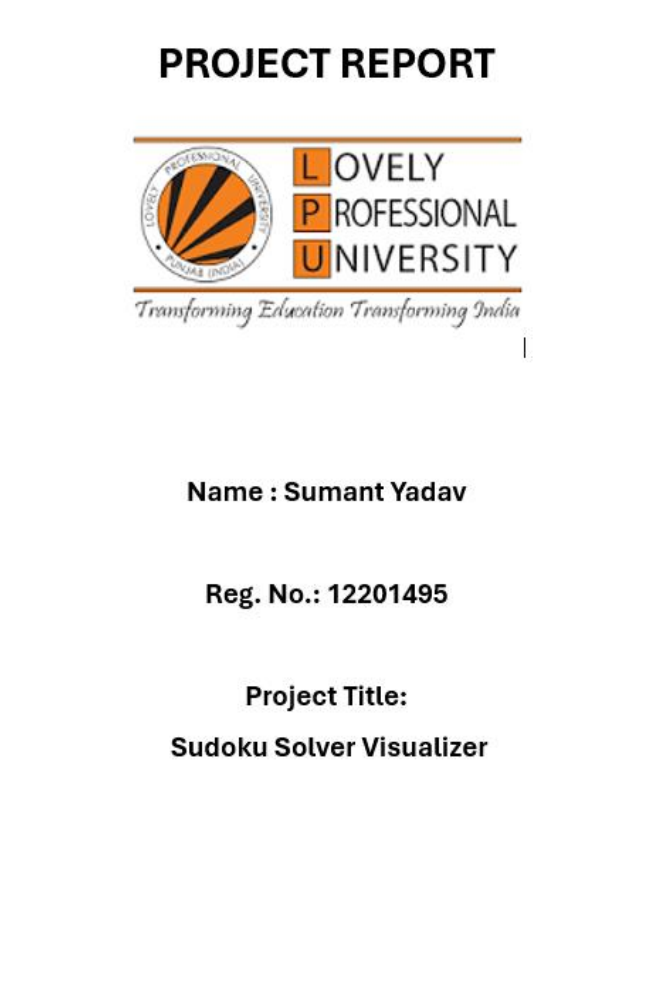
Sudoku Solver Visualizer

The Sudoku Solver Visualizer is an interactive Java application designed to solve Sudoku puzzles while providing a real-time visual representation of the solving process. This project aims to demonstrate the backtracking algorithm, showcase Java Swing's capabilities for GUI development, and offer insights into Sudoku solving techniques.



PROJECT REPORT

LOVELY
PROFESSIONAL
UNIVERSITY

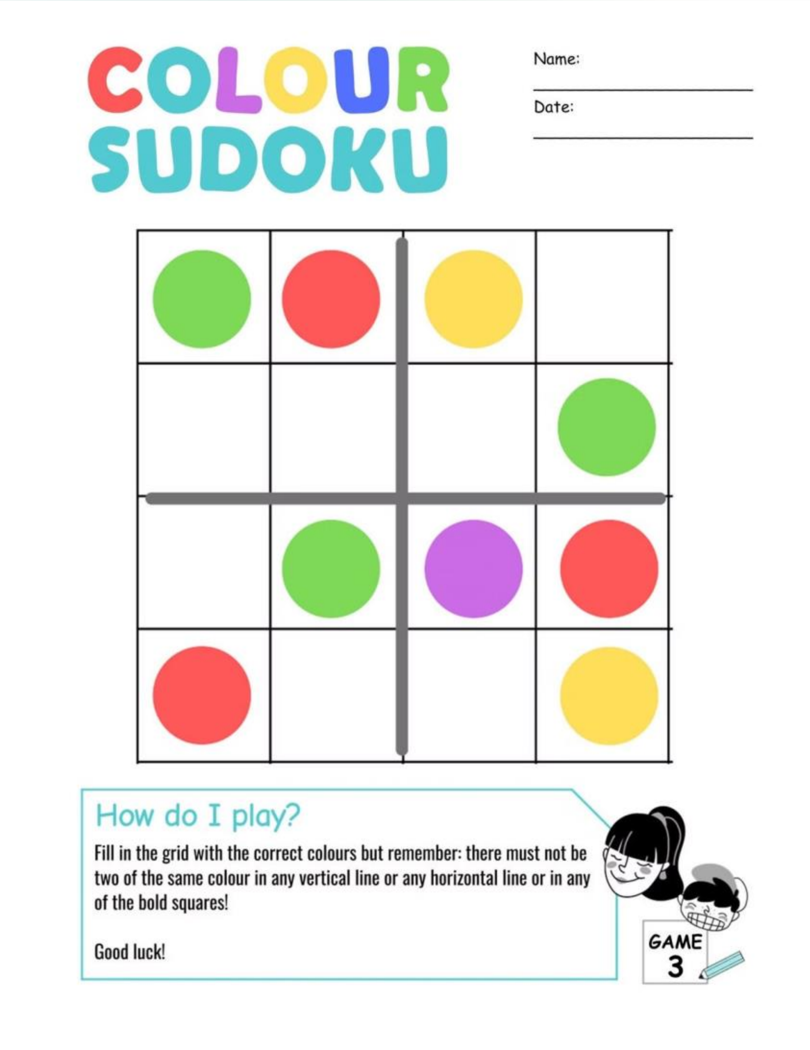
Transforming Education Transforming India

Name : Sumant Yadav

Reg. No.: 12201495

Project Title:
Sudoku Solver Visualizer

Project Objectives



COLOUR SUDOKU

Name: ____________

Date: ____________

🟢	🔴	🟡	
			🟢
	🟢	🟣	🔴
🔴			🟡

How do I play?

