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	AZKA SOHAIL	CS-002
S2	ESBAH SOHAIL	CS-008
S3	TUBA NAUSHAD	CS-021

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

COMPUTER PROGRAMMING PROJECT: HANGMAN

GROUP MEMBERS:

1.	AZKA SOHAIL	CS-002
2.	ESBAH SOHAIL	CS-008
3.	TUBA NAUSHAD	CS-021

GROUP IDs:

- G1
- GROUP ID 9

PROJECT DISCRIPTION:

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:

First, our program prompts the user to choose between playing as a user or making administrative modifications.

Our program randomly chooses a word from a text file of 5000 words for the user interface, and the user must guess each letter from a list of alphabets that is displayed to them.

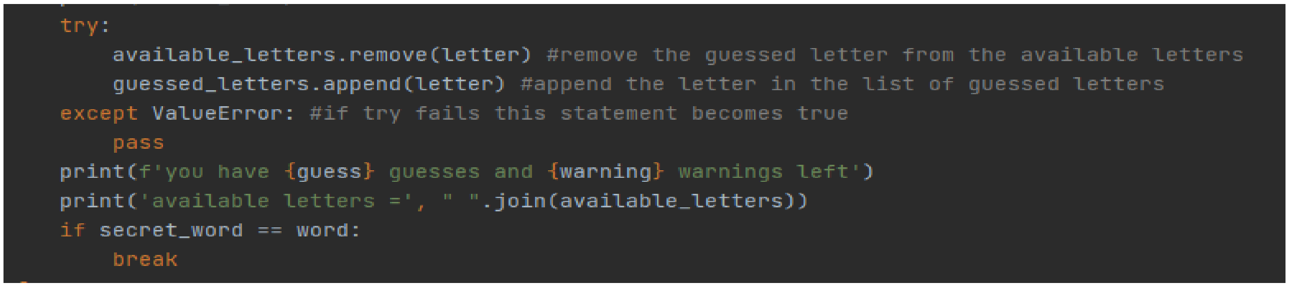
Our application prints a notification, if the letter predicted is incorrect, and the user loses one of their six guesses. The user is shown a list of alphabets before each turn that does not include the letter they just guessed, and our system also keeps track of the letter they guessed. User loses a guess when the number of warnings reaches zero. The software ends when the user correctly guesses the word or when the number of guesses and warnings reaches zero. The user's score and the highest score are printed at the end of the code.

In contrast, our program's admin interface will give the admin the option to either add a word to the text file or reset the high score file.

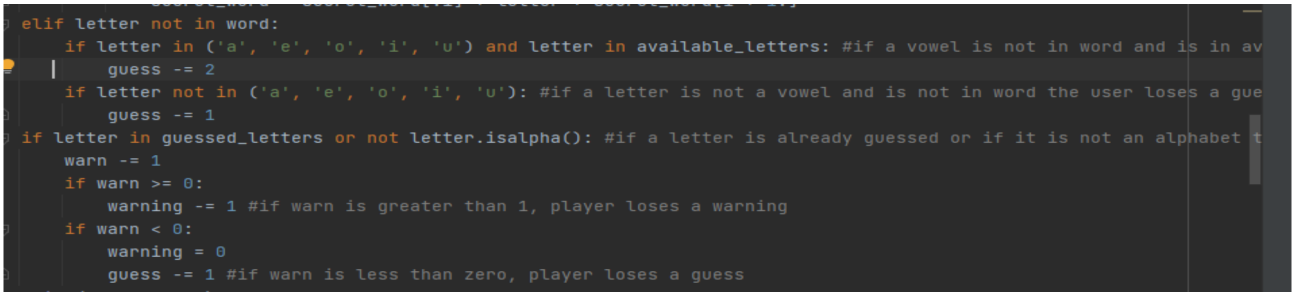
Following are the specific features of our program:

1. Our program logic is divided in to functions.
2. Game function allows user to play the game while admin mode allows the user to reset the high-score or to add word in the text file.
3. In user mode the user guesses the letter one by one, we initially provide user with 6 guesses and 3 warnings.
4. Our program automatically appends the letter in the list of guessed letters and also removes the letter from the set of available letters.
5. The player forfeits a guess if a letter is not in the word.
6. The player loses two guesses if they insert a vowel that is not present in the term.
7. If the player enters a vowel already guessed before, the player loses one guess.
8. The player forfeits a warning if they enter anything other than an alphabet.
9. The player forfeits a guess if the warning value falls to zero.
10. If the player successfully guesses the word, our application calls the player-score and high-score functions to print the player's score and high score respectively.
11. If the player guesses the word incorrectly, our application just displays the player's high score.
12. In admin mode, our program offers the user the option to play the game, add single or multiple words to a text file, or reset the high-score file.
13. In addition, we also used time and random functions and also imported lowercase alphabet from the string library. If the player enters anything besides an alphabet, the player loses a warning.

