**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.

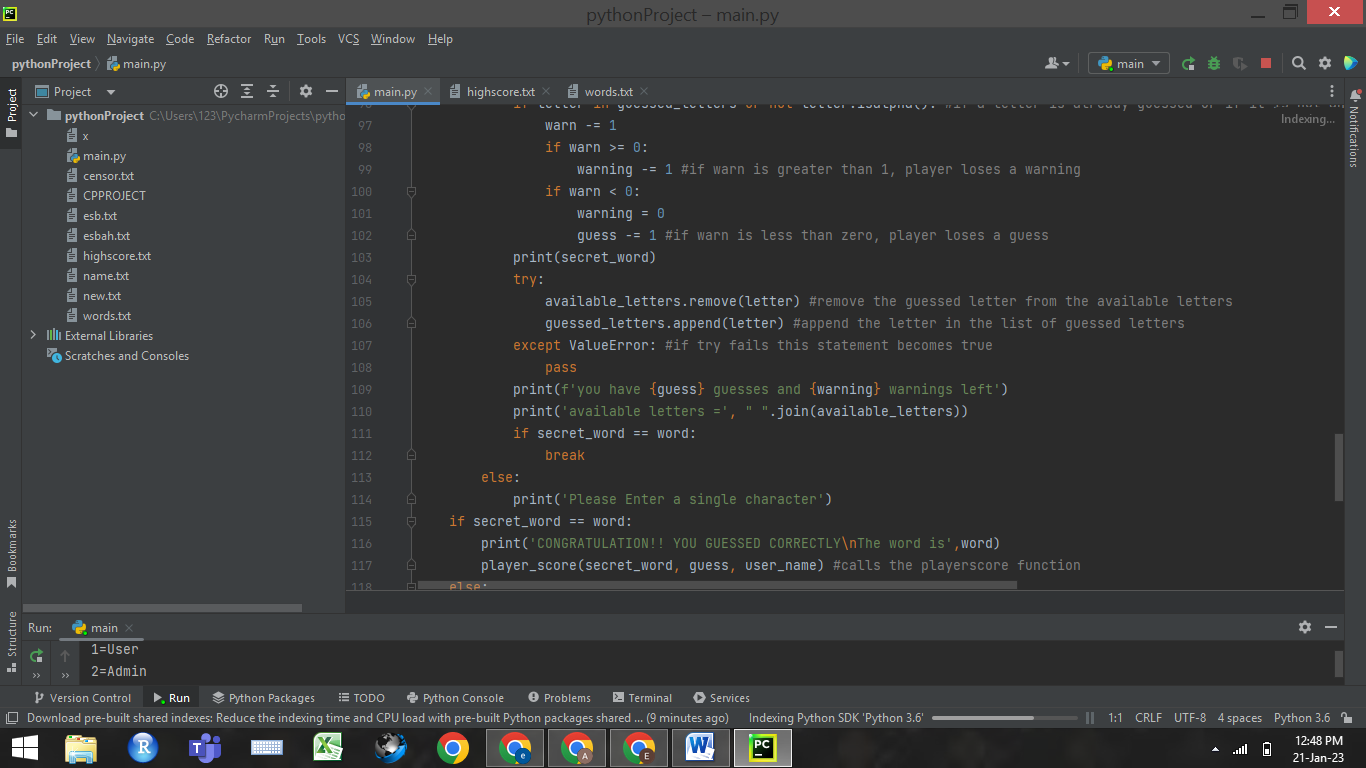
Our application prints a notification, if the letter predicted is incorrect, and the user loses one of their six guesses. The user is shown a list of alphabets before each turn that does not include the letter they just guessed, and our system also keeps track of the letter they guessed. User loses a 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.

