

*Lovely Professional University,Punjab.*

*Department of Computer Science and Engineering (CSE)*

*Semester: (Autumn, Year: 2024).*

**Sudoku Solver**

*Course Title: Data Structures*

*Course Code: CSES003*

*Section: 9SK01*

Students Details

|  |  |
| --- | --- |
| **Name** | **ID** |
|  |  |
| Aditya Kumar | 12221056 |
|  |  |

*Submission Date:* *12 June*

*Course Teacher’s Name: Rahul Singh*

**Lab Project Status**

**Marks:** **Signature:**

**Comments:** **Date:**

**Contents**

|  |  |  |  |
| --- | --- | --- | --- |
| **1 Introduction** | |  | **3** |
| 1.1 | Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 3 |
| 1.2 | Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 3 |
| 1.3 | Problem Definition . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 3 |
|  | 1.3.1 | Problem Statement . . . . . . . . . . . . . . . . . . . . . . . . | 3 |
|  | 1.3.2 | Complex Engineering Problem . . . . . . . . . . . . . . . . . . | 4 |
| 1.4 | Design Goals/Objectives . . . . . . . . . . . . . . . . . . . . . . . . . | | 4 |
| 1.5 | Application . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 4 |
|  | 1.5.1 | Output visualization . . . . . . . . . . . . . . . . . . . . . . . | 6 |

|  |  |  |  |
| --- | --- | --- | --- |
| **2 Design/Development/Implementation of the Project** | | | **8** |
| 2.1 | Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 8 |
| 2.2 | Project Details . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 8 |
|  | 2.2.1 | Key Features . . . . . . . . . . . . . . . . . . . . . . . . . . . | 8 |
| 2.3 | Implementation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 10 |
|  | 2.3.1 | User Interface: . . . . . . . . . . . . . . . . . . . . . . . . . . | 10 |
|  | 2.3.2 | Sudoku Generation Algorithm: . . . . . . . . . . . . . . . . . . | 10 |
|  | 2.3.3 | Sudoku Solving Algorithm: . . . . . . . . . . . . . . . . . . . | 11 |
|  | 2.3.4 | Visualization Techniques: . . . . . . . . . . . . . . . . . . . . | 12 |
|  | 2.3.5 | Implementation Details (With Screenshots) . . . . . . . . . . . | 12 |
| 2.4 | Algorithms . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 13 |

|  |  |  |  |
| --- | --- | --- | --- |
| **3 Performance Evaluation** | | | **17** |
| 3.1 | Simulation Environment/ Simulation Procedure . . . . . . . . . . . . . | | 17 |
| 3.2 | Results Analysis/Testing . . . . . . . . . . . . . . . . . . . . . . . . . | | 18 |
|  | 3.2.1 | Result\_Case\_1 . . . . . . . . . . . . . . . . . . . . . . . . . . | 18 |
|  | 3.2.2 | Result\_Case\_2 . . . . . . . . . . . . . . . . . . . . . . . . . . | 19 |

1

|  |  |  |  |
| --- | --- | --- | --- |
|  | 3.2.3 | Result\_Case\_3 . . . . . . . . . . . . . . . . . . . . . . . . . . | 20 |
| 3.3 | Results Overall Discussion . . . . . . . . . . . . . . . . . . . . . . . . | | 21 |
|  | 3.3.1 | Complex Engineering Problem Discussion . . . . . . . . . . . | 21 |
| **4 Conclusion** | |  | **23** |
| 4.1 | Discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 23 |
| 4.2 | Limitations . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 23 |
| 4.3 | Scope of Future Work . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 24 |
| 4.4 | References . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 25 |

2

**Chapter 1**

**Introduction**

**1.1** **Overview**

The Sudoku solver game is a puzzle where players fill a 9x9 grid with numbers from 1 to 9, ensuring each row, column, and 3x3 sub grid contains all numbers without repetition. Players input numbers, validate their choices, and a solver algorithm finds the solution using backtracking. Hints may be provided, and players can request assistance. Once the grid is filled, the game displays the solution. Sudoku offers a challenging experience that requires logical thinking and problem-solving skills.

**1.2** **Motivation**

The Sudoku solver game offers an engaging and enjoyable experience for players of all skill levels. It serves as a valuable tool to improve problem-solving abilities, learn Su-doku strategies, and develop logical reasoning skills. By observing the solver algorithm, players can enhance their problem-solving techniques and gain a sense of accomplish-ment. The game provides a platform to practice, learn, and experience the satisfaction of solving challenging Sudoku puzzles. Embrace the challenge, sharpen your mind, and enjoy the journey of Sudoku solving with the Sudoku solver game. [**?**].

**1.3** **Problem Definition**

**1.3.1** **Problem Statement**

When we chose this project, We were aware of facing some problems like:

* Input: The program should accept a 9x9 grid as input, representing a partially filled Sudoku puzzle. The input grid should contain numbers from 1 to 9, with 0 or empty cells denoting the blank spaces.
* Validation: The program should validate the input grid to ensure it follows the rules of Sudoku. Each row, column, and 3x3 subgrid should contain all numbers

3

from 1 to 9 without repetition. If the input grid violates the Sudoku rules, the program should indicate an invalid input.

* Solver Algorithm: The program should implement a backtracking algorithm to find a solution for the Sudoku puzzle. It should systematically explore different number combinations by placing valid numbers in empty cells and backtracking whenever an invalid configuration is reached.
* Solution Output: Upon successful solving of the Sudoku puzzle, the program should output the complete solution. The output should be a 9x9 grid where all empty cells are filled with valid numbers, satisfying the Sudoku rules.
* Efficiency: The solver program should efficiently find the solution to Sudoku puzzles of varying difficulty levels. It should minimize unnecessary computations and provide a timely response for both simple and complex puzzles.
* User Interface (optional): To enhance the user experience, the program may in-clude a user interface where users can input the Sudoku puzzle, view the solution, and interact with the solver algorithm. The user interface can provide features such as hints, error checking, and the ability to enter puzzles step by step.

The goal of the Sudoku solver program is to provide an automated solution for Sudoku puzzles, enabling users to solve puzzles quickly and effortlessly. The program should deliver accurate and efficient results while maintaining the integrity of the Sudoku rules.

**1.3.2** **Complex Engineering Problem**

**1.4** **Design Goals/Objectives**

* Solve Sudoku puzzles accurately and efficiently.
* Ensure valid input and reject invalid or unsolvable puzzles.
* Provide an intuitive and interactive user interface.
* Optimize the solver algorithm for quick solving.
* Handle invalid or unsolvable puzzles with appropriate feedback.
* Support multiple difficulty levels for varied challenges.
* Track and display performance metrics for a competitive experience.

