Introduction to Computing & Programming

C-CS111

***Fall 2023***

***[Word Search]***

Team Members

|  |  |
| --- | --- |
| Student Name | Student ID |
| Youssef Ashour | 23-101046 |
| Youssef Wael | 23-101183 |
| Adham Ayman | 23-10100 |
| Adam Hussein | 23-101081 |

**Contents**

[Abstract 2](#_Toc182165217)

[1. Introduction 3](#_Toc953214673)

[2. Motivation 3](#_Toc52289555)

[3. System Design 3](#_Toc750919886)

[ System modules and Functions 3](#_Toc441753981)

[ Hierarchy Chart 3](#_Toc775708408)

[4. Description 3](#_Toc399215330)

[5. Team’s Workload Distribution 3](#_Toc612722982)

[6. User Manual 3](#_Toc8928466)

[7. Testing and Validation 4](#_Toc1245160186)

[8. Challenges and Conclusions 4](#_Toc1452115452)

[References 4](#_Toc139903960)

# **Abstract**

This report presents a Python implementation of a word search game using the Tkinter library. The project was motivated by the aim to create an interactive and enjoyable gaming experience while incorporating key programming concepts. The design emphasizes user-friendly interfaces, category and difficulty level selection, and the ability to save and load game progress (still incomplete).

# **Introduction**

Welcome to word search by “Hello World”. It is a simple, yet fun and educational word search game that allows the player to search for various words through different levels. It allows the user to choose how difficult he/she wants the game to be. Written first for CLI then being transferred to a GUI, gives the user a better overall experience. The game offers 3 different difficulties starting up from easy, medium to hard. Our game challenges and entertains people of all ages. Whether you are playing to relax your mind or trying to improve your vocabulary, this game is the one to go with.

Through a user friendly and inbuilt interface, players can easily navigate through a grid full of words from specific lists of words. They can highlight letters in all directions, vertically, horizontally, diagonally and they can even highlight backwards.

Hello World’s word search game is not just a game, it’s a tool for learning and a source of entertainment. It’s a beautiful blend of joy and education together. Try the game yourself for an unforgettable experience.

# **Motivation**

The motivation behind this project stems from a desire to blend entertainment with educational elements. Word search games are not only enjoyable but also serve as a platform for enhancing vocabulary and cognitive abilities. By implementing a customizable and feature-rich word search game, we aim to provide users with an immersive and educational gaming experience.

# **System Design**

**word\_choice():**

* This function initializes the main Tkinter application loop where users enter their name, choose a category, select difficulty, and start the game.
* It creates input entries for the player's name and category choice, along with a menu for difficulty selection.
* Additionally, it allows users to load saved games (if available) through a menu button.

**create\_input\_entry(window, label\_text):**

* This function creates a label and an entry field under it in the specified window with the given label text.

**mode(window, difficulty):**

* This function starts the game based on the chosen difficulty level.
* The value passed for the difficulty parameter is used to find the suitable components for the chosen mode.
* It retrieves player inputs for name and category choice, validates them and reads words from the selected category file.
* The create\_word\_search function is called at its end to create the game grid based on the arguments given to this function.

**read\_words\_from\_file(file\_path):**

* This function reads lines from a text file and returns them as a list of strings.

**create\_word\_search(rows, cols, words\_to\_find, hint=None):**

* This function creates the main game window.
* It initializes a 2D list of empty labels for the word letters to be placed in.
* The first submission timer and total score are also initialized.
* The add\_words\_to\_grid, fill\_empty\_spaces and display\_grid functions are called at the end.

**add\_words\_to\_grid(game\_grid, words):**

* This function places words randomly in the game grid in different directions (horizontal, vertical, diagonal and reverse).
* It first chooses a random direction for the word to be placed in.
* Then, it calculates the exact coordinates (row,col) for each letter of the word.
* Additionally, it ensures that words don't overlap or are placed outside of the grid’s boundaries.

**fill\_empty\_spaces(game\_grid):**

* This function fills empty spaces in the grid with random letters.
* All labels that have an underscore as their text will be filled with a random character.

**display\_grid(game\_grid):**

* This function displays the grid of labels in the game window.
* It uses Tkinter's grid layout to position the labels in rows and columns.
* A button bind (left click) is also added to all labels with the on\_cell\_click command attached.

**on\_cell\_click(label):**

* This function changes the background color of a label when it's respective button binding is clicked.
* The program checks for the current background color of the label; if white, turn it yellow and vice versa.

**checkWords(game\_grid):**

* Command for submit button
* This function checks the selected letters in the grid to determine if they form a word.
* All yellow labels have their texts added to a set, which will be a compared to the list of sets retaining to the hidden words.
* If any of the sets match, point will be calculated and added to the score.
* Correct submissions are highlighted with different colors for a better visual experience
* Incorrect submissions will reduce the player’s remaining tries by one.
* Game is won when the number of words found matches the length of the list of word sets.
* Game is lost and terminated when player’s tries reach zero.

**on\_escape\_key(window):**

* This function handles the event when the Escape key is pressed.
* It creates a pause menu window allowing the player to save and exit or exit without saving, each corresponding to a different button.

**handle\_exit(primary, secondary, save):**

* This function handles the process of closing the game window.
* It closes both the primary and secondary windows based on the user's choice to save or not.

**save\_game\_to\_json(username):**

* This function saves game data to a JSON file using the provided username as the filename.

**get\_existing\_saves():**

* This function retrieves a list of existing saved game files based on a specific location, ending and extension.
* All file names will be available in the load menu button on the setup window.

# **Description**

The word search game is built upon the Tkinter library, a powerful and versatile tool for creating graphical user interfaces in Python. Tkinter provides a framework for constructing interactive and visually appealing applications using widgets, events, and various layout managers. In this project, Tkinter serves as the foundation for designing the game's user interface, handling user inputs, and facilitating the overall gaming experience.

The game logic revolves around the generation of a word search grid, category and difficulty level selection, and the implementation of scoring mechanisms. Here's an overview of the key components:

Category and Difficulty Level Selection

- The game offers users the choice of selecting a category from a predefined set (Animals, Capitals and Countries, Fruits and Vegetables, Sports).

- Difficulty levels (Easy, Medium, Hard) determine the number of words, grid size, and maximum allowed attempts, providing a customizable gaming experience.

Word Placement and Grid Generation

- The game dynamically reads word lists from text files corresponding to the selected category.

- Words are randomly placed on the grid, considering orientations such as horizontal, vertical, and diagonal.

