RKDF UNIVERSITY

Ranchi , Jharkhand

INTERNSHIP REPORT

Submitted by

Chandan Kumar Mahto (001CSE22GT026)

in partial fulfillment for the award of the degree

of

Bachelor of Engineering

in

Computer Science and Engineering

RKDF University

Ranchi , Jharkhand

An Autonomous Instituition

MAY 2025

BONAFIDE CERTIFICATE

Certified that this internship report

“ Mine Sweeper “

is the bonafide work of

Chandan Kumar Mahto (001CSE22GT026)

who carried out the project work under my supervision

SIGNATURE OF HOD	SIGNATURE OF SUPERVISIOR
Shubhangni Dey	**Abhishek Kumar Singh**
HEAD OF DEPARTMENT	SUPERVISOR
Computer science and Engineering	Computer science and Engineering
RKDF University , Ranchi , Jharkhand	**RKDF University , Ranchi , Jharkhand**

Submitted for the Autonomous End Semester Examination Internship Project

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

This project focuses on the development of a classic puzzle game, **Mine Sweeper**, implemented as a web-based application using HTML, CSS, and JavaScript. Mine Sweeper challenges the player to clear a grid of hidden mines without detonating any, using numerical clues provided by revealed cells indicating the count of adjacent mines. The project demonstrates the effective use of front-end web technologies to create an interactive and user-friendly gaming experience.

The structure of the game board is created using HTML, while CSS is employed to style the grid, cells, and game interface for clear visual distinction between different cell states such as unrevealed, revealed, flagged, and mined cells. JavaScript is responsible for the core functionality, including randomly placing mines, calculating adjacent mine counts, handling user interactions (left-click to reveal cells and right-click to flag mines), and determining game outcomes such as win or loss conditions.

Key programming concepts utilized in this project include DOM manipulation to dynamically update the game state and user interface, event handling to respond to player actions, and recursion to reveal empty adjacent cells automatically. The project not only provides entertainment but also serves as a practical exercise in logic, algorithm design, and web development skills.

The Mine Sweeper game developed here is scalable and can be enhanced with additional features such as difficulty levels, timers, and responsive design. Overall, this project is a comprehensive example of integrating HTML, CSS, and JavaScript to build an engaging browser-based game that tests the player's strategic thinking and problem-solving abilities.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to RKDF University for providing me with the opportunity and resources to work on this mini project titled **"Mine Sweeper."** This project has been a valuable learning experience that helped me enhance my skills in web development and problem-solving.

I am deeply thankful to my project guide, **Mr Jagarnath Reddy Sir** whose expert guidance, continuous encouragement, and constructive feedback were instrumental throughout the project development. Their patience and support motivated me to overcome challenges and complete the project successfully.

I also extend my heartfelt thanks to the entire faculty of the Computer Science and Engineering department for their valuable teachings and for fostering a supportive academic environment. Special thanks to my friends and classmates who offered their assistance, ideas, and moral support during this project.

Lastly, I appreciate the availability of various online resources and communities such as MDN Web Docs, W3Schools, and Stack Overflow, which provided me with essential information and examples that greatly aided the implementation of the game logic and user interface.

This project would not have been possible without the contribution and cooperation of all the above-mentioned individuals and institutions.

Chandan Kumar Mahto

(001CSE22GT026)

B.TECH(CSE)

Semester – VI

Table of Contents

1. INTRODUCTION

Mine Sweeper is a classic single-player puzzle game that tests the player's logical thinking and decision-making skills. The objective of the game is to clear a rectangular board containing hidden "mines" without triggering any of them. By uncovering cells, the player gains numerical clues that indicate how many mines are adjacent to that specific cell. Using these clues, the player must strategically deduce the locations of the mines and mark them accordingly.

This project implements the Mine Sweeper game as a web-based application using **HTML**, **CSS**, and **JavaScript**. HTML is used to structure the grid layout of the game board, CSS provides styling and visual feedback, and JavaScript handles the core logic, such as random placement of mines, counting adjacent mines, revealing cells, recursive clearing of empty areas, and determining win or loss conditions.

The project not only serves as a fun and interactive game but also acts as a practical application to understand essential front-end web development concepts. It involves real-time DOM manipulation, event handling, conditional logic, and recursive functions, making it a valuable educational tool for students learning JavaScript and UI/UX design principles.

