The system shall generate a board in less than 3 seconds
- The system shall send a message when the user completes a puzzle
- The system shall always fill in a minimum of 17 cells with numbers

2.1.2 Puzzle solver

2.2.2.1 Introduction

The system will allow the user to have the system solve the entire puzzle through pressing a button labeled “Auto-Solve”. This button shall be active throughout the puzzle's creation to its completion. After the user confirms they want the puzzle solved the system will fill in any unfilled cells on the board with the correct answers producing a fully solved puzzle that is displayed to the user. The board does this at once the board is generated but in order to save time the solution is stored for later use and when called on displays the completed board that had already been calculated. The entirety of the board will have numbers following sudoku rules ensuring the puzzle shows the correct unique solution with numbers 1-9 being present only once per row, column, and box. The system will take no longer than 1 seconds to return a complete sudoku puzzle back to the user after the initial request.

2.2.2.2 Stimulus

- User clicks “Auto-solve” button

2.2.2.3 Response

- System display the board with all cells filled in correctly

2.2.2.4 Functional Requirements

- The system shall take no longer than 1 second to display to the user a complete and correct board

- The system shall have an “Auto-solve” button
- The system shall display a completed board despite how much the user has completed
- The system shall have saved the completed board when it was first generated
- The system shall use dancing links algorithm to solve boards

2.2.3 Hint system

2.2.3.1 Introduction

The system will provide users with solutions to specific cells. The user may have trouble solving a sudoku puzzle and rather than have the system generate a new puzzle or outright solve the current one, the system will provide a number to an unfilled cell on a puzzle at the request of the user. The number that the system chooses to fill in must be unfilled on the board with the number chosen being one that adheres to the rules of sudoku. The system will take no longer than 1 seconds to provide a user with a hint.

2.2.3.2 Stimulus

- User selects cell
- User has no cell selected
- User presses the “Hint” button

2.2.3.3 Response

- System display selected cell hint
- System displays hint for random cell

2.2.3.4 Functional Requirements

- The system shall provide hint within 1 second from when the user requests it
- The system shall provide a hint for a specific cell when the user has a cell selected
- The system shall provide a hint even when a specific cell is not requested
- The system shall provide a hint in a unfilled cell
- The system shall indicate through a message that a hint was given

2.2.4 Sudoku Variations

2.2.4.1 Introduction

The application will support different variations of sudoku. Variations refer to puzzles with different sizes, conditions, and or shape, but this application will mainly cover the alternate conditions. The main variation that this application will work to achieve is the need for sub-sections such as the many 3x3 blocks in normal sudoku. These 3x3 blocks work as a condition that players must solve by finding unique numbers for each cell in the block such that no number is duplicated. A variation in the application will work to keep blocks but not restrict it to a 3x3 block format allowing the blocks to form in an irregular region in the board, but retains the condition that each region still has 9 cells assigned to it. This specific variation type is known as jigsaw sudoku and is known to increase difficulty by making normally used techniques harder to apply.

2.2.4.2 Stimulus

- User selects “Variation” option
- User selects “Jigsaw Sudoku”
- User presses the “Generate Puzzle” button

2.2.4.3 Response

- System displays variation name of the sudoku puzzle
- System displays generated sudoku puzzle with blocks being irregular

2.2.4.4 Functional Requirements

- The system shall provide puzzle within 3 seconds from when the user requests it
- The system shall provide a dropdown that allows the user to select variations
- The system shall provide sudoku variations to the user
- The system shall have a minimum of 17 cells filled in when the puzzle is generated
- The system shall display the current variation of the sudoku puzzle

2.3 Functions

- The system shall allow the user to move between cells
- The system shall allow the user to input numbers into cells
- The system shall allow the user to check if the number is correct or incorrect
- The system shall allow the user to generate a new sudoku puzzle
- The system shall allow the user to request from the system to return a hint for the current board
- The system shall have a section to teach the user the rules of sudoku
- The system shall have multiple variations of sudoku
- The system shall be able to allow the user to have the system solve the puzzle
- The system shall update the display of the board after each input from the user
- The system shall display a sudoku puzzle when the site is first opened
- The system shall have functioning buttons
- The system shall allow the user to resume their puzzle if they leave during one
- The system shall support desktop applications

