SUDOKU SOLVER

An

Human Computer Interface Course Project Report

in partial fulfilment of the degree

Bachelor of Technology

in

Computer Science & Engineering

By

K.Harsha Vardhan	2203A51051
T.Abhinay Kumar	2203A51066
E.Sidhardha Reddy	2203A51281
B.Karthik	2203A51337

Submitted to

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the **Human Computer Interface- Course Project** Report entitled **"**SUDOKU SOLVER**"** is a record of bonafide work carried out by the student K.Harsha vardhan ,T.Abhinaykumar,E.Sidhardhareddy and B.Karthik bearing Roll No(s) 2203A51051,2203A51066,2203A51281 and 2203A51337 during the academic year 2023-24 of II-I Semester CSE section-D in partial fulfillment of the award of the degree of ***Bachelor of Technology*** in **Computer Science & Engineering** by the SR University, Anantsagar.

Course faculty **Head of the Department**

TABLE OF CONTENTS

ABSTRACT

The Sudoku Solver Web Application is a user-friendly web tool created with HTML, CSS, and JavaScript. It enables users to input Sudoku puzzles of varying difficulty levels and swiftly provides solutions through an efficient JavaScript backtracking algorithm. The application offers responsive design, real-time validation, solution highlighting, and optional timer functionality, making it an ideal resource for Sudoku enthusiasts, fostering problem-solving skills, logical thinking, and entertainment.

2.Problem Statement:

Sudoku is a popular number puzzle game that requires logical thinking and problem-solving skills. The goal of this project is to create a web-based Sudoku solver that allows users to input a partially filled Sudoku puzzle and find a solution for it.

Designing the user interface (UI) and the overall layout of your Sudoku solver project is crucial for providing a pleasant user experience. Below is a simple design for the Sudoku solver that you can use as a starting point. Feel free to customize the design to your preferences:

Design Layout:

- Header:

Place the project title or logo at the top to identify your Sudoku solver. Add any navigation or information links (e.g., "How to Play," "About," "Instructions").

- Sudoku Grid:

Display a 9x9 grid for the Sudoku puzzle. Use a table structure or a combination of div elements to create the grid. Apply alternating background colors to make it easier to distinguish rows and columns. Provide space for each cell where users can input numbers. Use a larger font size for pre-filled numbers. Apply subtle borders to separate the cells.

- Number Input:

Include a numerical keypad or input field near the grid for users to enter numbers. Ensure that it's clear which cell in the grid the number input will affect.

- Action Buttons:

Add action buttons below the grid, such as "Solve," "Reset," and "Clear." Use contrasting colors to make these buttons stand out. Provide tooltips or labels for each button.

- Validation Feedback:

Display feedback messages below the action buttons to inform users if they enter an invalid number or make an incorrect move. Use different colors or icons to indicate whether the input is valid.

- Solution Display:

Once the puzzle is solved, highlight or display the solved numbers in the grid clearly. Consider using a different background color or font color to distinguish the solved numbers.

- Optional Features:

If you implement optional features like difficulty levels, include them in a user-friendly manner. Add settings or options sections for users to configure these features.

3.Software and Hardware Requirements:

Software and hardware requirements for your Sudoku solver project will depend on the complexity of your implementation and the technologies you choose. However, I can provide a basic set of requirements to get you started. You may need to adjust these requirements based on your project's specific needs and goals.

Software Requirements:

1.Web Browser:

Users should have access to a modern web browser to run the Sudoku solver. Ensure cross-browser compatibility for popular web browsers such as Chrome, Firefox, and Safari.

2.Code Editor:

You will need a code editor to write and edit your HTML, CSS, and JavaScript code. Some popular code editors include Visual Studio Code, Sublime Text, and Atom.

3. Version Control:

Consider using version control software like Git to track changes and collaborate with others if you are working in a team.

4.Web Server (optional):

While not strictly required for development, if you plan to deploy the Sudoku solver on a web server to make it accessible to users online, you'll need access to a web server environment. Common web servers include Apache, Nginx, and hosting platforms like Heroku or GitHub Pages.