In addition, the Mine Sweeper project provides an opportunity to explore more advanced ideas such as game state management, user feedback through dynamic styling, and scalable design. This implementation lays a strong foundation for more complex games and interactive applications in web development.

2. Tools and Technologies Used

The development of the **Mine Sweeper** game utilized fundamental web technologies and tools. Each technology played a specific role in the structure, design, and functionality of the game:

Technology / Tool	Purpose / Description
HTML (HyperText Markup Language)	Used to create the structure of the web page and the grid layout for the game board. Each cell of the game is rendered as a dynamic HTML element.
CSS (Cascading Style Sheets)	Used to style the game interface, including the layout, color schemes, hover effects, revealed cells, flagged cells, and overall visual appearance.
JavaScript	Core scripting language used to implement the game logic, including mine placement, user interactions (clicks and flags), recursive cell revealing, and win/loss detection.
DOM Manipulation	JavaScript is used to dynamically update the DOM based on user actions, such as revealing cells, placing flags, and updating messages.
Web Browser	Acts as the runtime environment where the HTML, CSS, and JavaScript are rendered and executed. The project runs entirely in the browser without requiring any external software.
Code Editor (e.g., VS Code, Sublime Text)	Used for writing and editing the source code of the project efficiently with syntax highlighting, extensions, and debugging tools.

3. System Design

The system design of the **Mine Sweeper** game outlines how various components of the application work together to deliver a fully interactive, browser-based puzzle game. The game follows a modular and event-driven architecture, separating concerns across the structure (HTML), style (CSS), and behavior (JavaScript). The key aspects of the system design are described below:

3.1. User Interface Design

The user interface consists of the following elements:

- **Game Title**: Displays the name of the game ("Mine Sweeper").
- **Game Board (Grid)**: A dynamically generated grid (e.g., 10x10) where each cell can be clicked or flagged.
- **Reset Button**: Allows the player to restart the game at any point.
- **Message Area**: Displays real-time game status updates such as “Game Over” or “You Won”.

The interface is styled using CSS to distinguish between:

- Unrevealed cells
- Revealed cells with numbers
- Flagged cells (suspected mines)
- Exploded mines (on game over)

3.2. Game Board Structure

- The board is represented as a **2D array** in JavaScript.
- Each cell in the array is an object containing:
    - mine (boolean): whether the cell has a mine
    - revealed (boolean): whether the cell has been revealed
    - adjacentMines (number): number of mines surrounding the cell
    - element: reference to the actual HTML element representing the cell

3.3. Functional Components

a. Mine Placement

- Random placement of a fixed number of mines (e.g., 15) at the start of the game.
- Ensures no duplicate placement and avoids mines at the first-click position (optional enhancement).

b. Adjacent Mine Counter

- For each non-mine cell, a function calculates the number of adjacent mines using a loop over its 8 neighboring cells.

c. User Interactions

- **Left-click**: Reveals a cell.
  - If the cell is a mine → triggers game over.
  - If the cell has 0 adjacent mines → recursively reveals neighboring empty cells.
- **Right-click**: Flags or unflags a cell as a suspected mine (prevents accidental reveal).

d. Game State Handling

- **Win Condition**: All non-mine cells are revealed.
- **Loss Condition**: A mine is clicked.
- Game status is displayed in the message area, and interactions are disabled after the game ends.

4. Implementations

The implementation of the **Mine Sweeper** game involves the integration of three core technologies: **HTML** for structure, **CSS** for styling, and **JavaScript** for logic and interactivity. The game is designed to be played directly in a web browser without requiring any additional software installations.

4.1. HTML Implementation

The HTML provides the basic structure of the game, including:

- A title (<h1>)
- A container for the grid ( )
- A reset button (<button id="reset">)
- A message area to show game status (<p id="message">)

Sample HTML Code:

html
CopyEdit

```
<!DOCTYPE html>
<html>
<head>
  <title>Mine Sweeper</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Mine Sweeper</h1>
  <button id="reset">Reset Game</button>
  <p id="message"></p>
  <div id="game-board"></div>
  <script src="script.js"></script>
</body>
</html>
```

4.2. CSS Implementation

CSS is used to style the grid and provide visual feedback to the user. Each cell is styled for default, revealed, flagged, and mine states.