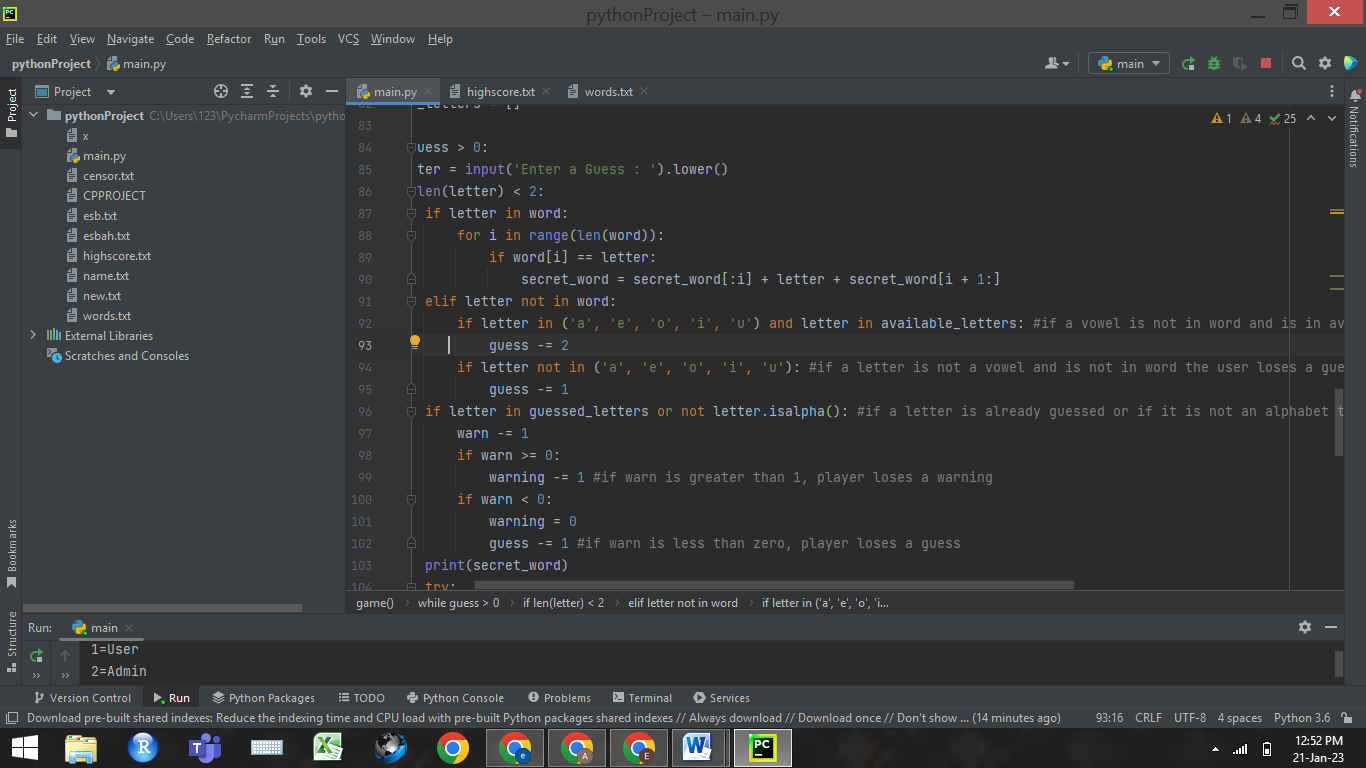
In contrast, our program's admin interface will give the admin the option to either add a word to the text file or reset the high score file.

