

# **SUDOKU SOLVER**

An Industry Oriented Project Report Submitted  
In partial fulfillment of the requirements for the award of the degree of

## **Bachelor of Technology in Computer Science and Information Technology**

by

<b>N. Chinna</b>	<b>23N31A3336</b>
<b>Mokshitha mahali</b>	<b>23N31A3363</b>
<b>A.Sukumar</b>	<b>23N31A3303</b>

Under the esteemed guidance of

**T.Shilpa**

Assistant Professor



**Department of Information Technology**

**Malla Reddy College of Engineering &  
Technology**

(Autonomous Institution- UGC, Govt. of India)

(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA & NAAC with  
'A' Grade)

Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100

website: [www.mrcet.ac.in](http://www.mrcet.ac.in)



# **Malla Reddy College of Engineering & Technology**

(Autonomous Institution- UGC, Govt. of India)

(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA & NAAC with 'A' Grade)

Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100

website: [www.mrcet.ac.in](http://www.mrcet.ac.in)

## **CERTIFICATE**

This is to certify that this is the bonafide record of the project entitled “SUDOKU SOLVER”, submitted by N.Chinna (23N31A3336), Mokshitha mahali (23N31A3363) and A.Sukumar (23N31A3303) of B.Tech in the partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science and Information Technology, Department of CSIT during the year 2024- 2025.

**Internal Guide**

**T.Shilpa**

**Assistant Professor**

**Head of the Department**

**Dr. G. Sharada**

**Professor**

**External Examiner**

## Table of Contents

<b><u>S.No</u></b>	<b><u>TITLE</u></b>	<b><u>PG.NO</u></b>
<b>1</b>	<b>INTRODUCTION</b>	
	1.1 DEFINITION AND SCOPE	01
	1.2 EXISTING AND PROPOSED SYSTEM	02
<b>2</b>	<b>SYSTEM ANALYSIS</b>	
	2.1 HARDWARE AND SOFTWARE REQUIREMENTS	04
<b>3</b>	<b>SYSTEM DESIGN</b>	
	3.1 ARCHITECTURE AND DIAGRAMS	05
<b>4</b>	<b>IMPLEMENTATION</b>	
	4.1 SOURCE CODE AND OUTPUT SCREENS	08
<b>5</b>	<b>CONCLUSION</b>	19
	<b>BIBLIOGRAPHY</b>	20

## **ABSTRACT**

The Sudoku Solver application is an innovative tool that leverages advanced computational algorithms to solve Sudoku puzzles of any complexity. Using methods like backtracking, constraint satisfaction, and heuristic search techniques, the app can efficiently process puzzles and generate solutions in a fraction of the time compared to manual solving. The application supports multiple input methods, allowing users to enter puzzles through a grid interface, upload pre-filled puzzles, or scan images directly. It also provides an option for users to see the step-by-step process, enabling both learning and entertainment.

Additionally, the app includes features such as puzzle generation, hint suggestions, and validation of user solutions, making it an all-in-one sudoku tool. The user-friendly interface allows for seamless navigation, providing adjustable difficulty levels and various puzzle formats to suit different preferences. The Sudoku Solver serves as an educational resource for beginners, a practice tool for enthusiasts, and a quick solution provide for advanced players, ensuring accessibility and efficiency for a

# **INTRODUCTION**

## **1.1 DEFINITION AND SCOPE**

### **Definition:**

The **Sudoku Solver Project** is a program designed to solve Sudoku puzzles of varying difficulty levels using algorithmic techniques, particularly backtracking. The primary goal of the project is to automate the process of solving a valid Sudoku puzzle, which follows the fundamental rule that each row, column, and 3x3 sub grid must contain unique numbers from 1 to 9. The project utilizes the backtracking algorithm, which attempts to fill in the puzzle by trying different number placements, and backtracks when an invalid placement is encountered, ensuring a correct solution is found.

The project is structured into several key components, including the main algorithm responsible for solving the puzzle, input handling for accepting Sudoku grids in different formats, and output formatting to display the solution in an easily readable format. If the project includes a user interface, it would provide a simple means for users to input their puzzles and view the results. The code also includes a test suite to verify the solver's accuracy and ensure the program functions as expected, even in edge cases.

### **Scope:**

The **scope of the Sudoku Solver Project** involves developing a software tool that efficiently solves Sudoku puzzles of varying complexities using algorithmic techniques, primarily focusing on the backtracking algorithm. The project begins with input handling, where the software will support various methods for entering Sudoku puzzles, such as text-based grids, matrices, or even graphical inputs via a user interface, if included. The core of the project revolves around implementing the backtracking algorithm, which recursively fills in the grid while adhering to Sudoku rules, ensuring that each row, column, and 3x3 sub grid contains unique numbers from 1 to 9. In addition to solving the puzzle, the program will include a validation step to ensure the input grid adheres to Sudoku rules and alerts the user to any invalid placements.

Once the puzzle is solved, the solution will be presented in a clear and readable format,

whether through text output, a graphical grid, or an interactive interface. The project will also include a testing framework to verify the solver's functionality across a range of puzzle difficulties and edge cases. If the project features a user interface, it will allow users to input their puzzles by typing or clicking directly into a grid, providing an intuitive and accessible experience. Furthermore, the solver will be optimized for performance, ensuring fast solution times even for more complex puzzles, with potential future optimizations such as constraint propagation to enhance solving efficiency. Overall, the scope of the project aims to create a robust, user-friendly Sudoku solver that can handle a variety of input formats, solve puzzles efficiently, and be easily expanded in the future.