Sample CSS Code:

```
css
CopyEdit
#game-board {
  display: grid;
  grid-template-columns: repeat(10, 30px);
  gap: 2px;
  margin-top: 20px;
}

.cell {
  width: 30px;
  height: 30px;
  background-color: #ccc;
  border: 1px solid #999;
  text-align: center;
  line-height: 30px;
  cursor: pointer;
  font-weight: bold;
}

.cell.revealed {
  background-color: #eee;
  cursor: default;
}

.cell.mine {
  background-color: red;
  color: white;
}

.cell.flagged {
  background-color: yellow;
}
```

4.3. JavaScript Implementation

JavaScript handles the logic for:

- Generating the board
- Randomly placing mines
- Calculating adjacent mine counts
- Handling left and right mouse clicks
- Checking for win/loss conditions
- Resetting the game

5. Features

The **Mine Sweeper** game developed using HTML, CSS, and JavaScript includes a range of detailed features that ensure an interactive, user-friendly, and logic-driven gameplay experience. Below are the core features explained in detail:

(i). Dynamic Grid Generation

- The game board is generated dynamically based on a fixed or user-defined size (e.g., 10x10).
- Each cell on the grid is created programmatically and assigned properties such as whether it contains a mine, whether it's revealed, and how many adjacent mines it has.
- This dynamic approach allows for easy customization or scaling to different levels.

(ii). Random Mine Placement

- A specific number of mines are placed randomly across the grid at the start of each game.
- The algorithm ensures that each mine is placed in a unique location.
- Optionally, the algorithm can be enhanced to avoid placing a mine on the first-clicked cell to improve user experience.

(iii). Adjacent Mine Counter

- For every non-mine cell, the game calculates the number of adjacent mines (up to 8 neighbors).
- This number is displayed when a cell is revealed, helping the player make logical decisions about which cell to uncover next.

(iv). Cell Revealing and Recursion

- Left-clicking a cell reveals its content.
- If the cell has zero adjacent mines, it triggers a **recursive reveal** of all neighboring empty cells until numbered or border cells are reached.
- This mimics the behavior of classic Mine Sweeper and improves playability.

(v). Flagging and Unflagging

- Right-clicking on a cell allows the player to **flag** it as a suspected mine.
- A flagged cell is visually marked (e.g., with a flag icon or color), preventing accidental reveals.
- Right-clicking again removes the flag, giving the player flexibility.

(vi) Game Over Detection

- If the player clicks on a mine, the game ends immediately.
- All mines are revealed, and a "Game Over" message is displayed.
- Further interaction is disabled after the game ends.

(vii). Win Condition Check

- The game continuously checks if all non-mine cells have been revealed.
- When this condition is met, a **"You Win"** message is displayed.
- All mines are flagged automatically to signify a successful game completion.

(viii). Reset / Restart Functionality

- A **Reset Game** button allows players to restart the game at any time.
- Clicking it clears the current board, reinitializes the grid, and places new mines for a fresh session.

6. Output and Screenshots

Code:

Index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Minesweeper</title>
<link rel="stylesheet" href="minesweeper.css">
<script src="minesweeper.js"></script>
</head>
<body>
<h1>Mines: <span id="mines-count">0</span></h1>
<div id="board"></div>
<br>
<button id="flag-button">□</button>
</body>
</html>
```

Style.css

```
body {
font-family: Arial, Helvetica, sans-serif;
font-weight: bold;
text-align: center;
}
#board {
width: 400px;
height: 400px;
border: 10px solid darkgray;
background-color: lightgray;
margin: 0 auto;
display: flex;
flex-wrap: wrap;
}
#board div {
width: 48px;
height: 48px;
border: 1px solid whitesmoke;
/* text */
font-size: 30px;
display: flex;
```

```
justify-content: center;
align-items: center;
}
.tile-clicked {
background-color: darkgrey;
}
.x1 {
color: blue;
}
.x2 {
color: green;
}
.x3 {
color: red;
}
.x4 {
color: navy;
}
.x5 {
color: brown;
}
.x6 {
color: teal;
```

```
}
.x7 {
color: black;
}
.x8 {
color: gray;
}
#flag-button {
width: 100px;
height: 50px;
font-size: 30px;
background-color: lightgray;
border: none;
}
```

Script.js

