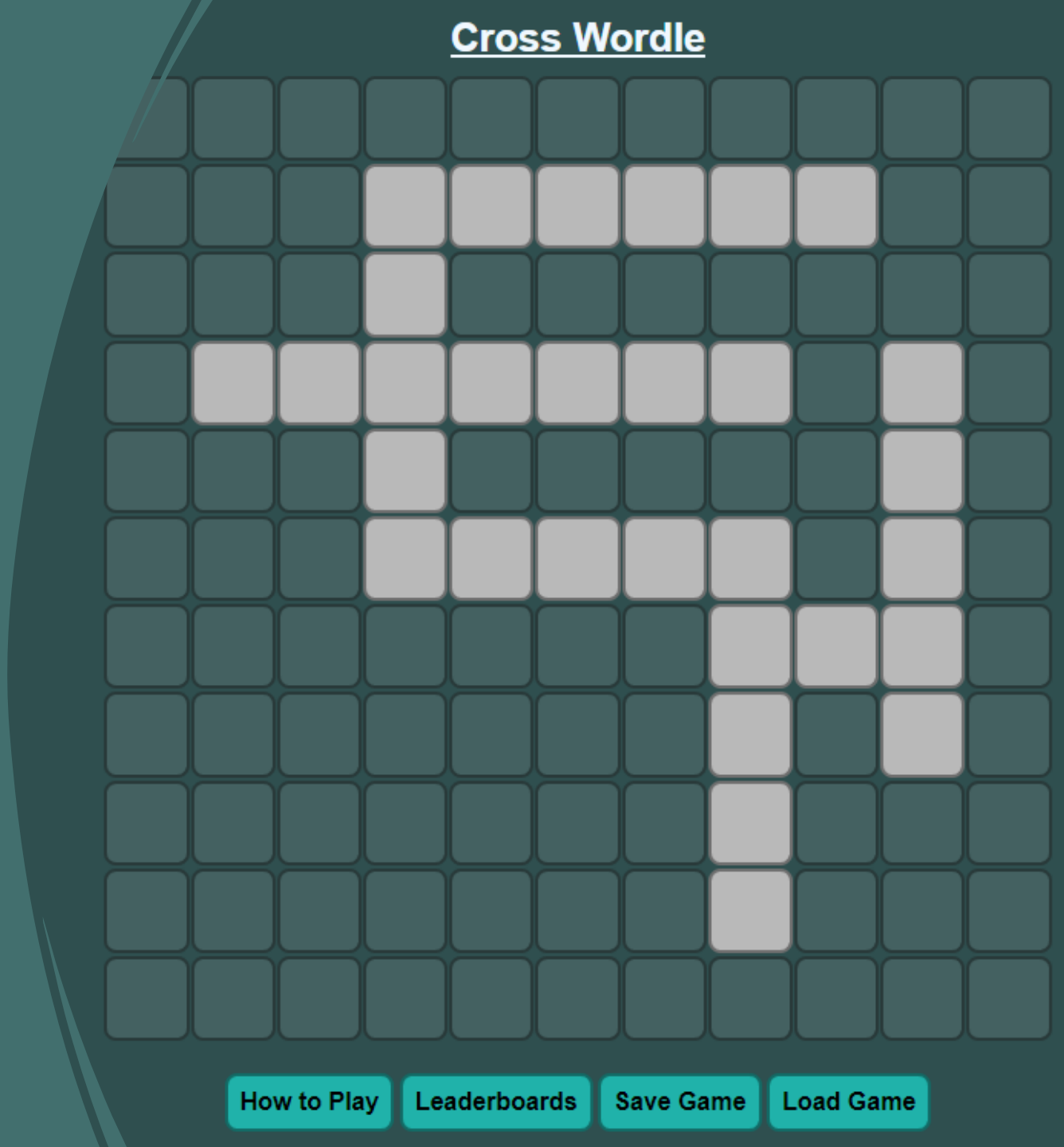

# Cross Wordle

*A Word Game by Matteo Salverio*

# My Game

- The game that I created is called **Cross Wordle**.
- It is like Wordle but takes the gameplay to the second dimension with a full crossword-style puzzle connecting words and ideas, each being a puzzle of their own.




# Programming Language

I chose to make this game using *JavaScript on an HTML page*.