## 1.2 EXISTING AND PROPOSED SYSTEM

### EXISTING SYSTEM :

Existing systems for Sudoku solvers typically rely on various algorithms and approaches to automate the solving process. The most common method is the **backtracking algorithm**, a depth-first search approach that fills the grid by placing numbers and backtracks when it encounters a conflict. Many systems enhance this with **constraint propagation**, which reduces the search space by eliminating invalid options early, making the solving process more efficient. Some advanced systems use the **exact cover problem** and **Dancing Links** technique, offering faster solutions by solving Sudoku as a mathematical problem. Additionally, some systems employ **genetic algorithms** or **heuristic methods** like **simulated annealing** to solve more complex puzzles, although these are not guaranteed to find the exact solution. Many solvers are also equipped with **graphical user interfaces (GUIs)**, allowing users to input puzzles and view the solving process dynamically. Web-based Sudoku solvers, which are widely available, also use backtracking and constraint propagation methods and often provide features like puzzle validation and hints. While these existing systems are effective for standard puzzles, they may struggle with more complex or computationally intensive instances. The **Sudoku Solver Project** can build on these existing systems by improving performance, offering optimization techniques, and providing a more intuitive user interface.

## PROPOSED SYSTEM :

The **proposed system** for the Sudoku Solver Project aims to build upon existing solvers by incorporating additional features and optimizations for better performance and user experience. The core algorithm will still utilize **backtracking** as the primary solving technique, but with **constraint propagation** enhancements to reduce the search space, improving efficiency and speed. The system will also implement more advanced techniques such as **constraint satisfaction algorithms** and **Dancing Links** for solving harder puzzles more effectively.

A key feature of the proposed system will be a **graphical user interface (GUI)** that provides a more interactive experience for users. The interface will allow users to input puzzles manually or through file uploads, with a clear visual representation of the puzzle. The system will offer functionality for users to view the step-by-step solving process, providing an educational tool to understand the algorithm. Users can also receive hints or check for errors in their input.

In addition to solving standard 9x9 grids, the system will have an option to support **larger Sudoku grids**, like 16x16 or 25x25, addressing the need for more flexibility. The system will also include a **difficulty classification** feature, allowing users to select puzzles based on their complexity, with the solver automatically adjusting its solving strategy. For performance optimization, the solver will be designed to handle large puzzles efficiently, solving even the most difficult grids within a reasonable time frame.

Furthermore, the proposed system will include robust **error handling** and **validation** features, ensuring that the input grid is valid and notifying users of any inconsistencies. It will also have the ability to generate puzzles with varying difficulty levels, giving users the opportunity to challenge themselves with custom puzzles.

Overall, the proposed system aims to offer a more **user-friendly, efficient, and feature-rich** solution compared to existing Sudoku solvers, focusing on both functionality and an enhanced user experience.

# **SYSTEM ANALYSIS**

## **2.1 SOFTWARE AND HARDWARE REQUIREMENTS**

### **SOFTWARE REQUIREMENTS:**

- **Frontend:** HTML, CSS, JavaScript (for user interface)
- **Backend:** Node.js, java (for server side logic)
- **Database:** MySQL (for database management)
- **Programming Language:** Java
- **Operating System:** Windows, macOS, or Linux (depending on the development environment)
- **IDE (Integrated Development Environment):** Visual Studio Code

### **HARDWARE REQUIREMENTS:**

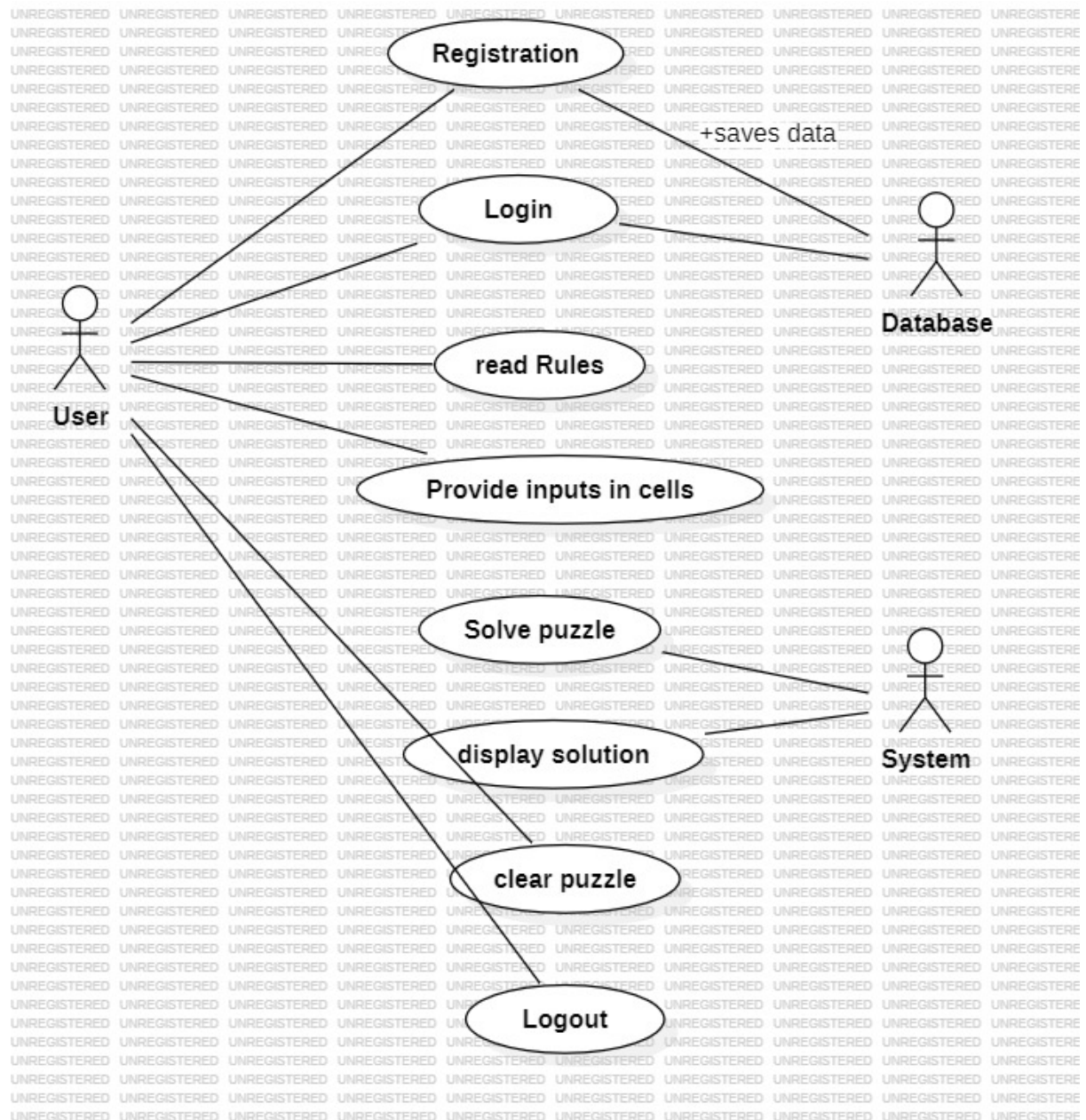
- **RAM :** 3GB RAM is sufficient
- **Processor:** Any Modern CPU with atleast 1 Core
- **Storage:** 500 MB is sufficient