```
let board = [];

let rows = 8;

let columns = 8;

let minesCount = 10;

let minesLocation = []; // "2-2", "3-4", "2-1"

let tilesClicked = 0; //goal to click all tiles except the ones containing mines

let flagEnabled = false;

let gameOver = false;

window.onload = function() {

startGame();

}

function setMines() {

// minesLocation.push("2-2");

// minesLocation.push("2-3");

// minesLocation.push("5-6");

// minesLocation.push("3-4");

// minesLocation.push("1-1");

let minesLeft = minesCount;

while (minesLeft > 0) {

let r = Math.floor(Math.random() * rows);

let c = Math.floor(Math.random() * columns);
```

```
let id = r.toString() + "-" + c.toString();
if (!minesLocation.includes(id)) {
minesLocation.push(id);
minesLeft -= 1;
}
}
}
function startGame() {
document.getElementById("mines-count").innerText = minesCount;
document.getElementById("flag-button").addEventListener("click", setFlag);
setMines();
//populate our board
for (let r = 0; r < rows; r++) {
let row = [];
for (let c = 0; c < columns; c++) {
//<div id="0-0"></div>
let tile = document.createElement("div");
tile.id = r.toString() + "-" + c.toString();
tile.addEventListener("click", clickTile);
document.getElementById("board").append(tile);
row.push(tile);
}
board.push(row);
```