Hardware Requirements:

1. Computer:

You will need a computer for development. The specific hardware requirements for your development machine depend on your chosen code editor and the complexity of your project. In general, a modern computer with sufficient RAM and a reasonably fast processor should suffice.

2. Internet Connection:

An internet connection is required for downloading libraries and resources, collaborating with others, and deploying the project online.

3. Input Devices:

A keyboard and mouse or other input devices are necessary for coding and testing the application.

4. Display:

A monitor or screen with adequate resolution for coding and testing is essential.

5. Memory and Storage:

Ensure your computer has enough memory (RAM) to run development tools and that you have sufficient storage space for your project files.

6. Browser Testing:

To ensure cross-browser compatibility, you may need access to multiple devices or virtual machines with different web browsers for testing.

4.system architecture :

The system architecture for a Sudoku solver web application involves outlining the components, their interactions, and the flow of data and operations within the system. Here's a high-level system architecture for your Sudoku solver project using HTML, CSS, and JavaScript:

System Components:

1. User Interface (UI):

- The user interface is the front end of the application that users interact with.
- It includes the Sudoku grid, number input, action buttons, validation feedback, and optional features.

2. HTML, CSS, and JavaScript:

- HTML is used for structuring the web page, creating the layout, and defining the elements.
- CSS is used for styling the user interface, making it visually appealing.
- JavaScript is used for implementing the logic, Sudoku solving algorithm, user interactions, and real-time validation.

3. Sudoku Solver Algorithm:

- The core of the application, the Sudoku solver algorithm, is implemented in JavaScript.
- It processes the user input and finds a solution for the Sudoku puzzle.

- The algorithm follows Sudoku rules to ensure a valid solution.

System Flow:

1. User Input:

- Users interact with the UI to input the initial Sudoku puzzle, enter numbers into the grid, and trigger actions like "Solve," "Reset," or "Clear."

2. Validation:

- JavaScript functions validate the user input in real time. They check whether the entered numbers comply with Sudoku rules and provide feedback to the user.

3. Solving Process:

- When the user clicks the "Solve" button, the Sudoku solving algorithm is invoked.
- The algorithm processes the partially filled Sudoku grid to find a valid solution.
- It uses techniques such as backtracking to explore possible solutions while adhering to Sudoku rules.

4. Solution Display:

- Once a solution is found, the solved Sudoku puzzle is displayed in the grid, highlighting the solved cells.

5. Optional Features:

- If optional features are included (e.g., difficulty levels, timer, hints, undo), they interact

with the UI and the solving algorithm as needed.

Data Flow:

- User input data flows from the UI to JavaScript for validation and processing.

- The Sudoku solver algorithm processes the data, searches for a solution, and sends the solution back to the UI for display.

- Feedback and messages from the validation and solving processes are displayed in the UI.

External Dependencies:

- The project may rely on external libraries or APIs for specific functionalities, such as UI frameworks (e.g., Bootstrap) for styling and user interface components, or third-party code for additional features.

- Deployment:
  - The application can be deployed on a web server to make it accessible to users through their web browsers.
- Security Considerations:
  - Ensure that the application is secure and doesn't have vulnerabilities, especially if it's deployed on the web.
- Testing and Quality Assurance:
  - Implement testing procedures to validate the functionality of the Sudoku solver and ensure that it works correctly in various scenarios.

This architecture provides a high-level overview of how the components of Sudoku solver system interact and function together.

5. OBJECTIVE:

The primary objective of the Sudoku Solver Web Application is to create a versatile and accessible tool for Sudoku enthusiasts. It aims to offer an efficient solution for solving Sudoku puzzles of varying difficulties through the use of a powerful JavaScript backtracking algorithm. This application provides a responsive, user-friendly interface designed with HTML and CSS, ensuring that users can engage with Sudoku puzzles on a wide range of devices. By offering real-time validation, solution highlighting, and the ability to select puzzle difficulty levels, it empowers users to learn, practice, and enjoy the game. Furthermore, the optional timer functionality adds a competitive element, allowing users to track their solving times, fostering a sense of accomplishment and improvement in Sudoku-solving skills.