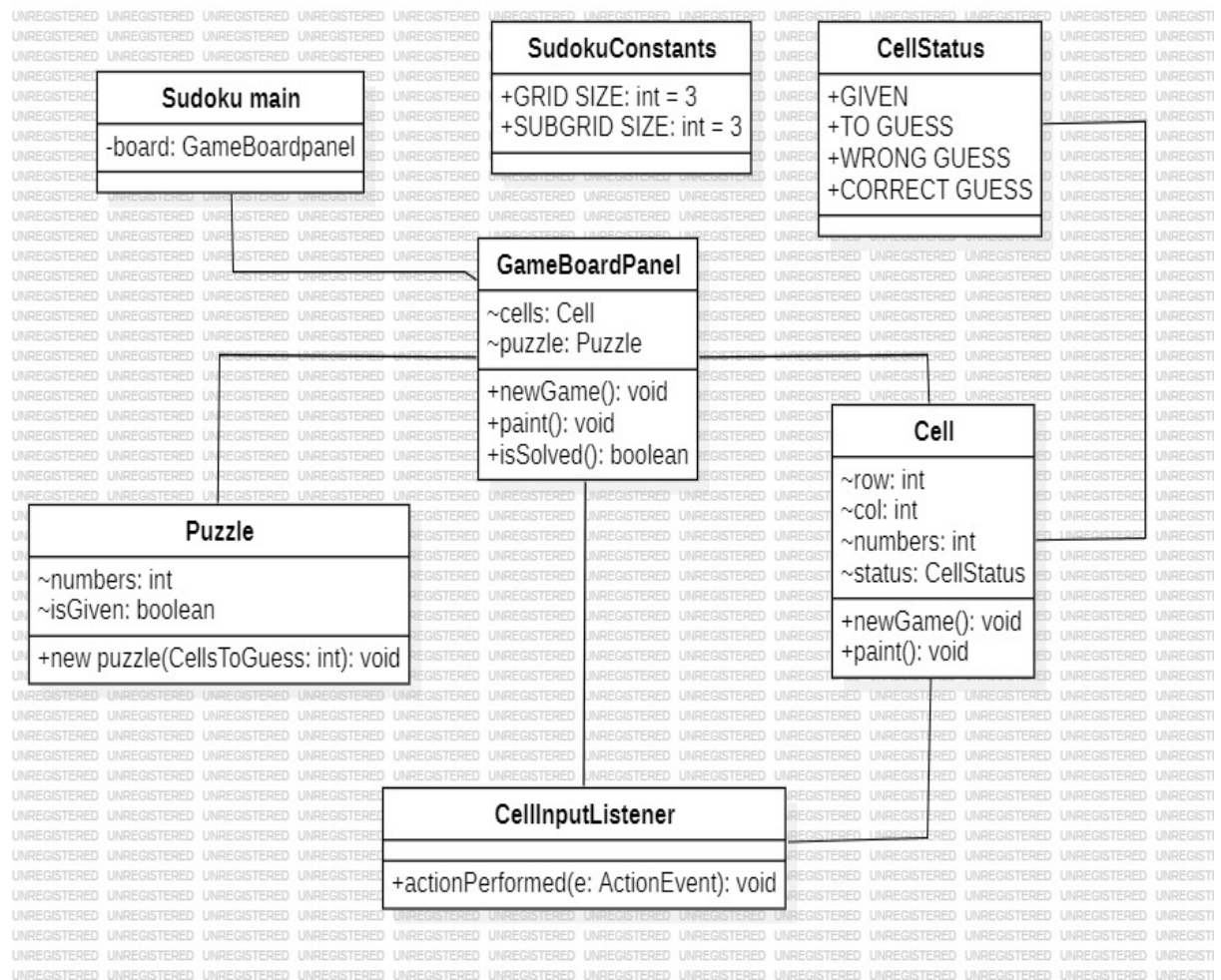
# SYSTEM DESIGN

## 3.1 ARCHITECTURE AND DIAGRAMS:

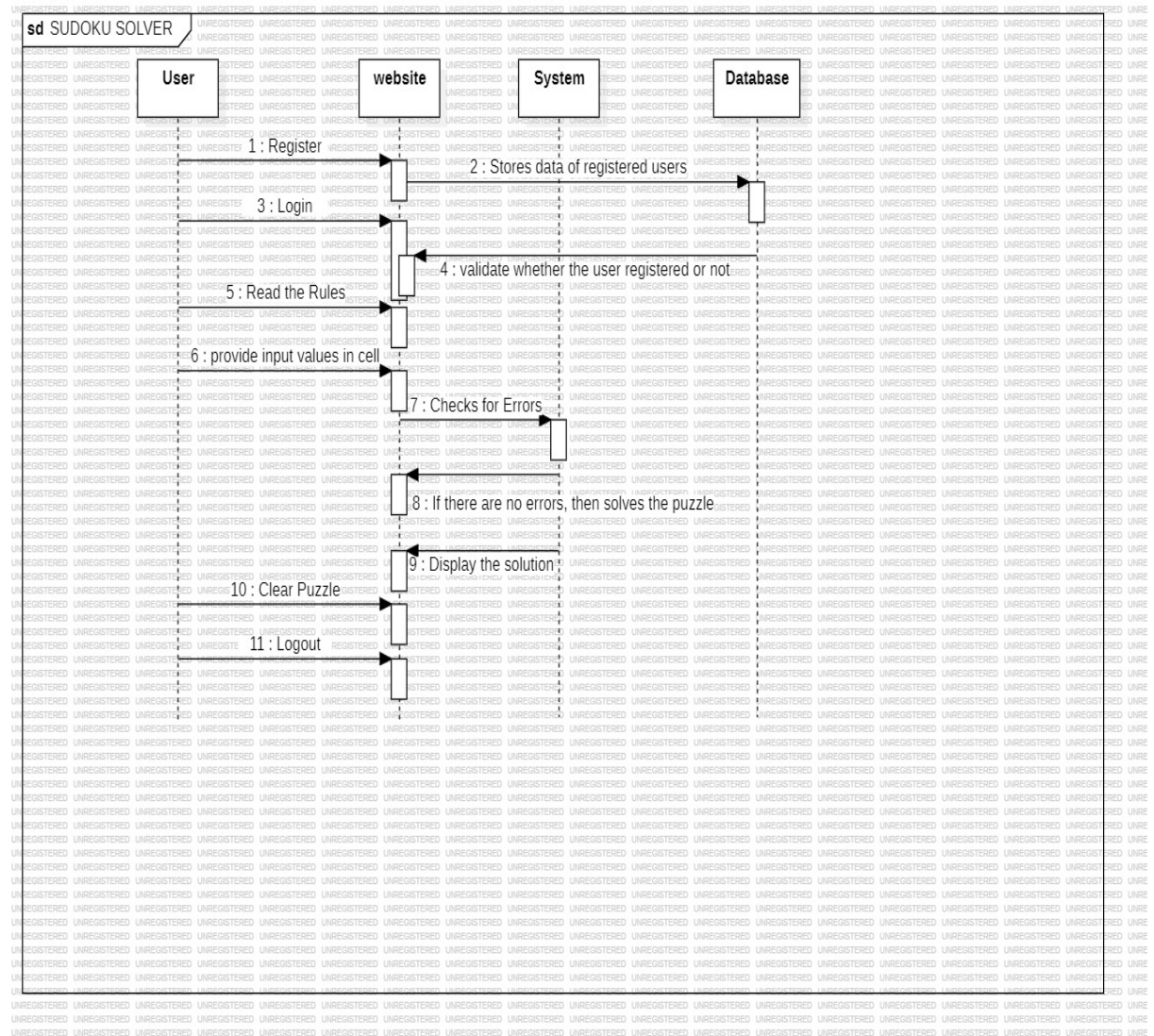
### USECASE DIAGRAM:



## CLASS DIAGRAM:



# SEQUENCE DIAGRAM:



# **IMPLEMENTATION**

## **4.1 SOURCE CODE AND OUTPUT SCREENS**

### **SOURCE CODE:**

#### **GAME DASHBOARD PAGE**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Sudoku Solver</title>
  <link rel="stylesheet" href="stylesss.css">
  <style>
    .input-cell {
      background-color: #4baaca;
    }
    .output-cell {
      background-color: #43df43;
    }

    .error-cell {
      background-color: red; /* Red background to highlight the duplicate cells */
      color: white; /* White text for better visibility */
      font-weight: bold; /* Bold text to make the error more noticeable */
      border: 2px solid yellow; /* Optional: Border to make the cell stand out */
    }

  </style>
```

&lt;/head&gt;

&lt;body&gt;

<div id="container">

# Sudoku Solver

&lt;table id="sudoku-board"&gt;

<colgroup><col><col><col></colgroup>
--------------------------------------

<colgroup><col><col><col></colgroup>
--------------------------------------

<colgroup><col><col><col></colgroup>
--------------------------------------

[illegible][illegible][illegible][illegible][illegible]

contenteditable="true"></td> <td contenteditable="true"></td> <td  
contenteditable="true"></td>