2.4 Performance Requirements

The system will be an application that supports Windows, Linux, and macOs. The application is expected to execute user commands under 1 second for inputting numbers, navigating cells, and requesting hints. Otherwise, the application is expected to take no longer than 3 seconds for actions such as generating or loading a saved puzzle from memory. The system is expected to handle one input at a time, through button presses or entering numbers into a cell. Local storage will be used to automatically save progress on puzzles so that when the user closes the application or chooses to save the puzzle, the current puzzle will be saved to local storage and can be resumed when the user returns. Unless the user decides to delete the files used to save the puzzle information the puzzle will remain in the same state the user last left it at.

2.5 Logical database Requirements

The system will store sudoku puzzle templates that will be used with algorithms to generate puzzles for the user. The system requires a difficulty to be chosen before generating the puzzle which will be used in puzzle, hint, and solving functions. The frequency at which the system generates a new puzzle depends on the user's activity regarding how often they request a new puzzle to be generated. Once the puzzle is generated the entire board is logged into the system where it can be altered by the system when a user requests a function to be performed. Due to the feature that allows users to resume their game of sudoku the system will need to store current information of the current board with data such as number locations, empty cells, and difficulty of that board. Each board the user plays will be unique and ensure only one solution which will remain the same after the board is stored and accessed again. The user should be allowed to continue their game if circumstances arise, thus keeping it stored is important. The feature will incorporate local storage to store the puzzle data until the user clears file data.

2.6 Design constraints

The application is designed in such a way that most systems will be able to run it with no problem. This application will involve no hardware and only rely on the software to perform proper calculations using algorithms researched for optimal time. The use of dancing links will be important for sudoku as it will assist in not repeating numbers for the areas sudoku rules state should only have one unique number for. Dancing links is an algorithm that deletes or adds nodes to a doubly linked list. The application needs to back track to ensure that sudoku rules are validated which may be resource intensive depending on the way it's implemented. The application must run on low intensive hardware as it is not very complex but the resources have to abide by making the application provide results to the user under a specified time limit. Overall, the application needs to quickly produce results to the user, make sure those results adhere to the rules of sudoku, and not require anything outside of the application itself.

2.6.1 Standard compliance

Data related to the application through user actions, puzzle generation, hints, difficulty, and auto-solving must keep consistent data values.

2.7 Software system attributes

2.7.1 Reliability

The application is expected to remain open 95% of the time with little downtime, but in the event the application goes down whether due to over flooding of users past the limit or other errors then the application should return within a reasonable time. Additionally, the user's progress on boards will remain saved on local storage to provide the user relief in the event the application does have trouble working. To deal with errors that may occur, an error handling system will be implemented that will provide users with error messages in different scenarios. Error messages will be made for functions the software fails to perform due to faulty data communication between the user and the application.

2.7.2 Availability

The application is expected to remain online 99% of the time with the 1% being factored into maintenance and updates. Due to the software not requiring complex features and little external dependencies, there is less risk of the software having faulty interactions with other systems. This will reduce the range of problems the software can encounter. The application will have a limit of users that will be closely monitored and properly adjusted to reflect users interest and demand. However, due the small scale of the application starting with lower limits can help gauge popularity, serve

2.7.3 Security

The system does not require sensitive user information to be stored on databases, but data encryption will be used to protect user saved data regarding saved puzzles. Transport Layer Security (TLS) encrypts data transmitted and since the application will allow saving puzzles then ensuring a secure connection reduces risk of data exposure no matter what type of data is transferred. The application will be available through secure distributors. This will provide relief to the user by ensuring the application download does not pose risk to the users hardware and/or software.

2.7.4 Maintainability

The software will be designed in a way that lowers coupling and promotes high cohesion. The application's functions should not rely too heavily on one another such that changes to one would lead to errors in another. By separating functions dependance on one another but not affecting cohesion will allow both implementation of new features or updating current ones to not impact the system as whole. This will help create and maintain each feature in the software reducing risk of interfering with functionality enabling developers to test more features and provide consistent updates to keep the application up to date. Python as the application programming language helps with implementing features due to its advantage at faster development and how straightforward it is to develop.