BIRZEIT UNIVERSITY

Faculty of Engineering and Technology Department of Electrical and Computer Engineering

Artificial Intelligence

ENCS3340

PROJECT 1

MAGNETIC CAVE GAME

Prepared by :

Eng.Dana Hammad -1191568

Eng.Maisam Alaa -1200650

Instructor: Dr. Yazan Abu Farha &

Dr.Ismael

Section 2 & 3

20 /6/ 2023

Abstract:

In this report, software that implements a two-player board game on a grid resembling a chessboard is discussed. Players can take turns using the application to set their respective symbols on the grid's squares. Creating a line of five consecutive symbols (horizontally, vertically, or diagonally) before your opponent does is the goal of the game.

JavaScript, HTML, and CSS were used to create the software. The chessboard grid is defined by the HTML framework, which also creates the squares on demand. The squares and the symbols (X and O) have their appearances customized using CSS style. The game's rules and interactive elements are handled via JavaScript logic.

Initialization, determining if a win condition exists, and validating movements are among the program's primary tasks. When a player clicks on a square while playing, the software determines if the move is legal, changes the square with the player's symbol, and adds the appropriate style. Additionally, a winning condition or a tie game is checked. When a winning condition is satisfied, the game is over and an alert is shown.
The given HTML code may be copied into an HTML file and viewed in a web browser to execute the application. For a game of competition, players can click on squares in turn.

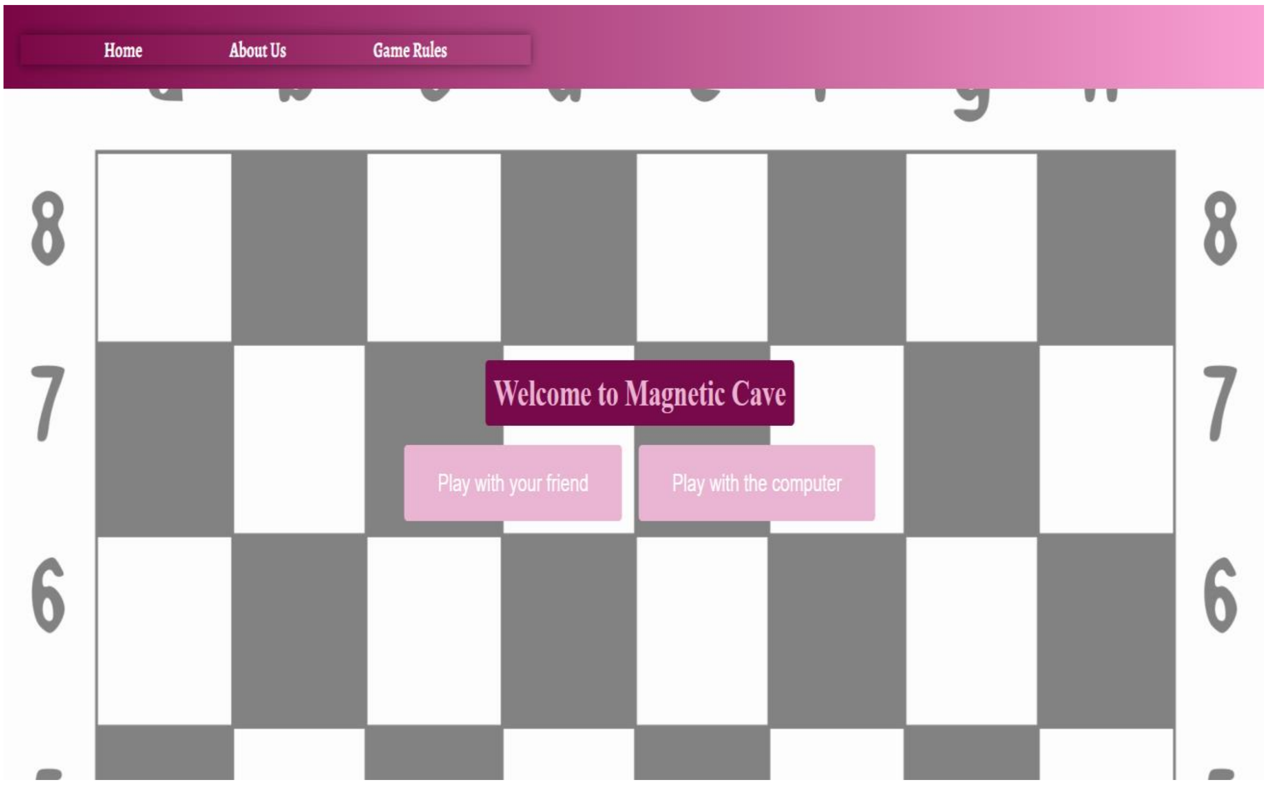
The game shows how to leverage web technologies to provide a straightforward yet fun experience that is both interactive and engaging for two players.

