DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING
BACHELORS IN COMPUTER SYSTEMS ENGINEERING
Course Code: CS-115
Course Title: Computer Programming

Complex Engineering Problem

FE Batch 2022, Fall Semester 2022
Grading Rubric
TERM PROJECT

Group Members:

Student No.	Name	Roll No.
S1	**Sardar Abdul Moiz Khan**	**CS-22049**
S2	**Syed Muhammad Mujtaba**	**CS-22045**
S3		

CRITERIA AND SCALES				**Marks Obtained**		
				S1	**S2**	**S3**
Criterion 1: Does the application meet the desired specifications and produce the desired outputs? (CPA-1, CPA-3) **[8 marks]**						
1	2	3	4			
The application does not meet the desired specifications and is producing incorrect outputs.	The application partially meets the desired specifications and is producing incorrect or partially correct outputs.	The application meets the desired specifications but is producing incorrect or partially correct outputs.	The application meets all the desired specifications and is producing correct outputs.			
Criterion 2: How well is the code organization? **[2 marks]**						
1	2	3	4			
The code is poorly organized and very difficult to read.	The code is readable only to someone who knows what it is supposed to be doing.	Some part of the code is well organized, while some part is difficult to follow.	The code is well organized and very easy to follow.			
Criterion 3: How friendly is the application interface? (CPA-1, CPA-3) **[2 marks]**						
1	2	3	4			
The application interface is difficult to understand and use.	The application interface is easy to understand and but not that comfortable to use.	The application interface is very easy to understand and use.	The application interface is very interesting/ innovative and easy to understand and use.			
Criterion 4: How does the student performed individually and as a team member? (CPA-2, CPA-3) **[4 marks]**						
1	2	3	4			
The student did not work on the assigned task.	The student worked on the assigned task, and accomplished goals partially.	The student worked on the assigned task, and accomplished goals satisfactorily.	The student worked on the assigned task, and accomplished goals beyond expectations.			
Criterion 5: Does the report adhere to the given format and requirements? **[4 marks]**						
1	2	3	4			
The report does not contain the required information and is formatted poorly.	The report contains the required information only partially but is formatted well.	The report contains all the required information but is formatted poorly.	The report contains all the required information and completely adheres to the given format.			
			Total Marks:			

Teacher's Signature

COMPLEX ENGINEERING PROBLEM : HANGMAN GAME

PROBLEM DESCRIPTION :

We were asked by our course teacher to make hangman which should contain user interface as well as admin interface.

Develop a software application in Python using the basic concepts and structures of computer programming.

Your application will allow the user to play the classic word game Hangman against the computer. You application maintains two interfaces: one for the player and one for the administrator, as shown in the following flow diagram.

For the game, the computer picks a word, randomly form a list of available words, and the player tries to guess letters in the word. The player is given a certain number of guesses at the beginning. The game is interactive; as the player inputs his/her guess, the computer either:

- reveals the letter if it exists in the secret word
- penalize the user and updates the number of guesses remaining.

The game ends when either the user guesses the secret word, or the user runs out of guesses.

DISTINGUISHING FEATURES OF OUR PROJECT :

Using GUI and its functions is the most prominent feature of our project. An on-screen keyboard that will allow the user to select the letters and if the letter is correct, it will turn green. The password entered by the admin will be shown as "*" (asterisk). An easy-to-use Admin mode that will allow the admin to add multiple words and if the word entered by the admin is already in the file(words.txt), it will not be written and after adding all the words a temporary Label will appear that will show the success message.

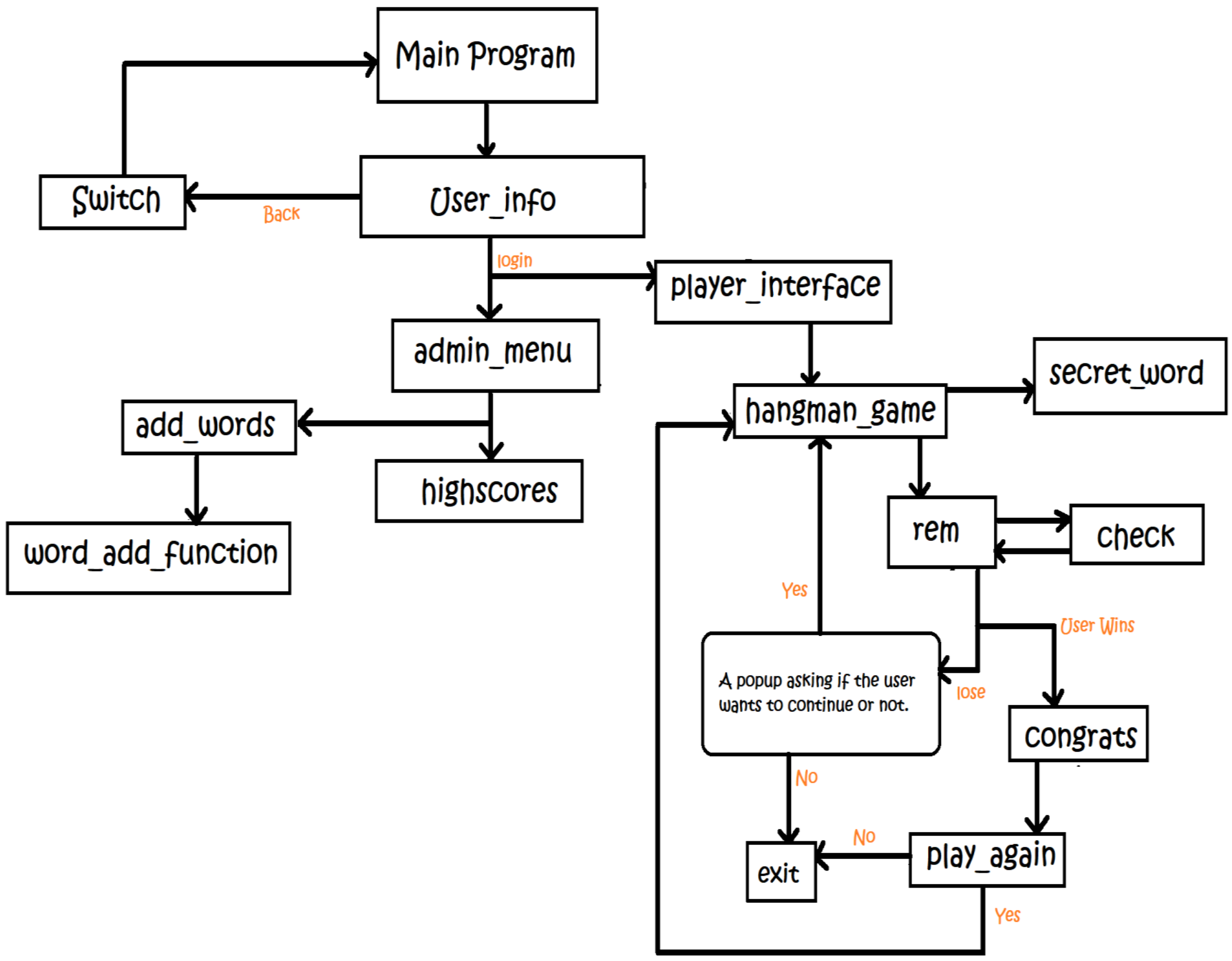
FLOW OF OUR PROJECT :

FUNCTIONS AND THEIR WORKING:

- **Main Program:** Actually, it is not a function instead this shows the flow of execution for the program. Here a window is created, in which some Labels and Buttons are placed. It also contains the Radio Buttons for the Player or Admin's interface and a Continue Button.
- **user_info :** This Function is called when the user has selected the interface and pressed the continue button. This Function disables the interface selection and has If-else condition to determine the User's selection of the interface. Then it creates and displays the Information asking widgets (Labels and Entries), with two buttons Login and Back.
- **Switch :** If the User has selected the wrong interface, then he can click on the Back button to go to the Main Program for re-selecting the interface. This function also destroys the widgets created by user_info function.
- **admin_menu :** This Function is called when the User has selected the admin mode and has pressed the Login Button. This Function checks if the user has entered the correct name and password for the administrator access. Then, it creates the Layout of the admin mode, displaying 2 Buttons named ADD More Words and Reset Scores. If the user had entered the wrong name or password, it will display a pop-up message saying "Invalid Name or Password".