ELEMENTS USED IN THE PROJECT:

- HTML:
  - HTML (Hypertext Markup Language) is the foundation of the project, defining the structure and content of the web page. It includes elements like headings, tables, forms, and divs to organize and present the Sudoku grid and user interface.
- CSS:
  - CSS (Cascading Style Sheets) is used to define the project's visual style and layout. It's employed to create a responsive design, format text, and enhance the user experience with styling and animations.
- JavaScript:
  - JavaScript is the programming language responsible for the core functionality of the application. It includes elements such as event listeners, functions, and algorithms to handle user interactions, solve Sudoku puzzles, and provide real-time validation.
- Sudoku Grid:
  - The Sudoku grid is implemented using HTML tables or div elements to create a 9x9 grid structure where users can input numbers and view the solved puzzle.

- User Input Fields:
  - These are input fields within the Sudoku grid where users can click and enter numbers to create their puzzle.

- Solver Algorithm:
  - The JavaScript algorithm is a critical element that uses a backtracking approach to solve Sudoku puzzles programmatically.

- Difficulty Level Selector:
  - A dropdown or menu that allows users to choose the difficulty level of Sudoku puzzles they want to solve.

- Solution Highlighting:
  - JavaScript code to highlight the solved puzzle for users to compare with their input.

- Real-Time Validation:
  - JavaScript validation functions that check the user's input to ensure it adheres to Sudoku rules, preventing invalid entries.

- Clear and Reset Buttons:
  - Buttons or links to clear the grid or reset it to the initial state, allowing users to start over.

- Responsive Design Elements:
  - HTML and CSS elements like media queries and flexible layouts that ensure the application adapts to different screen sizes and devices.

These elements work together to create a seamless and enjoyable user experience in the Sudoku Solver Web Application, allowing users to interact with Sudoku puzzles, learn, and improve their skills.

7.IMPLEMENTATION

7.1. CODE

Main.html :

```
<!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="styles.css">
</head>
<body>
    <div id="container">
        <h1 class = "padd">Sudoku Solver</h1>
        <table id="sudoku-board">

          <colgroup><col><col><col>
          <colgroup><col><col><col>
          <colgroup><col><col><col>
          <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>
          <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>
          <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>
```

Styles.css :

```
 @importurl('https://fonts.googleapis.com/css2?family=Water+Brush&display=swap');
body {
    background-color: #f6cce5;
}

#container {
  text-align: center;
}

.padd {
    font-size: 5.6rem;
    font-family: 'Water Brush', cursive;
}

table {
  border-collapse: collapse;
  font-size: 2em;
  margin: 0 auto;
}

colgroup,
tbody {
  border: solid medium;
}
```

```
td {
  border: solid thin;
  height: 1.4em;
  width: 1.4em;
  text-align: center;
  padding: 0;
}

.btns{
    display: flex;
    text-align: center;
    justify-content: center;
}

button {
  margin-top: 15px;
  font-size: 1.5em;
  background-color: #A288E3;
  color: white;
  margin-left: 5px;
  border: #A288E3;
  border-radius: 5px;
  padding: 5px 10px;
}

button:hover{
    color: #A288E3;
    background-color: white;
    outline: #A288E3;
    transition: all 1s ease;
    cursor: pointer;
}
```

```
@media screen and (max-width: 460px) {
    .padd {
        font-size: 4.2rem;
    }

    table {
      font-size: 1.5em;
    }

    button{
        font-size: 1rem;
    }
}
```