The task of this project is to implement the minimax algorithm to run the game automatically. The program is able to run in manual mode, in which players can enter moves manually, and in automatic mode, in which the program determines the next move within a time period of 3 seconds.

Explanation of interfaces and their work:

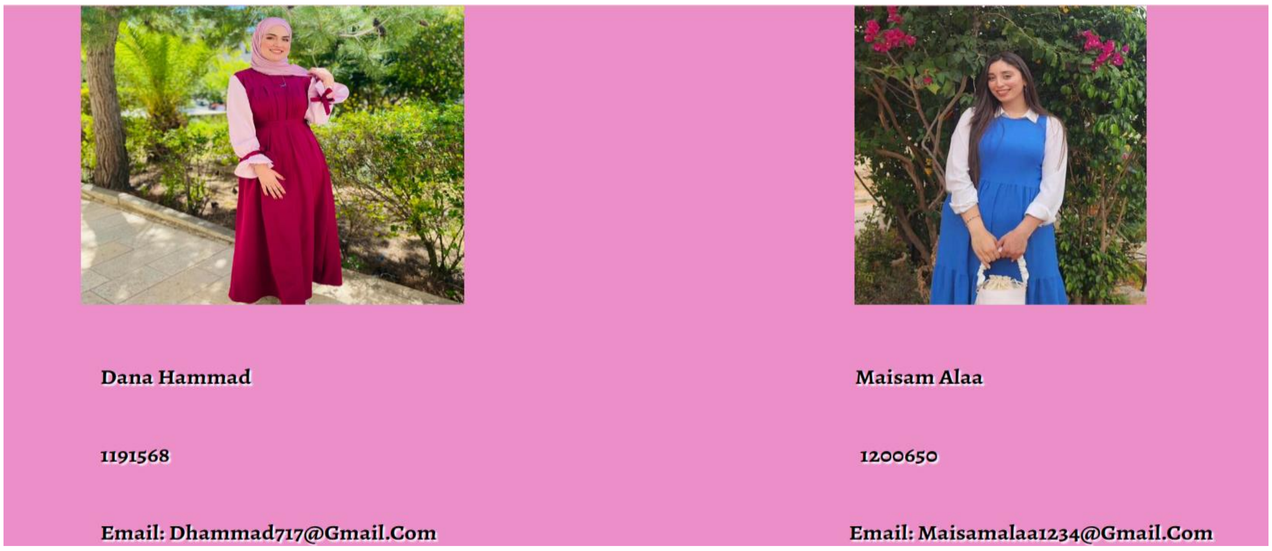
-Main page:

The primary interface of the website, developed using HTML, features a navigation menu positioned in the upper section. Additionally, the interface includes two distinct buttons that afford users the option to select between multiplayer gameplay or engaging in a game against a computer opponent.



-About us:

Have information contact with us



Dana Hammad

1191568

Email: Dhammad717@Gmail.Com

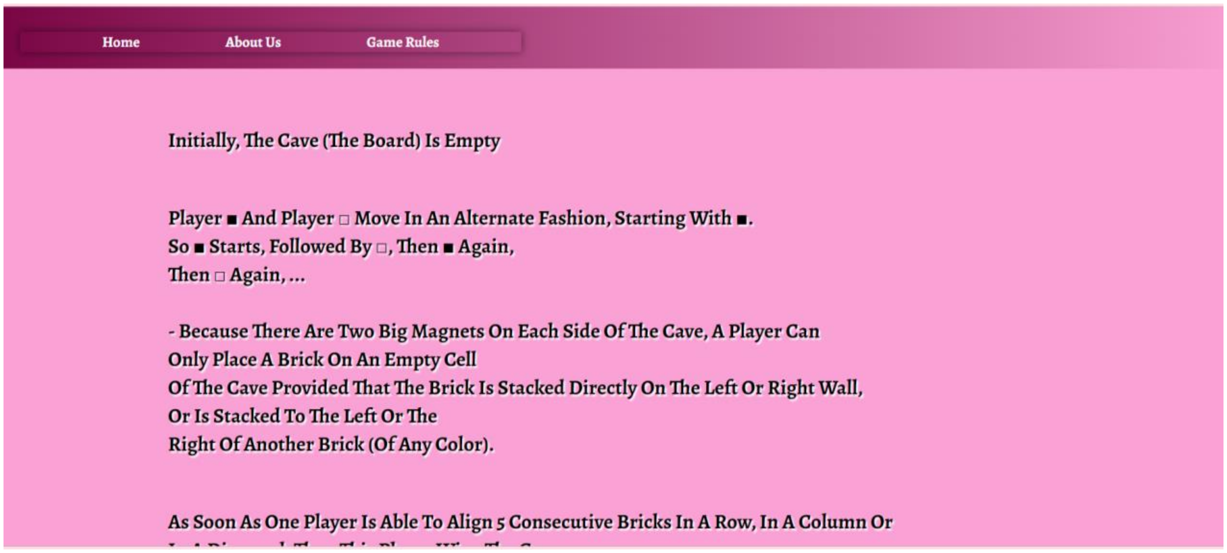
Maisam Alaa

1200650

Email: Maisamalaa1234@Gmail.Com

-Rule page:

Explanation of the rules of the game



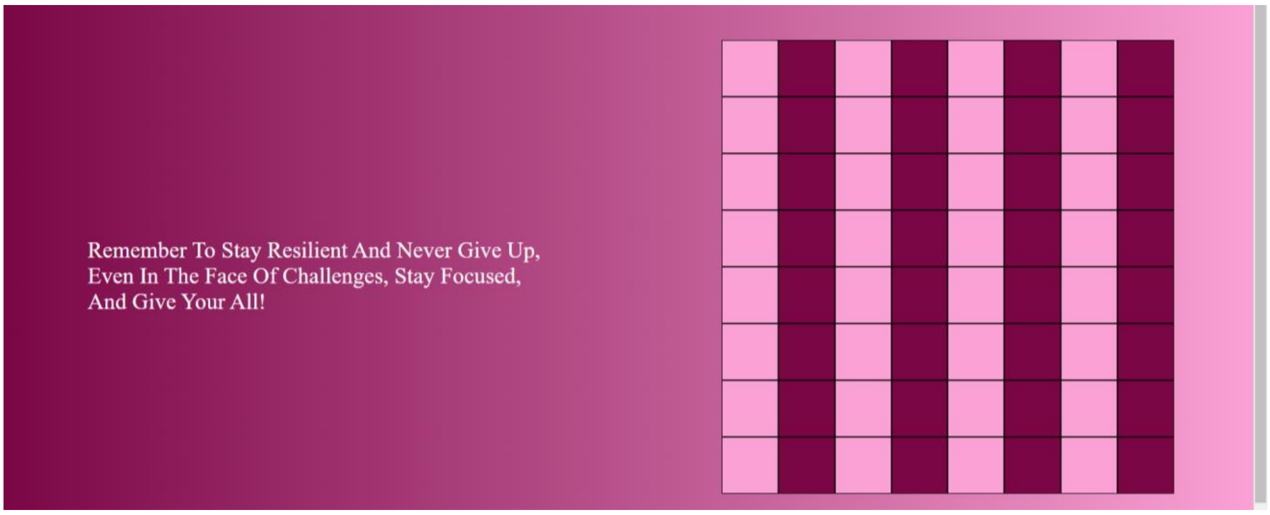
Initially, The Cave (The Board) Is Empty

Player ■ And Player □ Move In An Alternate Fashion, Starting With ■.
So ■ Starts, Followed By □, Then ■ Again,
Then □ Again, ...

- Because There Are Two Big Magnets On Each Side Of The Cave, A Player Can
Only Place A Brick On An Empty Cell
Of The Cave Provided That The Brick Is Stacked Directly On The Left Or Right Wall,
Or Is Stacked To The Left Or The
Right Of Another Brick (Of Any Color).

As Soon As One Player Is Able To Align 5 Consecutive Bricks In A Row, In A Column Or

-play with friend:



Remember To Stay Resilient And Never Give Up,
Even In The Face Of Challenges, Stay Focused,
And Give Your All!