```
}
console.log(board);
}
function setFlag() {
if (flagEnabled) {
flagEnabled = false;
document.getElementById("flag-button").style.backgroundColor = "lightgray";
}
else {
flagEnabled = true;
document.getElementById("flag-button").style.backgroundColor = "darkgray";
}
}
function clickTile() {
if (gameOver || this.classList.contains("tile-clicked")) {
return;
}
let tile = this;
if (flagEnabled) {
if (tile.innerText == "") {
tile.innerText = "□";
}
else if (tile.innerText == "□") {
```

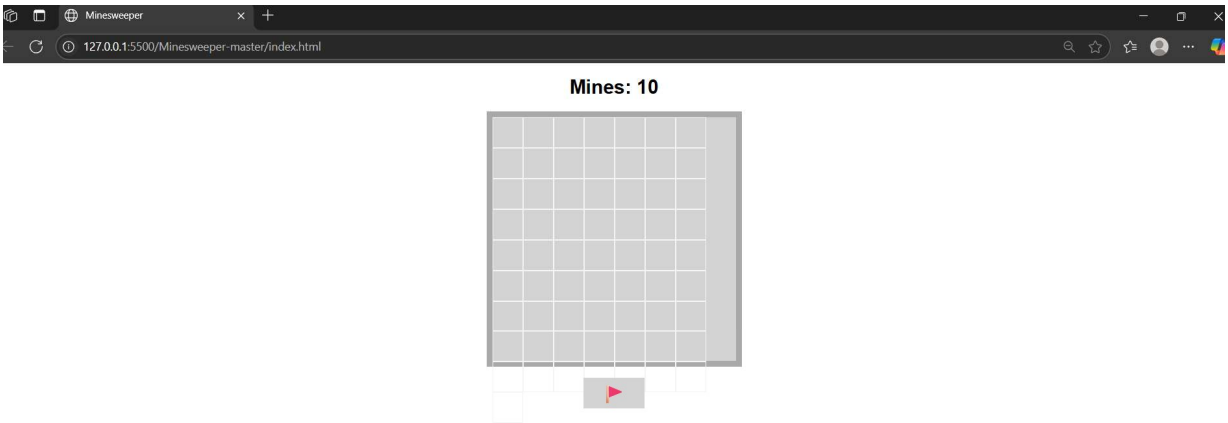
```
tile.innerText = "";
}
return;
}
if (minesLocation.includes(tile.id)) {
// alert("GAME OVER");
gameOver = true;
revealMines();
return;
}
let coords = tile.id.split("-"); // "0-0" -> ["0", "0"]
let r = parseInt(coords[0]);
let c = parseInt(coords[1]);
checkMine(r, c);
}
function revealMines() {
for (let r= 0; r < rows; r++) {
for (let c = 0; c < columns; c++) {
let tile = board[r][c];
if (minesLocation.includes(tile.id)) {
tile.innerText = "□";
tile.style.backgroundColor = "red";
}
```

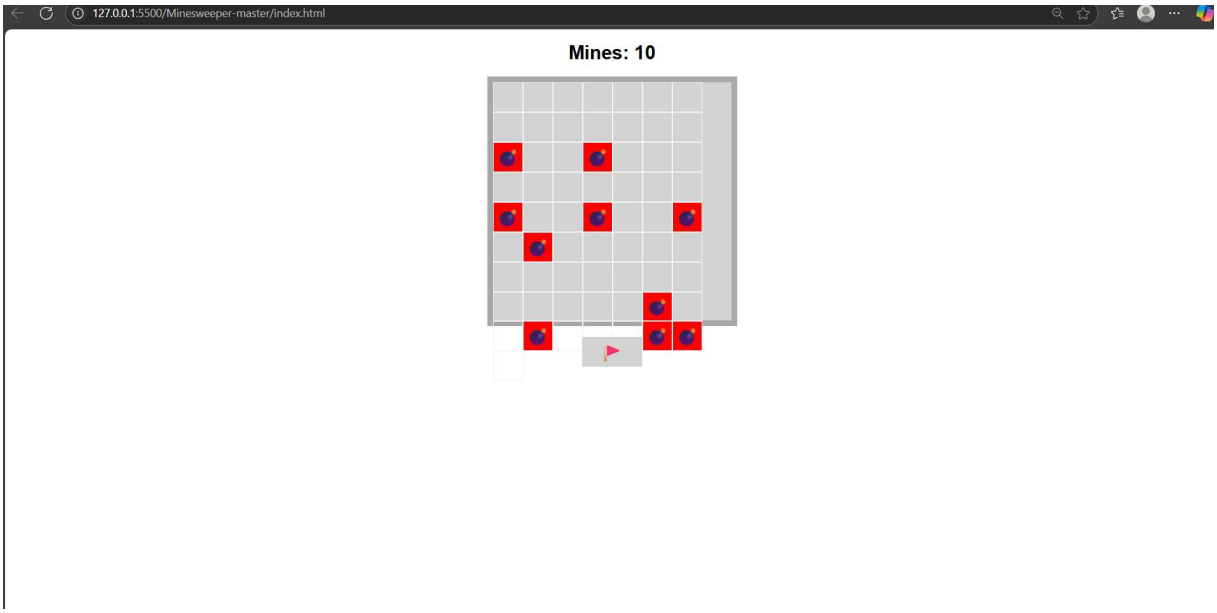
```
}
}
}
function checkMine(r, c) {
if (r < 0 || r >= rows || c < 0 || c >= columns) {
return;
}
if (board[r][c].classList.contains("tile-clicked")) {
return;
}
board[r][c].classList.add("tile-clicked");
tilesClicked += 1;
let minesFound = 0;
//top 3
minesFound += checkTile(r-1, c-1); //top left
minesFound += checkTile(r-1, c); //top
minesFound += checkTile(r-1, c+1); //top right
//left and right
minesFound += checkTile(r, c-1); //left
minesFound += checkTile(r, c+1); //right
//bottom 3
minesFound += checkTile(r+1, c-1); //bottom left
minesFound += checkTile(r+1, c); //bottom
```

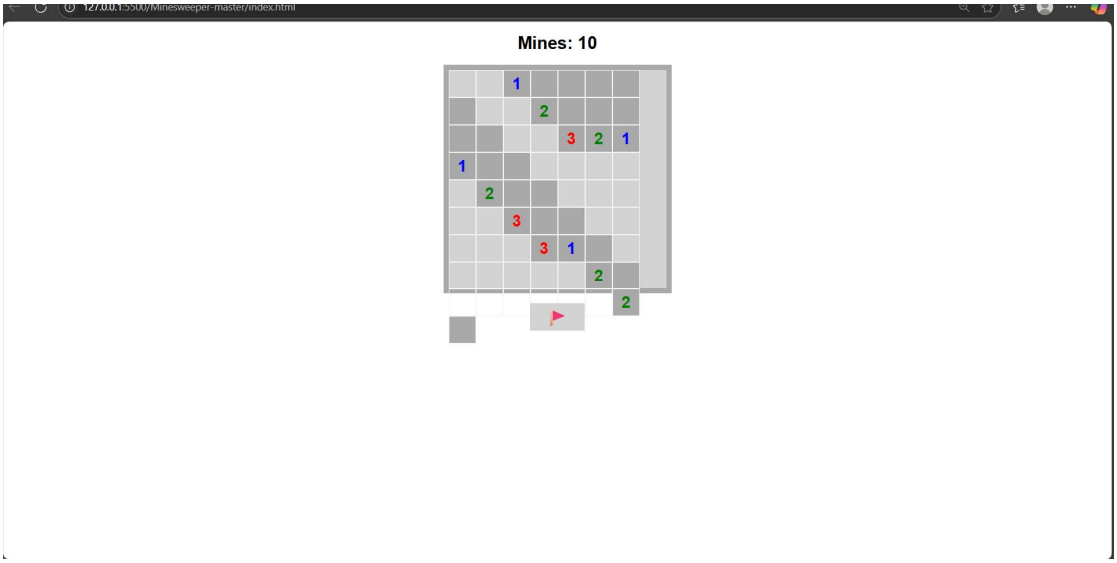
```
minesFound += checkTile(r+1, c+1); //bottom right
if (minesFound > 0) {
board[r][c].innerText = minesFound;
board[r][c].classList.add("x" + minesFound.toString());
}
else {
board[r][c].innerText = "";
//top 3
checkMine(r-1, c-1); //top left
checkMine(r-1, c); //top
checkMine(r-1, c+1); //top right
//left and right
checkMine(r, c-1); //left
checkMine(r, c+1); //right
//bottom 3
checkMine(r+1, c-1); //bottom left
checkMine(r+1, c); //bottom
checkMine(r+1, c+1); //bottom right
}
if (tilesClicked == rows * columns - minesCount) {
document.getElementById("mines-count").innerText = "Cleared";
gameOver = true;
}
```