- **add_words :** If the User presses the button named ADD More Words, then this function will be called. It creates and display a Button named OK. This Function also Creates an Entry Widget for the User, in which he can input multiple words with comma separation, and once the user has inputted all the words, he can press the OK button.
- **word_add_function :** Once the OK button is pressed, this Function will be called. This function has the Logic for writing all the new words in the file(words.txt) instead of re-writing the same word. When all the words are written on the file, it creates a Label for informing the User about the task succession.
- **Highscores :** This Function takes one argument(the argument have default value) and can perform two operations. One, When called from the Admin mode its default value changes then it can reset the scores. Meaning it will overwrite all the scores saved in the file(“score.txt”). But if it’s called without an argument, it will open the file(“score.txt”) and read all the data. Then, it will compare all the scores to find the highest one and stores it in a variable. So that this high-score can be displayed in the game.
- **player_interface :** This function is called when the User has selected the Player interface and entered a name or alias in the entry widget, then had pressed the Login button. It also creates a Play button. If the Login button is pressed without entering any name it will display an error message and will not allow the user to play the game.
- **secret_word :** This Function has the logic for choosing a random word from the file(“word.txt”) and returns a random word as a string which then, is stored in a variable.
- **rem :** This function basically creates the Buttons of the Alphabets, from which the user can guess a letter by pressing its Button. It also checks if the pressed letter is guessed or not, and if that letter is in the secret-word then it will turn the Button of that letter to “Green”. All the Buttons created have their specific values that is passed as an argument when pressed to “check” function.
- **hangman_game :** This Function is called when the User presses the Play button. This function creates the Actual Game Layout with this Labels. This function has the variables such as guesses_left, warnings, character_picture, words_left and much more. It main task is to display all the Label and Button widgets for the Hangman game. It also calls the “secret_word” and “rem” function for the word and remaining letters for the game. It also has a button named Quit that will end the game and closes the window if pressed.
- **check :** This Function is called when the User presses any alphabet Button displaying on the screen. Then, it takes the value of that Button as argument and checks if that letter is in the secret-word or not. If the letter is in the word, then it replaces the “_” (underscore) with that letter and if the letter is not in the word, then it will subtract the guesses left. It also checks if the user has pressed the letter again, then it will subtract the warnings. It also contains the logic of subtracting two guesses(when a vowel is guessed that is not in the word) and subtracting the guesses once all the warnings are used. After all the updating it will display the updated values of all the Labels. It will also call the “congrats” function when the user has guessed all the letters, and if the user loses, it will also display a dialog box showing the secret-word and asking the user if he wants to play again or not.
- **congrats :** When called this function will destroy the Game window, to show the congratulatory message and the score of the user as well as two Buttons named Play Again and Quit. This Function also contains an Inner-Function. The Quit button will destroy the game window and will terminate the program, while the Play Again button will call the Inner-Function named play_again. It will also check if the user score is the best score of the game or not. If it is, then an additional Label will be displayed. Then it will write the user score with its name in the file(“score.txt”).
    - **play_again :** If the user presses the Play Again button, then this inner-function will be called. This function will destroy the congratulatory Labels and all the other

widgets created by the “congrats” function. Then, it will call the “hangman_game” function and the cycle continues.



MOST CHALLENGING PART WHILE WORKING ON THIS PROJECT :

Creating the GUI with its own logic for the Hangman game was the challenging part and took most of the time. The command option of Button widget was the one of the most frustrating part as it requires a function to pass the arguments and without knowing that it gave me the hard time. Also, tkinter’s PhotoImage created many problems and performed differently while calling from different functions; this problem remained until I moved to the latest version of the python. Also learning the functions and their behavior when used with GUI. After sorting all these issues, Warnings and guesses acquiring negative values took most of our time.

ANY NEW THING LEARNT WHILE WORKING ON THIS PROJECT :

We learnt a lot of things through-out our time in working on this project. Well, learning GUI was the best part of working in this project. Exception Handling, File Handling and using different libraries and its functions are a bonus point.

INDIVIDUAL CONTRIBUTIONS OF THE MEMBERS IN OUR GROUP :

- **Sardar Abdul Moiz Khan – CS22049 :** Created the GUI, and Playing Logic for the game.
- **Syed Muhammad Mujtaba – CS22045 :** Created the Function that chooses a random word from the file. Also, Wrote the documentation and report.

FUTURE EXPANSIONS :

An account creating option, that separates the new and existing users, with that they can use their accounts to play, and their score history will be saved. An option for the admin to delete some specific words from the word-list as well as specific player's account with their scores. A separate section for the Top Scores of the game and sound-effects is also part of future expansion.

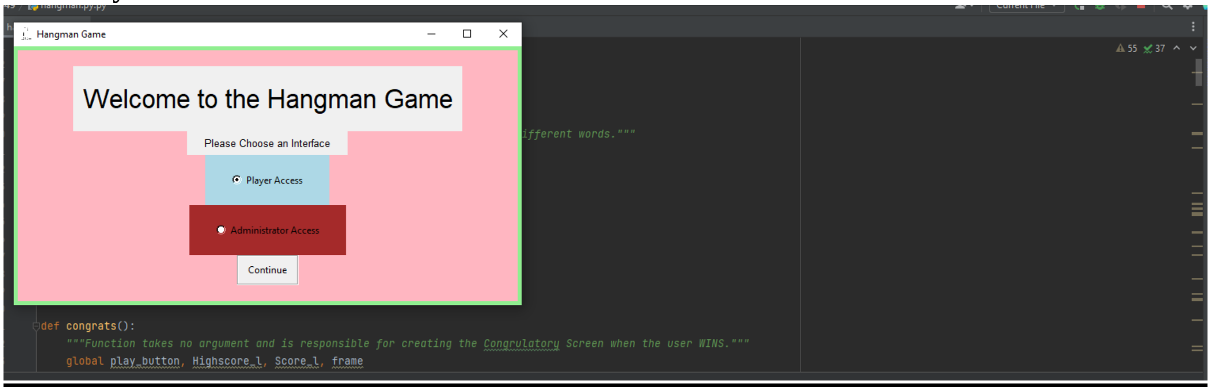
REFERENCES :

[Stack Overflow - Where Developers Learn, Share, & Build Careers]