How the code work?

The provided program presents an implementation of a two-player board game that takes place on a grid resembling a chessboard. Its purpose is to enable two players to engage in turns, strategically placing their respective symbols on the grid's squares. The primary objective of the game is to achieve a sequence of five consecutive symbols, either horizontally, vertically, or diagonally, before the opponent accomplishes the same.

Let's examine the program and its principal components:

HTML Structure: The program employs HTML to define the game board's structure. It constructs a chessboard-like grid using a <div> element with the class "chessboard" and dynamically generates 64 squares within this container.

CSS Styling: To enhance visual appeal and functionality, the program applies CSS styling to the squares. It specifies properties such as background colors, sizes, and distinct styles for the two symbols (X and O).

JavaScript Logic: JavaScript is utilized to implement the game's logic and interactivity. The program incorporates the following key functions:

a. Initialization: Upon the loading of the Document Object Model (DOM) content, the program dynamically creates the squares, assigns event listeners to them, and initializes necessary variables. It also establishes the squares array, responsible for storing references to individual square elements, and the selectedSquares array, which tracks the selected squares.

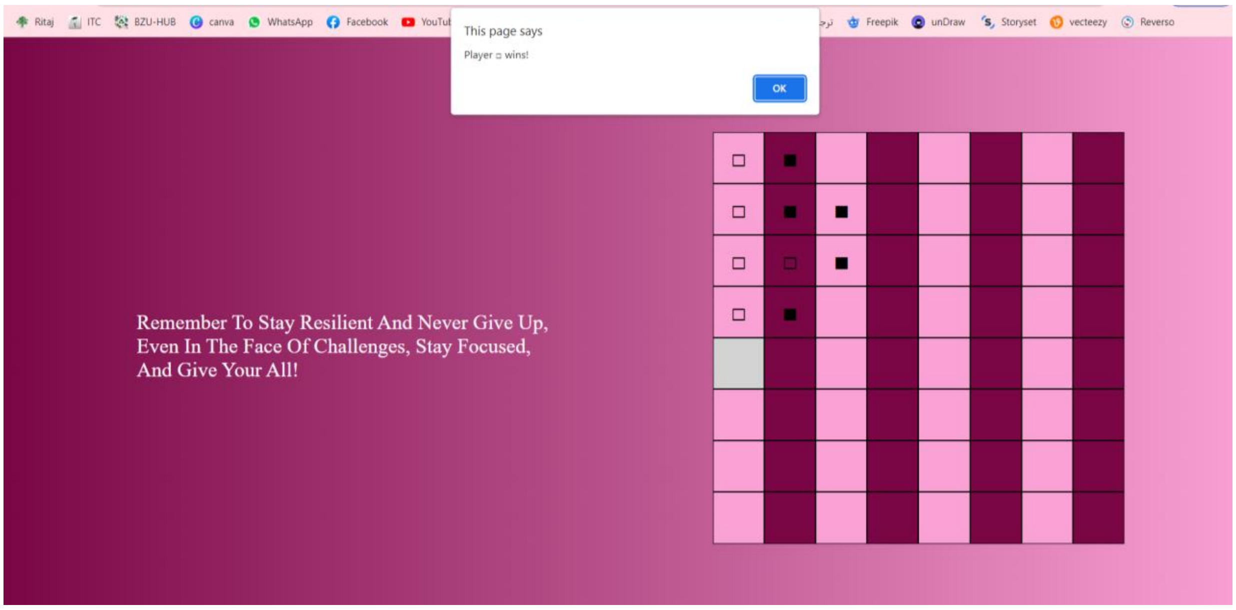
b. CheckWin: This function examines all possible winning lines on the board to determine if a player has won. It iterates over the winningLines array, which holds the indices of squares forming a winning line. The function verifies if any of these lines contain five consecutive symbols of the current player and returns true if a winning condition is detected.

c. CheckValid: This function ensures that a move is valid by checking if the selected square has already been played and adheres to the game's rules. It examines the selectedSquares array and returns true if the move is considered valid.

Gameplay: The program facilitates player turns by allowing them to click on the grid's squares. When a square is clicked, the program first employs the checkValid function to verify the move's validity. If the move is deemed valid, the program updates the corresponding square with the current player's symbol and applies the appropriate CSS class for styling. It then utilizes the checkWin function to determine if the current player has won. If a winning condition is met, an alert is displayed, signaling the end of the game. If no winning condition is satisfied and the number of moves reaches 64 (indicating a tie), the game is also considered over. Otherwise, the turn is passed to the next player.

