**Unit testing report**

**Submitted by**

**Group 5**

**Github link : https://github.com/Sayeed6424/PRT\_580\_Unit\_testting**

* **Introduction**

This work details the creation of a Python-based game that determines a player's Scrabble score depending on the words they enter. To get the most incredible score based on Scrabble letter values, players must input words of a certain length within a set amount of time in this challenging game. Python was chosen for the project's implementation because of its adaptability and threading capabilities, which were crucial for the countdown timer. The unit test framework was utilized to integrate automated testing to verify functionality. This paper describes the development process, the testing strategy, and the main results.

* **Process**

The Scrabble scoring system, in which each letter has a fixed point value, served as the foundation for the game. Letters like A, E, I, O, U, and L, for instance, are worth one point, whereas uncommon letters like Q and Z are worth ten. The primary objective is to ask the user to enter words with a count of letters that is created at random, assess the words' validity, and calculate a score based on the letter values. The game also includes a countdown timer to enhance a sense of urgency and difficulty.

* **the main components of the process**

1. **creating the letter scoring system**

A dictionary was defined to store the Scrabble values for each letter. This provided a simple and efficient way to retrieve the score for each letter during gameplay.

1. **Countdown Timer with Multithreading:**

The countdown functionality was achieved using the threading library, which allows two tasks (the countdown and the user input) to run simultaneously. One thread handles the countdown, while the other handles user input, ensuring that both tasks can occur concurrently.

1. **User input validation**

The game prompts users to enter a word of a specific length, which is randomly generated at the start of each round. The input is validated to ensure it contains only alphabetical characters and matches the required length. If the user enters an invalid word, they are prompted to try again, and they are given multiple attempts

1. **Calculate score**

The game adds up the values of each letter in a word after a valid word has been entered to get the overall score. The word that the user typed is shown with the score. Additionally, the application ends the countdown when the user enters a proper word.

* **Writing test**

Several unit tests were built using Python's unit test framework to guarantee the Scrabble game's functioning. Crucial elements, including score computation, input validation, and the countdown timer, were the focus of the testing. The tests verified that the software accurately calculated the total score for different words according to their letter values in order to calculate the score. The purpose of input validation tests was to confirm that the game only recognized alphabetic letters that were the right length and rejected other types of input. The countdown timer was also checked to make sure it matched user input accurately, pausing when a legitimate word was submitted. These examinations contributed to ensuring the game's dependability and precision.

* **Running test**

The unit test framework was used to automate the validation process while running the tests. Every test was created to confirm a particular feature of the game, and the framework made it simple to find errors or inconsistencies.

For example, the software was tested to make sure the letter values were added up correctly when the score calculation test was run, regardless of whether the input word was in capital or lowercase. Regarding input validation, tests were conducted on the software to make sure it rejected terms that needed to be longer or shorter.

* **Conclusion**

Test-driven creation (TDD) techniques and threading for real-time interaction were successfully applied to create the Scrabble-like Python game. The main problem was ensuring that the user input and countdown processes could operate simultaneously without interfering. The game operates efficiently, offering players an enjoyable and engaging experience thanks to multithreading and meticulous.  
  
The game's essential features, such as input validation and score computation, were all thoroughly tested thanks to the usage of the unit test framework. Future improvements include implementing a more powerful word-checking function that compares words against the whole English lexicon.