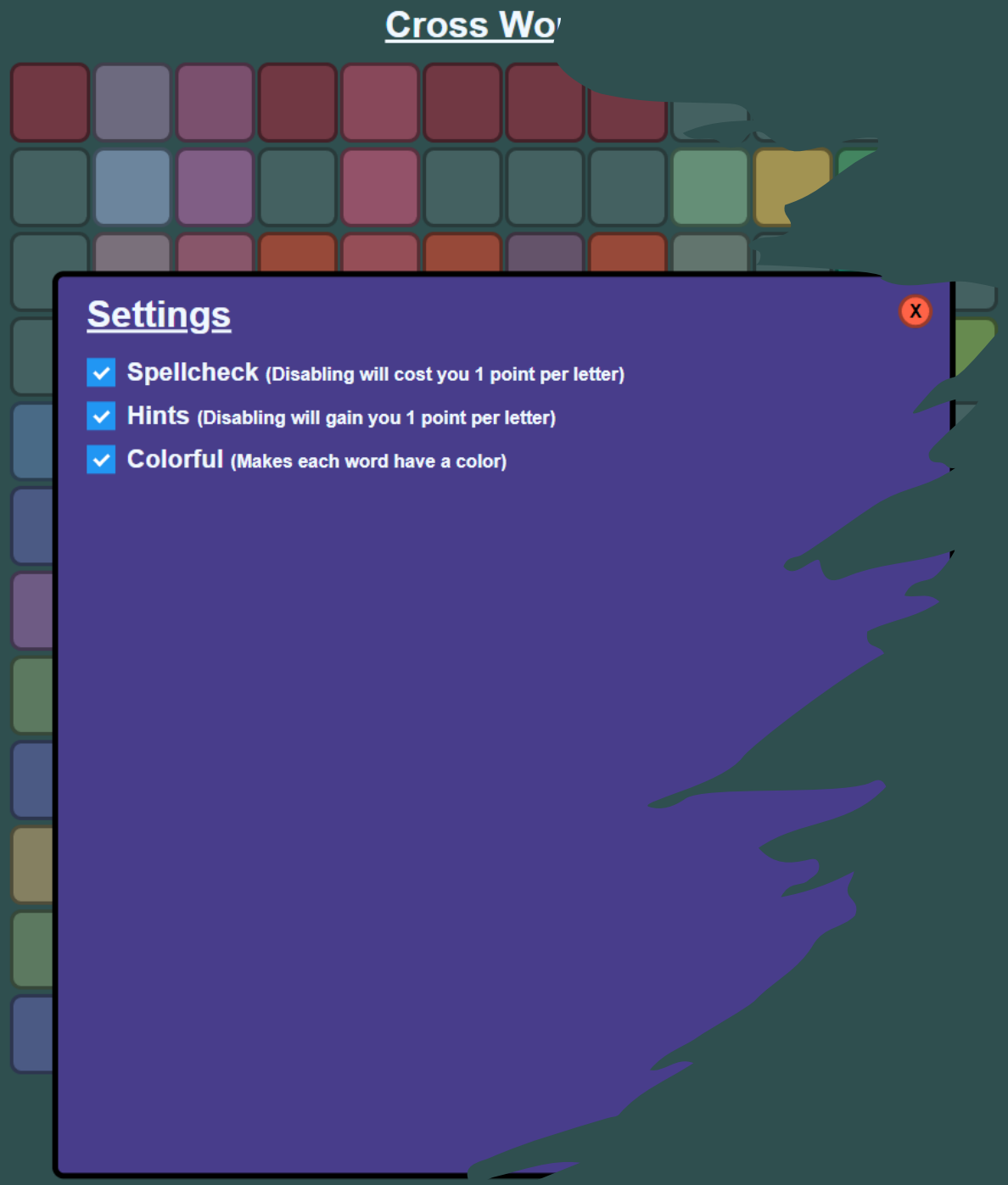
- It can utilize HTML design features.
- Seamless communication between front and back-end.

# Gameplay Summary

- The game is played mostly like a crossword puzzle.
- Takes the simplicity of Wordle but adds some extra challenge to it.

# Getting Started

- Upon opening the site for the first time, the player is greeted with a *"How to Play"* screen that gives a quick explanation of the game.