To execute the program, copy the HTML code into an HTML file and open it in a web browser. The game will be presented, enabling you to click on the squares and engage in gameplay against another player.

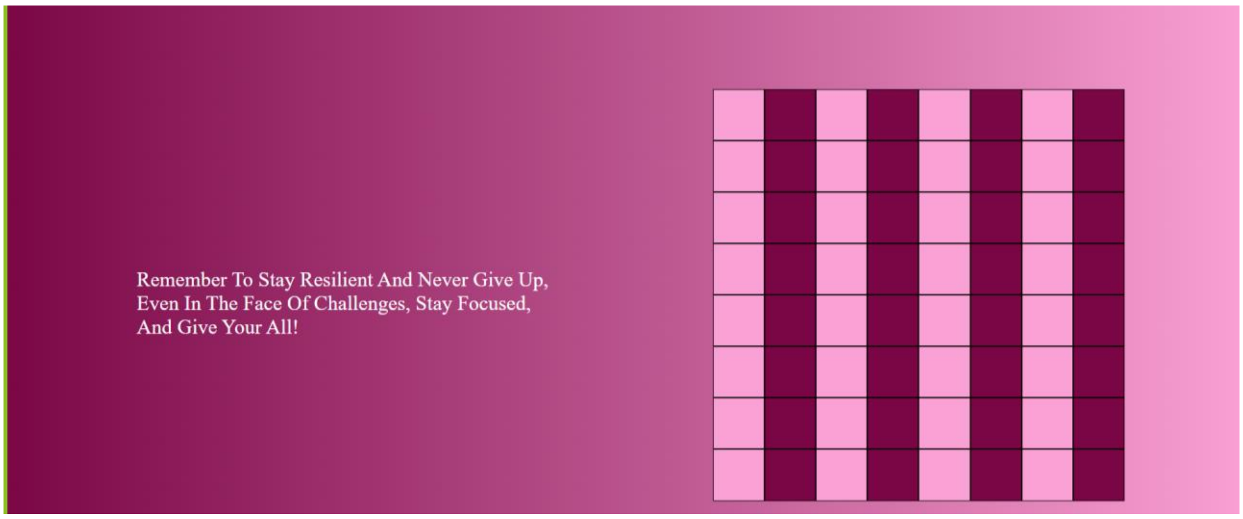
Play with a friend and the score :


This page says

Player □ wins!

Remember To Stay Resilient And Never Give Up, Even In The Face Of Challenges, Stay Focused, And Give Your All!

Play with a computer:



Remember To Stay Resilient And Never Give Up, Even In The Face Of Challenges, Stay Focused, And Give Your All!

The software simulates a two-player board game played on a chessboard-like grid. Players can take turns placing their symbols on the grid's squares. In order to win, you must create a line of five consecutive symbols that runs either horizontally, vertically, or diagonally before your opponent.
Simply open the HTML file in a web browser to start the application. The UI for the game will appear, with a grid that resembles a chessboard. To move, players simply click on the squares.

The program's primary operations and data structures are as follows:
Startup Process: As the page loads, the software dynamically generates the grid's squares and implements event listeners for user interactions. Critical variables such as squares (an array that keeps track of square elements), selectedSquares (an array signifying the chosen squares), currentPlayer (the symbol representing the current player), moves (the count of executed moves), gameOver (an indicator flag signaling the end of the game), and currentMode (the game mode, deciding whether the moves are performed manually or automatically) are also initialized.

Event Monitors: The software pays heed to click events on the grid's squares. Whenever a square is clicked, the function makeManualMove gets invoked, handling the procedure of making a move manually.

Move Verification: The function checkValid ascertains whether a move is acceptable based on the chosen square. It verifies whether the square has been previously selected and complies with the game's principles.