<tr> <td contenteditable="true"></td> <td contenteditable="true"></td> <td  
contenteditable="true"></td> <td contenteditable="true"></td> <td  
contenteditable="true"></td> <td contenteditable="true"></td> <td  
contenteditable="true"></td> <td contenteditable="true"></td> <td  
contenteditable="true"></td>

<tbody>

<tr> <td contenteditable="true"></td> <td contenteditable="true"></td> <td  
contenteditable="true"></td> <td contenteditable="true"></td> <td  
contenteditable="true"></td> <td contenteditable="true"></td> <td  
contenteditable="true"></td> <td contenteditable="true"></td> <td  
contenteditable="true"></td>

<tr> <td contenteditable="true"></td> <td contenteditable="true"></td> <td  
contenteditable="true"></td> <td contenteditable="true"></td> <td  
contenteditable="true"></td> <td contenteditable="true"></td> <td  
contenteditable="true"></td> <td contenteditable="true"></td> <td  
contenteditable="true"></td>

<tr> <td contenteditable="true"></td> <td contenteditable="true"></td> <td  
contenteditable="true"></td> <td contenteditable="true"></td> <td  
contenteditable="true"></td> <td contenteditable="true"></td> <td  
contenteditable="true"></td> <td contenteditable="true"></td> <td  
contenteditable="true"></td>

</table>

<div class="btns">

<div>

<button id="solve-button">Solve</button>

</div>

<div>

<button id="clear-button">Clear board</button>

</div>

</div>

</div>