**1.5** **Application**

The Sudoku-solving game has various applications that cater to puzzle enthusiasts, edu-cational institutions, and anyone looking to improve their problem-solving skills. Here are some key applications of the Sudoku-solving game:

4

* Entertainment and Recreation: The Sudoku-solving game provides a fun and en-gaging activity for individuals of all ages. It offers a challenging puzzle-solving experience that can be enjoyed during leisure time or as a mental exercise to un-wind and relax.
* Skill Development: The Sudoku solving game serves as a valuable tool for devel-oping and honing problem-solving skills. It enhances logical reasoning, critical thinking, pattern recognition, and strategic planning abilities. Regularly engaging in Sudoku solving can lead to improved cognitive skills and mental agility.
* Educational Tool: The Sudoku solving game finds application in educational set-tings, such as schools and educational institutions. It can be used to introduce and teach concepts like logic, deductive reasoning, and number patterns. It encour-ages students to think analytically and boosts their problem-solving capabilities.
* Personal Growth: The Sudoku solving game promotes personal growth by fos-tering perseverance, patience, and the ability to handle challenges. It encourages individuals to overcome obstacles, learn from mistakes, and strive for continuous improvement. The game cultivates a growth mindset and a sense of accomplish-ment upon successfully solving puzzles.
* Brain Training: The Sudoku solving game is often used as a brain-training exer-cise to keep the mind sharp and active. Regular practice helps maintain mental acuity and memory retention. It offers a stimulating workout for the brain and can be part of a comprehensive brain-training regimen.
* The Sudoku solving game is widely available on various platforms, including mobile devices and online websites. Its accessibility allows users to engage in puzzle-solving anytime, anywhere, providing convenience and flexibility.
* Project Coordination and Management: AutoCAD’s project management features allow the team to track and manage the progress of the design project. They can create schedules, set deadlines, and assign tasks to team members, ensuring efficient coordination and completion of the design tasks.

In summary, the Sudoku solving game has versatile applications, ranging from enter-tainment and skill development to educational use and personal growth. It offers an en-joyable and beneficial experience that challenges the mind, promotes problem-solving skills, and enhances cognitive abilities.

5

**1.5.1** **Output visualization**

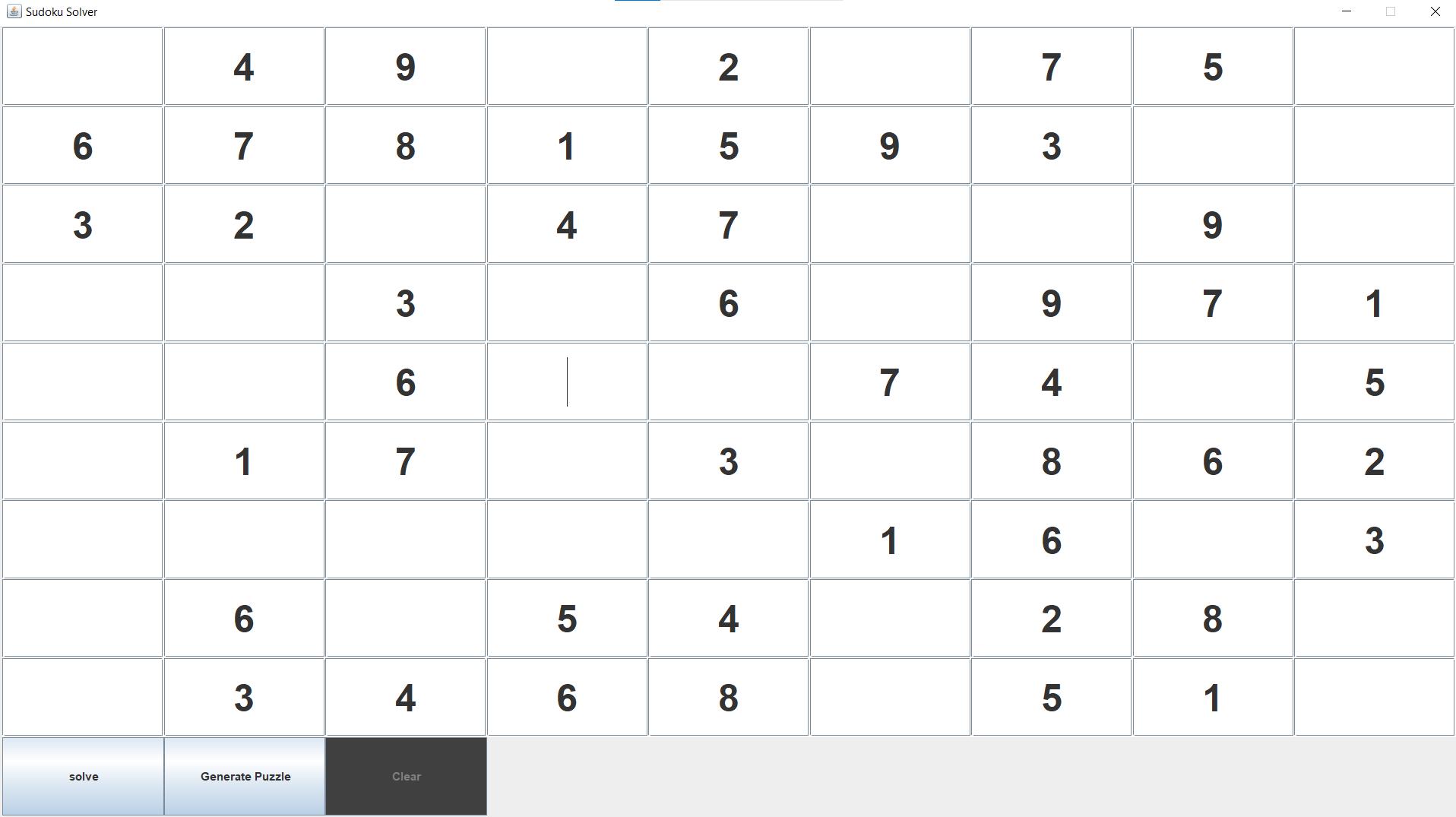


Figure 1.1: Visualizer

6

Table 1.1: Summary of the attributes touched by the mentioned projects

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Name of the P Attributes** |  | **Explain how to address** |  |  |
|  |  |  |  |  |  |
|  | **P1:** Depth of knowledge required |  | Familiarity with logical reasoning, number pat- |  |  |
|  |  |  |  |
|  |  |  | terns, and problem-solving techniques is essen- |  |  |
|  |  |  | tial to effectively solve Sudoku puzzles. |  |  |
|  |  |  |  |  |  |
|  | **P2:** Range of conflicting require- |  | Balancing efficiency and accuracy, while pro- |  |  |
|  | ments |  | viding user-friendly features and maintaining |  |  |
|  |  |  | adherence to Sudoku rules, presents a range of |  |  |
|  |  |  | conflicting requirements in Sudoku solving. |  |  |
|  |  |  |  |  |  |
|  | **P3:** Depth of analysis required |  | The depth of analysis required for Sudoku solv- |  |  |
|  |  |  | ing involves employing advanced logical de- |  |  |
|  |  |  | duction techniques, including complex patterns |  |  |
|  |  |  | and inter-dependencies, to uncover hidden rela- |  |  |
|  |  |  | tionships and make precise choices. |  |  |
|  |  |  |  |  |  |
|  | **P4:** Familiarity with issues |  | Developing familiarity with common issues in |  |  |
|  |  |  | Sudoku solving, such as invalid input, unsolv- |  |  |
|  |  |  | able puzzles, and efficient algorithms, helps en- |  |  |
|  |  |  | sure accurate solutions and a smooth solving |  |  |
|  |  |  | experience |  |  |
|  |  |  |  |  |  |
|  | **P5:** Extent of applicable codes |  | The applicable codes for Sudoku solving en- |  |  |
|  |  |  | compass algorithms for puzzle generation, val- |  |  |
|  |  |  | idation, backtracking, and user interface imple- |  |  |
|  |  |  | mentation. |  |  |
|  |  |  |  |  |  |
|  | **P6:** Extent of stakeholder involve- |  | Balancing the input and expectations of di- |  |  |
|  | ment and conflicting requirements |  | verse stakeholders while managing conflicting |  |  |
|  |  |  | requirements to create a Sudoku solving solu- |  |  |
|  |  |  | tion that satisfies various user preferences and |  |  |
|  |  |  | objectives. |  |  |
|  |  |  |  |  |  |
|  | **P7:** Interdependence |  | The successful solving of Sudoku puzzles re- |  |  |
|  |  |  | lies on the interdependence between logical de- |  |  |
|  |  |  | duction, pattern recognition, and the ability to |  |  |
|  |  |  | eliminate possibilities through iterative testing |  |  |
|  |  |  | and backtracking. |  |  |
|  |  |  |  |  |  |