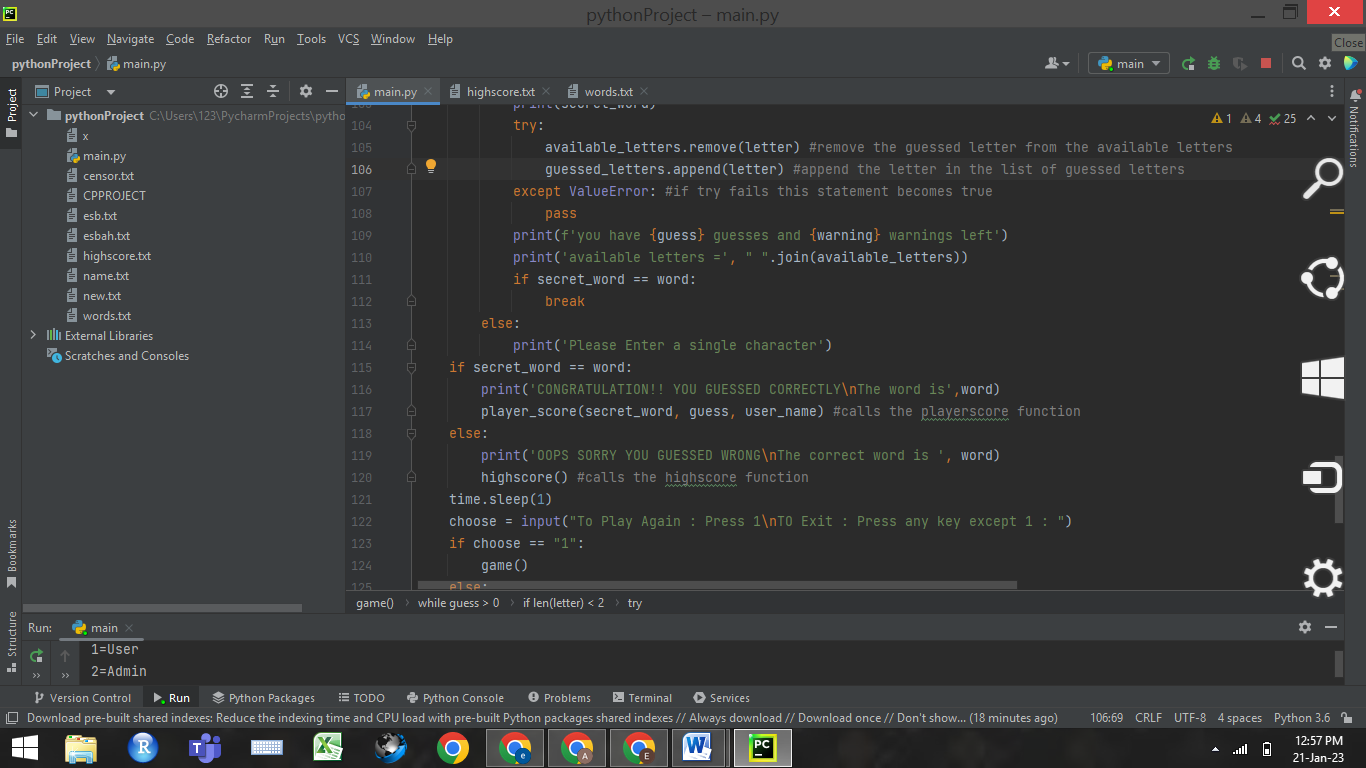
Following are the specific features of our program:

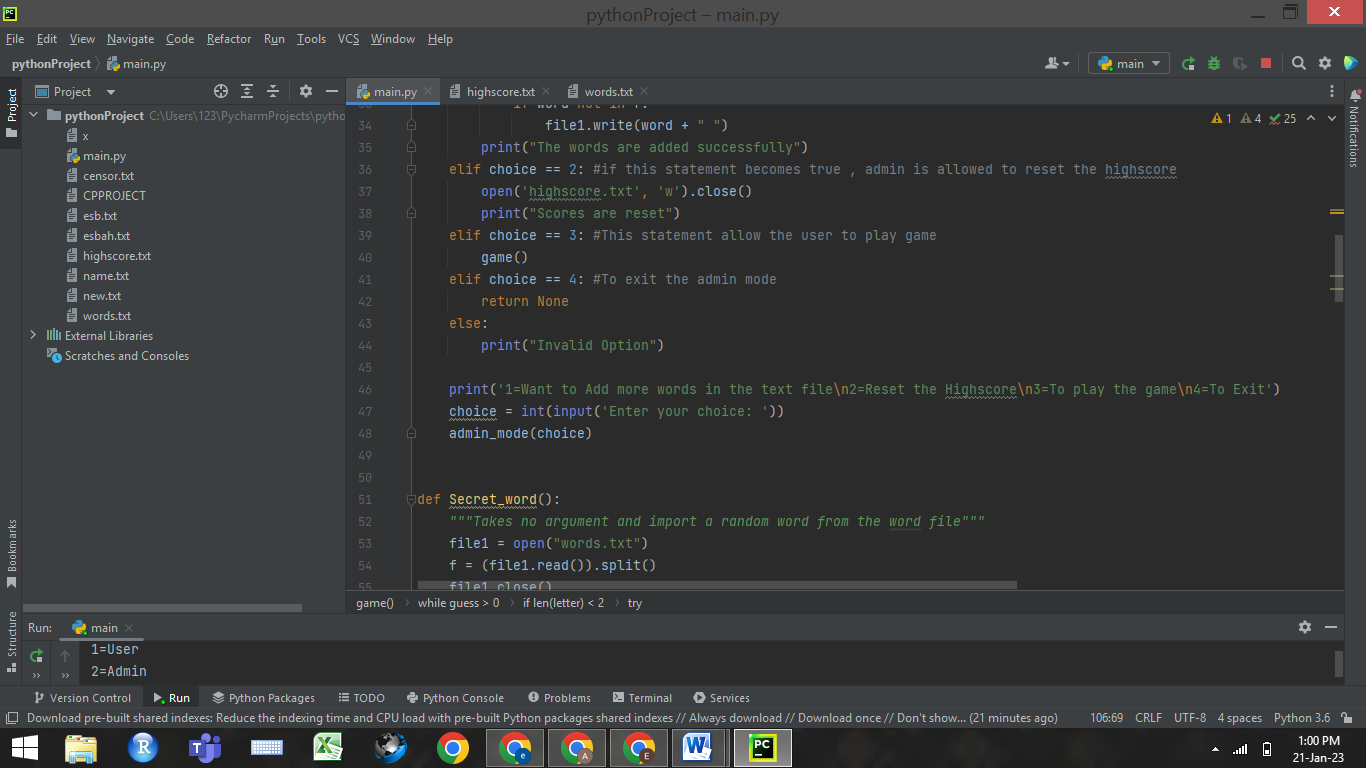
1. Our program logic is divided in to functions.
2. Game function allows user to play the game while admin mode allows the user to reset the high-score or to add word in the text file.
3. In user mode the user guesses the letter one by one, we initially provide user with 6 guesses and 3 warnings.
4. 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.