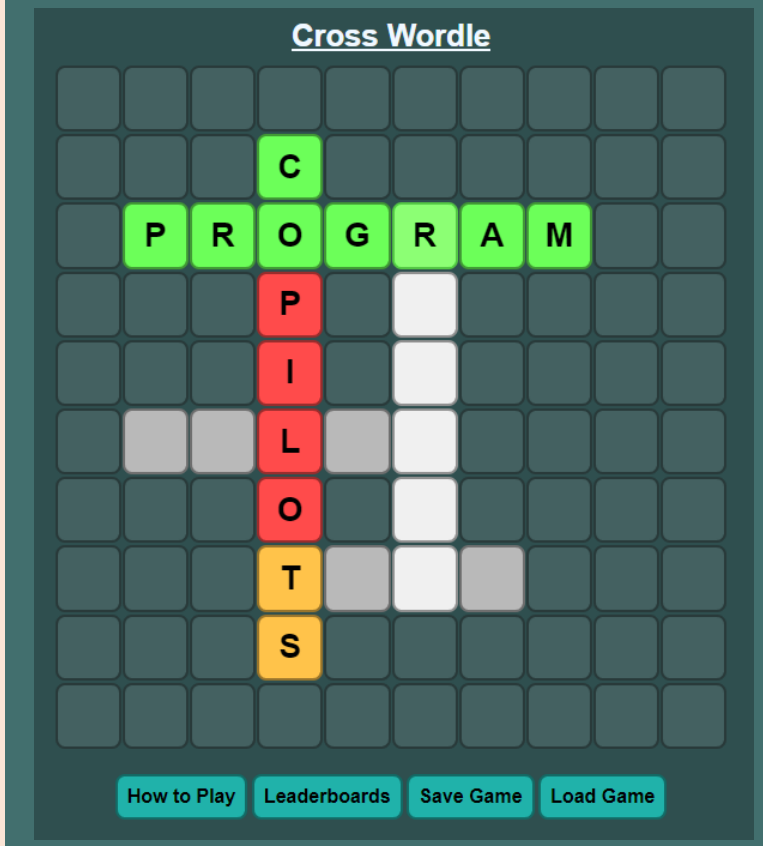


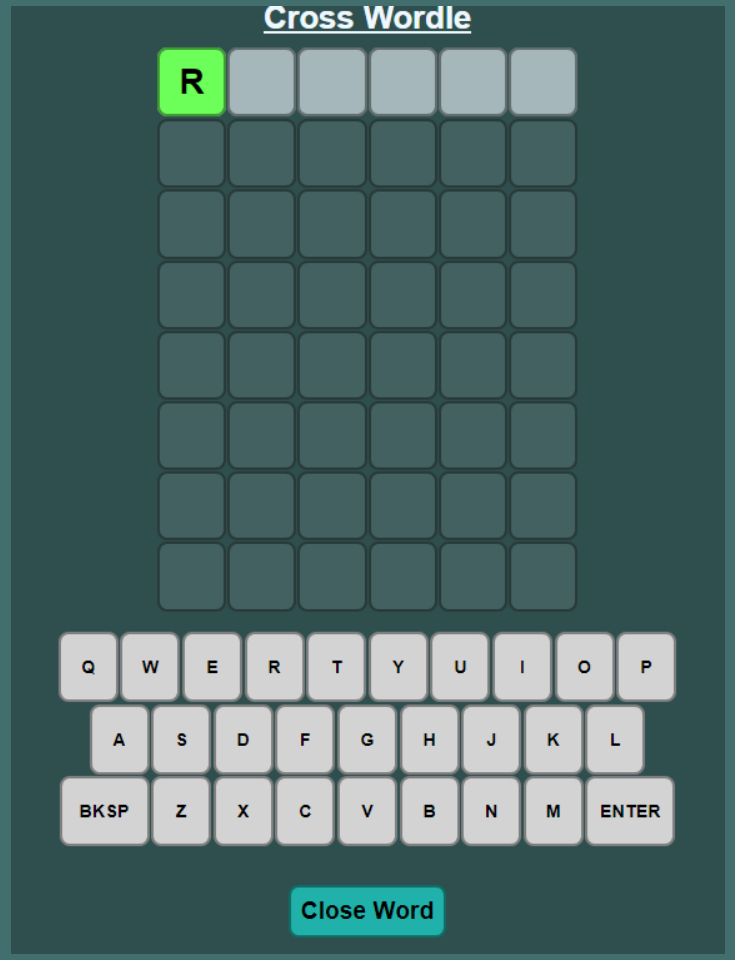

# Settings Menu

- A settings panel is provided.
- Helpful features and options to improve experience.