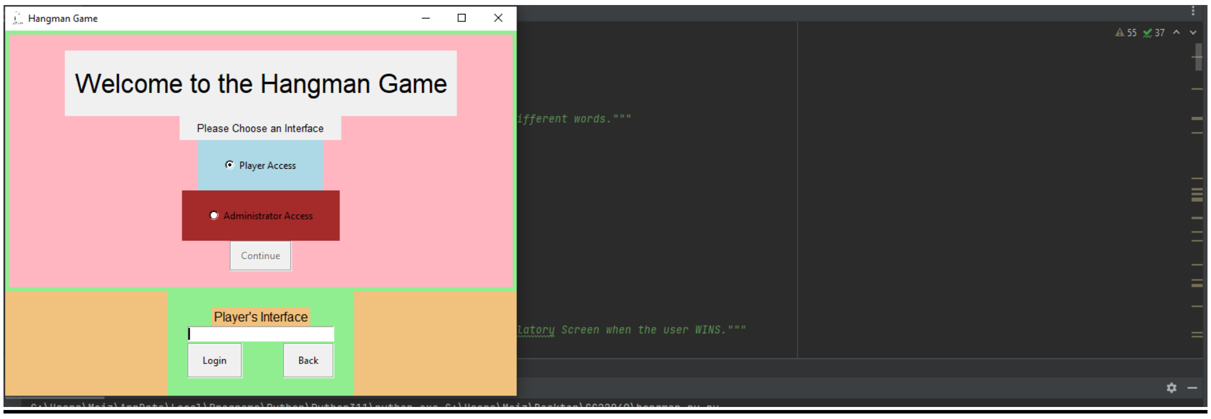
[Python GUI Programming (w3schools.in)]

TEST RUNS :

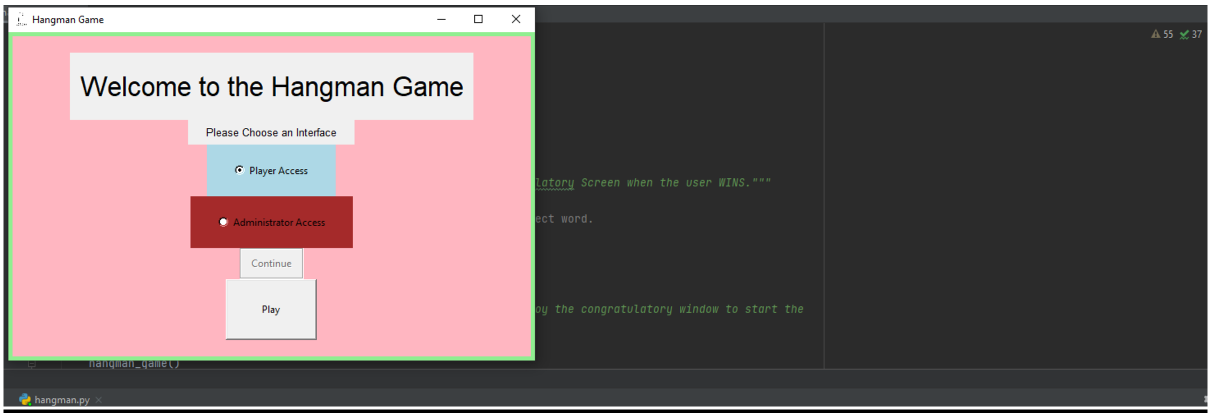
Main Layout :



```
ifferent words."""

def congrats():
    """Function takes no argument and is responsible for creating the Congrulatory Screen when the user WINS."""
    global play_button, Highscore_l, Score_l, frame
```

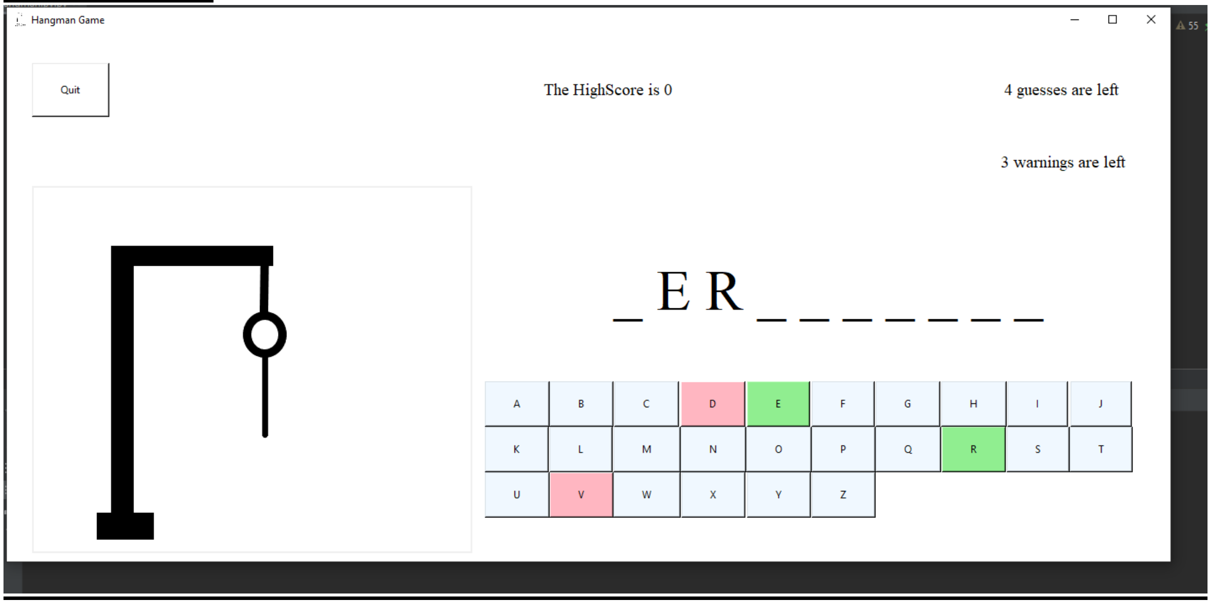


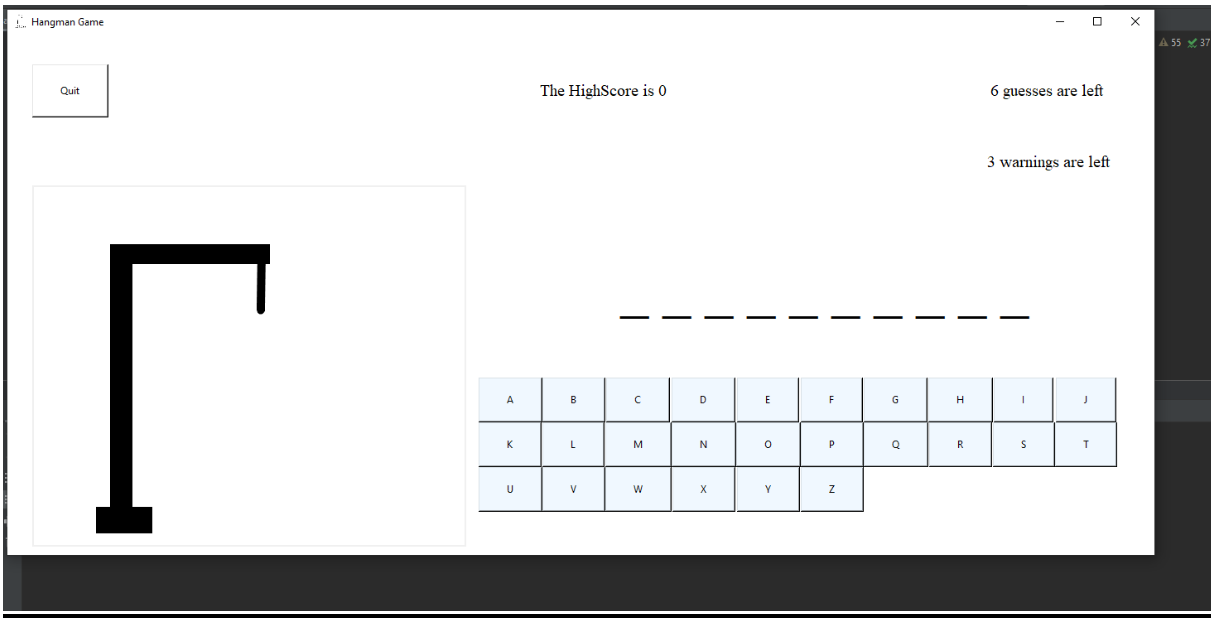
```
ifferent words."""
latory Screen when the user WINS."""
```