- The remaining grid cells are filled with random letters to create a visually appealing and challenging game board.

Scoring Mechanism

- Players earn points based on the length of each word found and the speed at which they locate them.

- A time bonus is applied for finding words quickly, incentivizing players to complete the game efficiently.

- The total score accumulates as players successfully identify words within the grid.

User Interaction and Visual Elements

- The graphical user interface features interactive elements, including input entries for name and category selection.

- The game grid is displayed as a set of labeled Tkinter labels, each representing a cell in the word search grid.

- Users can select letters by clicking on the corresponding labels, and the selected letters are visually highlighted.

Save and Load Functionality

- The game allows users to save their progress during an ongoing session and load it later for continued play.

- Saved games are stored as JSON files, incorporating player names and found word counts for comprehensive progress tracking.

# **User Manual**

*Instructions to Run the Game*

* Ensure you have Python installed on your machine.
* Run the code using a Python interpreter.
* A setup window will appear, prompting you to enter your name and choose a category and difficulty level.
* Click on "New Game" to start the game.
* Use your mouse to select words in the grid by clicking on the letters. Click "SUBMIT" to check your selected words.
* The game provides feedback on correct and incorrect answers and keeps track of your score.
* You can save your progress and exit the game by pressing the ESC key.

*Features*

* Category Selection: Choose from categories such as Animals, Capitals and Countries, Fruits and Vegetables, and Sports.
* Difficulty Levels: Select from Easy, Medium, and Hard difficulty levels, each with a different number of words and grid size.
* Save Game: Save your progress with your username in a JSON file.
* Load Game: Load your saved game and continue playing (work in progress).
* Scoring: Earn points based on the length of words found and the time taken to complete the game.
* Responsive UI: The game window is responsive and adjusts to the screen size.

*How to Play*

* Enter your name in the input field.
* Choose a category by entering the corresponding number (1 for Animals, 2 for Capitals and Countries, 3 for Fruits and Vegetables, 4 for Sports).
* Select a difficulty level (Easy, Medium, or Hard).
* Click "New Game" to start the word search game.
* Use the mouse to select letters in the grid and form words.
* Click "SUBMIT" to check your selected words.
* The game provides feedback on correct and incorrect answers.
* Save your progress and exit the game by pressing the ESC key.

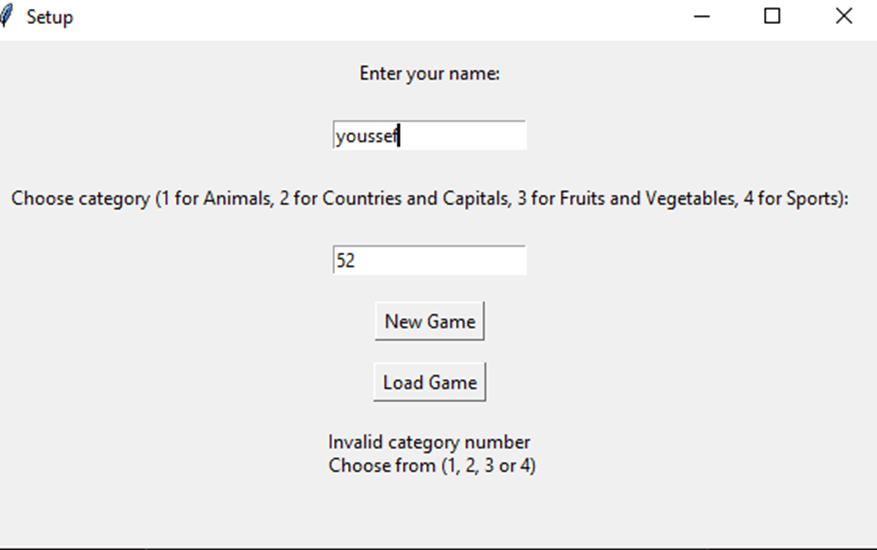
***\*\*NOTE\*\****

Loading is still not fully implemented.

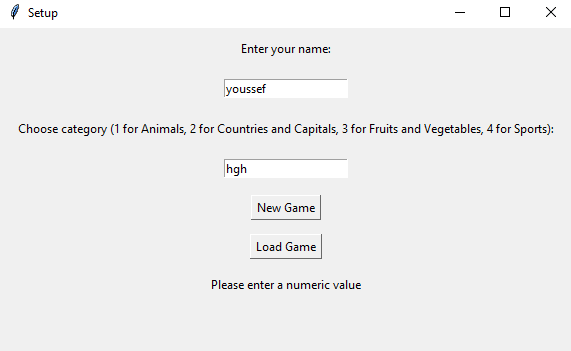
# **Testing and Validation**

**User Input Validation:**

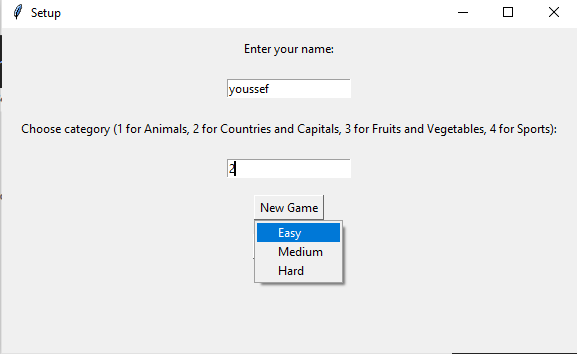
*User entered a category number choice that is not within the range:*