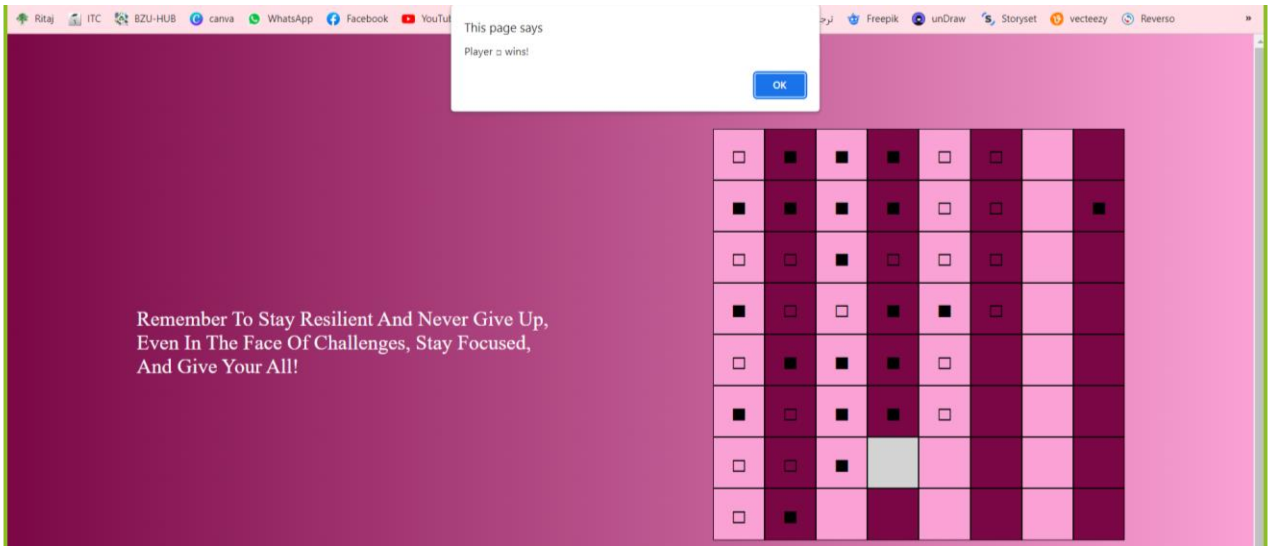
Game Mechanics: The software utilizes the checkWin function to ascertain a win condition. It investigates all plausible winning sequences on the board to find out if a player has composed a line of five sequential symbols.

Automated Moves: During automatic mode, the software employs the minimax algorithm with alpha-beta pruning to determine the most advantageous move for the computer-controlled player. The function makeComputerMove simulates and assesses possible moves, picking the one with the highest value.

The software continuously refreshes the grid with the players' symbols throughout the game, verifies for a win situation or draw, and transfers the turn to the following player. The game proceeds until a player triumphs or a draw transpires.

The tournament's outcomes hinge on the players' strategies and decision-making skills. Victory or defeat can be traced back to factors like executing optimal moves, foreseeing the opponent's moves, and adjusting to fluctuating game circumstances. Reviewing the game's progress, appraising potential moves, and applying effective tactics can influence the final result of the competition.

-In the first case the computer win :



This page says

Player □ wins!

Remember To Stay Resilient And Never Give Up, Even In The Face Of Challenges, Stay Focused, And Give Your All!

Conclusion:

The "Magnetic Cave Game" project presents a comprehensive exploration of integrating JavaScript, HTML, and CSS to create an engaging and interactive two-player board game. Through the creation of a grid that mimics a chessboard and an accompanying user interface, it offers a simple yet stimulating gaming experience.

The software successfully integrates key elements of game design such as initial setup, win condition verification, move validation, and automated moves using minimax algorithms. This sophisticated combination allows both manual play where individuals can interact, and automatic mode that tests the effectiveness of computer-based strategies within a 3-second decision window.

The game stands as a testament to the versatility of web technologies, demonstrating how they can be harnessed to create applications that are not only functional but also engaging. It shows the power of combining basic technologies such as HTML and CSS with more advanced JavaScript logic to create a dynamic, interactive user experience.

Importantly, the project serves as an educational tool, illustrating the application of various algorithms and principles in a practical, easy-to-understand way. Through its clear explanation of how the game's rules and structure are implemented in code, it offers valuable insights into the logic and programming techniques that underpin game development.

While providing an enjoyable gaming experience, "Magnetic Cave Game" also contributes to a deeper understanding of web development and artificial intelligence concepts. Its accessibility and strategic depth make it a valuable addition to the field of computer-based board games. Moving forward, the project could be further enhanced with additional features such as a more advanced AI, multiplayer functionality over the network, and customizable game rules, paving the way for future explorations in the field of game development.