7

**Chapter 2**

**Design/Development/Implementation of the Project**

**2.1** **Introduction**

The Sudoku Solving project is a user-friendly application centred around a 9x9 grid panel and three key buttons: "Sudoku Generate," "Solve," and "Clear." It provides an intuitive interface for generating, solving, and clearing Sudoku puzzles. The "Sudoku Generate" button creates valid and solvable puzzles, displaying them on the grid panel. The "Solve" button employs efficient algorithms to determine the solution, updating the grid dynamically. The "Clear" button resets the grid for new challenges. The project prioritizes user experience and interface design, offering a sleek platform for Sudoku enthusiasts.

**2.2** **Project Details**

The Sudoku Solving project is designed to provide a user-friendly interface for solving Sudoku puzzles. It utilizes a 9x9 grid panel and three buttons: "Sudoku Generate," "Solve," and "Clear." This document provides a detailed overview of the project, in-cluding key features, implementation details, and visualization techniques.

**2.2.1** **Key Features**

* **Sudoku Generation:** The "Sudoku Generate" button generates a new Sudokupuzzle with a unique solution. It ensures that the puzzle adheres to the rules of Sudoku, such as having no repeating digits in rows, columns, or smaller 3x3 sub-grids.
* **Sudoku Solver:** The "Solve" button employs an efficient algorithm to solve thegenerated Sudoku puzzle. It uses a backtracking technique to explore possible solutions and finds the correct solution that satisfies all the Sudoku rules. The solver also identifies invalid or unsolvable puzzles, providing appropriate feed-back to the user.

8

* **Clear Functionality:** The "Clear" button resets the Sudoku grid to its initial state,allowing the user to start a new puzzle or erase their progress on the current one. results in a clear and intuitive manner.

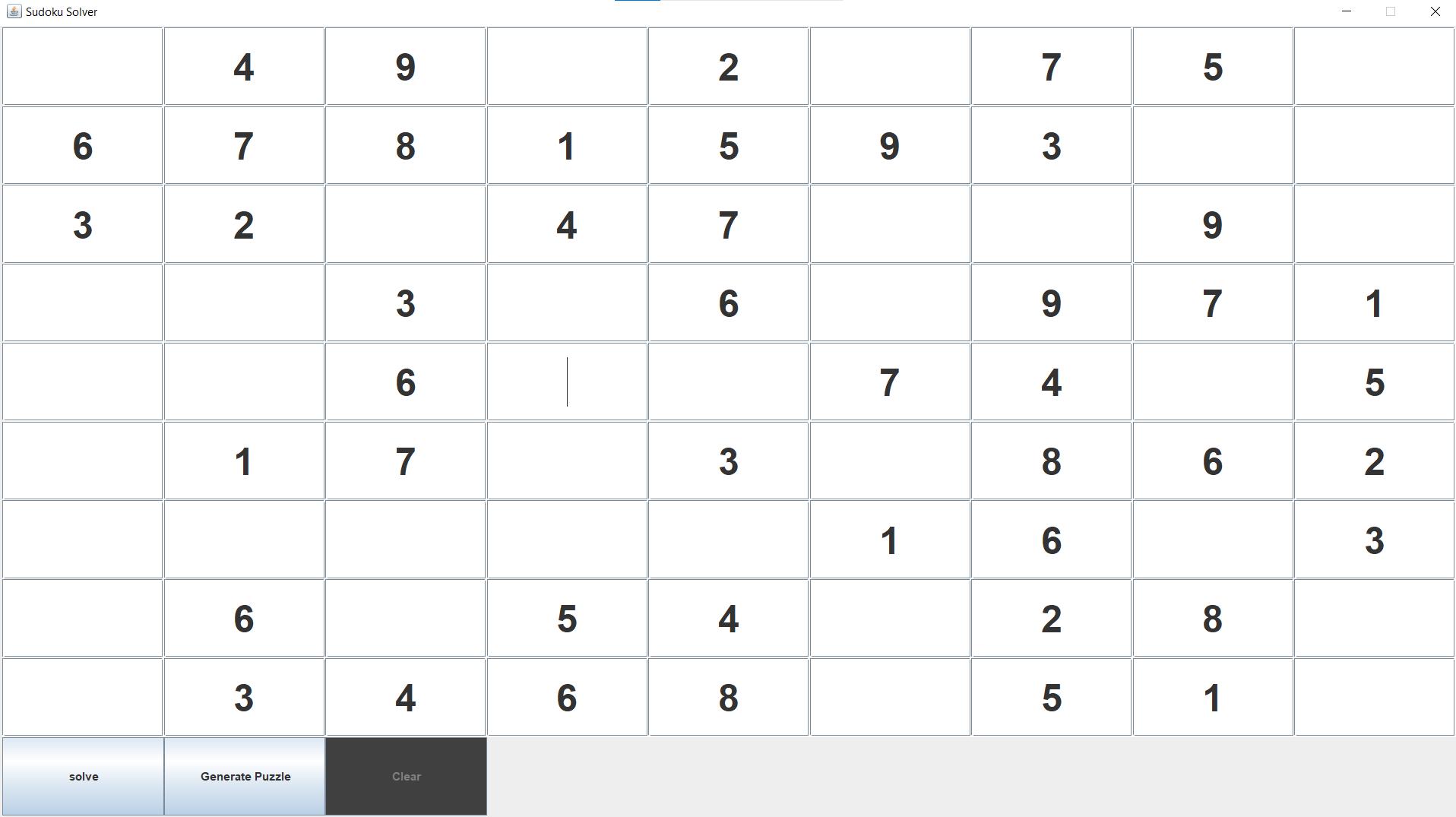


Figure 2.1: User Interface Of Sudoku solver

9

**2.3** **Implementation**

The project employs a graphical user interface (GUI) that includes a 9x9 grid panel representing the Sudoku puzzle. Each cell in the grid can be populated with a digit from 1 to 9, or left empty for the user to fill. The interface is intuitive and user-friendly, with clearly labeled buttons and visual cues to guide the user’s interactions.