```
}
function checkTile(r, c) {
if (r < 0 || r >= rows || c < 0 || c >= columns) {
return 0;
}
if (minesLocation.includes(r.toString() + "-" + c.toString())) {
return 1;
}
return 0;
}
```

Output:




127.0.0.1:5500/Minesweeper-master/index.html
Mines: 10


Mines: 10

7.CONCLUSION

The development of the **Mine Sweeper** game using **HTML, CSS, and JavaScript** has been a rewarding and insightful project that not only highlights the power of front-end web technologies but also strengthens fundamental programming and problem-solving skills. This classic logic-based puzzle game has been successfully implemented as an interactive, browser-based application, demonstrating the effective use of web development techniques.

Throughout the course of the project, key objectives were met—ranging from dynamic board creation and random mine placement to user interactions, flagging functionality, and win/loss condition detection. The application provides immediate feedback and ensures an engaging experience through visual cues, such as revealing cells, highlighting mines, and displaying endgame messages. The game also supports reusability through a reset feature, encouraging repeated play without reloading the page.

From a technical perspective, the project showcases the use of **DOM manipulation**, **event handling**, **recursive logic**, and **modular code structure**. It also serves as an excellent example of how a real-world game can be implemented entirely using client-side scripting without relying on any external libraries or frameworks.

Moreover, working on this project has helped reinforce best practices in coding, debugging, and UI design. It has proven to be an excellent way to transition theoretical knowledge into a practical, functional product that is both entertaining and educational.

In conclusion, the **Mine Sweeper** game is a complete and efficient application that demonstrates strong command over front-end web development. It offers a solid foundation for future enhancements such as adding difficulty levels, sound effects, animations, timer-based scoring, and mobile responsiveness, making it a scalable and extensible web project.

8.References

- **MDN Web Docs – Mozilla Developer Network**
https://developer.mozilla.org/
Used for understanding JavaScript functions, DOM manipulation, and event handling.

- **W3Schools Online Web Tutorials**
https://www.w3schools.com/
Referenced for basic syntax and examples of HTML, CSS, and JavaScript.

- **Stack Overflow**
https://stackoverflow.com/
Community-driven platform used for troubleshooting specific errors and logic-related queries.

- **GeeksforGeeks – JavaScript Tutorials**
https://www.geeksforgeeks.org/javascript/
Consulted for understanding recursion, grid generation, and logical implementation.

- **YouTube Tutorials on JavaScript Game Development**
Various YouTube channels were referred for inspiration and ideas on implementing mine sweeping logic and UI behavior.

- **CodePen / JSFiddle**
https://codepen.io/
https://jsfiddle.net/
Used for quick testing and prototyping of components during development.

- **Github -** https://github.com/Chandan-890/Mine-sweeper.git