```
<script src="script.js"></script>
</body>
</html>
```

## **JAVASCRIPT**

```
document
.getElementById("sudoku-board")
.addEventListener("keyup", function (event) {
  if (event.target && event.target.nodeName == "TD") {
    var validNum = /[1-9]/;
    var tdEl = event.target;
    if (tdEl.innerText.length > 0 && validNum.test(tdEl.innerText[0])) {
      tdEl.innerText = tdEl.innerText[0];

      tdEl.classList.add("input-cell");
      tdEl.classList.remove("output-cell");
    } else {
      tdEl.innerText = "";

      tdEl.classList.remove("input-cell", "output-cell");
    }
    highlightErrors();
  }
});
```

```
document
.getElementById("solve-button")
.addEventListener("click", function (event) {
  var boardString = boardToString();
  console.log("Board String for solving: ", boardString);

  if (boardString === "-".repeat(boardString.length))
  {
```

```

    alert("Please fill the input numbers in the sudoku board before solving!");
    return;
}
var solution = SudokuSolver.solve(boardString);
if (solution) {
    stringToBoard(solution);
} else {
    alert("Invalid Board! please check there shouldn't be same number in (same row/same
col/in nonet)");
}
highlightErrors();
});

```

```

function highlightErrors() {
    var tds = document.getElementsByTagName("td");
    for (var i = 0; i < tds.length; i++) {
        tds[i].classList.remove("error-cell");
        tds[i].removeAttribute("title"); // Remove previous error highlighting
    }
}

```

```

var boardArray = boardToString().split("");

```

```

// Check for duplicates in rows, columns, and nonets
highlightDuplicates(boardArray);
}
document.getElementById("clear-button").addEventListener("click", clearBoard);

```

```

function clearBoard() {
    var tds = document.getElementsByTagName("td");
    for (var i = 0; i < tds.length; i++) {
        tds[i].innerText = "";
        tds[i].classList.remove("input-cell", "output-cell");
        tds[i].classList.remove("error-cell");
    }
}

```



```
}
```

```
function boardToString() {  
    var string = "";  
    var validNum = /[1-9]/;  
    var tds = document.getElementsByTagName("td");  
    for (var i = 0; i < tds.length; i++) {  
        if (validNum.test(tds[i].innerText[0])) {  
            string += tds[i].innerText[0];  
        } else {  
            string += "-";  
        }  
    }  
    console.log("Board String: ", string);  
    return string;  
}
```

```
function stringToBoard(string) {  
    var currentCell;  
    var validNum = /[1-9]/;  
    var cells = string.split("");  
    var tds = document.getElementsByTagName("td");  
    for (var i = 0; i < tds.length; i++) {  
        currentCell = cells.shift();  
        if (validNum.test(currentCell)) {  
            tds[i].innerText = currentCell;  
        }  
    }  
    highlightErrors();  
}
```

```
("use strict");
```

```

var EASY_PUZZLE =
  "1-58-2---9--764-52--4--819-19--73-6762-83-9-----61-5---76---3-43--2-5-16--3-89--";
var MEDIUM_PUZZLE =
  "-3-5--8-45-42---1---8--9---79-8-61-3-----54---5-----78-----7-2---7-46--61-3--5--";
var HARD_PUZZLE =
  "8-----36-----7--9-2---5---7-----457-----1---3---1----68--85---1--9----4--";

var TESTABLE = true;

```

```

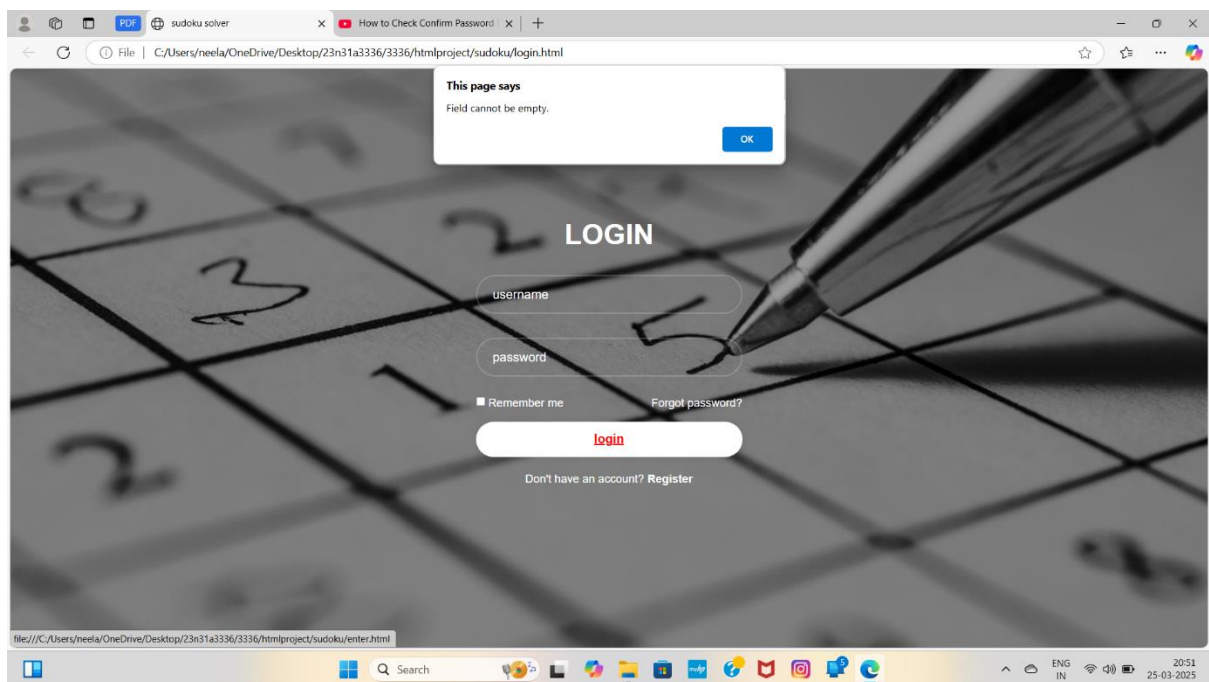
var SudokuSolver = (function (testable) {
  var solver;