**2.3.1** **User Interface:**

The project employs a graphical user interface (GUI) that includes a 9x9 grid panel representing the Sudoku puzzle. Each cell in the grid can be populated with a digit from 1 to 9, or left empty for the user to fill. The interface is intuitive and user-friendly, with clearly labeled buttons and visual cues to guide the user’s interactions.

**2.3.2** **Sudoku Generation Algorithm:**

To generate a new Sudoku puzzle, the project employs an algorithm that ensures a unique solution. It starts with an empty grid and progressively fills cells while adher-ing to the Sudoku rules. The algorithm employs techniques like randomization and constraint satisfaction to create diverse puzzles.

* **Empty Grid Initialization:** The algorithm begins with an empty 9x9 Sudokugrid where all cells are initialized to be empty.
* **Randomized Cell Population:** To create a unique puzzle, the algorithm popu-lates a random selection of cells in the grid. The number of initially filled cells can vary, depending on the desired difficulty level of the puzzle. For example, easier puzzles may have more initially filled cells, while harder ones have fewer.
* **Constraint Satisfaction** After populating a set of cells, the algorithm performsconstraint satisfaction techniques to ensure that the puzzle remains solvable and adheres to Sudoku rules. It checks that no initial conflict exists between the pre-filled cells, ensuring that no digit violates the constraints within its row, column, or 3x3 sub-grid.
* **Backtracking Approach:** The algorithm then employs a backtracking strategyto complete the puzzle by filling the remaining empty cells. It iterates through each empty cell and attempts to place a valid digit (1 to 9) that does not violate any Sudoku rules. If a digit leads to a conflict, the algorithm backtracks and tries a different digit until a valid placement is found.
* **Solution Uniqueness:** Throughout the puzzle generation process, the algorithmensures that the solution obtained is unique. After completing the puzzle, it ver-ifies that there is only one possible solution by attempting to solve the generated puzzle using a Sudoku solver algorithm. If multiple solutions are found, the al-gorithm modifies the filled cells or backtracks to create a puzzle with a unique solution.

10

**2.3.3** **Sudoku Solving Algorithm:**

The Sudoku-solving algorithm is responsible for finding the solution to a given Sudoku puzzle. It employs an efficient approach that systematically explores possible digit placements in empty cells while ensuring that all Sudoku rules are satisfied. Here are more details about the Sudoku-solving algorithm:

* **Backtracking:** The solving algorithm utilizes a backtracking technique, which isa recursive search strategy that explores all possible solutions by making informed guesses and backtracking when a conflict or inconsistency arises. This approach guarantees finding the correct solution by systematically exploring the solution space.
* **Empty Cell Selection** The algorithm starts by selecting an empty cell in the Su-doku grid to fill. It can follow various strategies for selecting the next cell, such as choosing the top-leftmost empty cell or selecting the cell with the fewest pos-sibilities.
* **Digit Placement:** For the selected empty cell, the algorithm attempts to place adigit (1 to 9) that satisfies the Sudoku constraints. It checks if the chosen digit violates any rules within the cell’s row, column, or 3x3 sub-grid. If a conflict arises, the algorithm backtracks and tries a different digit.
* **Recursive Exploration:** After placing a digit in the current cell, the algorithmrecursively explores the next empty cell, repeating the digit placement process. This recursive exploration continues until all cells are filled, and a valid solution is found. If a conflict occurs in any step, the algorithm backtracks to the previous cell and tries a different digit until a solution is ultimately found.
* **olution Verification:** Once a solution is obtained, the algorithm verifies its cor-rectness by ensuring that it satisfies all Sudoku constraints. It checks that no digit is repeated in any row, column, or 3x3 sub-grid. If the solution is invalid, the algorithm continues the backtracking process to find an alternative solution or reports that the puzzle is unsolvable.
* **Optimizations:** To improve efficiency, the solving algorithm can incorporate var-ious optimizations. For example, it can use techniques like constraint propagation or constraint satisfaction to reduce the number of possibilities and narrow down potential digit placements. Additionally, the algorithm can employ strategies like "minimum remaining values" or "least-constraining value" heuristics to prioritize cells with the fewest possibilities, improving the search efficiency.

11

**2.3.4** **Visualization Techniques:**

The project incorporates a time delay feature to showcase the solving process gradually. By introducing a pause between each step, the user can observe how the algorithm progresses through the puzzle. The time delay can be adjusted based on the desired speed of the animation, allowing the user to follow the solving process at their own pace.

**Tools and Libraries**

* Programming Language: Java.
* Integrated Development Environment (IDE): visual St
* Visualization Libraries: Java Swing.
* User Interface (UI) Libraries or Frameworks: Java AWT, Java Swing,

**2.3.5** **Implementation Details (With Screenshots)**

**User Interaction:**

The Sudoku Solving project offers a user-friendly interface for generating and solving Sudoku puzzles. Upon launching the application, a 9x9 grid panel is displayed, repre-senting the Sudoku puzzle. The user can click the "Sudoku Generate" button to create a new puzzle, which is populated with a unique solution. Manual input is optional, allow-ing the user to enter digits into empty cells. By clicking the "Solve" button, the solving algorithm starts and gradually fills the cells using a backtracking approach. Time delay visualization and color highlighting enhance the user experience, showcasing the solv-ing process step by step. Once a solution is found, the completed puzzle is displayed on the grid. The "Clear" button allows the user to reset the grid. The project provides an intuitive interface for users to interact with Sudoku puzzles, making it enjoyable and engaging.