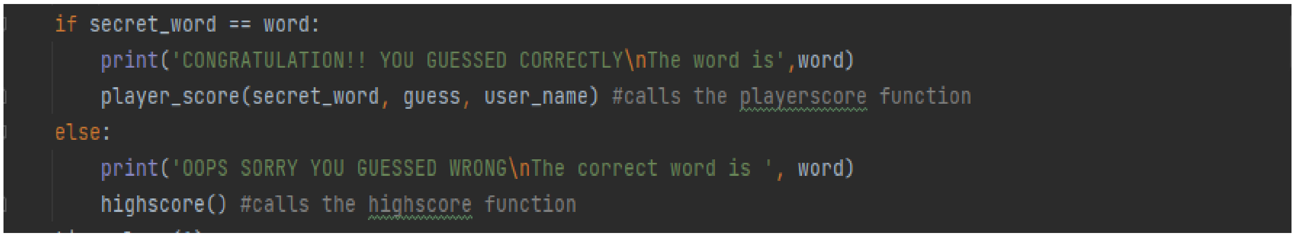
.



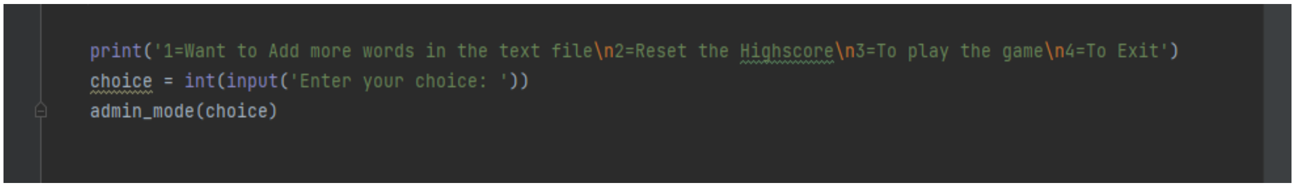
```
    try:
        available_letters.remove(letter) #remove the guessed letter from the available letters
        guessed_letters.append(letter) #append the letter in the list of guessed letters
    except ValueError: #if try fails this statement becomes true
        pass
    print(f'you have {guess} guesses and {warning} warnings left')
    print('available letters =', " ".join(available_letters))
    if secret_word == word:
        break
```



```
elif letter not in word:
    if letter in ('a', 'e', 'o', 'i', 'u') and letter in available_letters: #if a vowel is not in word and is in av
        guess -= 2
    if letter not in ('a', 'e', 'o', 'i', 'u'): #if a letter is not a vowel and is not in word the user loses a gue
        guess -= 1
if letter in guessed_letters or not letter.isalpha(): #if a letter is already guessed or if it is not an alphabet t
    warn -= 1
    if warn >= 0:
        warning -= 1 #if warn is greater than 1, player loses a warning
    if warn < 0:
        warning = 0
        guess -= 1 #if warn is less than zero, player loses a guess
```