Fill in the grid with the correct colours but remember: there must not be two of the same colour in any vertical line or any horizontal line or in any of the bold squares!

Good luck!

1 **Algorithmic Demonstration**

Visualizes the backtracking algorithm to provide an intuitive understanding of its workings.

2 **Interactive Visualization**

Users can watch the solving process with color-coded feedback for placed numbers (cyan) and backtracked attempts (red).

3 **Java Swing Utilization**

Demonstrates creating a grid-based layout, using JLabels, and updating GUI components in real-time.

4 **Problem-Solving Insight**

Offers insight into logical steps required to solve complex Sudoku puzzles.

Key Features

Interactive GUI

Built with Java Swing, ensuring platform independence. Uses GridLayout for a 9x9 grid, with JLabels representing each cell.

Real-Time Visualization

Users can observe the solving process, including number placements and backtracking. Color-coding: Cyan for successfully placed numbers, Red for backtracked cells, Light Gray for initial state and final solution.

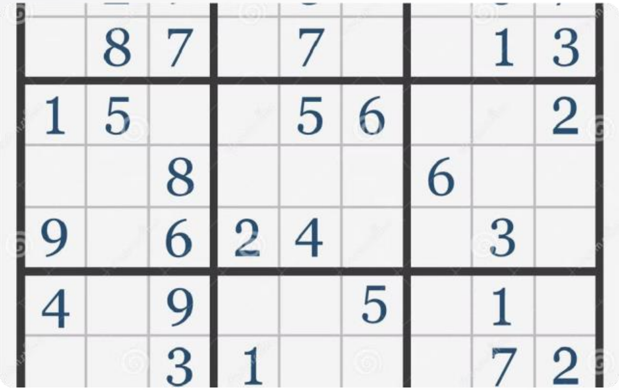
Backtracking Algorithm

Efficient backtracking algorithm explores potential solutions and backtracks when encountering invalid states. Offers a clear view of the decision-making process during puzzle solving.

Technical Implementation

Programming Language and Framework	Java, Java Swing
Core Components	Sudoku Solver Class, GUI Elements
Algorithm Details	isSafe Method, findSolution Method, solveSudoku Method
Visualization	Color Coding, Real-Time Updates
Data Structures	2D Arrays, Predefined Puzzles

User Interface

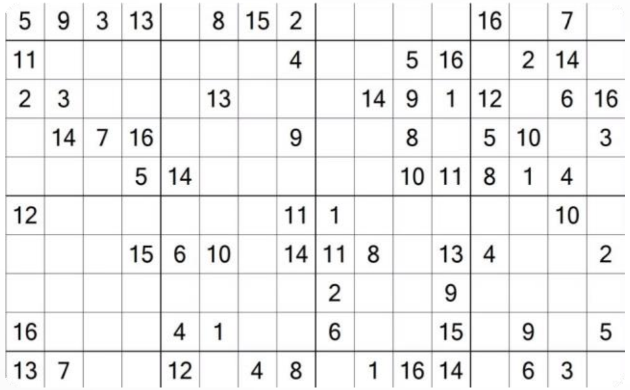


Main Window

JFrame Configuration: 500x500 pixels, GridLayout(9, 9), and JFrame.EXIT_ON_CLOSE. Set to visible after all components are added.

Sudoku Grid

Each cell represented by a JLabel. Custom borders to visually separate 3x3 subgrids.



Dynamic Updates

Real-Time Visualization: Updates for each cell change. Backtracking Visualization: Red cells indicate backtracked attempts.

Performance Considerations

1

Algorithm Efficiency

Efficient for most Sudoku puzzles with pruning techniques and early termination.