12

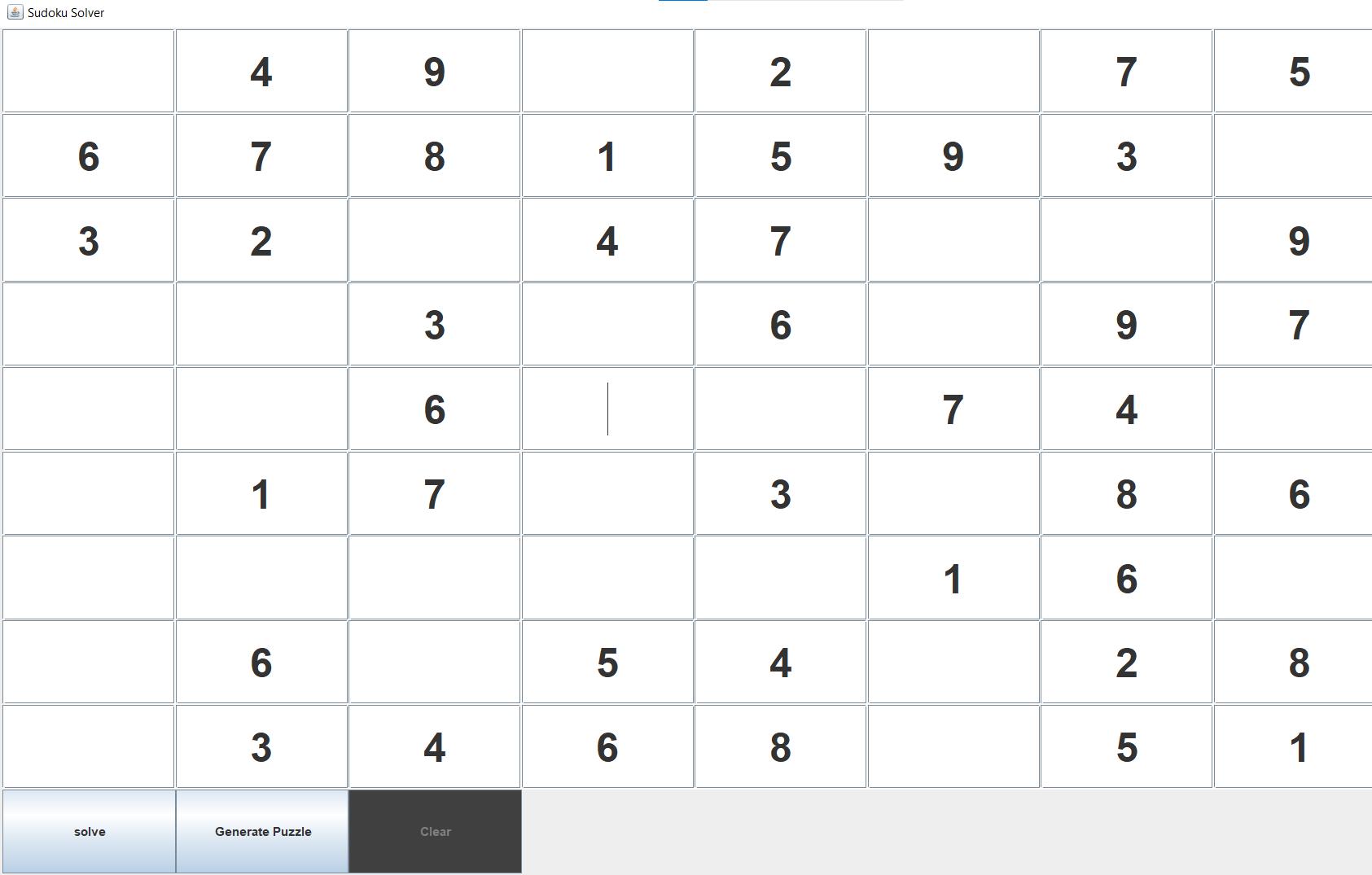


Figure 2.2: User Interface / Buttons Of Visualizer

**2.4** **Algorithms**

Algorithm is the prepossess to solve a problem.It is noting but a step by step procedure for the full program.Here is the main algorithm part for my project .

13

**Algorithm 1:** Sudoku Solving Algorithm

**Input:** Sudoku grid

**Output:** Boolean indicating if Sudoku is solved

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| /\* Recursive function | | to solve the Sudoku puzzle | | \*/ |
| /\* Find the first | empty | | location in the Sudoku grid | \*/ |
| /\* If no empty location | | | is found, the Sudoku is already |  |
| solved |  |  |  | \*/ |
| /\* Initialize row | and | column variables | | \*/ |

* **int** row = 0; **int** col = 0; **int[]** a =FindEmptyLocation(row, col);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | /\* If there are no empty locations, Sudoku is already solved | | | |  |
|  | \*/ | | |  |  |
| **2** | **if** a[0] == 0 **then** | | |  |  |
| **3** |  |  | **return true**; |  |  |
|  |  |  |  |
|  |  |  |  |  |  |
|  | /\* Get the position of the empty row and column | | | \*/ |  |
| **4** | row = a[1]; col = a[2]; | | |  |  |
|  | /\* Attempt to fill numbers from 1 to 9 into the empty | | |  |  |
|  |  |  | location | \*/ |  |

* **for** INTi = 1; i≤SIZE; i++ **do**

/\* Check if number i can be safely filled into the empty

location \*/

* **if** NumberIsSafe(i, row, col) **then**

/\* Fill the number i into the empty location \*/

* get [row][col] = i

/\* Recursive call to solve the Sudoku with the updated

grid \*/

* **if** public boolean solveSudoku()() **then**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **9** |  |  |  | **return true** |  |
|  |  |  |  |  |  |
|  | /\* | | | Backtracking | \*/ |
|  | /\* | | | If the solution is wrong, reassign the value | \*/ |

1. get [row][col] = 0
2. **return false**;
3. public boolean solveSudoku() ();

14

**Algorithm 2:** Find Empty Location in Sudoku Grid Algorithm

**Input:** Sudoku grid, row, col

**Output:** Array with flag, row, and column values

* **public int[]** FindEmptyLocation((INTrow,INTcol))
* int flag = 0; // Iterate through each cell in the Sudoku grid for (int i = 0; i < SIZE; i++) for (int j = 0; j < SIZE; j++) // Check if the cell is empty if (get

[i][j] == 0) // Assign the row and column values of the empty cell row = i; col = j; flag = 1;

* // Create an array with flag, row, and column values **int[]** a = flag, row, col;

**return** a;

* // If no empty cell is found, return an array with flag = 0 and row = -1, col = -1 **int[]** a = flag, -1, -1; **return** a;
* // Example usage of the FindEmptyLocation function **int** row = 0; **int** col = 0; **int[]** result =FindEmptyLocation(row, col);

**Algorithm 3:** Check if Number is Safe to Place in Sudoku Grid Algorithm

**Input:** Sudoku grid, number n, row r, column c

**Output:** Boolean indicating if number n is safe to place at position (r, c)

* **public boolean** NumberIsSafe((INTn,INTr,INTc))
* // Check if number n already exists in the same column **for** INT i = 0; i < SIZE; i++ **do**
* **if** get[r][i] == n **then**
* **return false**
* // Check if number n already exists in the same row **for** INT i = 0; i < SIZE; i++

**6**

**7**

**do**

**if** get[i][c] == n **then**

**return false**

* // Check if number n already exists in the corresponding sub-matrix **int**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | rowst art = (r/3) 3; **int**colst art = (c/3) 3; | | | | | |  |
| **9** | **for** INTi = rowst art;i<rowst art+3;i+ + **do** | | | | | |  |
| **10** | **for** INTj = colst art;j<colst art+3;j+ + **do** | | | | | |  |
| **11** |  |  |  | **if** get[i][j] == n **then** | | |  |
| **12** |  |  |  |  |  | RETURN FALSE |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

1. RETURN TRUE
2. // Example usage of the NumberIsSafe function INT number = 5; INT row = 2;

INT col = 4; BOOLEAN isSafe = NumberIsSafe (number, row, col);

**15**

15

**Algorithm 4:** Sudoku Solving Algorithm

**Input:** Sudoku grid

**Output:** Boolean indicating if Sudoku is solved

* solveSudoku()
  + Recursive function to solve the Sudoku puzzle
  + Find the first empty location in the Sudoku grid
  + If no empty location is found, the Sudoku is already solved
  + Initialize row and column variables
* row ← 0 col ← 0 []a ← FindEmptyLocation(row, col)
  + If there are no empty locations, Sudoku is already solved **3 if** a[0] == 0 **then**

**4return true**

* + Get the position of the empty row and column
* row ← a[1] col ← a[2]

// Attempt to fill numbers from 1 to 9 into the empty

location

* **for** i←1**to**SIZE **do**
* Check if number i can be safely filled into the empty location
* **if** NumberIsSafe(i, row, col) **then**