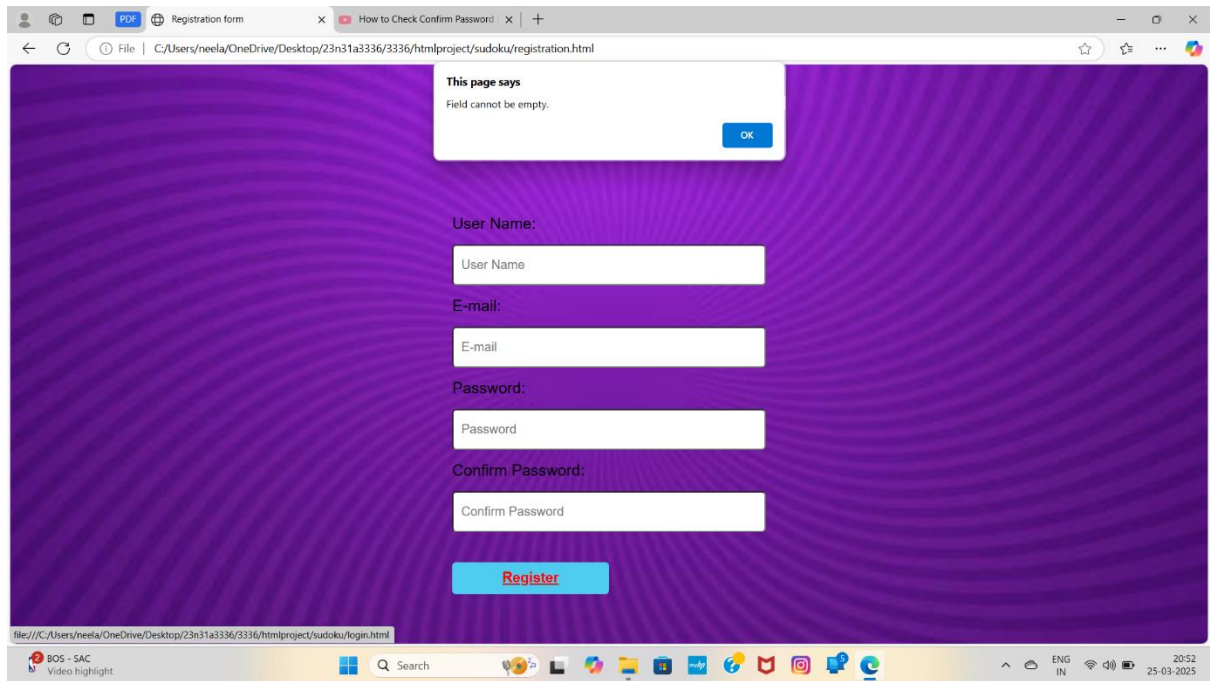
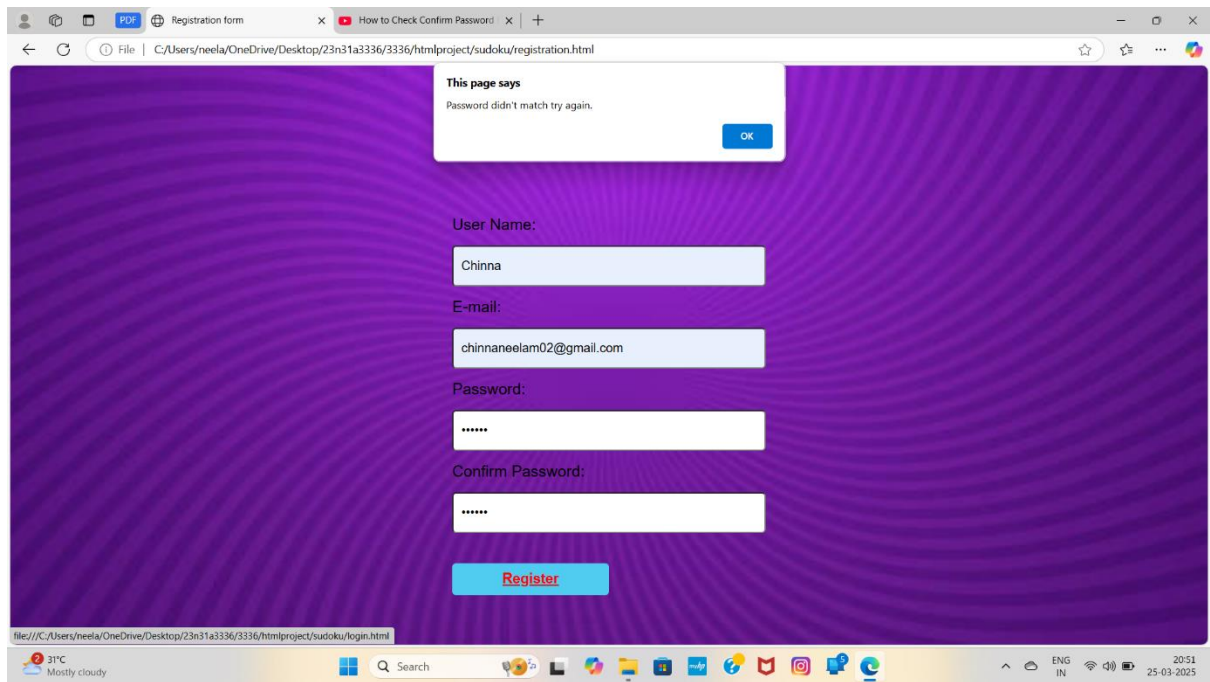
  function solve(boardString) {
    var boardArray = boardString.split("");