Script.js :

```
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];
      } else {
        tdEl.innerText = "";
      }
    }
  });

document
  .getElementById("solve-button")
  .addEventListener("click", function (event) {
    var boardString = boardToString();
    var solution = SudokuSolver.solve(boardString);
```

```
    if (solution) {
      stringToBoard(solution);
    } else {
      alert("Invalid board!");
    }
  });

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 = "";
  }
}

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;
    } else {
      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;
    }
  }
}

("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("");
    if (boardIsInvalid(boardArray)) {
      return false;
    }
    return recursiveSolve(boardString);
  }

  function solveAndPrint(boardString) {
    var solvedBoard = solve(boardString);
    console.log(toString(solvedBoard.split("")));
```

```
  return solvedBoard;
}

function recursiveSolve(boardString) {
  var boardArray = boardString.split("");
  if (boardIsSolved(boardArray)) {
    return boardArray.join("");
  }
  var cellPossibilities = getNextCellAndPossibilities(boardArray);
  var nextUnsolvedCellIndex = cellPossibilities.index;
  var possibilities = cellPossibilities.choices;
  for (var i = 0; i < possibilities.length; i++) {
    boardArray[nextUnsolvedCellIndex] = possibilities[i];
    var solvedBoard = recursiveSolve(boardArray.join(""));
    if (solvedBoard) {
      return solvedBoard;
    }
  }
  return false;
}

function boardIsInvalid(boardArray) {
  return !boardIsValid(boardArray);
}

function boardIsValid(boardArray) {
  return (
    allRowsValid(boardArray) &&
    allColumnsValid(boardArray) &&
    allBoxesValid(boardArray)
  );
}

function boardIsSolved(boardArray) {
```

```
  for (var i = 0; i < boardArray.length; i++) {
    if (boardArray[i] === "-") {
      return false;
    }
  }
  return true;
}

function getNextCellAndPossibilities(boardArray) {
  for (var i = 0; i < boardArray.length; i++) {
    if (boardArray[i] === "-") {
      var existingValues = getAllIntersections(boardArray, i);
      var choices = ["1", "2", "3", "4", "5", "6", "7", "8", "9"].filter(
        function (num) {
          return existingValues.indexOf(num) < 0;
        }
      );
      return { index: i, choices: choices };
    }
  }
}

function getAllIntersections(boardArray, i) {
  return getRow(boardArray, i)
    .concat(getColumn(boardArray, i))
    .concat(getBox(boardArray, i));
}

function allRowsValid(boardArray) {
  return [0, 9, 18, 27, 36, 45, 54, 63, 72]
    .map(function (i) {
      return getRow(boardArray, i);
    })
    .reduce(function (validity, row) {
```