// Fill the number i into the empty location

* get [row][col] ← i
* Recursive call to solve the Sudoku with the updated grid
* **if** solveSudoku() **then**

|  |  |  |
| --- | --- | --- |
| **10** |  | **return** TRUE |
|  |  |  |

* Backtracking
* If the solution is wrong, reassign the value

1. get [row][col] ← 0
2. **return** FALSE
3. solveSudoku ()

16

**Chapter 3**

**Performance Evaluation**

**3.1** **Simulation Environment/ Simulation Procedure**

The simulation environment was developed using Java programming language, specif-ically utilizing the Java Swing library for creating the graphical user interface (GUI). The environment provides a user-friendly interface for Sorting Algorithm Visualizer. The development environment used for this project includes:

* Java Development Kit (JDK) version 7 or higher
* Integrated Development Environment (IDE) such as Visual Studio code. this case, I have used Visual Studio code editor

**Device**

* **Brand**: Dell
* **Model Name**: Inspiron 15 3501
* **Screen Size**: 15.6 inches
* **Colour**: Silver
* **SSD**: 512 GB
* **CPU Model**: i5-1135G7
* **RAM Memory Installed Size**: 8 GB
* **Operating System**: Windows 10 Pro
* **Graphics Card Description**: NVIDIA GeForce MX330 with 2GB DDR5

17

**3.2** **Results Analysis/Testing**

The Sudoku Solving project aims to provide a user-friendly interface for generating and solving Sudoku puzzles. It utilizes a 9x9 grid panel and incorporates three main buttons: "Sudoku Generate," "Solve," and "Clear." The "Sudoku Generate" button gen-erates a new Sudoku puzzle with a unique solution, offering users fresh challenges. The "Solve" button triggers the solving algorithm, visually displaying the step-by-step solving process on the grid. It employs time delay visualization to enhance the user experience. The "Clear" button allows users to reset the grid, providing a clean slate for new puzzles or starting over. Together, these buttons create an interactive environment for users to generate, solve, and engage with Sudoku puzzles..