# Selecting a Word

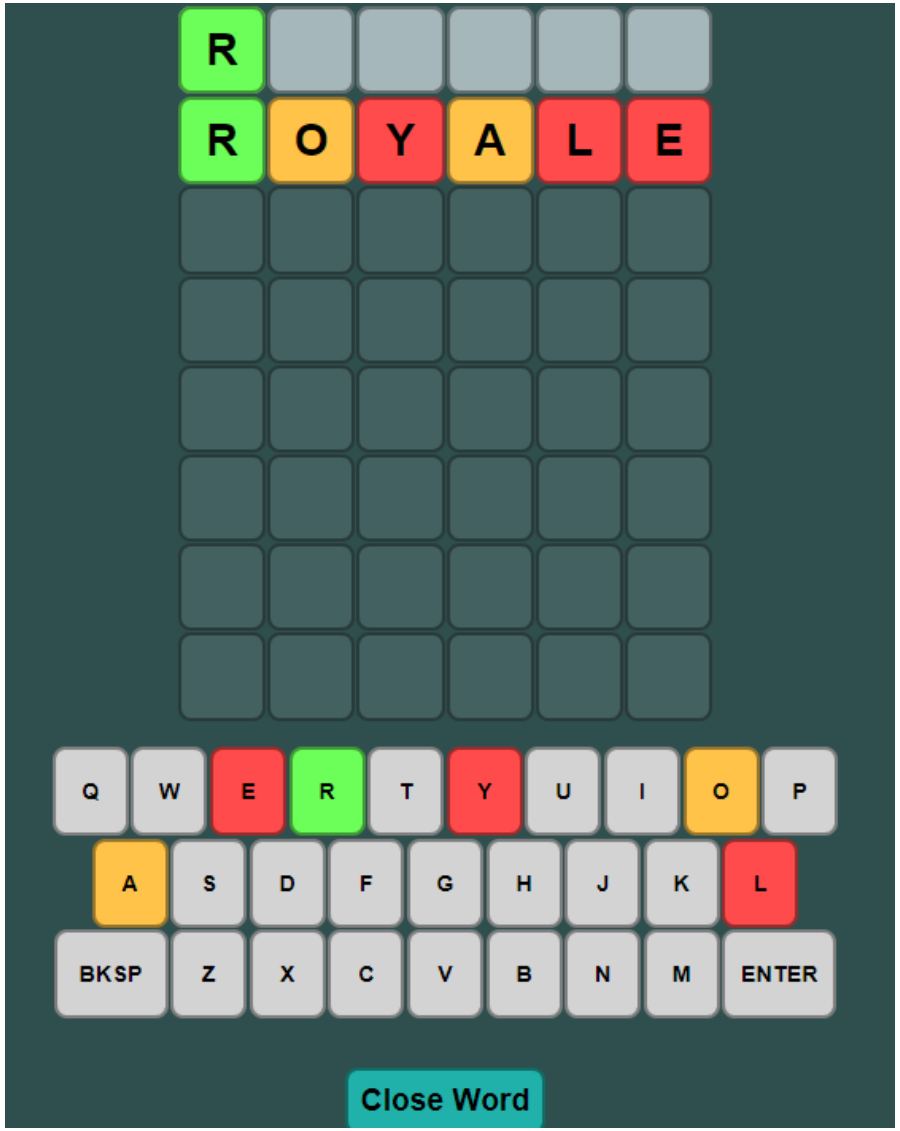
- Hover and click on desired word.
- Wordle-like panel for each word.



# Making a Guess

- The player may type using their physical keyboard, or the on-screen keyboard to enter letters.
- Guess will be spellchecked and submitted as an attempt.
- Green letters are correct, orange letters are in the incorrect place, and red letters are incorrect.
- The player does not have to remember what they have already found.



# Finishing the Game

- Score is calculated as follows:
  - **Green Letters:** 5 points
  - **Orange Letters:** 1 point
  - **Red Letters:** 0 points

- If the player is within the top five players based on their score earned, they will be displayed on the leaderboard.

# Letter Correctness Detection

- Letters are checked for their correctness relative to the word.

- I will now explain how this process works and the code behind it.