*User entered a non-numeric value for the category choice:*

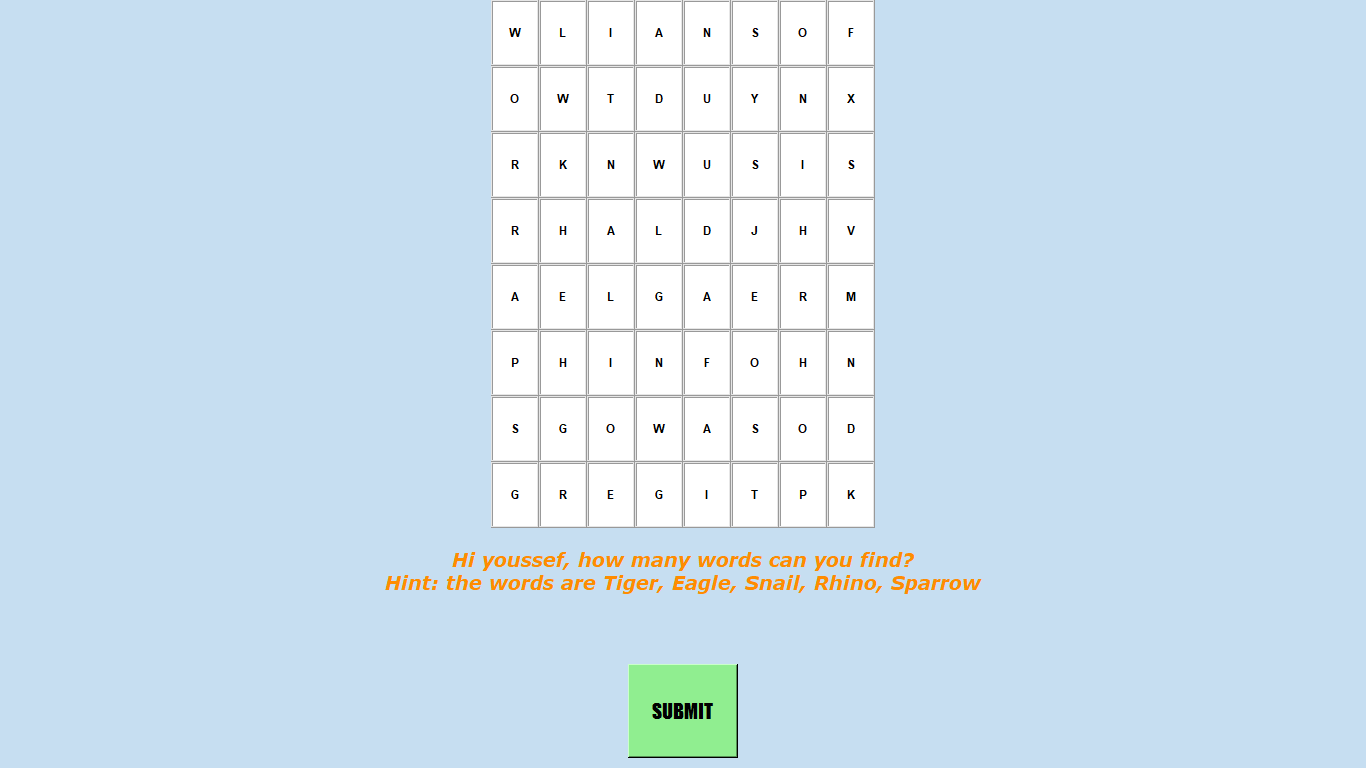


*User entered name and valid category choice:*