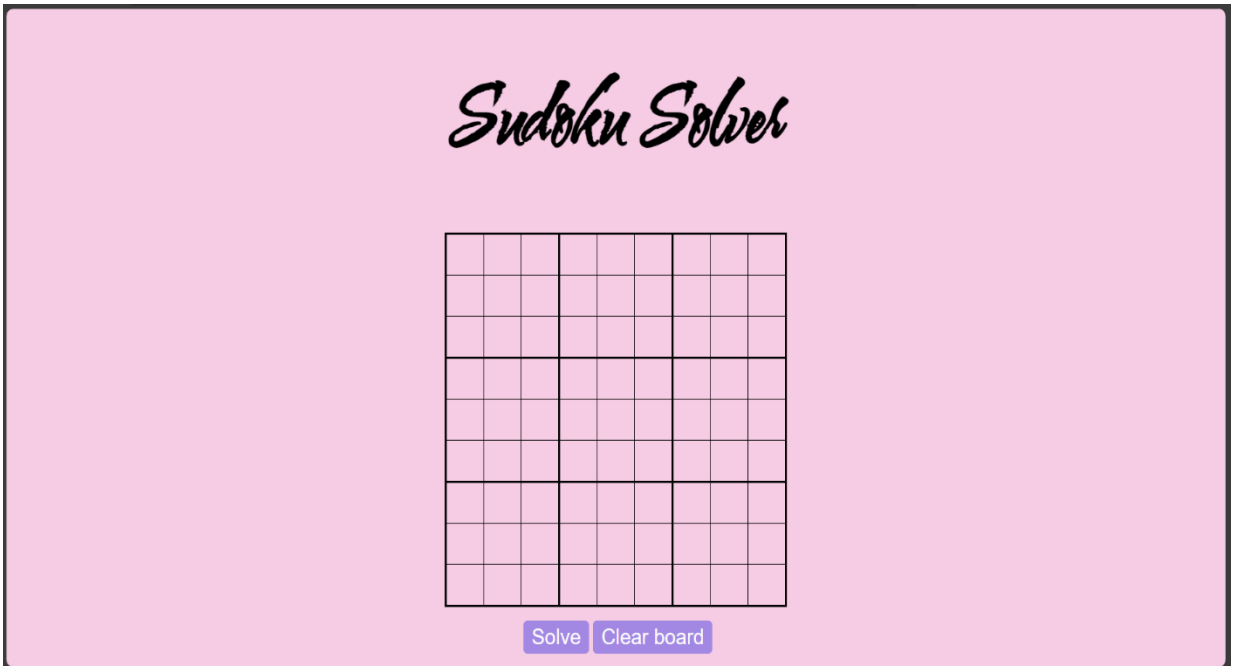
```
    return collectionIsValid(row) && validity;
  }, true);
}

function getRow(boardArray, i) {
 var startingEl = Math.floor(i / 9) * 9;
 return boardArray.slice(startingEl, startingEl + 9);
}

function allColumnsValid(boardArray) {
 return [0, 1, 2, 3, 4, 5, 6, 7, 8]
   .map(function (i) {
    return getColumn(boardArray, i);
   })
   .reduce(function (validity, row) {
    return collectionIsValid(row) && validity;
   }, true);
}

function getColumn(boardArray, i) {
 var startingEl = Math.floor(i % 9);
 return [0, 1, 2, 3, 4, 5, 6, 7, 8].map(function (num) {
  return boardArray[startingEl + num * 9];
 });
}

function allBoxesValid(boardArray) {
 return [0, 3, 6, 27, 30, 33, 54, 57, 60]
   .map(function (i) {
    return getBox(boardArray, i);
   })
   .reduce(function (validity, row) {
    return collectionIsValid(row) && validity;
   }, true);
```

```
}

function getBox(boardArray, i) {
  var boxCol = Math.floor(i / 3) % 3;
  var boxRow = Math.floor(i / 27);
  var startingIndex = boxCol * 3 + boxRow * 27;
  return [0, 1, 2, 9, 10, 11, 18, 19, 20].map(function (num) {
    return boardArray[startingIndex + num];
  });
}

function collectionIsValid(collection) {
  var numCounts = {};
  for (var i = 0; i < collection.length; i++) {
    if (collection[i] != "-") {
      if (numCounts[collection[i]] === undefined) {
        numCounts[collection[i]] = 1;
      } else {
        return false;
      }
    }
  }
  return true;
}

function toString(boardArray) {
  return [0, 9, 18, 27, 36, 45, 54, 63, 72]
    .map(function (i) {
      return getRow(boardArray, i).join(" ");
    })
    .join("\n");
}

if (testable) {
```