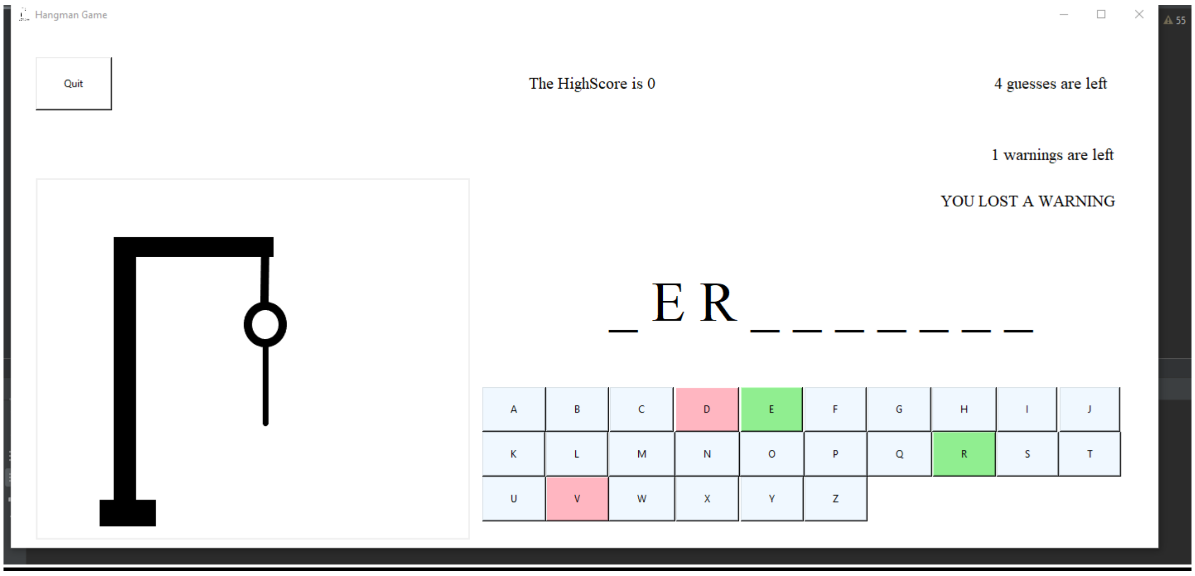


```
latory Screen when the user WINS."""
ect word.
oy the congratulatory window to start the
hangman_game()
```

TEST RUN #1:







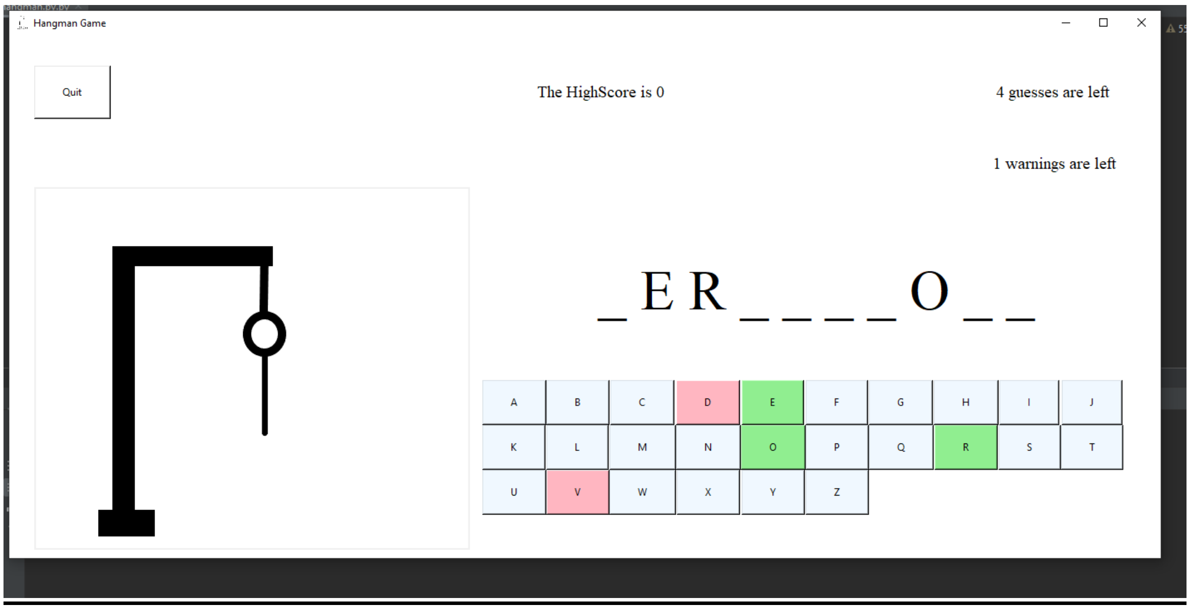
The HighScore is 0

4 guesses are left

1 warnings are left

YOU LOST A WARNING

_ E R _ _ _ _ _ _ _

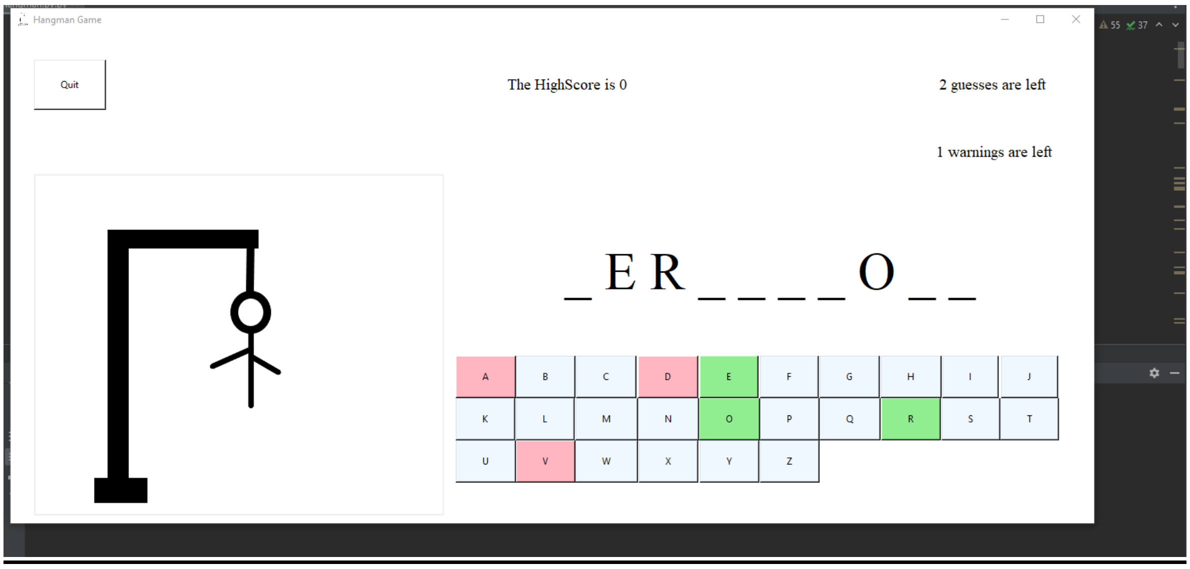


The HighScore is 0

4 guesses are left

1 warnings are left

_ E R _ _ _ _ O _ _

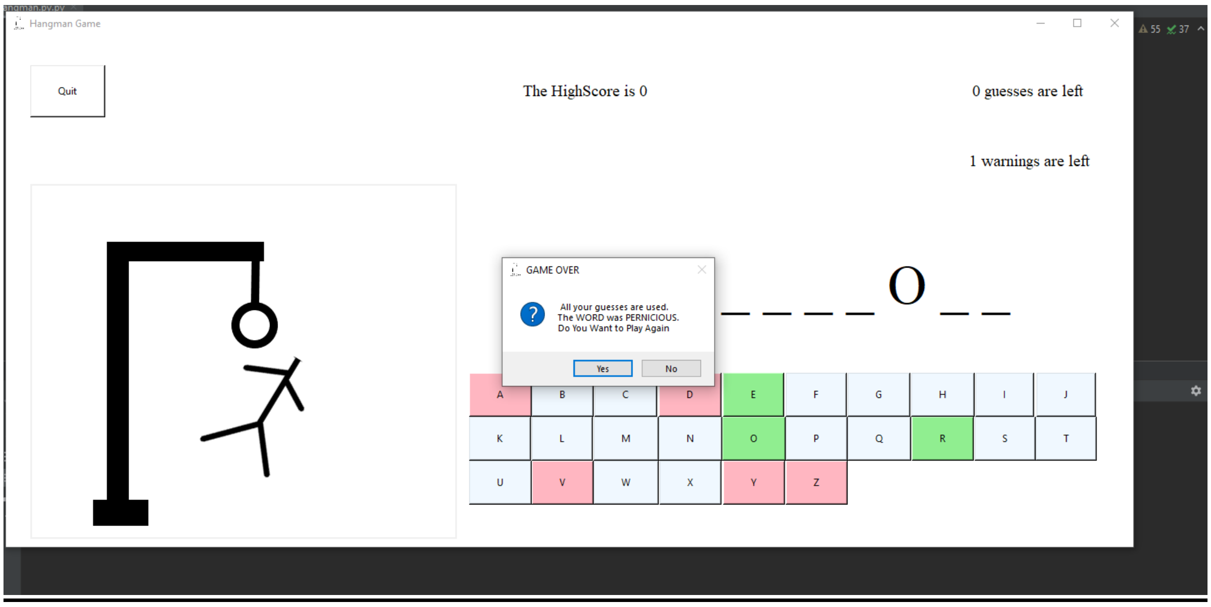


The HighScore is 0

2 guesses are left

1 warnings are left

_ E R _ _ _ _ O _ _



The HighScore is 0

0 guesses are left

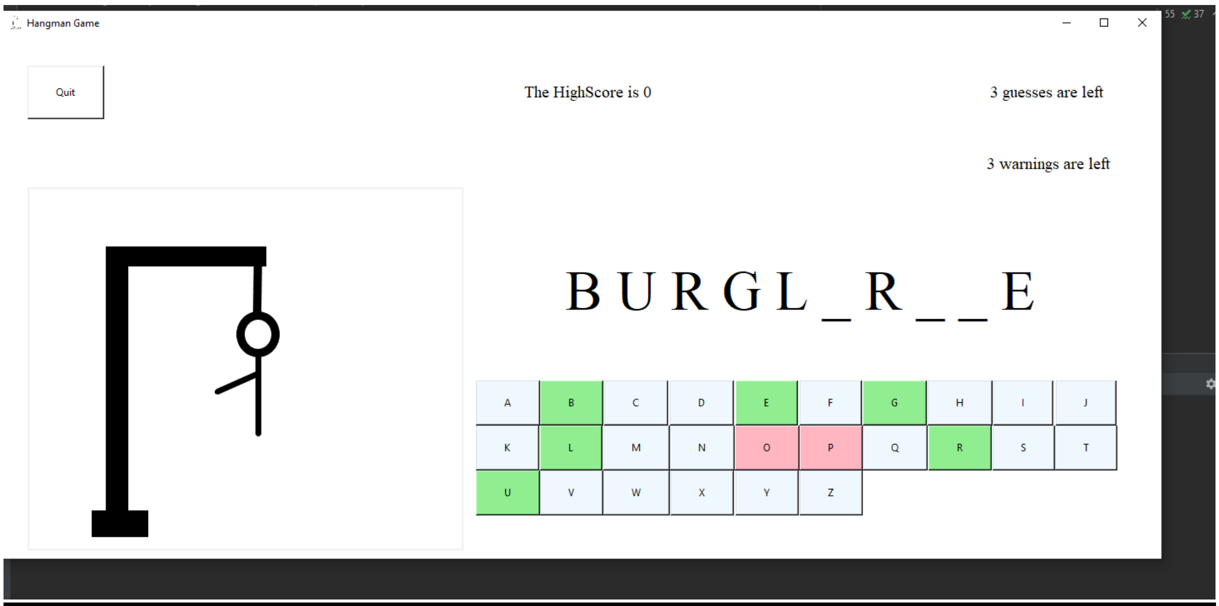
1 warnings are left

GAME OVER

All your guesses are used.
The WORD was PERNICIOUS.
Do You Want to Play Again

_ _ _ _ O _ _

TEST RUN #2:

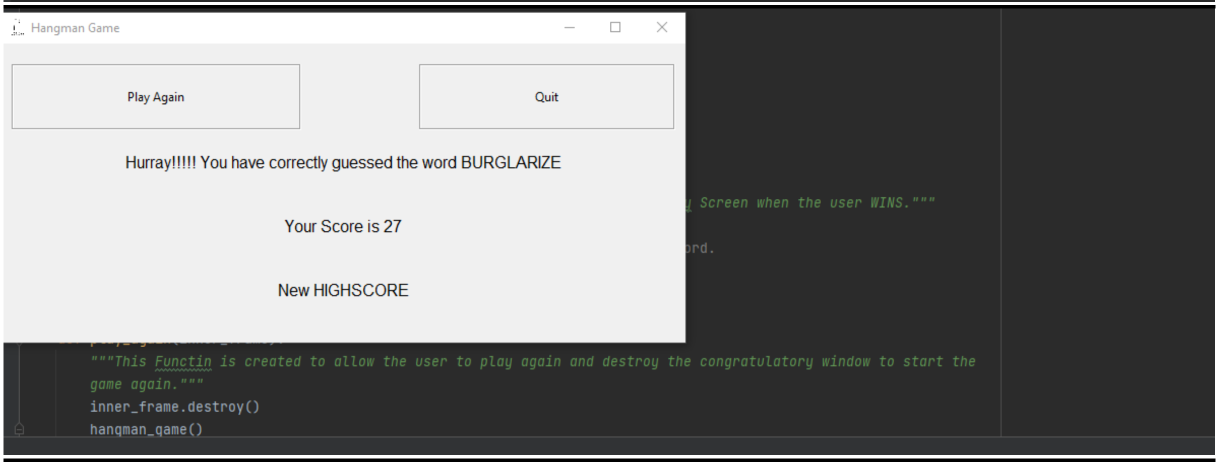


The HighScore is 0

3 guesses are left

3 warnings are left

B U R G L _ R _ _ E



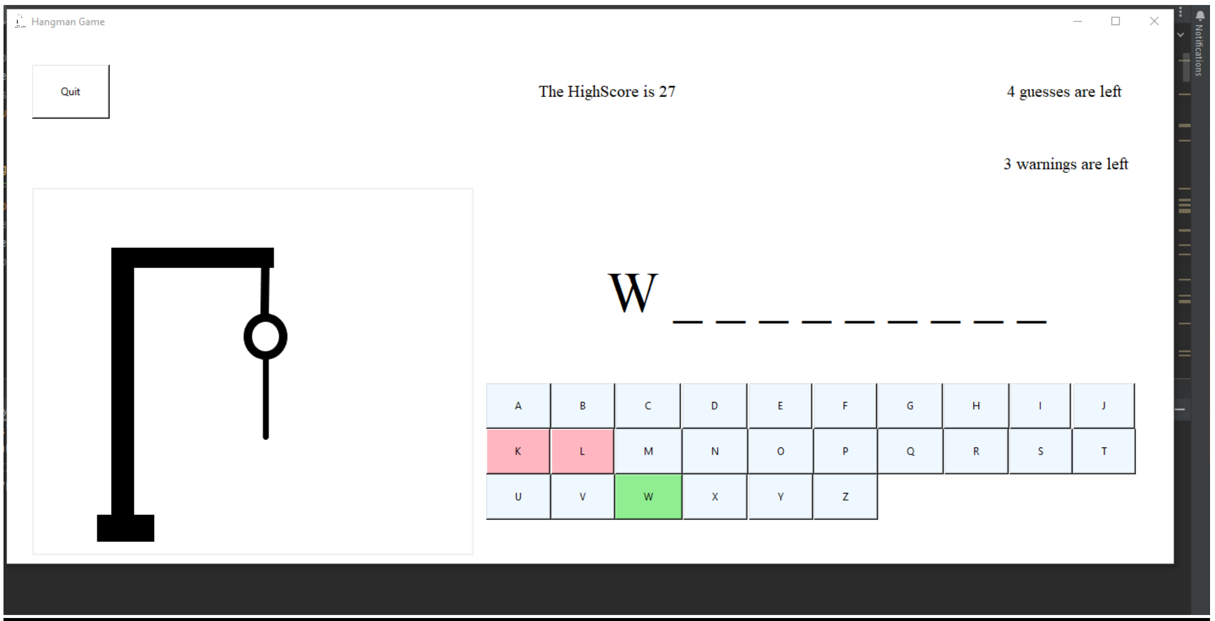
Hurray!!!!! You have correctly guessed the word BURGLARIZE

Your Score is 27

New HIGHSCORE

```
                                    Screen when the user WINS."""
                                    ord.
    """This Functin is created to allow the user to play again and destroy the congratulatory window to start the
    game again."""
    inner_frame.destroy()
    hangman_game()
```

TEST RUN #3:

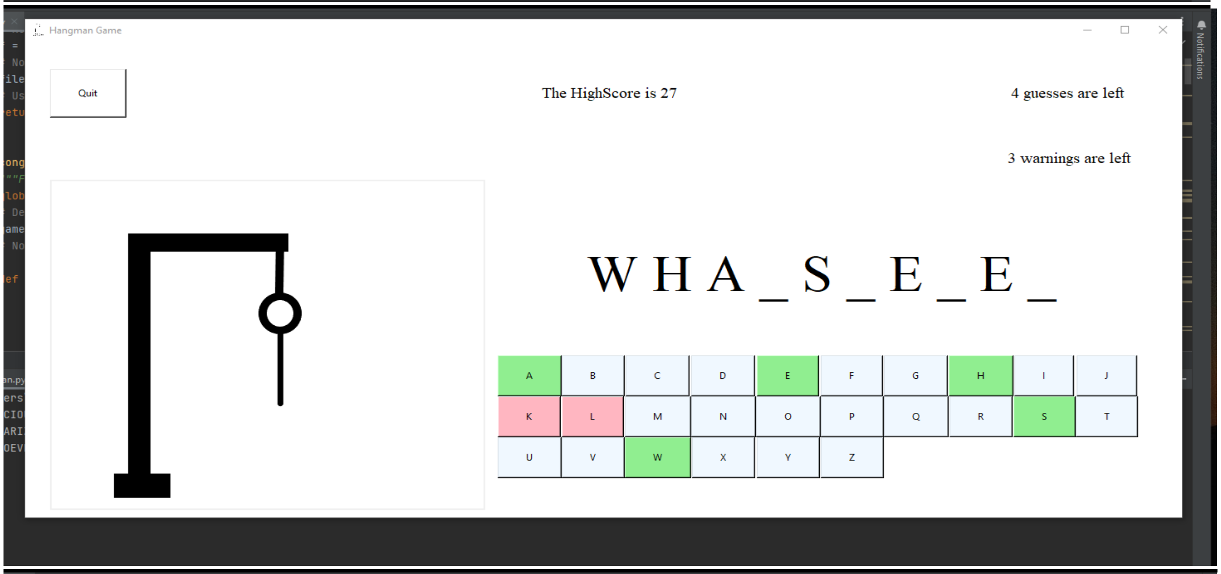


The HighScore is 27

4 guesses are left

3 warnings are left

W _ _ _ _ _ _ _ _ _

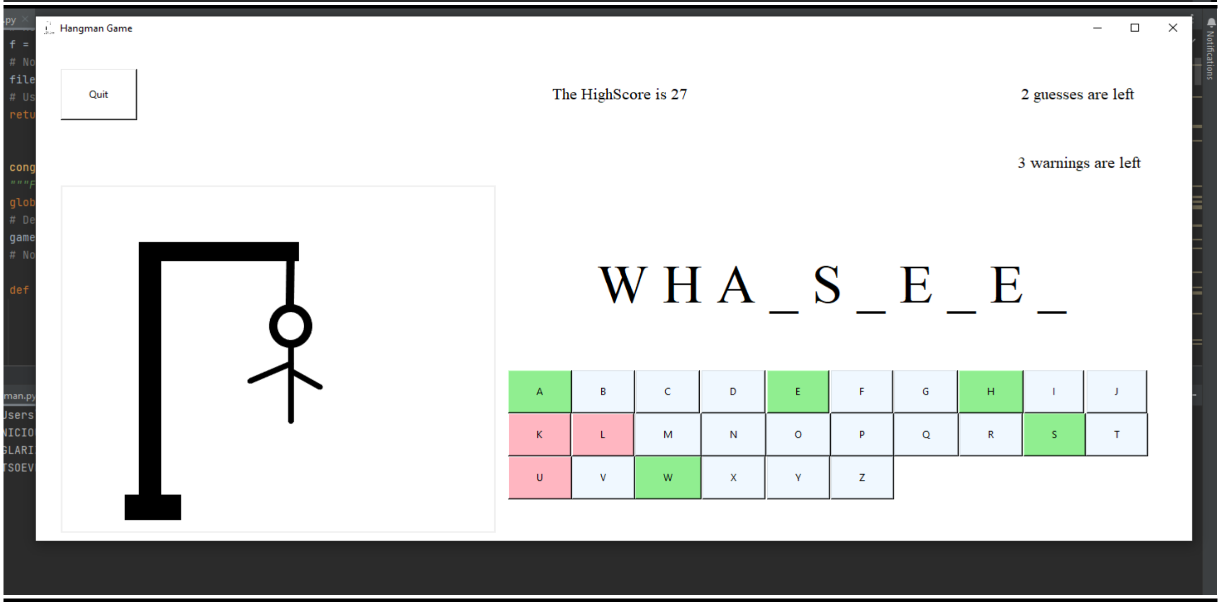


The HighScore is 27

4 guesses are left

3 warnings are left

W H A _ S _ E _ E _

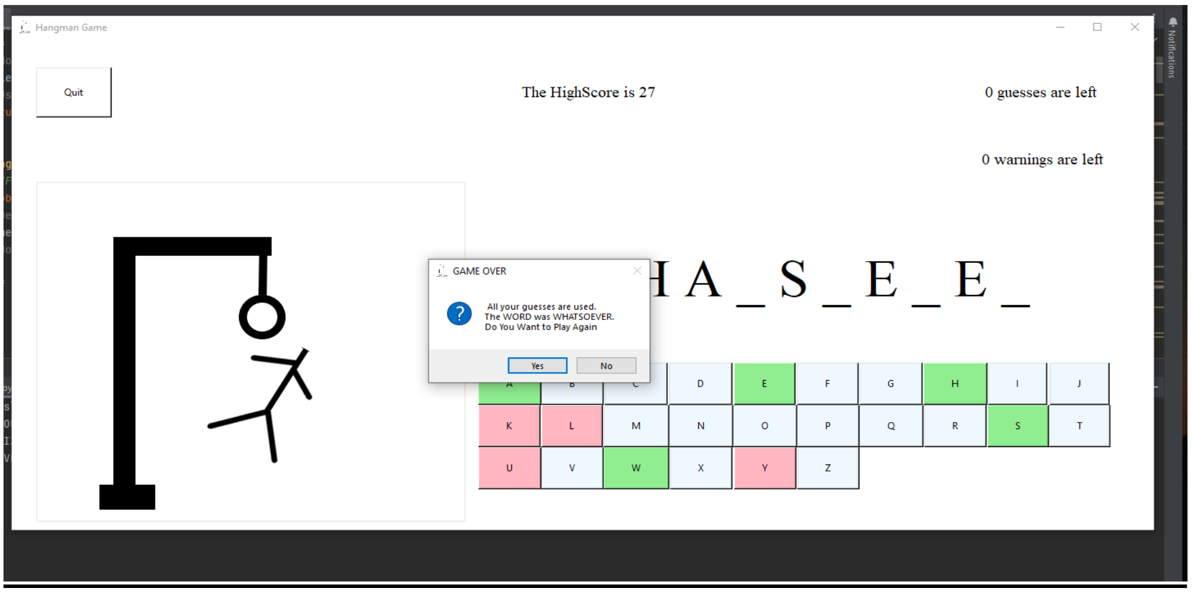


The HighScore is 27

2 guesses are left

3 warnings are left

W H A _ S _ E _ E _


Hangman Game

The HighScore is 27

0 guesses are left

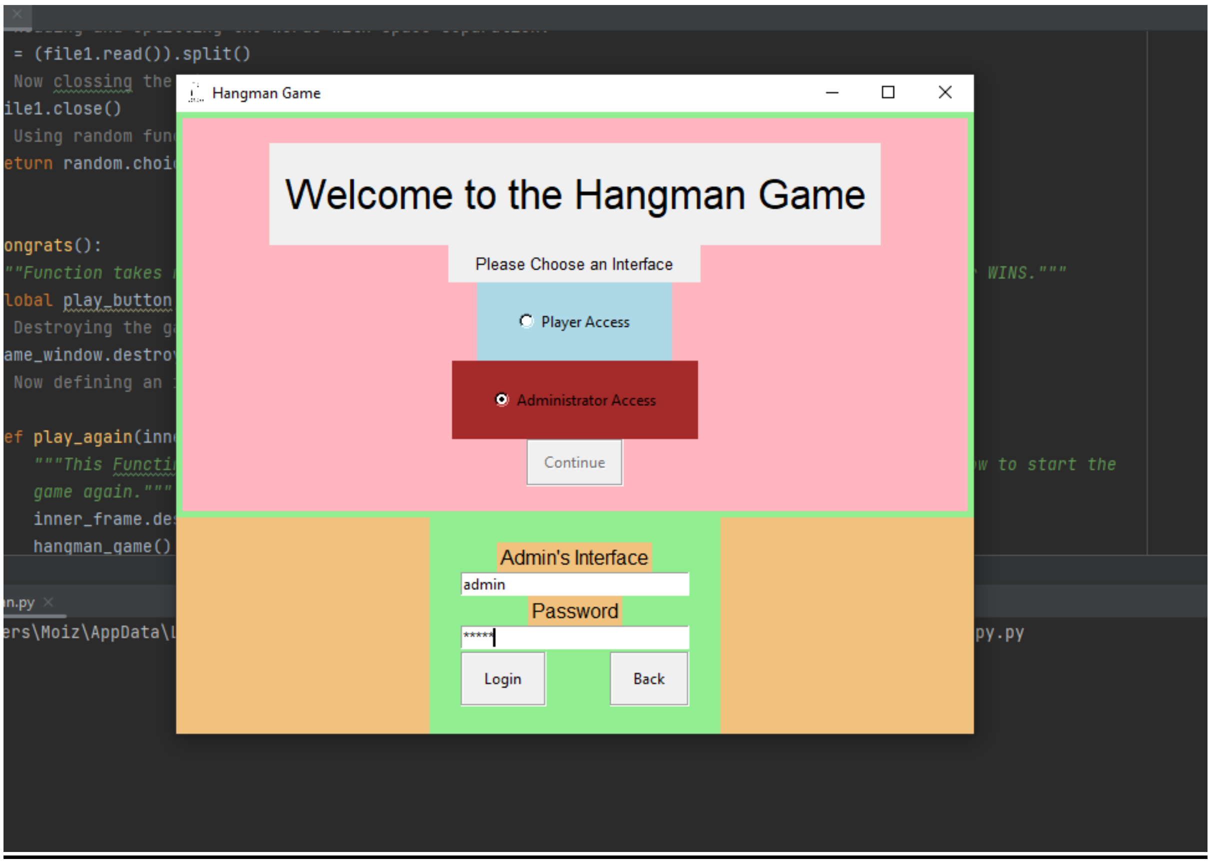
0 warnings are left

H A _ S _ E _ E _

GAME OVER

All your guesses are used.
The WORD was WHATSOEVER.
Do You Want to Play Again

Admin Layout:


```
= (file1.read()).split()
 Now clossing the
ile1.close()
 Using random fun
eturn random.choi

ongrats():
""Function takes                                   WINS."""
lobal play_button
 Destroying the g
ame_window.destro
 Now defining an

ef play_again(inn
   """This Functi                                  w to start the
   game again."""
   inner_frame.de
   hangman_game()
```