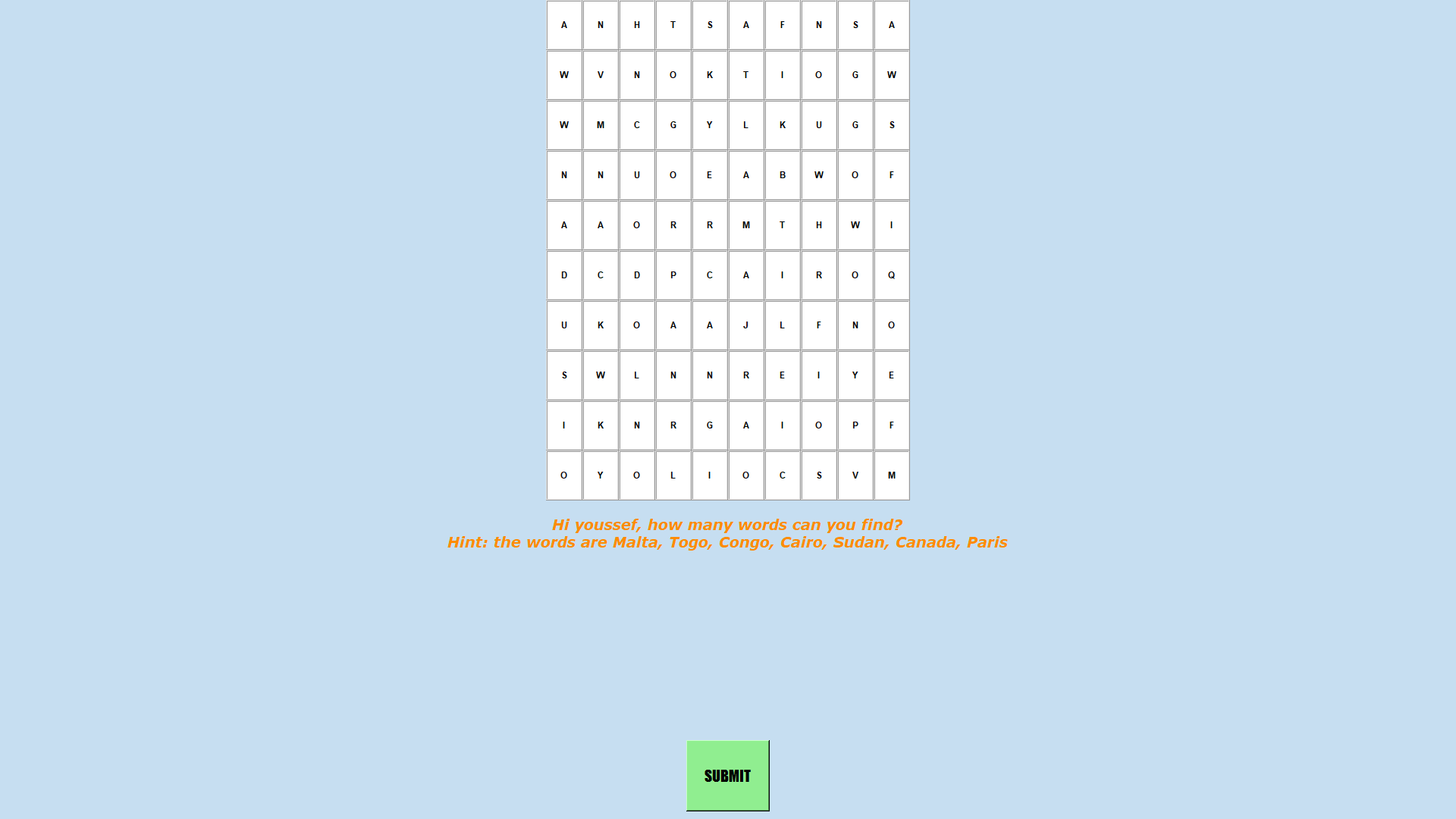


**Difficulties & Grid:**

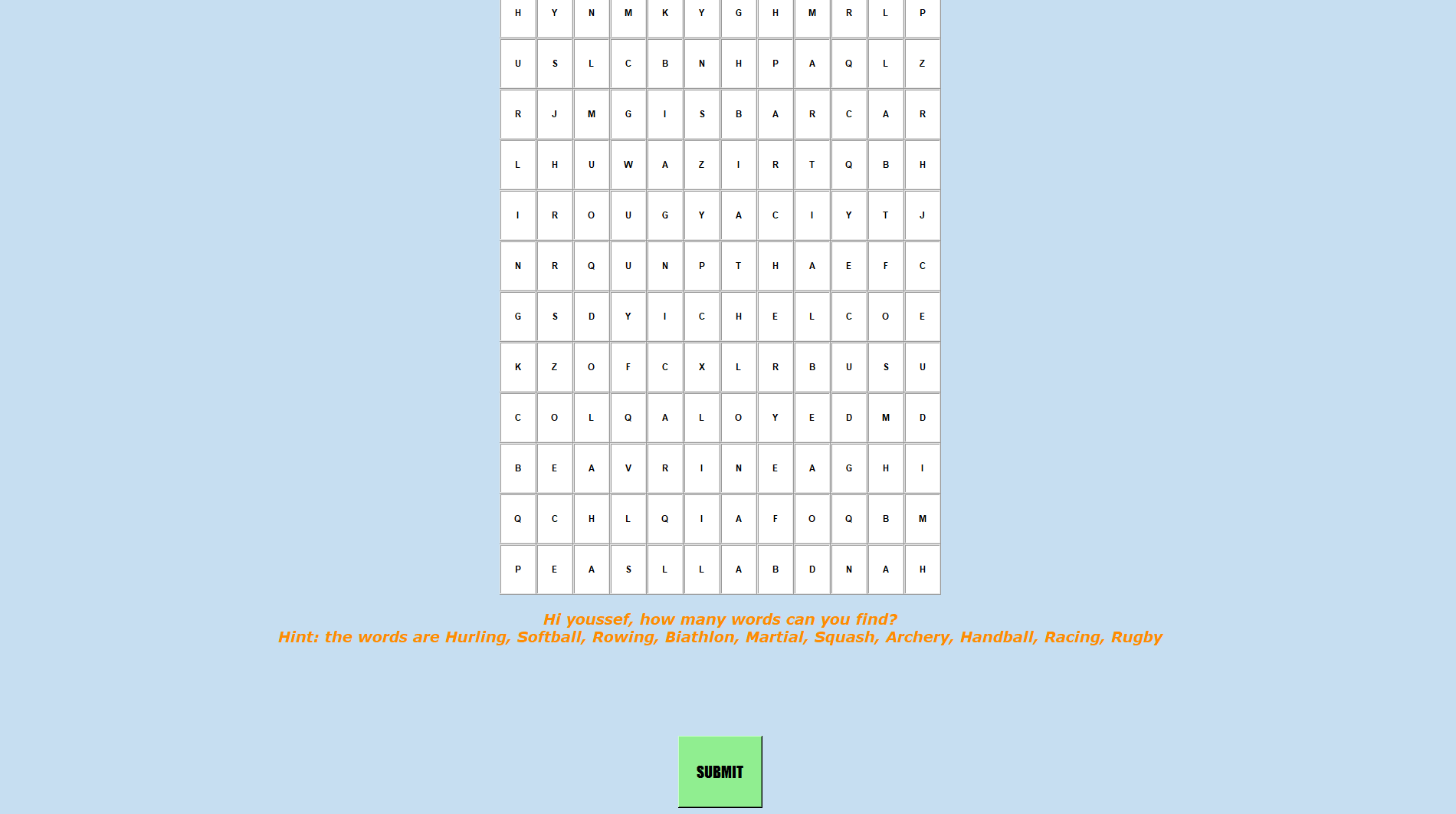
*Easy difficulty window & its 8x8 grid and five hidden words:*