# Example 1

# If-Else Tree Example

```
// Returns an array of colors for a guess on a given word
function checkGuess(wordId, guess) {
    let arr = []; // Array of colors
    let word = dataList.words[wordId].word; // Current word

    for (let i = 0; i < word.length; i++) { // Cycle through each letter
        if (word[i] == guess[i]) // If the guess letter matches the word letter
            arr.push(green); // The letter is correct

        else if (word.indexOf(guess[i]) > -1) // If the word contains the guess letter
            arr.push(orange); // The letter is in the word

        else // If the word does not contain the letter
            arr.push(red); // The letter is incorrect
    }

    return arr; // Return the array of colors to be used when displaying the guess
}
```

# Code Snippet

```
// Returns an array of colors for a guess on a given word
function checkGuess(wordId, guess) {
    let arr = []; // Array of colors
    let word = dataList.words[wordId].word; // Current word
    let letters = []; // Count of each letter in the word
    let lettersFound = []; // Cound of how many of each letter the guess contains

    // Fill the arrays from 65 (the ascii value for 'a') to 90 (the ascii value for 'z')
    for (let i = 65; i <= 90; i++) {
        letters[i] = 0; // Set the count to zero
        lettersFound[i] = 0;
    }

    // Count each letter in the word
    for (let i = 0; i < word.length; i++)
        letters[word[i].charCodeAt()]++;

    // Correct letters are prioritized
    for (let i = 0; i < word.length; i++) { // For each letter in the word
        if (guess[i] == word[i]) { // If letters match
            lettersFound[guess[i].charCodeAt()]++; // Add one to the count of that letter
            arr[i] = green; // The letter is correct
        }
    }

    for (let i = 0; i < word.length; i++) { // Count all other letters after
        if (guess[i] == word[i]) // If letters match,
            continue; // Skip this time because they were already checked

        else if (word.indexOf(guess[i]) > -1) { // If the letter is in the word
            lettersFound[guess[i].charCodeAt()]++; // Add one to the count of that letter

            // If the amount of the letter is greater in the guess than in the word:
            if (lettersFound[guess[i].charCodeAt()] > letters[guess[i].charCodeAt()])
                arr[i] = (red); // The letter is incorrect

            else
                arr[i] = (orange); // The letter is in the word somewhere
        }

        else // If the letter is not in the word
            arr[i] = (red); // The letter is incorrect
    }

    return arr; // Return the array of colors to be used when displaying the guess
}
```