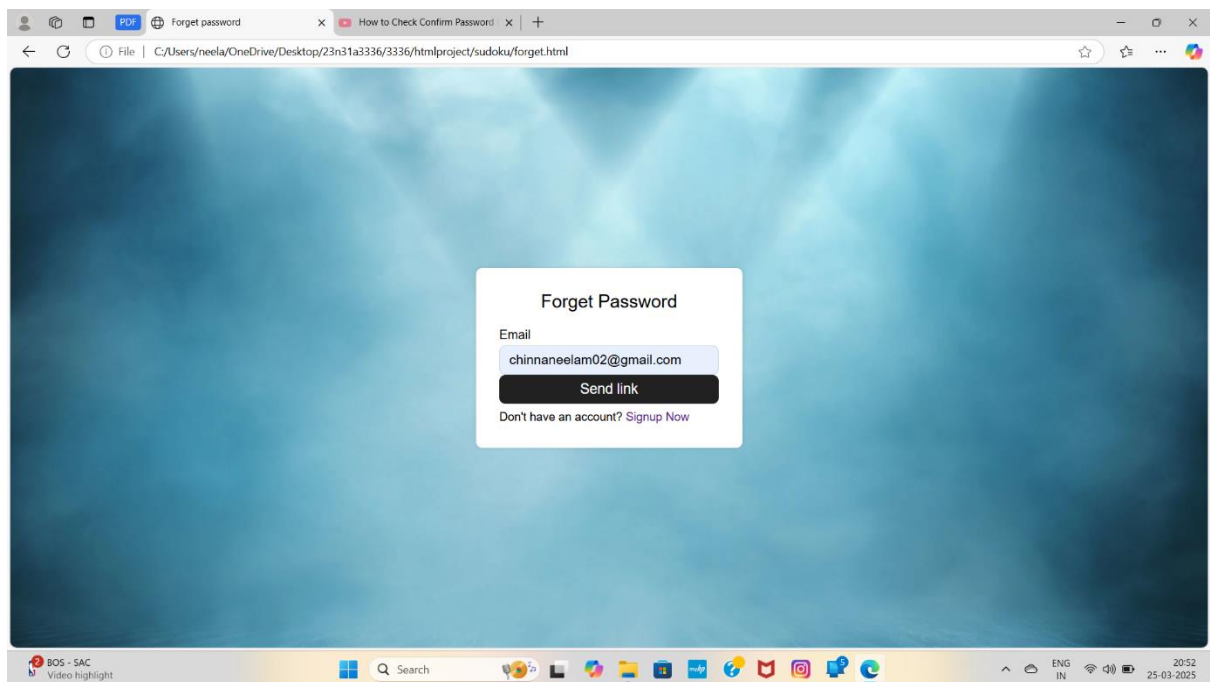
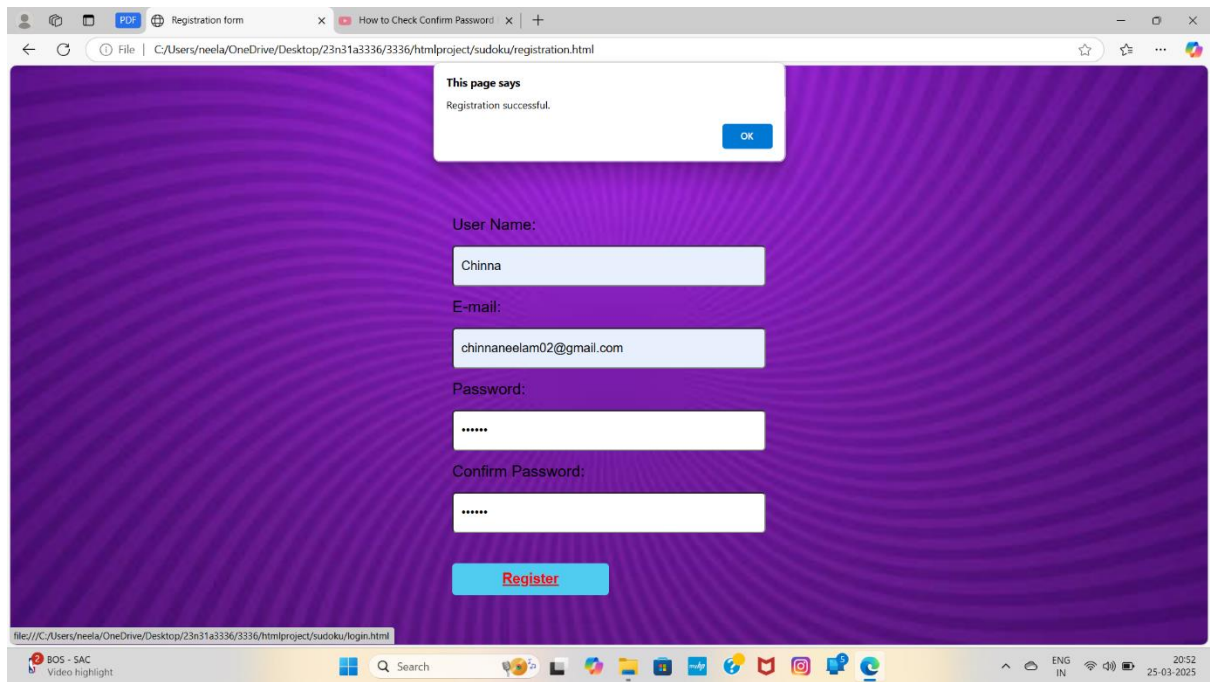
    console.log("Board Array before validation: ",boardArray);
  }

```

## OUTPUT SCREENS:





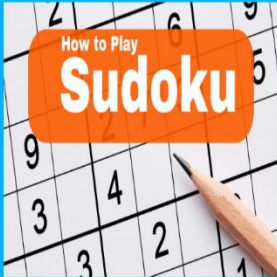


Rules

How to Check Confirm Password

C:/Users/neela/OneDrive/Desktop/23n31a3336/3336/htmlproject/sudoku/enter.html

# RULES OF SUDOKU SOLVER



- Every Row must contain numbers from 1 to 9 without repeating.
- Every Column must contain numbers from 1 to 9 without repeating.
- Every block(3x3) must contain numbers from 1 to 9 without repeating.
- Use the numbers given at the start of the game as clues.
- The sum of every single row,column and nonet must equal to 45.

next

BOS - SAC  
Video highlight

Search

ENG IN 20:53 25-03-2025

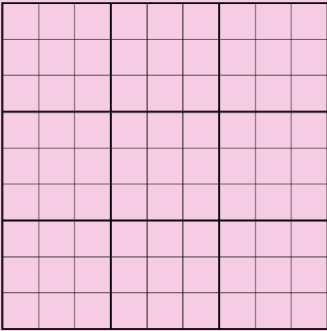
Sudoku Solver

How to Check Confirm Password

C:/Users/neela/OneDrive/Desktop/23n31a3336/3336/htmlproject/sudoku/index.html

This page says  
Please fill the input numbers in the sudoku board before solving!

OK

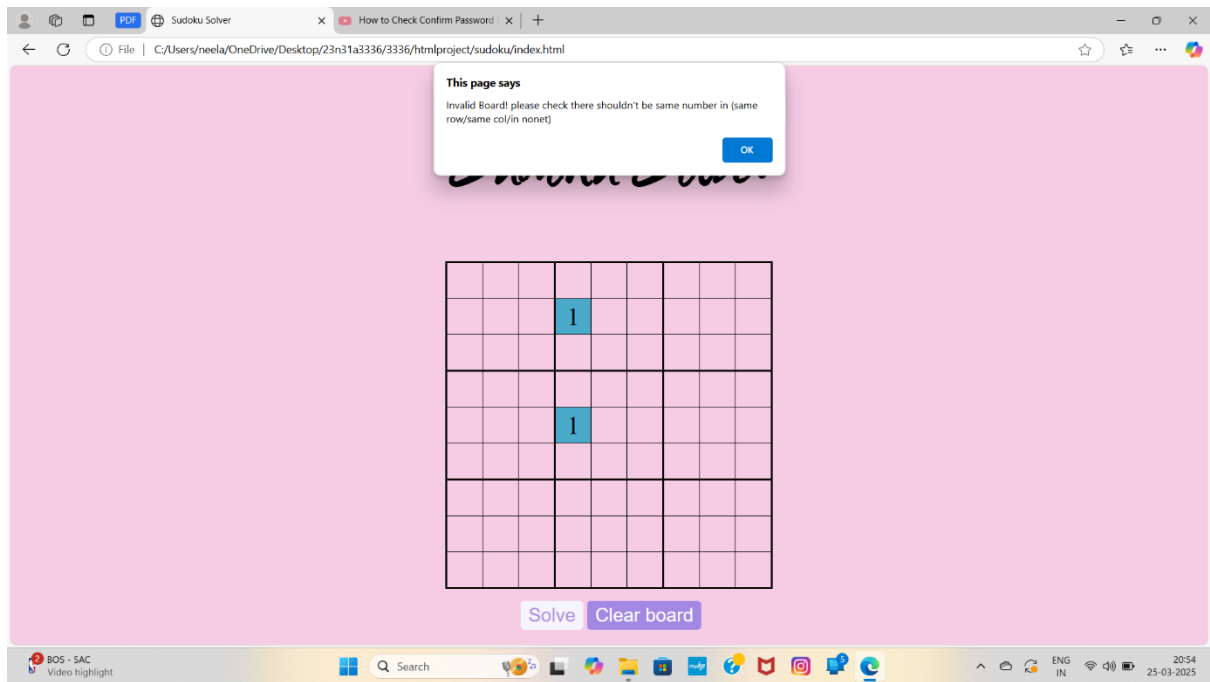


Solve Clear board

BOS - SAC  
Video highlight

Search

ENG IN 20:53 25-03-2025



## CONCLUSION

The **Sudoku Solver** project successfully integrates algorithmic problem-solving with a user-friendly interface, providing an efficient platform for solving Sudoku puzzles. By utilizing algorithms like **backtracking**, **constraint propagation**, or **brute force**, the system automates the solving process, offering a faster and more reliable solution than manual methods. The project is