.







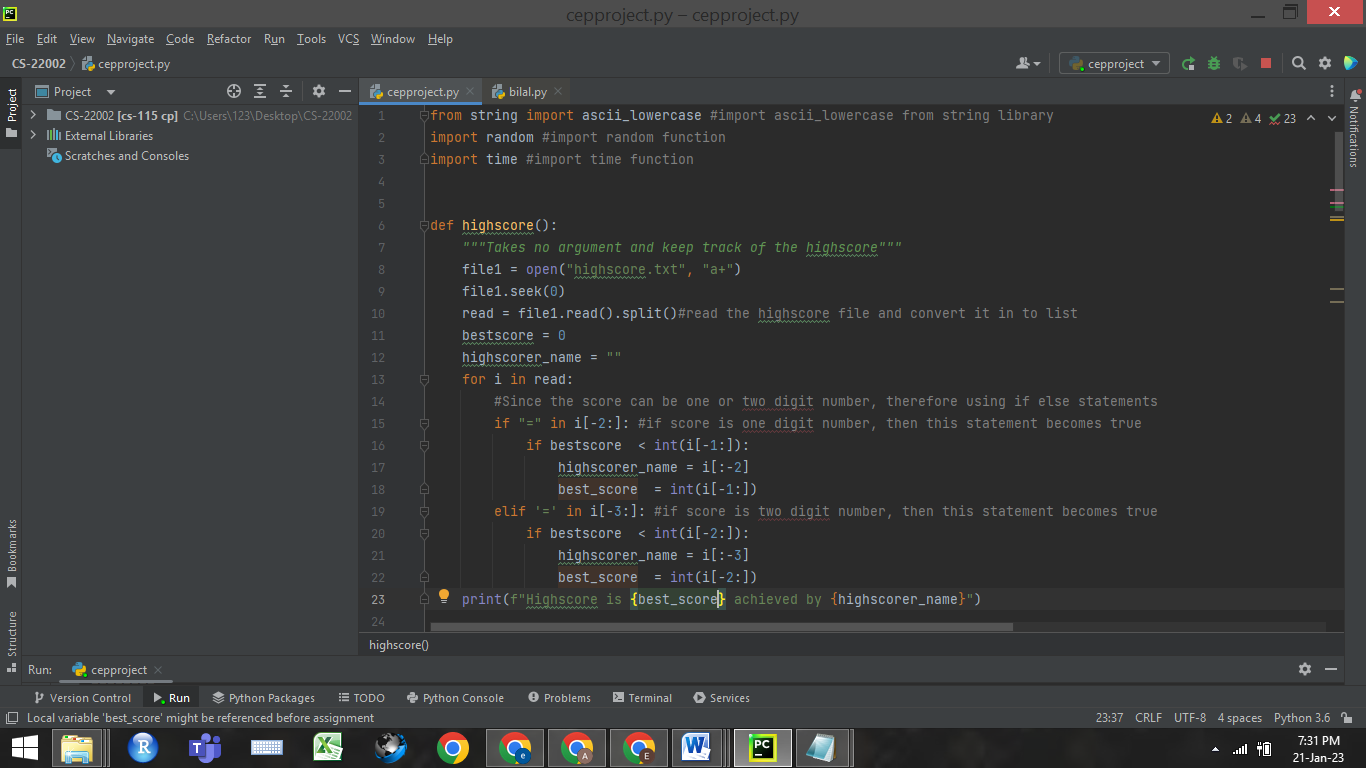


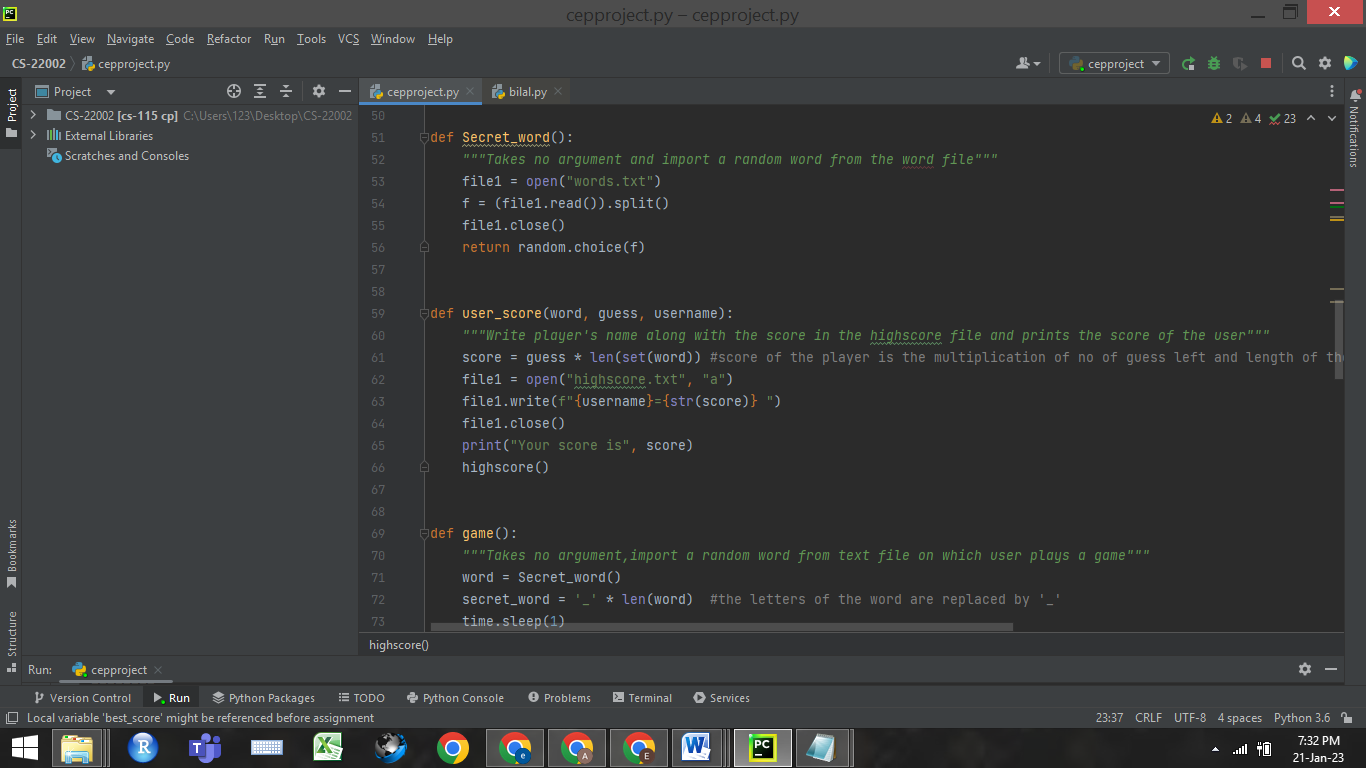
## FLOW OF OUR PROJECT:

**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





## 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:**

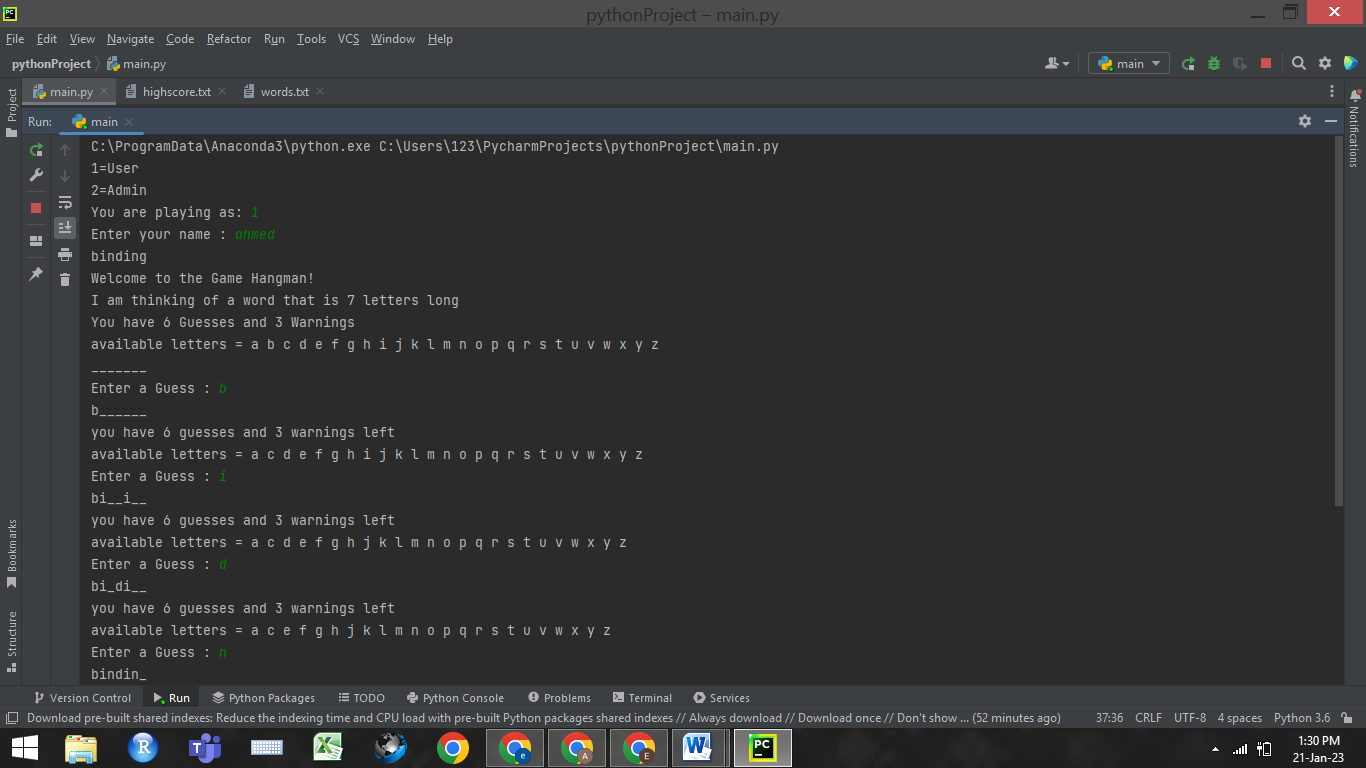
* Selecting random word from the file (secret word function).
* Appending word in text file.
* Resetting the high score.
* Keeping track of the guessed letters.