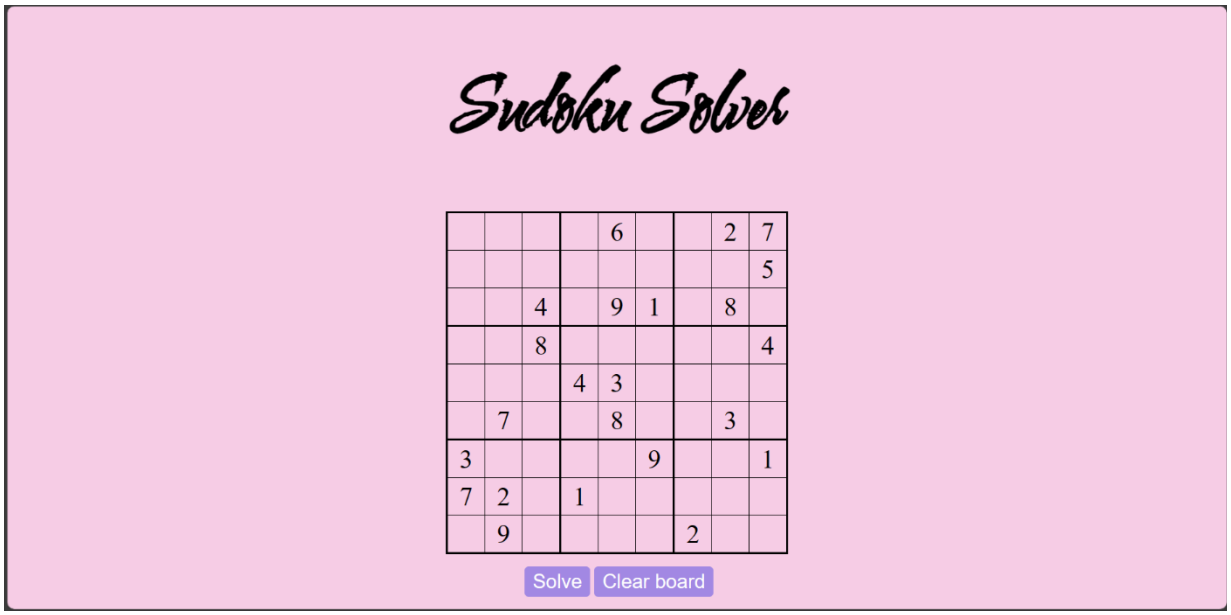
```
    solver = {
      solve: solve,
      solveAndPrint: solveAndPrint,
      recursiveSolve: recursiveSolve,
      boardIsInvalid: boardIsInvalid,
      boardIsValid: boardIsValid,
      boardIsSolved: boardIsSolved,
      getNextCellAndPossibilities: getNextCellAndPossibilities,
      getAllIntersections: getAllIntersections,
      allRowsValid: allRowsValid,
      getRow: getRow,
      allColumnsValid: allColumnsValid,
      getColumn: getColumn,
      allBoxesValid: allBoxesValid,
      getBox: getBox,
      collectionIsValid: collectionIsValid,
      toString: toString,
    };
  } else {
    solver = { solve: solve, solveAndPrint: solveAndPrint };
  }

  return solver;
})(TESTABLE);
```

RESULT SCREENS

Sudoku Table:



Data Entry in Sudoku Table:



Result:


Sudoku Solver
9 1 3 8 6 5 4 2 7
6 8 7 2 4 3 9 1 5
2 5 4 7 9 1 6 8 3
5 3 8 9 1 2 7 6 4
1 6 2 4 3 7 5 9 8
4 7 9 5 8 6 1 3 2
3 4 5 6 2 9 8 7 1
7 2 6 1 5 8 3 4 9
8 9 1 3 7 4 2 5 6
Solve
Clear board

9.CONCLUSION

The Sudoku Solver Web Application, developed with HTML, CSS, and JavaScript, provides a user-friendly and educational platform for Sudoku enthusiasts. The project's core, a robust JavaScript backtracking algorithm, automates Sudoku puzzle-solving, eliminating manual calculations. Real-time validation, solution highlighting, and an optional timer feature enhance the educational and competitive aspects, catering to users of all skill levels. This application goes beyond entertainment; it cultivates logical thinking and problem-solving skills, promoting cognitive development. The responsive design ensures seamless usability on various devices. In a rapidly evolving technological landscape, this project serves as a prime example of how web technologies can simplify complex problems, making them more accessible and enjoyable for a broader audience.

By bringing Sudoku into the digital era, the Sudoku Solver Web Application facilitates learning, enjoyment, and growth, ultimately contributing to the popularity of this classic game while empowering users to enhance their analytical thinking skills, fostering a deeper appreciation for the world of puzzles and logic.