designed with a focus on enhancing user experience, featuring **login**, **registration**, and **forgot password** functions for secure user access, along with an intuitive **game dashboard** where users can input puzzles, solve them, and reset the grid.

Additionally, the project serves as an educational tool, helping users understand the underlying algorithms involved in solving Sudoku puzzles. It offers scalability, allowing for future features such as different difficulty levels, custom puzzle creation, or more advanced solving techniques. The system can be further expanded to include AI/ML-based approaches or even multiplayer modes, allowing users to engage with the puzzle-solving process in new ways.

In conclusion, this project demonstrates the powerful combination of algorithmic efficiency and user interface design, providing both a fun and educational experience. It successfully solves real-world puzzles while offering opportunities for future development and innovation, making it a valuable tool for both puzzle enthusiasts and those interested in exploring problem-solving techniques.

## **BIBLIOGRAPHY**

➤ **PROJECT REFERRED LINK:**

<https://www.codewithfaraz.com/content/269/sudoku-solver-with-html-css-and-javascript>

The Existing Project consists in the above link.

➤ **CODE LINK:**

[https://1drv.ms/f/c/eb809b85b4263c82/EuVr3rq\\_IZ5PqCCK7EM2JD4B\\_ODwB3IsgLXTvqABwgniEQ?e=bMepGG](https://1drv.ms/f/c/eb809b85b4263c82/EuVr3rq_IZ5PqCCK7EM2JD4B_ODwB3IsgLXTvqABwgniEQ?e=bMepGG)

Code Execution is present in the above link.