*Medium difficulty window & its 10x10 grid and seven hidden words:*

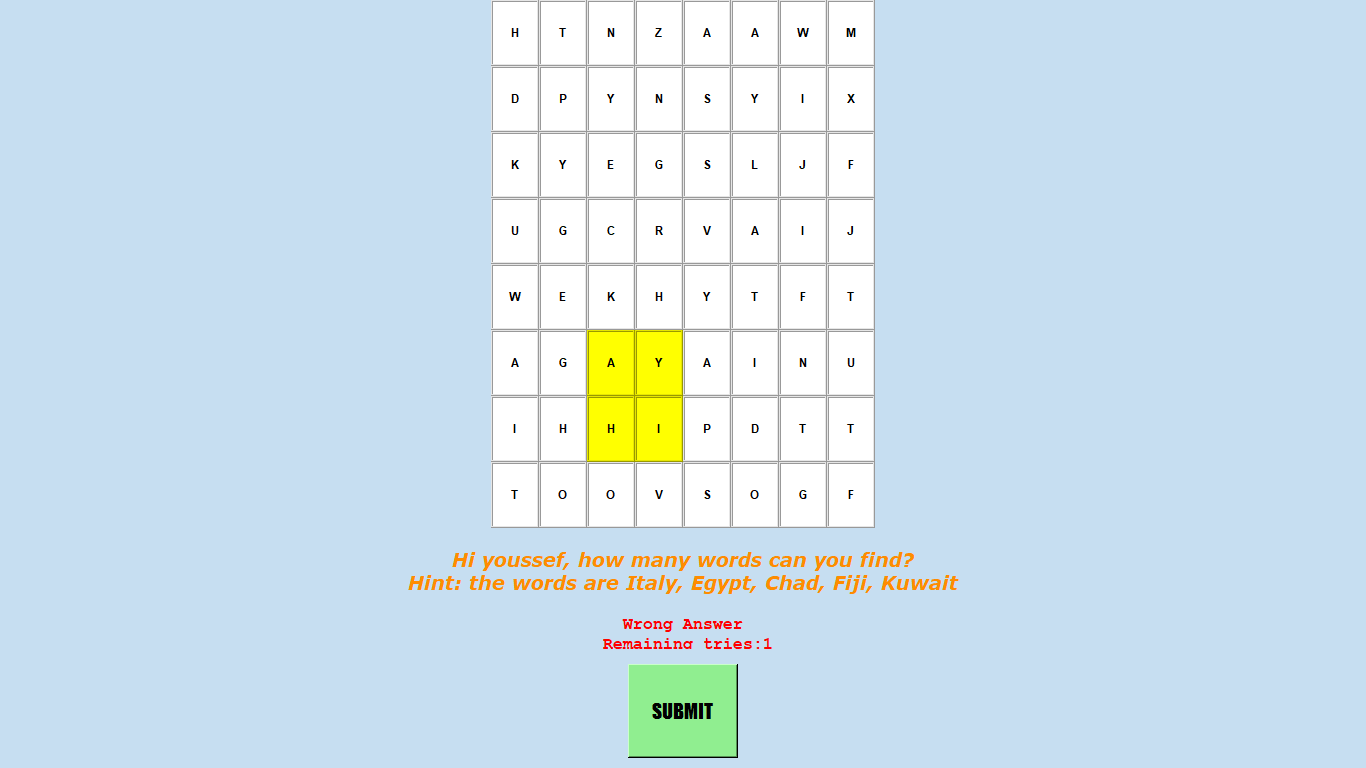


*Hard difficulty window & its 12x12 grid and ten hidden words:*

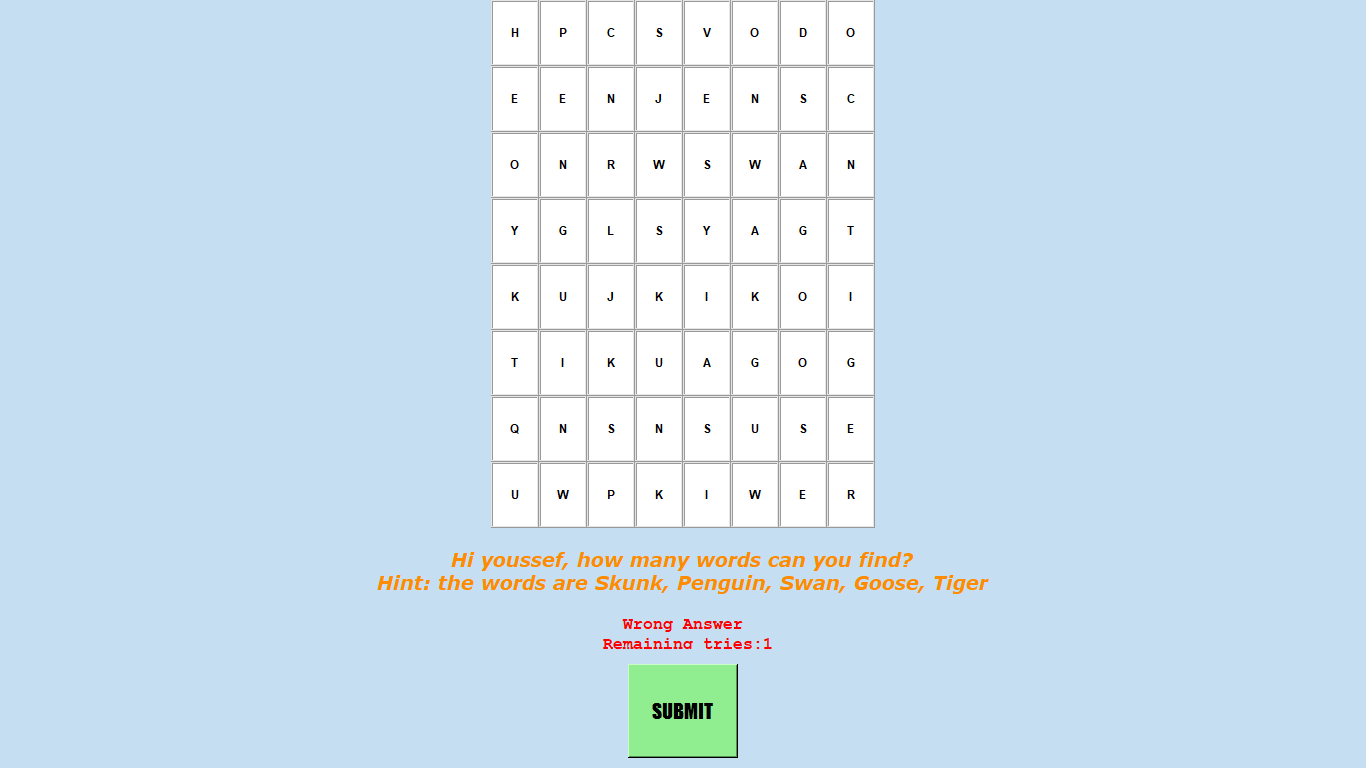


**Gameplay:**

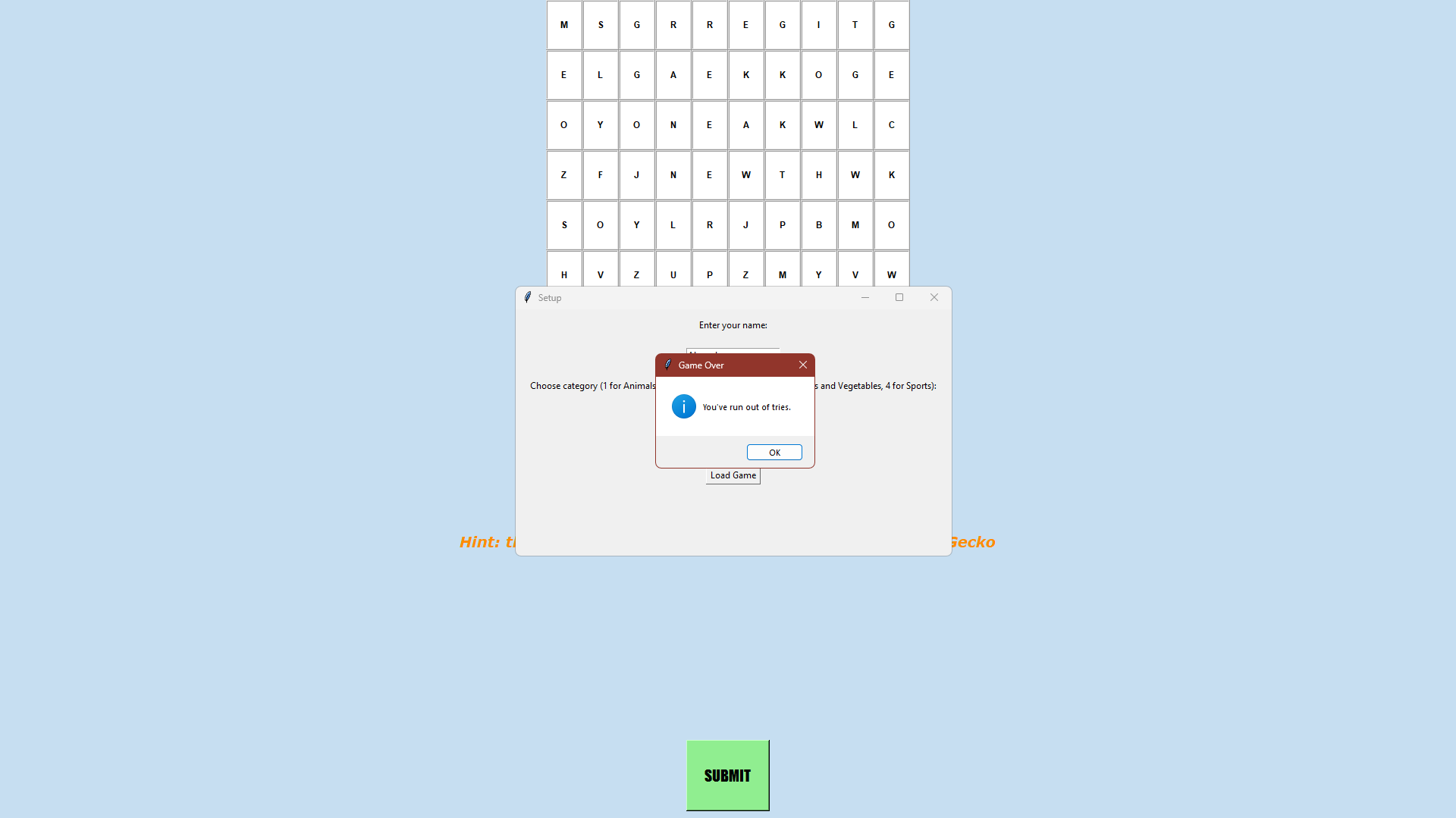
*Player submits wrong answer:*



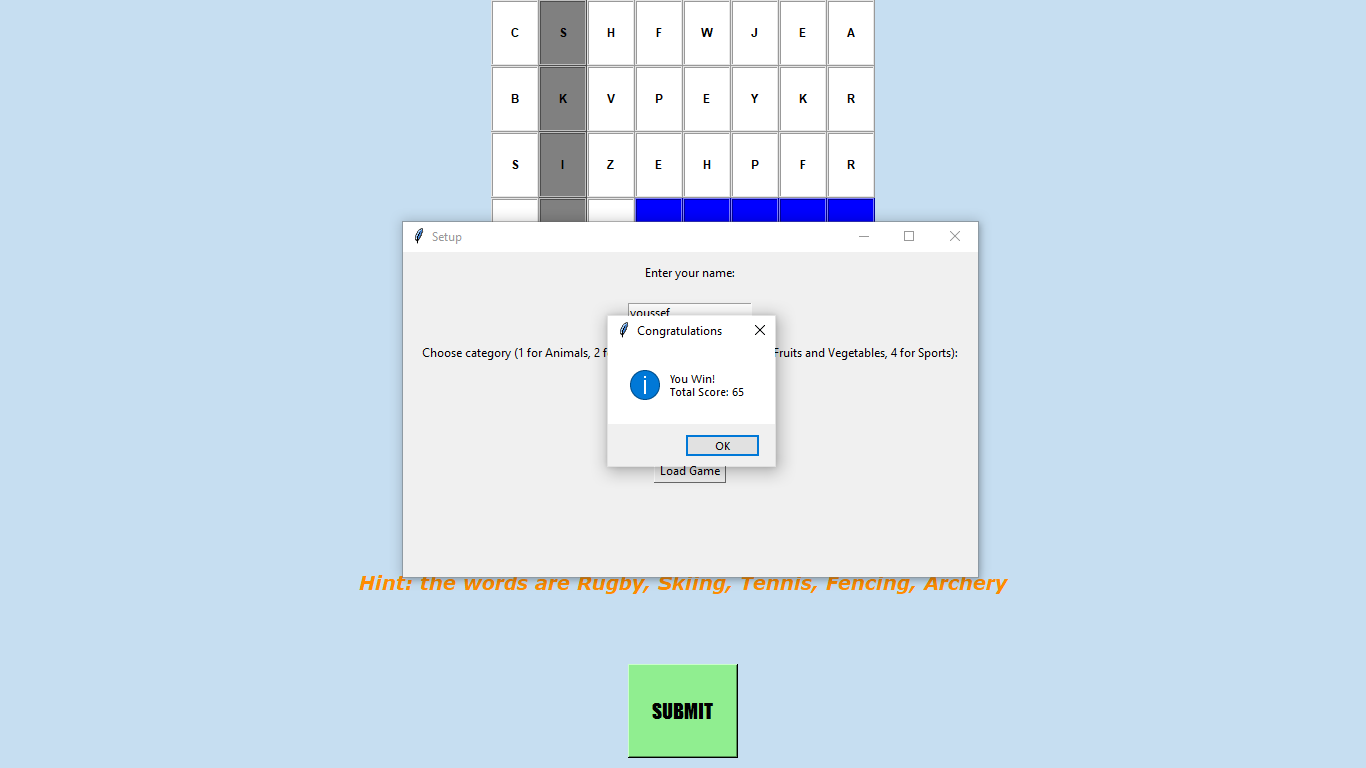
*Player submits empty answer:*



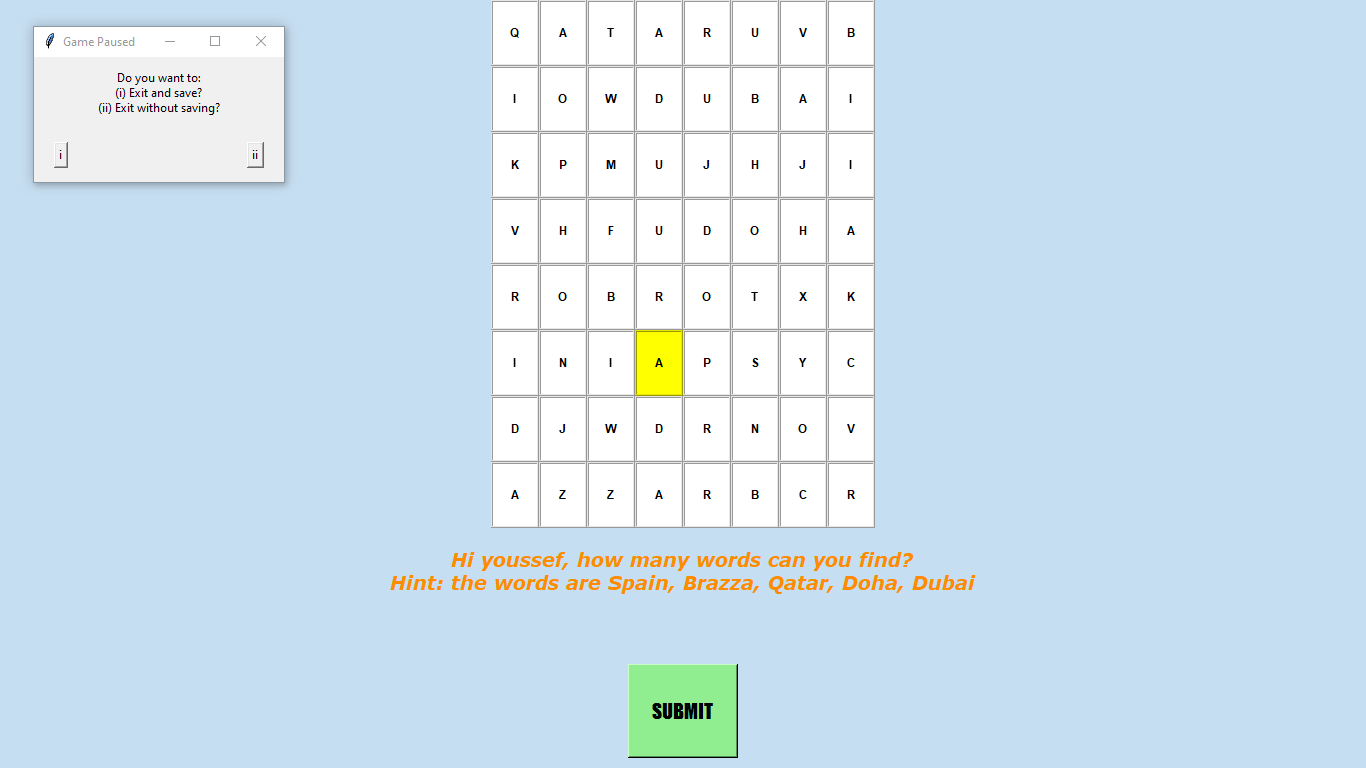
*Player runs out of tries:*



*Player finds all words and wins (scores appears in message box):*

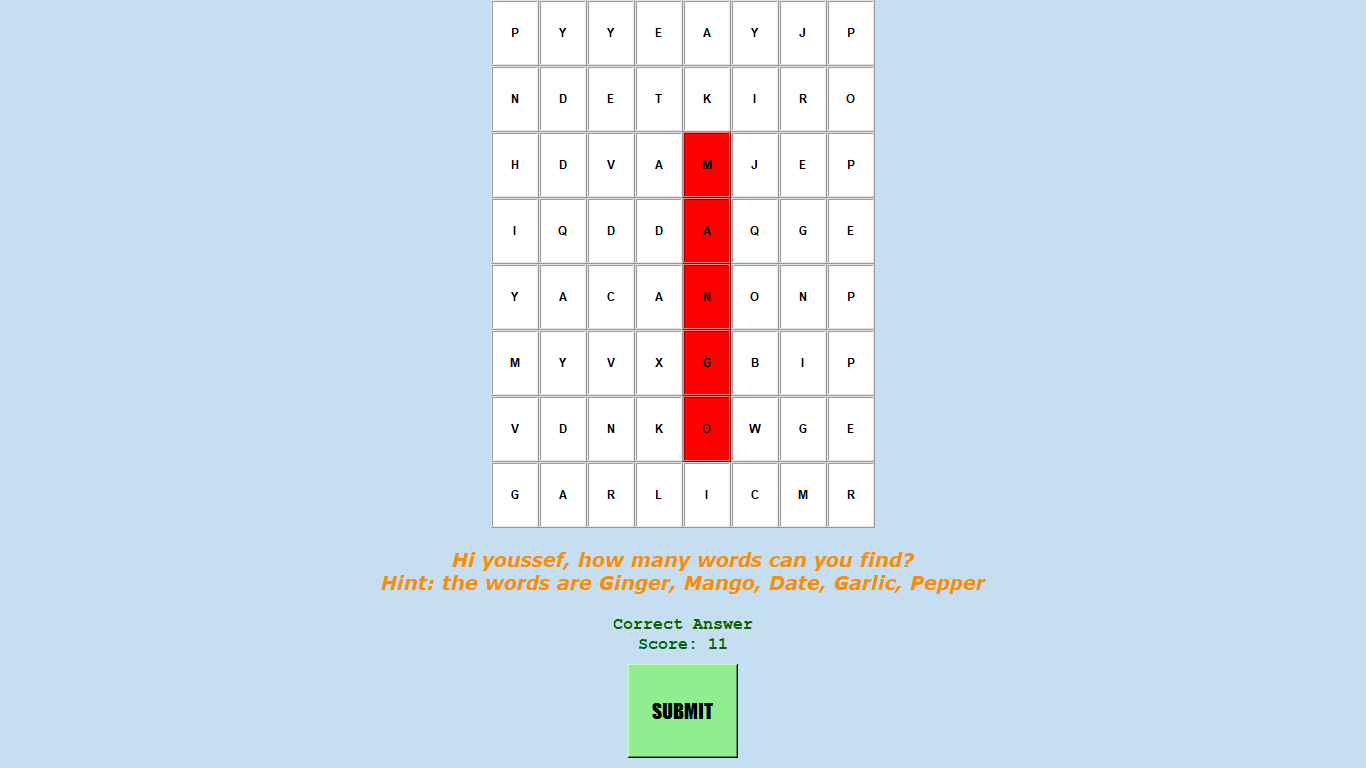


*Pause screen:*

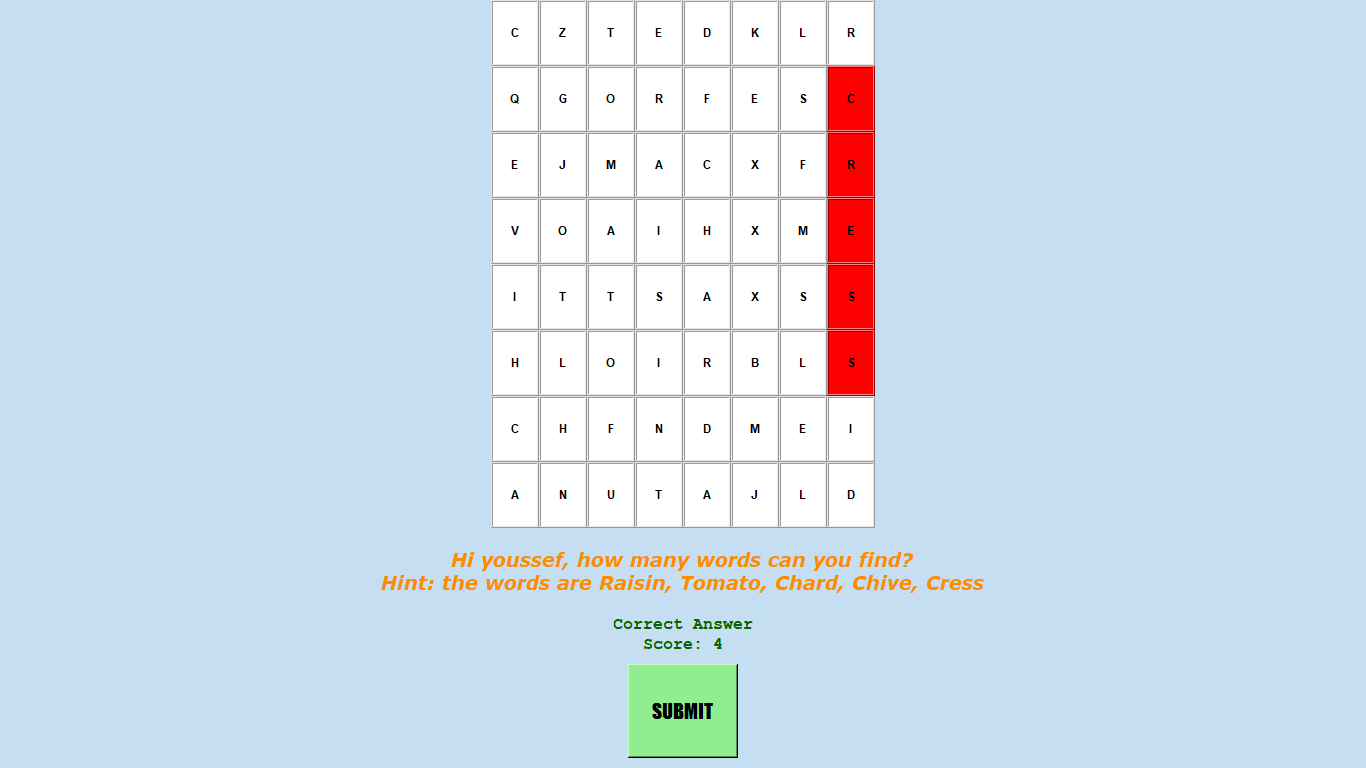


**Scoring Mechanism:**

*Player submits correct answer in less than 15 seconds (bonus is acquired):*

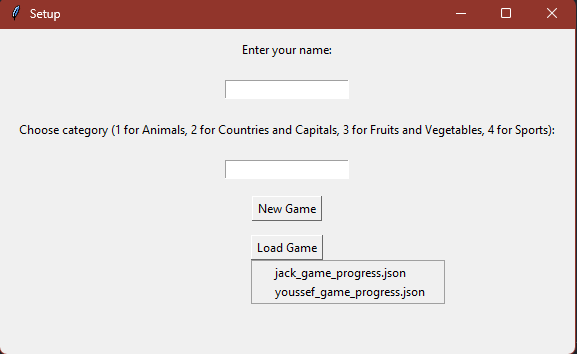


*Player submits correct answer in more time than 15 seconds (no bonus is given):*

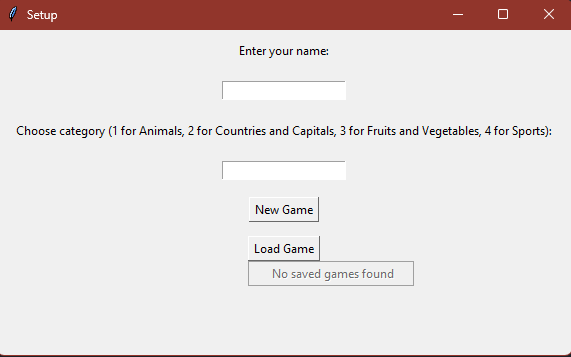