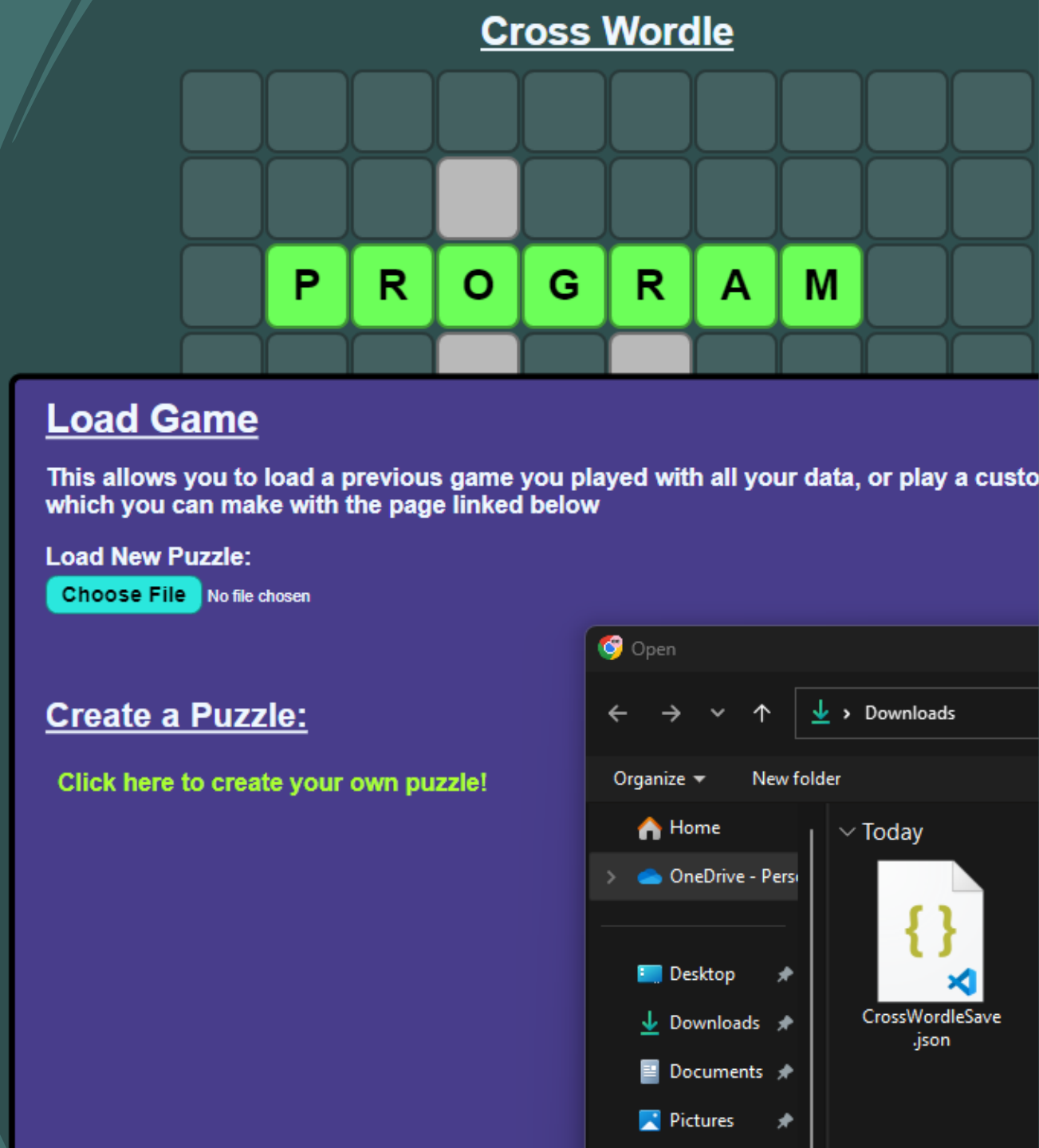
# Example 2

# Art style

- Simplistic, yet colorful design.
- Neutral background color makes the letter spaces really stand out.
- Simple design allows for better focus.

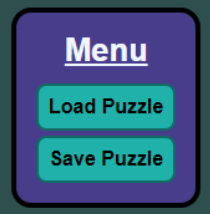
# Saving/Loading Game

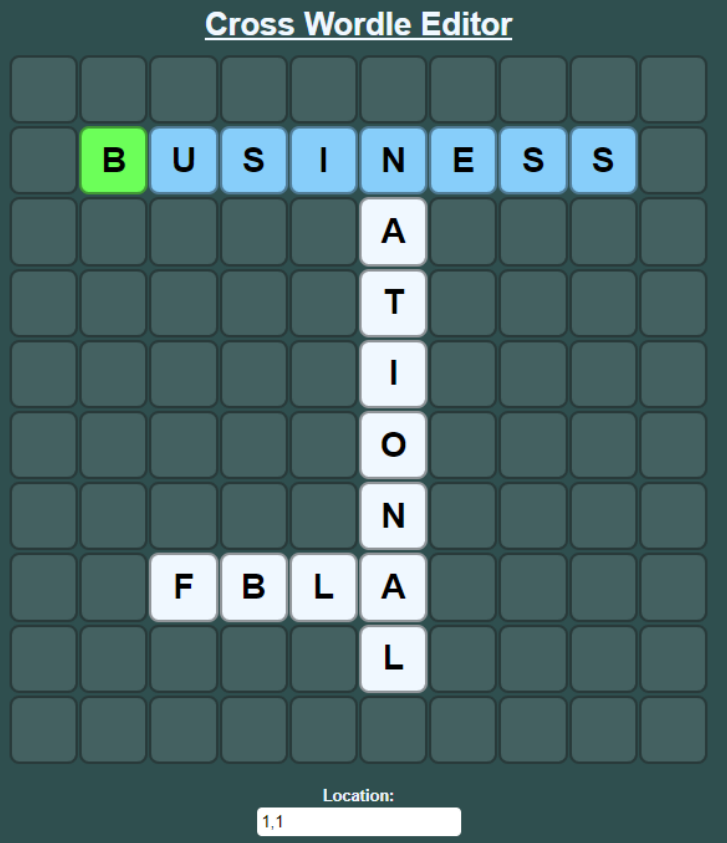
- You can also save your game state as a file, so that you can leave and return to play later.
- This feature also allows you to load custom puzzles, which can be created using the feature I will show next.