2

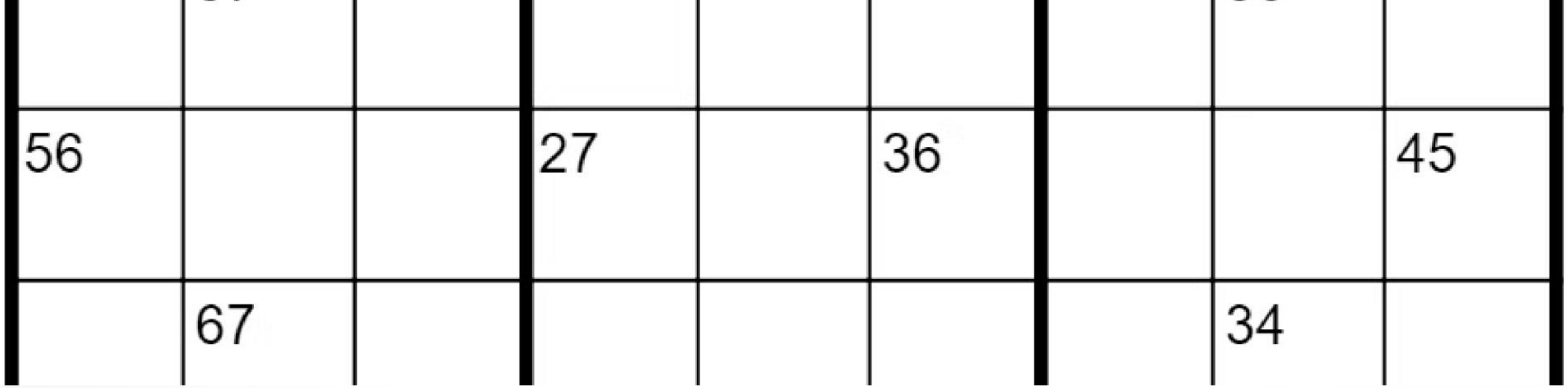
GUI Update Frequency

Potential performance issues for very fast solves.

3

Scalability

Currently handles only 9x9 grids. Static nature limits solving multiple puzzles simultaneously.



Future Enhancements

User Input for Custom Puzzles

Input Mechanism: Text input field, clickable grid, or file upload. Input Validation: Ensure valid Sudoku grids.

Difficulty Levels

Puzzle Generation: Generate puzzles of varying difficulties. Difficulty Rating: Rate and display puzzle difficulties.

Adjustable Visualization Speed

Speed Control: Slider or buttons for adjusting visualization speed. Pause and Resume: Functionality to control the solving process.

Step-by-Step Explanation

Textual Descriptions: Provide written explanations for each step. Explanation Panel: Display explanations alongside the visual grid.

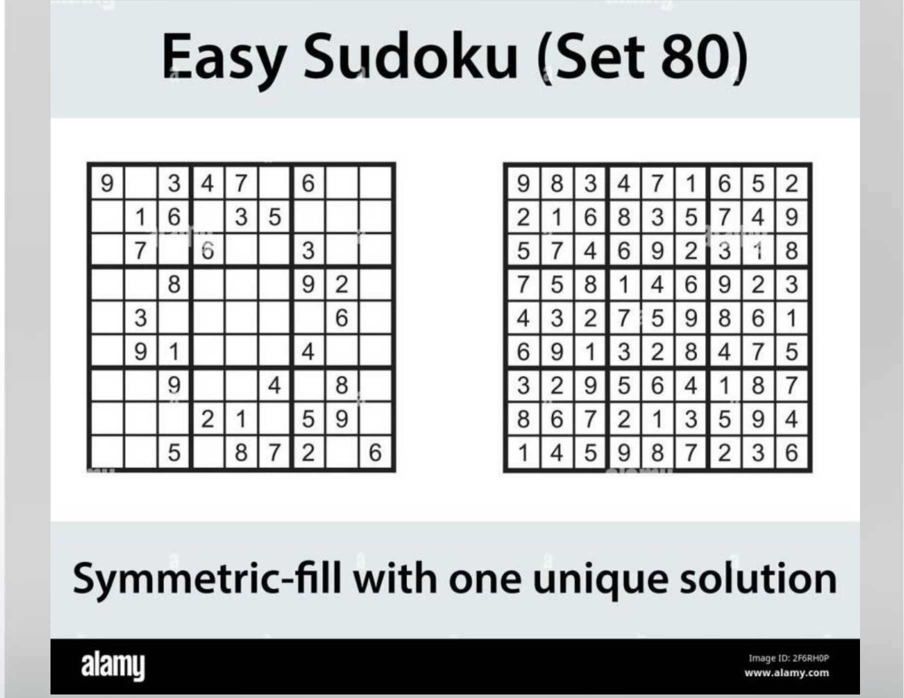
Conclusion

Achievement of Project Goals

Solving Capability: Efficiently solves standard 9x9 Sudoku puzzles. Visualization Effectiveness: Provides a clear, real-time view of the solving process. Educational Value: Serves as a powerful tool for demonstrating algorithmic concepts.

Future Potential

Educational Platform: Evolve into a comprehensive tool for teaching algorithms. Sudoku Training Tool: Enhance with user-solving capabilities and hints. Algorithm Comparison Framework: Compare different solving algorithms visually.



Easy Sudoku (Set 80)

9		3	4	7		6		
	1	6		3	5			
	7		6			3		
		8				9	2	
	3						6	
	9	1				4		
		9			4		8	
			2	1		5	9	
		5		8	7	2		6

9	8	3	4	7	1	6	5	2
2	1	6	8	3	5	7	4	9
5	7	4	6	9	2	3	1	8
7	5	8	1	4	6	9	2	3
4	3	2	7	5	9	8	6	1
6	9	1	3	2	8	4	7	5
3	2	9	5	6	4	1	8	7
8	6	7	2	1	3	5	9	4
1	4	5	9	8	7	2	3	6

Symmetric-fill with one unique solution