# Sudoku Solver Visualizer Report

# 1. Introduction

# 1.1 Purpose

The purpose of this project is to develop a Sudoku solver visualizer using Java Swing. The visualizer will demonstrate the solving process of a Sudoku puzzle step-by-step, making it easier to understand the backtracking algorithm used to solve the puzzle.

# 1.2 Scope

This report covers the design, implementation, and testing of the Sudoku solver visualizer. It provides an overview of the architecture, key components, and features of the application.

# 2. System Overview

# 2.1 Architecture

The system is built using Java and Swing for the graphical user interface. The application follows a Model-View-Controller (MVC) architecture:

- **Model** : Represents the Sudoku board and the solving algorithm.
- **View** : Displays the Sudoku board and the solving process.
- **Controller** : Handles user inputs and updates the view accordingly.

# 2.2 Key Components

- **SudokuBoard** : Represents the Sudoku grid and provides methods for checking the validity of numbers.
- **Solver** : Implements the backtracking algorithm to solve the Sudoku puzzle.
- **SudokuPanel** : A Swing component that visualizes the Sudoku board.
- **MainFrame** : The main application window that integrates all components.

# 3. Design and Implementation

# 3.1 SudokuBoard Class

The `SudokuBoard` class stores the current state of the Sudoku puzzle and provides methods to:

- Check if a number can be placed in a cell.

- Retrieve and set numbers in the grid.

- Verify if the puzzle is solved.

# 3.2 Solver Class

The `Solver` class contains the backtracking algorithm to solve the puzzle. Key methods include:

- `solve()`: Initiates the solving process.

- `isSafe(int row, int col, int num)`: Checks if placing a number in a specific cell is valid.

- `solveSudoku()`: Recursive method to solve the puzzle.

# 3.3 SudokuPanel Class

The `SudokuPanel` class extends `JPanel` and is responsible for:

- Drawing the Sudoku grid.

- Updating the grid as the solving process progresses.

- Highlighting the current cell being solved.

# 3.4 MainFrame Class

The `MainFrame` class sets up the main application window and integrates the `SudokuBoard`, `Solver`, and `SudokuPanel`. It also provides controls for:

- Starting the solving process.

- Resetting the puzzle.

- Loading a new puzzle.

# 4. Features

# 4.1 Interactive Visualization

The visualizer updates the grid in real-time, showing the step-by-step process of solving the puzzle. Cells being processed are highlighted to indicate progress.

# 4.2 User Controls

- **Start/Stop** : Allows users to start or pause the solving process.

- **Reset** : Resets the board to its initial state.

- **Load Puzzle** : Enables users to load a new Sudoku puzzle.

# 4.3 Speed Control

Users can adjust the speed of the solving visualization, allowing them to slow down or speed up the process as needed.

# 5. Testing

# 5.1 Unit Testing

Individual components, such as `SudokuBoard` and `Solver` , were tested to ensure correctness. Edge cases, such as invalid puzzles and fully solved puzzles, were also tested.

# 5.2 Integration Testing

The integration of the `SudokuBoard` , `Solver` , and `SudokuPanel` was tested to ensure smooth interaction and correct visualization.

# 5.3 User Testing

User testing was conducted to gather feedback on the usability and functionality of the visualizer. Adjustments were made based on this feedback to improve the user experience.

# 6. Conclusion

The Sudoku solver visualizer provides an effective way to understand the backtracking algorithm used to solve Sudoku puzzles. The interactive and visual nature of the application makes it a valuable educational tool. Future improvements could include additional solving algorithms, better user interface elements, and more customization options for puzzles.