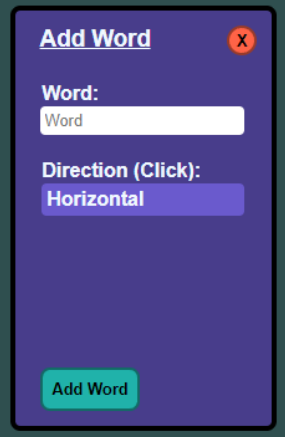


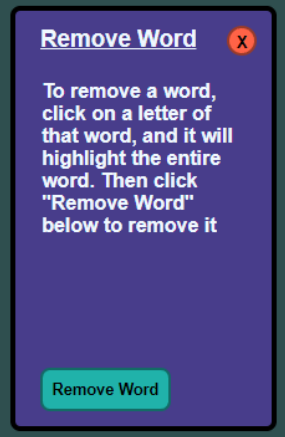
# Creating Custom Puzzles

- Using a tool that I created alongside the game; you can create your own puzzles for others to try out.
- You can also edit existing puzzles, to make your ideal puzzle.

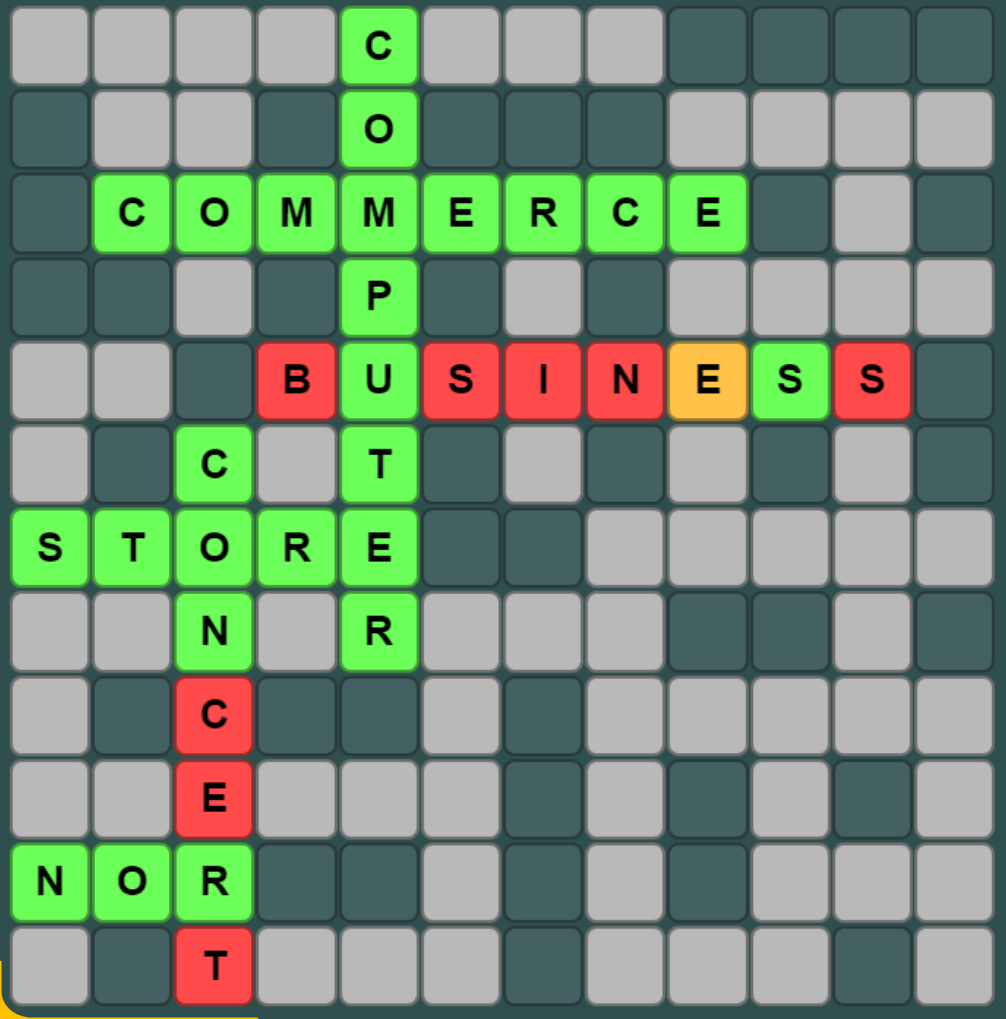



Matteo Salverio - FBLA 2023




# Custom Puzzles Create Scalability

- Near limitless puzzle possibilities.
- You can create truly challenging puzzles with little effort.


# Thank You