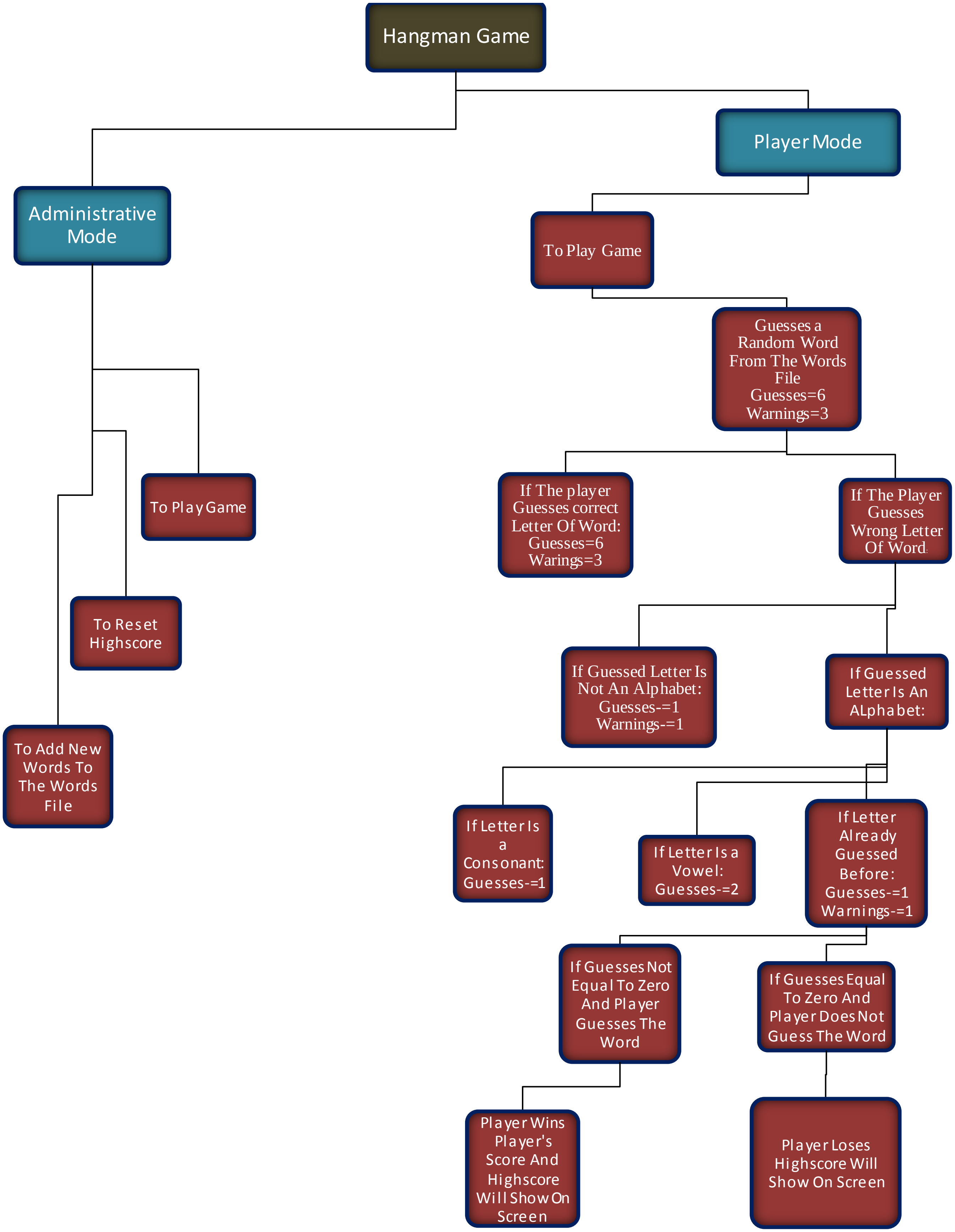


```
if secret_word == word:
    print('CONGRATULATION!! YOU GUESSED CORRECTLY\nThe word is',word)
    player_score(secret_word, guess, user_name) #calls the playerscore function
else:
    print('OOPS SORRY YOU GUESSED WRONG\nThe correct word is ', word)
    highscore() #calls the highscore function
```



```
print('1=Want to Add more words in the text file\n2=Reset the Highscore\n3=To play the game\n4=To Exit')
choice = int(input('Enter your choice: '))
admin_mode(choice)
```

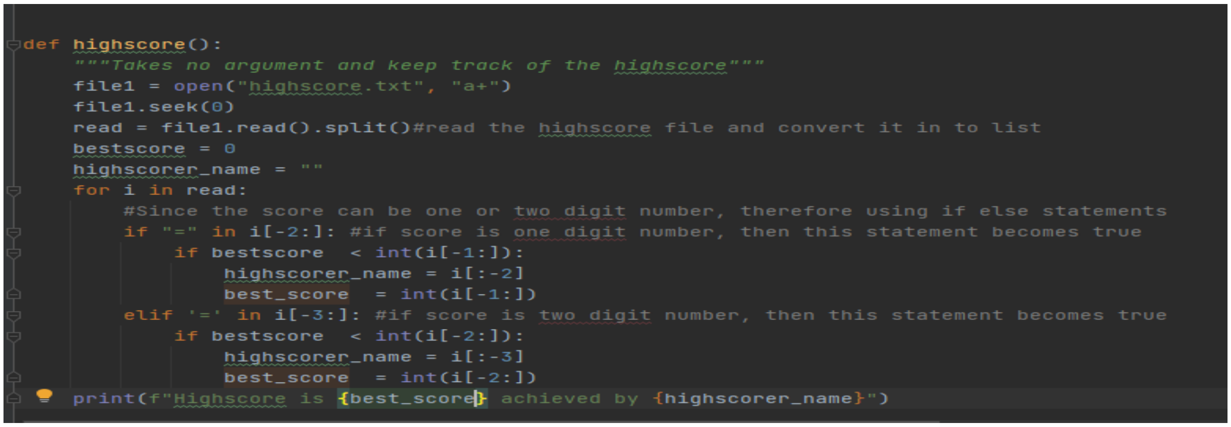
FLOW OF OUR PROJECT:


Hangman Game
Administrative Mode
Player Mode
To Play Game
To Play Game
To Reset Highscore
To Add New Words To The Words File
Guesses a Random Word From The Words File Guesses=6 Warnings=3
If The player Guesses correct Letter Of Word: Guesses=6 Warings=3
If The Player Guesses Wrong Letter Of Word
If Guessed Letter Is Not An Alphabet: Guesses-=1 Warnings-=1
If Guessed Letter Is An ALphabet:
If Letter Is a Consonant: Guesses-=1
If Letter Is a Vowel: Guesses-=2
If Letter Already Guessed Before: Guesses-=1 Warnings-=1
If Guesses Not Equal To Zero And Player Guesses The Word
If Guesses Equal To Zero And Player Does Not Guess The Word
Player Wins Player's Score And Highscore Will Show On Screen
Player Loses Highscore Will Show On Screen