**Save and loading buttons:**

*Load button with existing files:*



*Load button with no existing files:*



# **Challenges and Conclusions**

Tkinter Complexity: Tkinter is a versatile but complex library for GUI development. Managing different widgets, layouts, and event handling was challenging, especially for someone new to GUI programming.

Grid Generation and Display: Generating a random letter grid, arranging words within it, and displaying it was complex. The process of thinking about the logic behind word placement and arrangement was exceptionally difficult.

Event Handling: Implementing click events for letter selection, tracking user input, and responding to those events was also tricky. We were unfamiliar with bindings and buttons commands.

File Handling: Reading word lists from text files and handling file paths correctly posed some challenges. Proper error handling was necessary to ensure the process was executed perfectly.

Data Management: Managing global variables, such as counters, across different functions and modules was not easy. Maintaining the state of the game and ensuring data consistency are important.

Save and Load Mechanism: Implementing a reliable save and load mechanism was the hardest part. We chose to implement JSON, but we unfortunately failed to produce complete loading and saving functionality. Saving is partially working as the player can have a JSON file with their name if they choose, but further work is needed to load the game from this file.

Future Improvements:

As mentioned above, a core part is still not working in the program: loading. All future efforts will be spent to find the best way to add this feature. We are still trying to find out the necessary data that needs to be saved from the player’s game state onto the JSON file. Once figured out, we will start working on the loading function that loads the saved data onto the game window for the player to continue playing on an identical grid.

Moreover, a key oversight in the code is the overuse of global variables. The team will try to minimize their usage by passing more values as parameters to their respective functions.

# **References**

<https://theswissbay.ch/pdf/Gentoomen%20Library/Programming/Python/Starting%20Out%20with%20Python%20%282009%29.pdf>

<https://stackoverflow.com>

<https://www.youtube.com/playlist?list=PLCC34OHNcOtoC6GglhF3ncJ5rLwQrLGnV>

<https://www.youtube.com/watch?v=YXPyB4XeYLA&t=1433s&pp=ygUUdGtpbnRlciBmcmVlY29kZWNhbXA%3D>

<https://youtu.be/KGw7ppL1ntc?si=PuGyvLWDz-piM0Yh>

<https://www.youtube.com/watch?v=nnG8fN-2xlI&t=49s&pp=ygUYd29yZCBzZWFyY2ggam9obiBoYW1tb25k>

<https://youtu.be/XKHEtdqhLK8?si=immffs6hUWonel9u>