Hangman Game

Welcome to the Hangman Game

Please Choose an Interface

○ Player Access

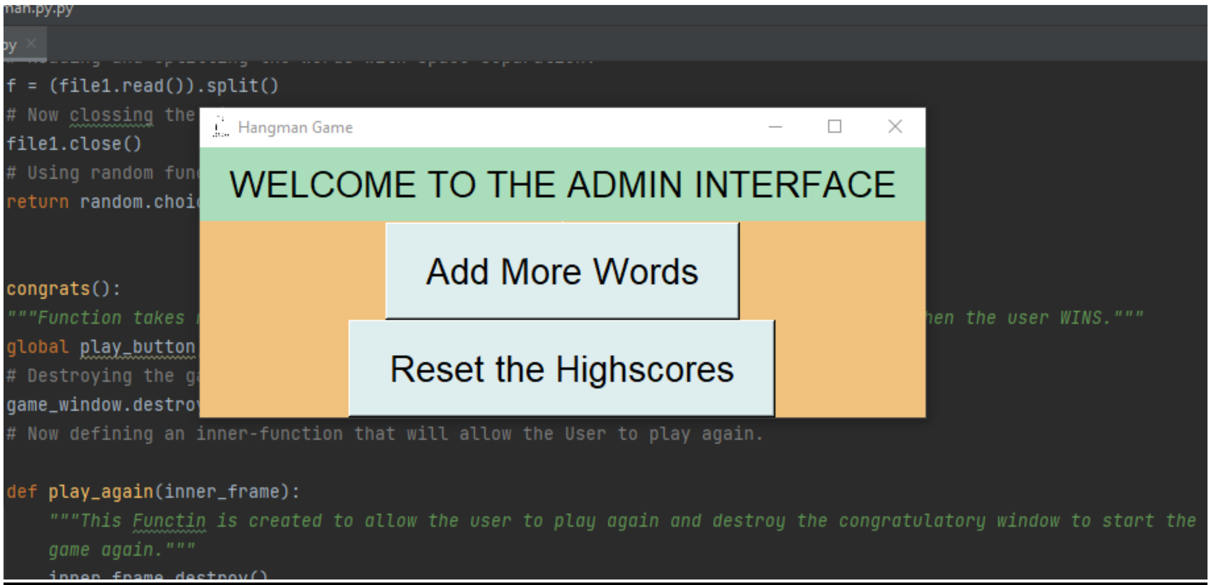
● Administrator Access

Admin's Interface

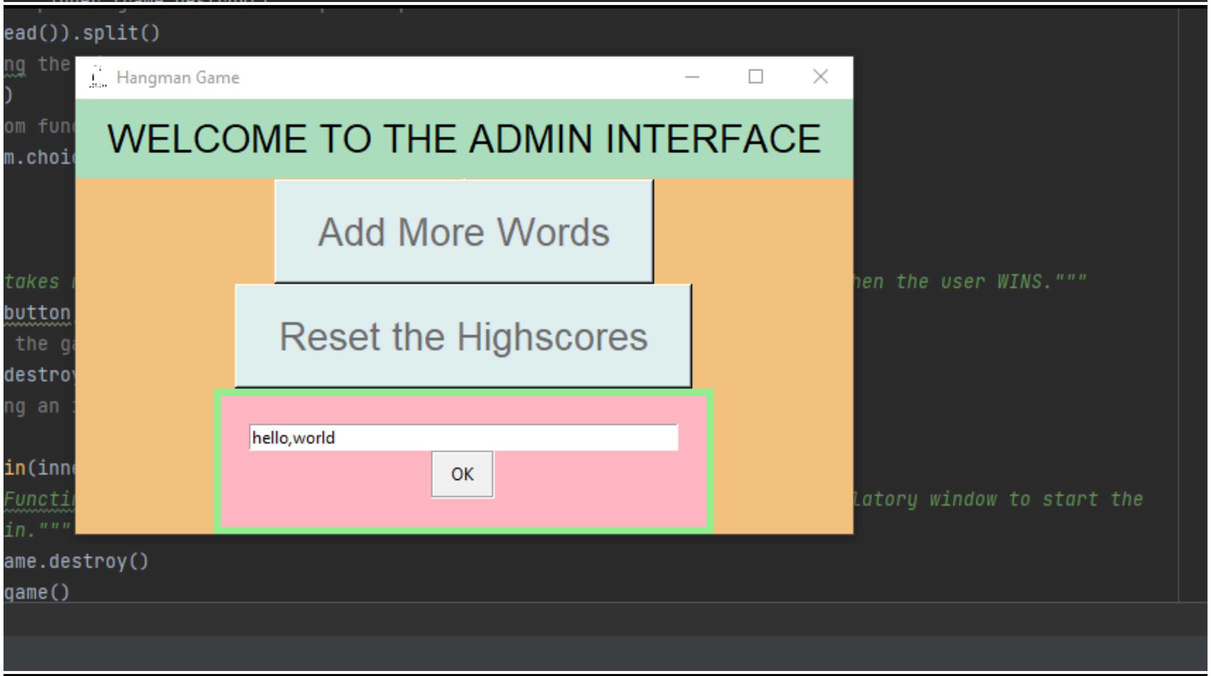
admin

Password

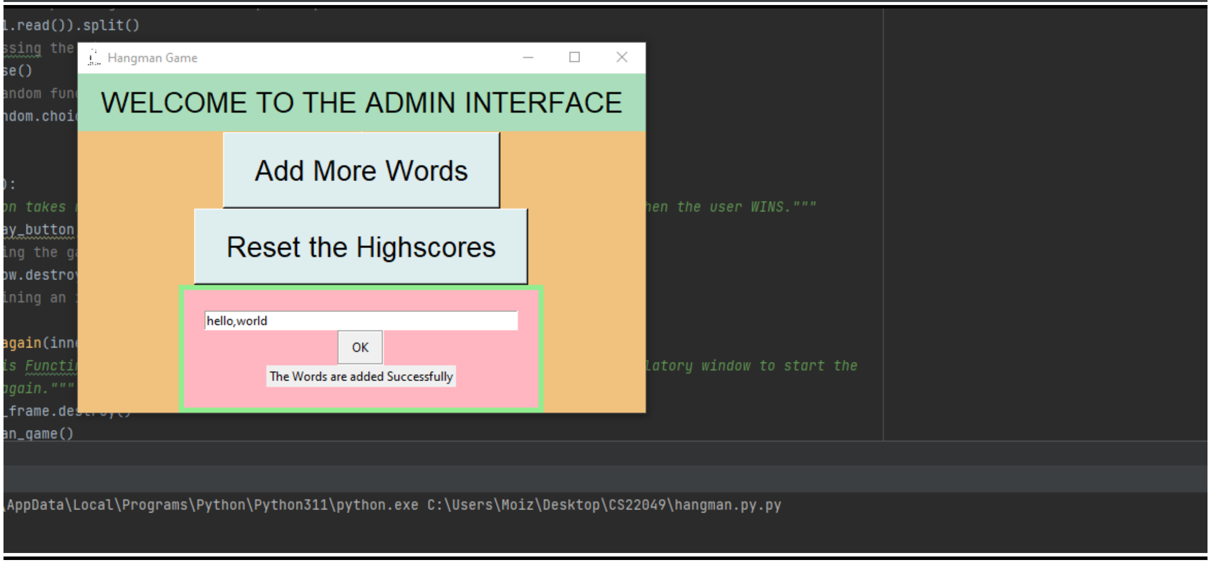
*****



```
man.py.py
f = (file1.read()).split()
# Now clossing the
file1.close()
# Using random func
return random.choic

congrats():
"""Function takes                                                   hen the user WINS."""
global play_button
# Destroying the ga
game_window.destroy
# Now defining an inner-function that will allow the User to play again.

def play_again(inner_frame):
    """This Functin is created to allow the user to play again and destroy the congratulatory window to start the
    game again."""
    inner_frame.destroy()
```



```
ead()).split()
ng the
)
om func
m.choic

takes                                               hen the user WINS."""
button
 the ga
destroy
ng an

in(inn
Functi                                              latory window to start the
in."""
ame.destroy()
game()
```



```
1.read()).split()
sing the
se()
andom fun
ndom.choi

):
n takes                                    hen the user WINS."""
ay_button
ing the g
ow.destro
ining an

again(inn
is Functi                                  latory window to start the
gain."""
_frame.destroy()
an_game()

AppData\Local\Programs\Python\Python311\python.exe C:\Users\Moiz\Desktop\CS22049\hangman.py.py
```

The Words are added Successfully