**3.2.1** **Result\_Case\_1**

**Sudoku Generate Button:**

This is the picture of the Sudoku Generating Button work

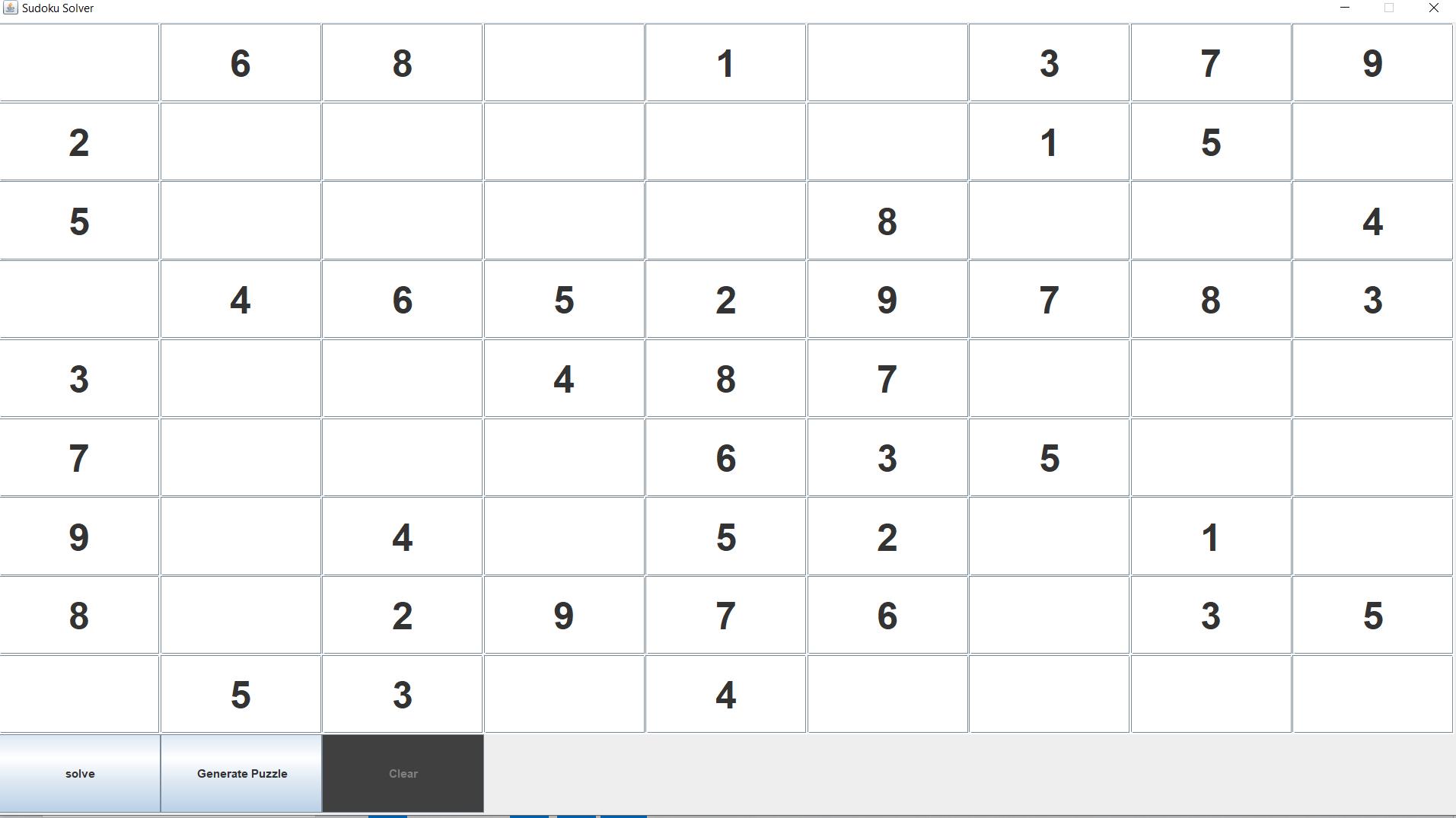


Figure 3.1: Generate Button

18

**3.2.2** **Result\_Case\_2**

**Sudoku solving Button:**

This is the picture of the Sudoku-solving Button work

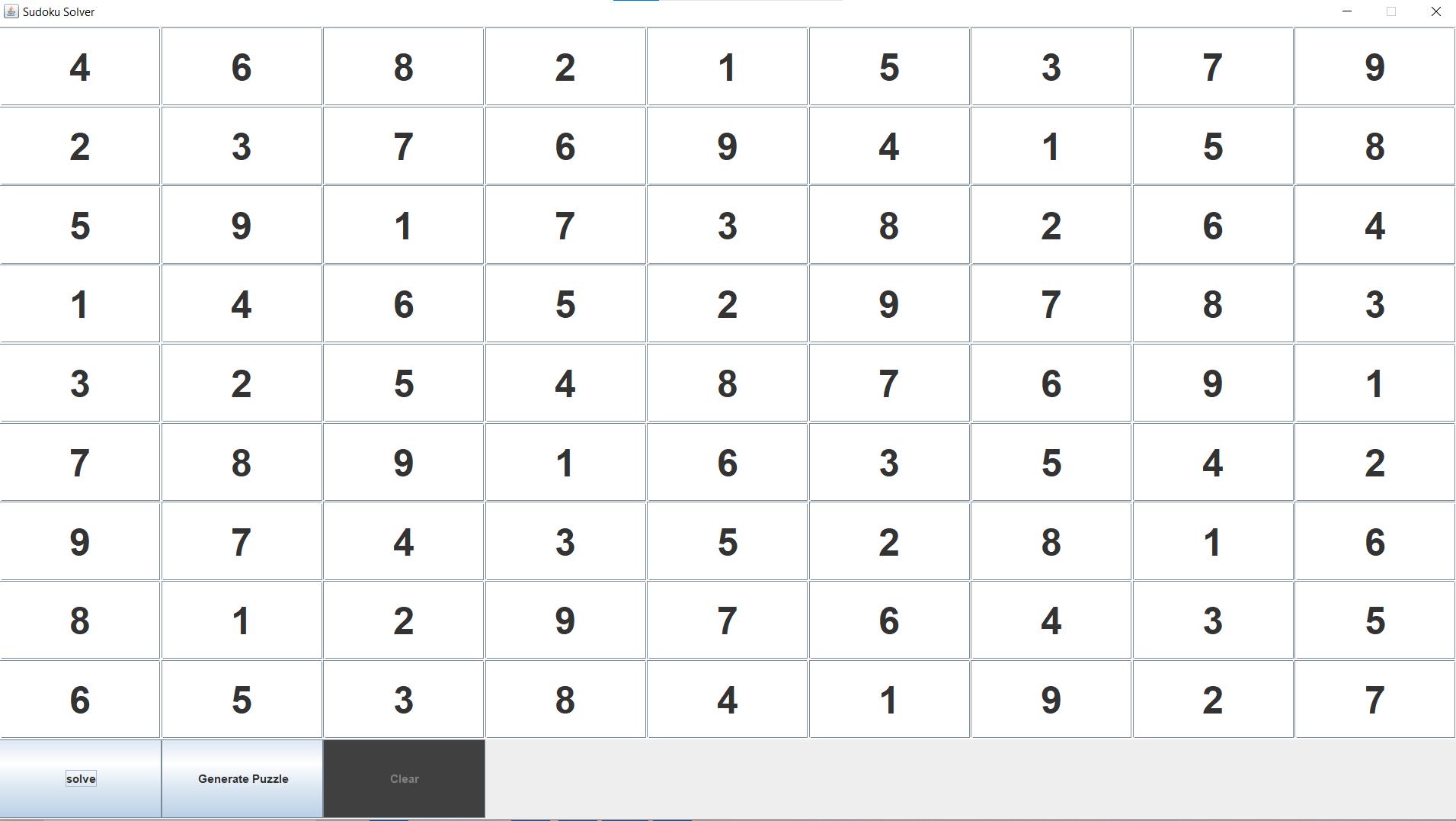


Figure 3.2: Marge Sort Visualization

19

**3.2.3** **Result\_Case\_3**

**Clear Button:**

This is the picture of clear Button work

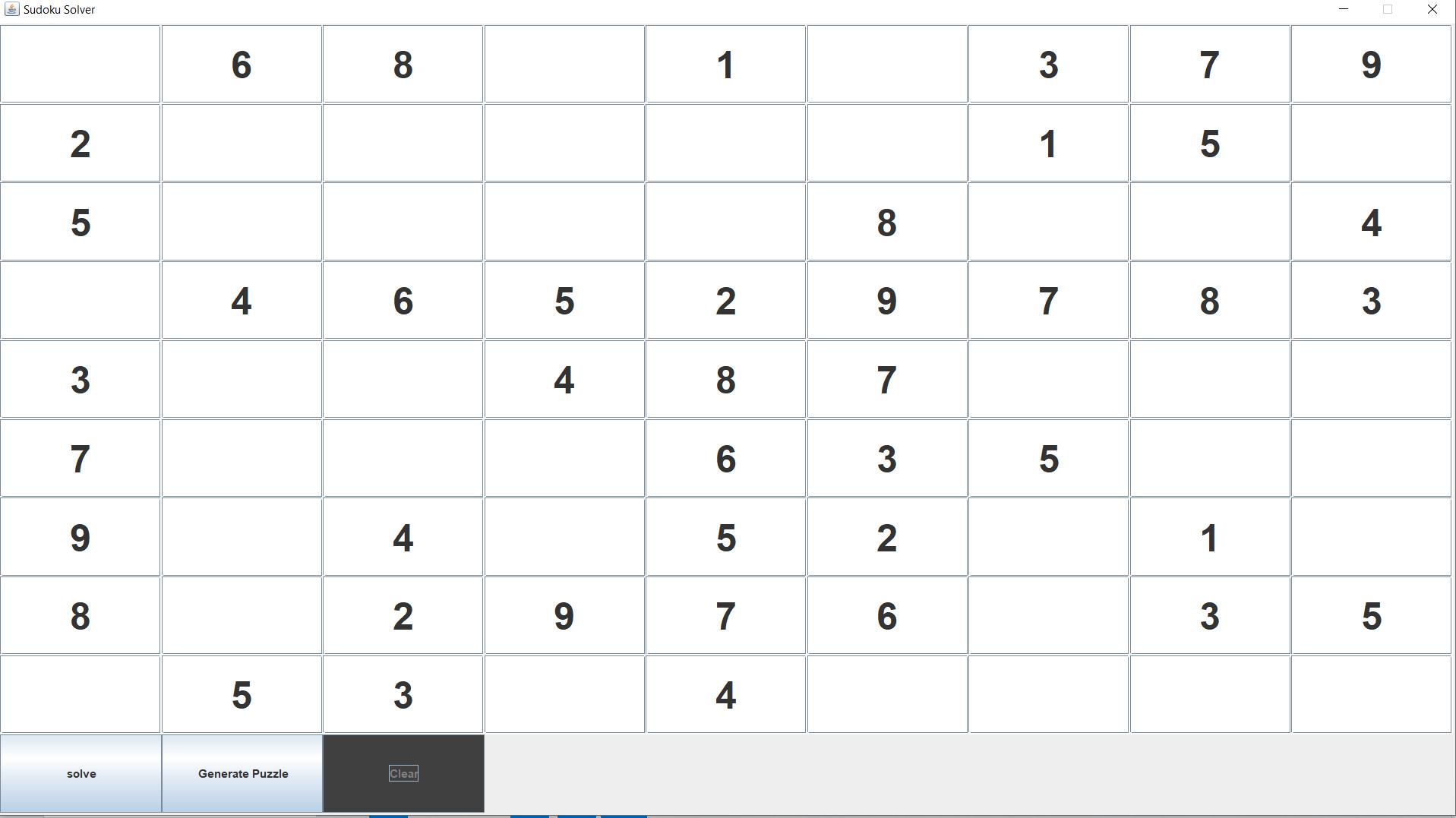


Figure 3.3: Selection Sort Visualization

20

**3.3** **Results Overall Discussion**

The Sudoku Solving project offers a user-friendly interface for generating and solving Sudoku puzzles. With a 9x9 grid panel and three main buttons, namely "Sudoku Gener-ate," "Solve," and "Clear," the project provides a comprehensive experience for Sudoku enthusiasts.