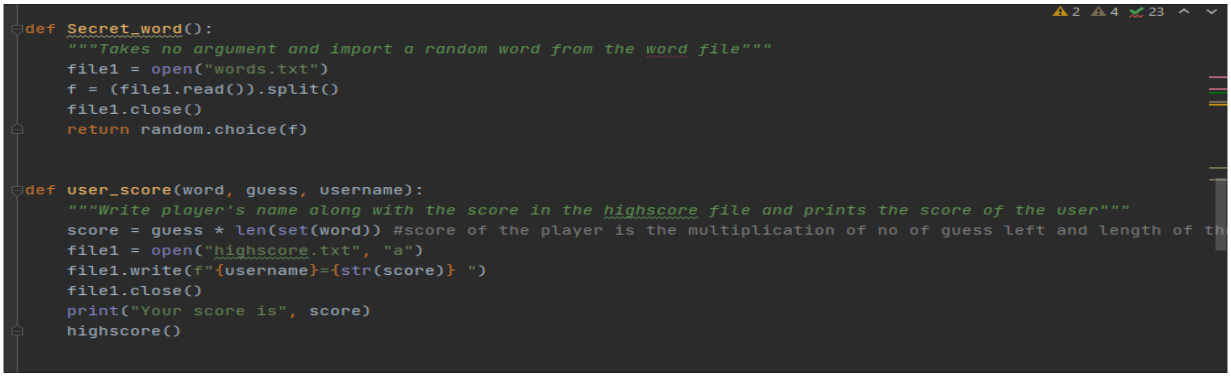
CHALLENGING PART FOR US WHILE MAKING THE PROJECT:

- We found the high-score logic to be the most difficult aspect of the project to create. In order to make our program more efficient, we first simply intended to record the player's score in the high-score file. However, by adding the player's name as well as their score, we were forced to contend with the challenge of comparing the scores in order to maintain the high-score.
- We also faced some complications in the replacement of (-) by the correctly guessed letter It took us a while, but we figured out it was to be done on indices.

We worked on this for a while, but we eventually succeeded in creating this logic



```
def highscore():
    """Takes no argument and keep track of the highscore"""
    file1 = open("highscore.txt", "a+")
    file1.seek(0)
    read = file1.read().split()#read the highscore file and convert it in to list
    bestscore = 0
    highscorer_name = ""
    for i in read:
        #Since the score can be one or two digit number, therefore using if else statements
        if "=" in i[-2:]: #if score is one digit number, then this statement becomes true
            if bestscore  < int(i[-1:]):
                highscorer_name = i[:-2]
                best_score  = int(i[-1:])
        elif '=' in i[-3:]: #if score is two digit number, then this statement becomes true
            if bestscore  < int(i[-2:]):
                highscorer_name = i[:-3]
                best_score  = int(i[-2:])
    print(f"Highscore is {best_score} achieved by {highscorer_name}")
```



```
def Secret_word():
    """Takes no argument and import a random word from the word file"""
    file1 = open("words.txt")
    f = (file1.read()).split()
    file1.close()
    return random.choice(f)


def user_score(word, guess, username):
    """Write player's name along with the score in the highscore file and prints the score of the user"""
    score = guess * len(set(word)) #score of the player is the multiplication of no of guess left and length of th
    file1 = open("highscore.txt", "a")
    file1.write(f"{username}={str(score)} ")
    file1.close()
    print("Your score is", score)
    highscore()
```

NEW LEARNING WHILE MAKING THIS PROJECT:

The two distinct things we discovered while working on this project were the time function and random function. Our project is more effective thanks to the time function, which allow us to import time in seconds on any statement in the program that we choose. The random function selects a word at random from a collection of several words on which the player plays the game.

We also explored maximum use of function and how to call one function in another function.

We learned to assign same value to two different variables and to modify one while the other is fixed.

CONTRIBUTION OF EVERY GROUP MEMBER WHILE MAKING THIS PROJECT:

CONTRIBUTION OF AZKA:

- The logic of letters in words replaced by '-'.
- Replacement of underscore (-) by correctly guessed letter and save the changes.
- If warning becomes 0 player loses a guess by assigning two variables (warn and warning) to the number of warnings.
- Calculating the score of the player (player score function).

CONTRIBUTION OF TUBA:

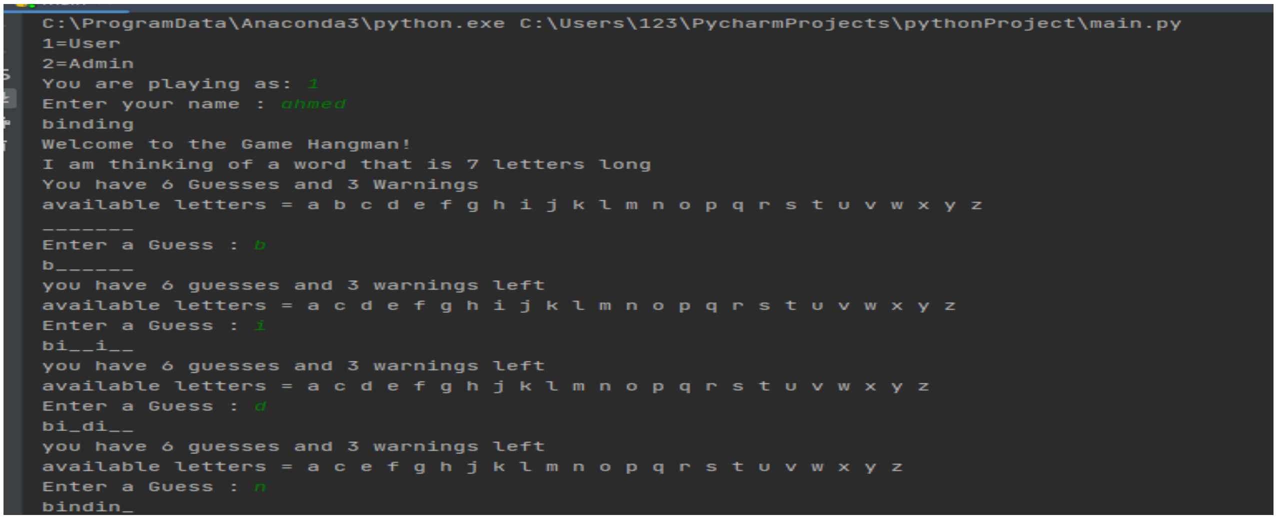
- Selecting random word from the file (secret word function).
- Appending word in text file.
- Resetting the high score.
- Keeping track of the guessed letters.

CONTRIBUTION OF ESBAH:

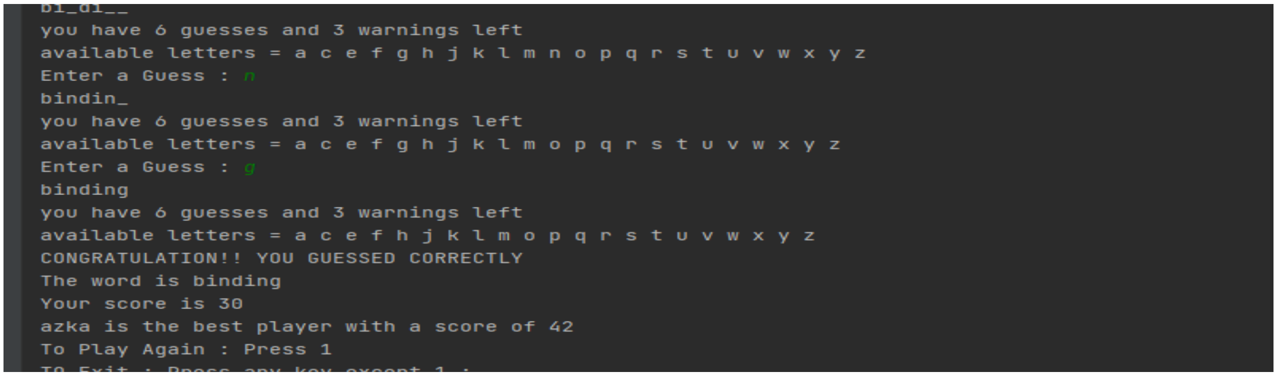
- Removing letter from the list of available letter.
- Comparing the score of the players
- Printing the score along with user name.
- Keeping track of the high score
- Printing high score along with the name of the user achieving it (high score function)

TEST CASE RUNS:

WHEN USER WINS THE GAME:



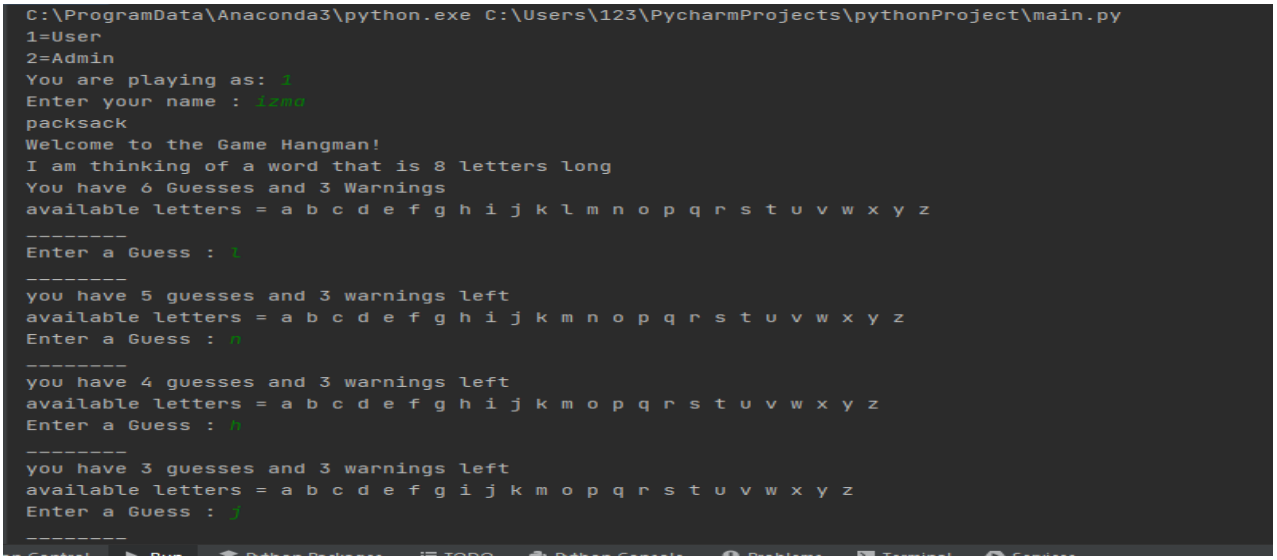
```
C:\ProgramData\Anaconda3\python.exe C:\Users\123\PycharmProjects\pythonProject\main.py
1=User
2=Admin
You are playing as: 1
Enter your name : ahmed
binding
Welcome to the Game Hangman!
I am thinking of a word that is 7 letters long
You have 6 Guesses and 3 Warnings
available letters = a b c d e f g h i j k l m n o p q r s t u v w x y z
_______
Enter a Guess : b
b______
you have 6 guesses and 3 warnings left
available letters = a c d e f g h i j k l m n o p q r s t u v w x y z
Enter a Guess : i
bi__i__
you have 6 guesses and 3 warnings left
available letters = a c d e f g h j k l m n o p q r s t u v w x y z
Enter a Guess : d
bi_di__
you have 6 guesses and 3 warnings left
available letters = a c e f g h j k l m n o p q r s t u v w x y z
Enter a Guess : n
bindin_
```



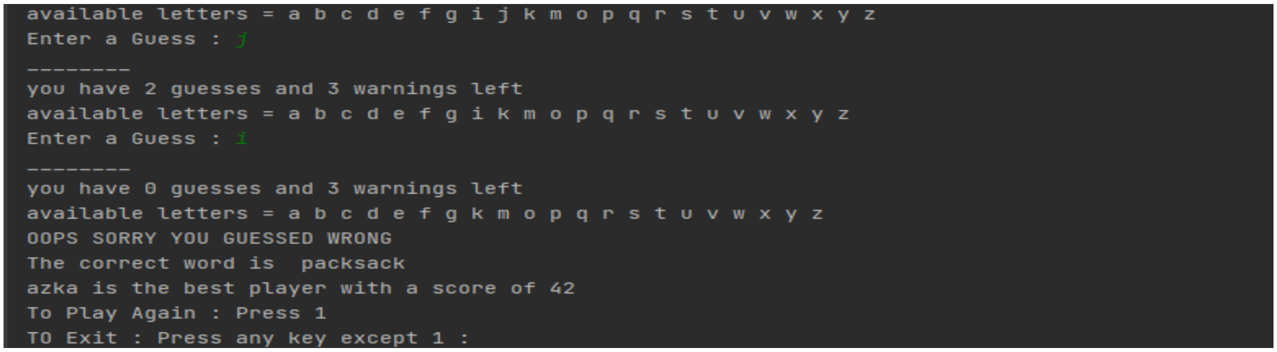
```
bi_di__
you have 6 guesses and 3 warnings left
available letters = a c e f g h j k l m n o p q r s t u v w x y z
Enter a Guess : n
bindin_
you have 6 guesses and 3 warnings left
available letters = a c e f g h j k l m o p q r s t u v w x y z
Enter a Guess : g
binding
you have 6 guesses and 3 warnings left
available letters = a c e f h j k l m o p q r s t u v w x y z
CONGRATULATION!! YOU GUESSED CORRECTLY
The word is binding
Your score is 30
azka is the best player with a score of 42
To Play Again : Press 1
TO Exit : Press any key except 1 :
```

When user wins the game the program print a statement “CONGRATULATION!! YOU GUESSED CORRECTLTY”. It shows the score along with the name of the player and the high-score. It then allows the user to either exit the game or to play it again.

WHEN USER LOSES THE GAME:



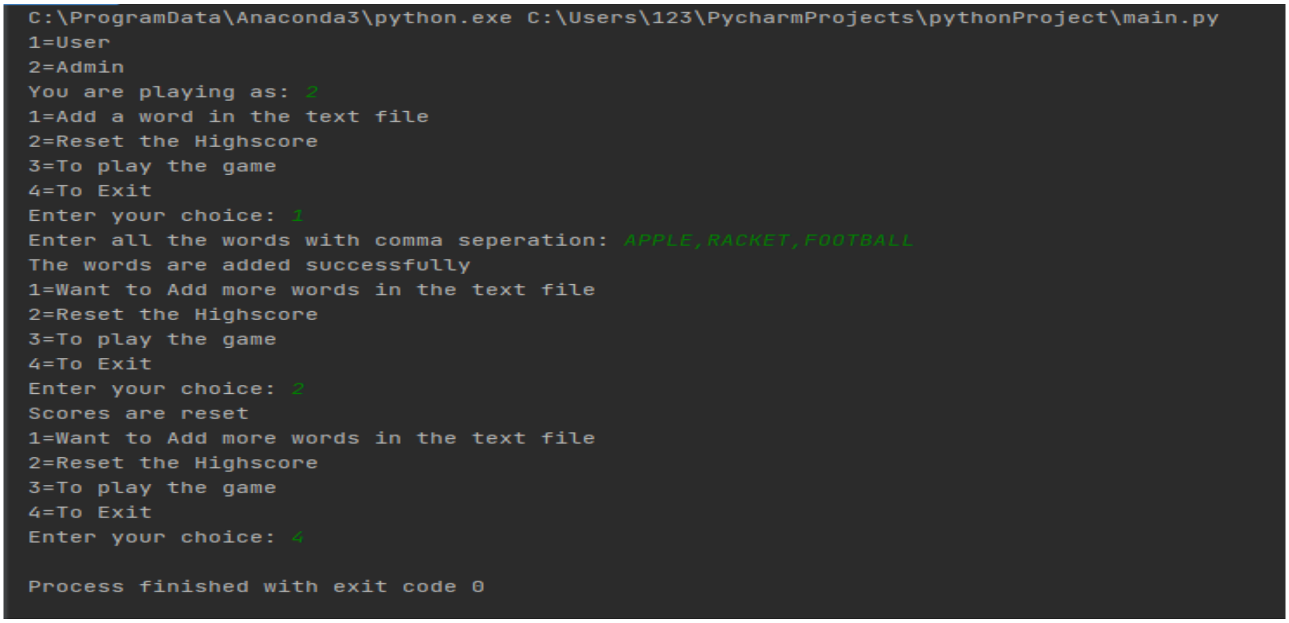
```
C:\ProgramData\Anaconda3\python.exe C:\Users\123\PycharmProjects\pythonProject\main.py
1=User
2=Admin
You are playing as: 1
Enter your name : izma
packsack
Welcome to the Game Hangman!
I am thinking of a word that is 8 letters long
You have 6 Guesses and 3 Warnings
available letters = a b c d e f g h i j k l m n o p q r s t u v w x y z
--------
Enter a Guess : l
--------
you have 5 guesses and 3 warnings left
available letters = a b c d e f g h i j k m n o p q r s t u v w x y z
Enter a Guess : n
--------
you have 4 guesses and 3 warnings left
available letters = a b c d e f g h i j k m o p q r s t u v w x y z
Enter a Guess : h
--------
you have 3 guesses and 3 warnings left
available letters = a b c d e f g i j k m o p q r s t u v w x y z
Enter a Guess : j
--------
```



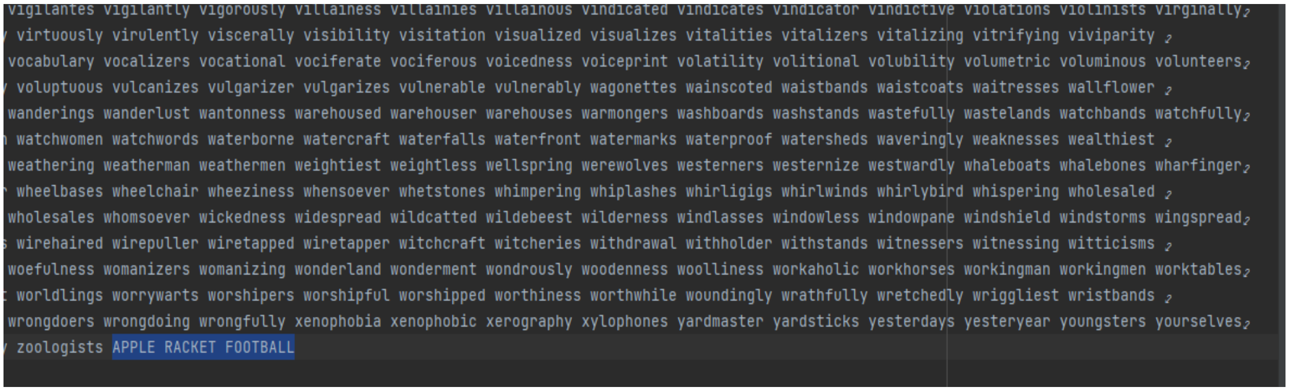
```
available letters = a b c d e f g i j k m o p q r s t u v w x y z
Enter a Guess : j
--------
you have 2 guesses and 3 warnings left
available letters = a b c d e f g i k m o p q r s t u v w x y z
Enter a Guess : i
--------
you have 0 guesses and 3 warnings left
available letters = a b c d e f g k m o p q r s t u v w x y z
OOPS SORRY YOU GUESSED WRONG
The correct word is  packsack
azka is the best player with a score of 42
To Play Again : Press 1
TO Exit : Press any key except 1 :
```

If the user guessed the word wrong, the program print a statement "OOPS SORRY YOU GUESSED WRONG" . It reveals the secret word at the end and shows the high-score. It then allows the user to either exit the game or to play it again.