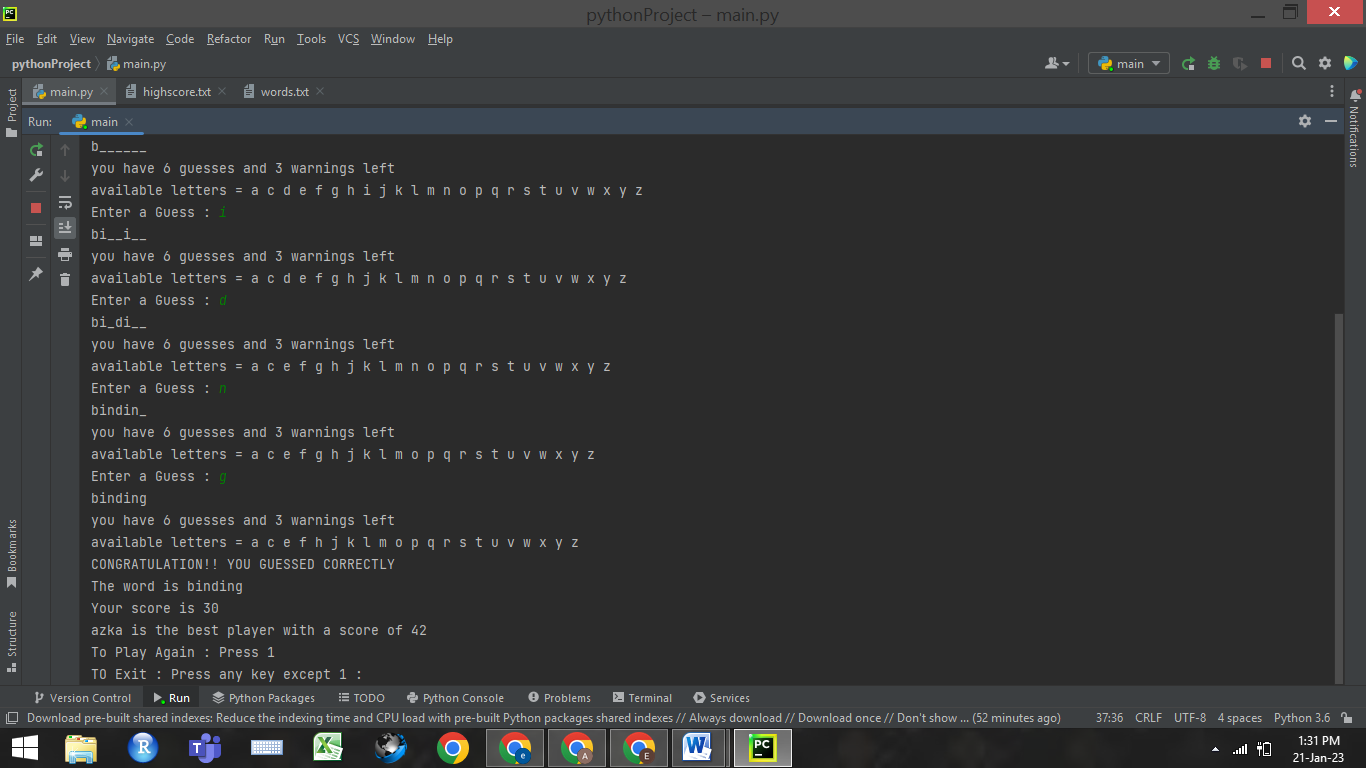
**CONTRIBUTION OF ESBAH:**

* 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:

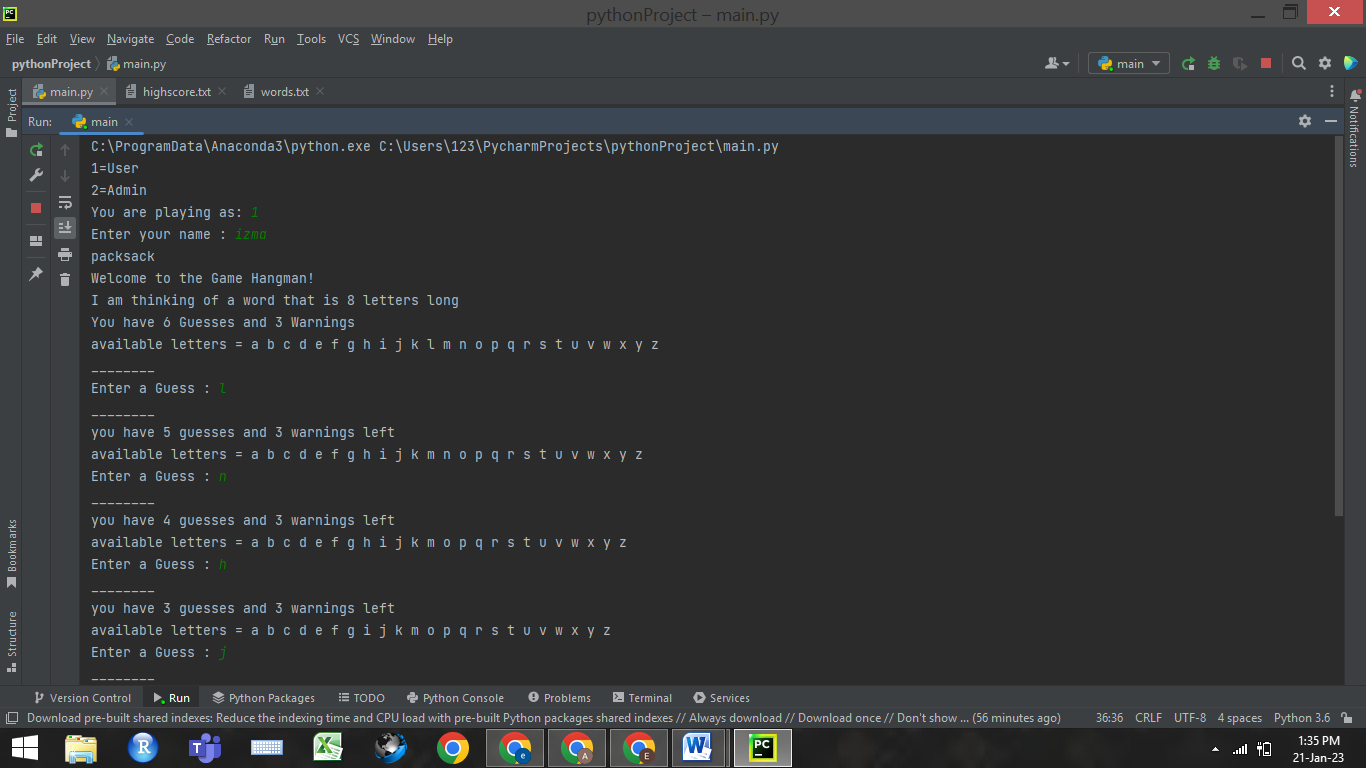
**WHEN USER WINS THE GAME:**

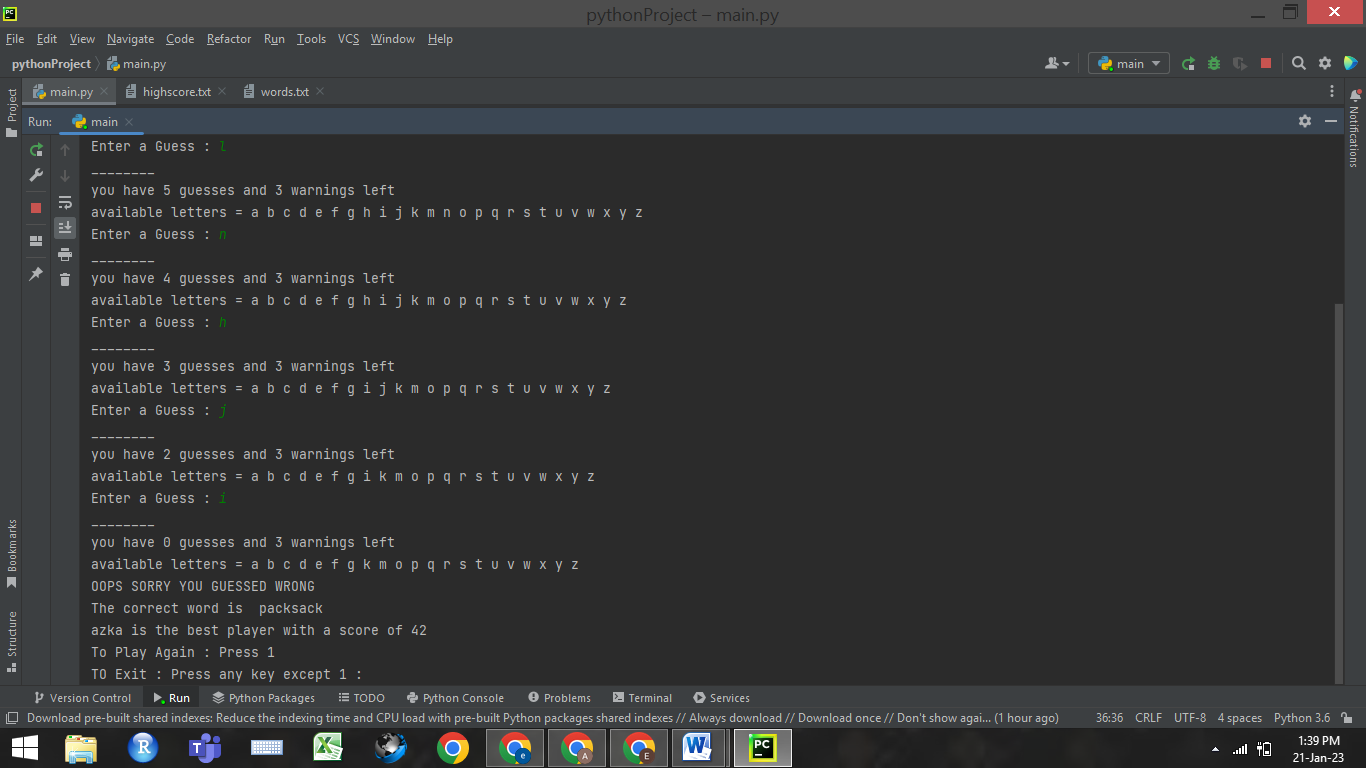




When user wins the game the program print a statement “CONGRATULATION!! YOU GUESSED CORRECTLTY”. It shows the score along with the name of the player and the high-score. It then allows the user to either exit the game or to play it again.

**WHEN USER LOSES THE GAME:**





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:**