The "Sudoku Generate" button generates new puzzles, ensuring a unique solution each time. The Sudoku generation algorithm populates a random selection of cells in the grid, resulting in puzzles with varying levels of difficulty. This feature allows users to have a continuous supply of fresh challenges.

The "Solve" button triggers the Sudoku solving algorithm, which employs a back-tracking approach to fill the empty cells systematically. The solving process is visually displayed on the grid, with a time delay feature and color highlighting for enhanced visualization. Users can observe each step of the algorithm and gain insights into the solving process.

The "Clear" button provides a convenient option for users to reset the grid to its initial state. This feature allows users to start fresh with a new puzzle or erase their progress on the current one, ensuring flexibility and convenience.

By combining these functionalities, the Sudoku Solving project offers an interactive and engaging experience. Users can generate new puzzles, solve them with the assis-tance of the algorithm, and reset the grid as needed. The project provides a user-friendly interface, making it accessible to Sudoku enthusiasts of all skill levels.

**3.3.1** **Complex Engineering Problem Discussion**

Several complex engineering problems across different categories.

1. **Depth of Knowledge Required:** The Sudoku Solving project requires a depthof knowledge in various areas. A strong understanding of Sudoku rules and logic is necessary, along with proficiency in algorithmic problem solving to implement the Sudoku solving algorithm effectively. Competence in a programming lan-guage, such as Python, Java, C++, or JavaScript, is essential for developing the project. Familiarity with data structures and grid representation is required to manipulate and store Sudoku puzzles. Basic knowledge of user interface design principles and development frameworks like HTML, CSS, JavaScript, Swing, or JavaFX is necessary to create an intuitive user interface. Understanding how to incorporate time delay and visualization techniques to demonstrate the solving process is important. Proficiency in testing and debugging is crucial for ensur-ing the correctness and reliability of the project. This broad range of knowledge ensures the successful development and implementation of the Sudoku Solving project.
2. **Range of Conflicting Requirements:** The Sudoku Solving project encompassesa range of conflicting requirements that must be carefully addressed. On one hand, the project aims to generate Sudoku puzzles that are challenging and di-verse, requiring a sophisticated algorithm for puzzle generation. On the other

21

hand, the solving algorithm must be efficient and capable of solving puzzles in a reasonable amount of time. The user interface should be intuitive and visually appealing while also providing clear feedback and visualizations of the solving process. Balancing these conflicting requirements requires careful consideration, such as finding the right balance between generating complex puzzles and en-suring solvability, optimizing the solving algorithm for speed without sacrificing accuracy, and designing a visually appealing and informative interface. Striking the right balance is crucial to create a Sudoku Solving project that offers enjoy-able and challenging puzzles while providing a seamless user experience.

1. **Depth of Analysis Required:** The Sudoku Solving project requires a depth ofanalysis in various areas. This includes analyzing the puzzle generation algorithm to ensure unique solutions and varying difficulty levels, assessing the correctness and efficiency of the solving algorithm, evaluating user interaction and interface usability, conducting performance analysis to optimize speed and responsiveness, analyzing error handling and validation mechanisms, assessing cross-platform compatibility, and evaluating overall usability and user experience. Thorough analysis in these areas is essential to refine and improve the project, ensuring it meets high standards of puzzle generation, solving accuracy, user interaction, performance, error handling, compatibility, and user satisfaction.
2. **Extent of Applicable Codes:** The Sudoku Solving project involves the imple-mentation of various codes. This includes code for puzzle generation to create Sudoku puzzles of different difficulty levels, code for the solving algorithm to find solutions using backtracking or other search algorithms, code for the user interface to provide an interactive grid, buttons, and visual feedback, code for validation and error handling to ensure the entered puzzles are valid and solvable, code for time delay and visualization techniques to demonstrate the solving pro-cess gradually, code for performance optimization to enhance efficiency, and code for testing to validate the implemented functionalities. The extent of applicable codes for the project spans puzzle generation, solving, user interface, validation, time delay, visualization, performance optimization, and testing, all of which con-tribute to the successful implementation and functionality of the Sudoku Solving project.

22

**Chapter 4**

**Conclusion**

**4.1** **Discussion**

The Sudoku Solving project utilizes a user-friendly interface with three buttons: "Su-doku Generate," "Solve," and "Clear." The "Sudoku Generate" button generates a unique Sudoku puzzle, while the "Solve" button applies an algorithm to solve the puzzle step-by-step, visually displaying the progress. The "Clear" button resets the grid. Overall, the project offers an enjoyable Sudoku-solving experience with puzzle generation, solv-ing algorithm implementation, and user interaction.

**4.2** **Limitations**

limitations include:

* **Complexity of Puzzle Generation:** Generating Sudoku puzzles with unique so-lutions can be a computationally intensive task, especially when aiming to provide puzzles of varying difficulty levels. As a result, the puzzle generation process may take longer for more challenging puzzles, which can impact user experience.
* **Inability to Solve Invalid Puzzles:** The solving algorithm assumes that the in-put puzzle is a valid Sudoku configuration. If an invalid or unsolvable puzzle is provided, the algorithm may run indefinitely or produce incorrect results. It is important to ensure that the project includes appropriate error handling and validation mechanisms to prevent such scenarios.
* **Lack of Advanced Solving Techniques:** The The current implementation of thesolving algorithm relies solely on backtracking. While backtracking is an effec-tive method for solving Sudoku puzzles, it may not utilize more advanced solving techniques like constraint propagation or advanced heuristics. Incorporating ad-ditional solving techniques could enhance the algorithm’s efficiency and provide more optimized solutions.

23

**4.3** **Scope of Future Work**

Scope of Future Work for the Sudoku Solving Project:

* **Advanced Solving Techniques:** The project can be expanded to incorporatemore advanced solving techniques, such as constraint propagation, hidden sin-gles, naked pairs, and other advanced strategies used in Sudoku solving. Imple-menting these techniques would enhance the solving algorithm’s efficiency and provide faster and more optimized solutions.
* **Puzzle Validation and Error Detection:** Enhancing the project to include puzzlevalidation and error detection mechanisms would help identify invalid or unsolv-able puzzles at the input stage. This would prevent the solving algorithm from running indefinitely on incorrect puzzles and provide users with immediate feed-back on the validity of their inputs.
* **Mobile and Cross-Platform Compatibility:** Adapting the project for mobiledevices and ensuring cross-platform compatibility would expand its reach and make it accessible to a wider audience. This could involve developing a respon-sive user interface or creating dedicated mobile applications for iOS and Android platforms.
* **Additional Grid Sizes and Variations:** Extending the project to support dif-ferent grid sizes and variations of Sudoku, such as 4x4, 16x16, or irregularly shaped grids, would provide users with more puzzle options and challenges. Im-plementing additional grid sizes would require modifying the solving algorithm and adapting the user interface to accommodate the changes.
* **Multiplayer and Online Challenges:** Implementing multiplayer functionalitywould enable users to compete against each other in solving Sudoku puzzles. This could involve real-time puzzle synchronization, leaderboards, and timed challenges, adding a social and competitive aspect to the project.

24

**4.4** **References**

Please Visit the Following Link For More Information:

* https://youtu.be/tRj4VlVTat8
* https://youtu.be/qH9mWpYMtYU

25