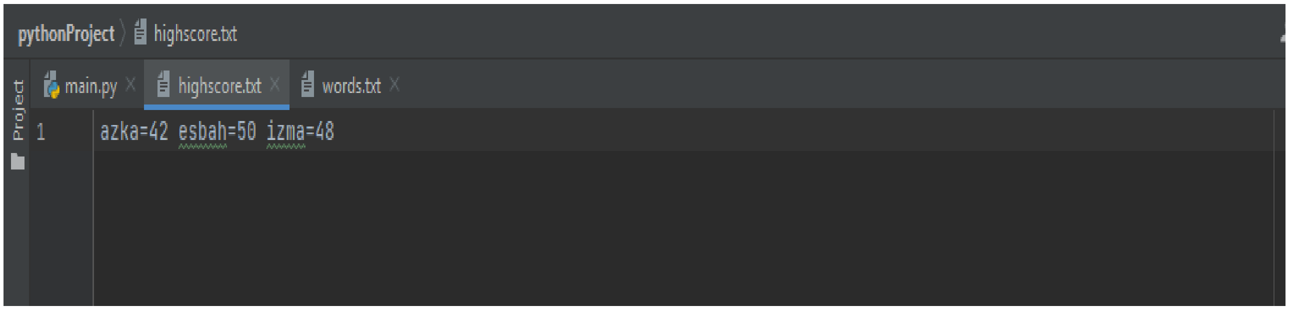
IN ADMIN MODE:



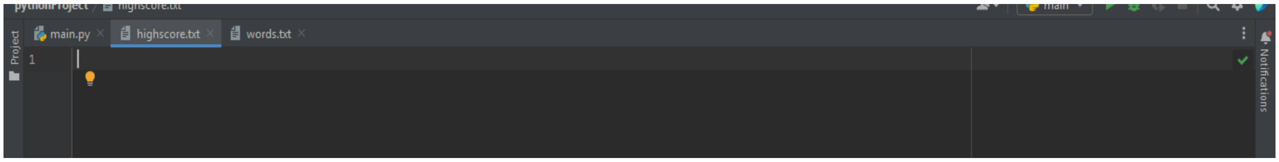
```
C:\ProgramData\Anaconda3\python.exe C:\Users\123\PycharmProjects\pythonProject\main.py
1=User
2=Admin
You are playing as: 2
1=Add a word in the text file
2=Reset the Highscore
3=To play the game
4=To Exit
Enter your choice: 1
Enter all the words with comma seperation: APPLE,RACKET,FOOTBALL
The words are added successfully
1=Want to Add more words in the text file
2=Reset the Highscore
3=To play the game
4=To Exit
Enter your choice: 2
Scores are reset
1=Want to Add more words in the text file
2=Reset the Highscore
3=To play the game
4=To Exit
Enter your choice: 4

Process finished with exit code 0
```



```
vigilantes vigilantly vigorously villainess villainies villainous vindicated vindicates vindicator vindictive violations violinists virginally
virtuously virulently viscerally visibility visitation visualized visualizes vitalities vitalizers vitalizing vitrifying viviparity
vocabulary vocalizers vocational vociferate vociferous voicedness voiceprint volatility volitional volubility volumetric voluminous volunteers
voluptuous vulcanizes vulgarizer vulgarizes vulnerable vulnerably wagonettes wainscoted waistbands waistcoats waitresses wallflower
wanderings wanderlust wantonness warehoused warehouser warehouses warmongers washboards washstands wastefully wastelands watchbands watchfully
watchwomen watchwords waterborne watercraft waterfalls waterfront watermarks waterproof watersheds waveringly weaknesses wealthiest
weathering weatherman weathermen weightiest weightless wellspring werewolves westerners westernize westwardly whaleboats whalebones wharfinger
wheelbases wheelchair wheeziness whensoever whetstones whimpering whiplashes whirligigs whirlwinds whirlybird whispering wholesaled
wholesales whomsoever wickedness widespread wildcatted wildebeest wilderness windlasses windowless windowpane windshield windstorms wingspread
wirehaired wirepuller wiretapped wiretapper witchcraft witcheries withdrawal withholder withstands witnessers witnessing witticisms
woefulness womanizers womanizing wonderland wonderment wondrously woodenness woolliness workaholic workhorses workingman workingmen worktables
worldlings worrywarts worshipers worshipful worshipped worthiness worthwhile woundingly wrathfully wretchedly wriggliest wristbands
wrongdoers wrongdoing wrongfully xenophobia xenophobic xerography xylophones yardmaster yardsticks yesterdays yesteryear youngsters yourselves
zoologists APPLE RACKET FOOTBALL
```



```
azka=42 esbah=50 izma=48
```


pythonProject / highscore.txt
main
main.py
highscore.txt
words.txt
Project
Notifications