UNIT NAME: SOFTWARE ENGINEERING: PROCESS AND TOOLS (PRT582)

PROF: Charles Yeo

Assessment 1

SUBMITTED BY:

STUDENT NAME: Victor Dinakar Kankipati

STUDENT NUMBER: S343722

STUDY LOCATION: Casuarina Campus

Scrabble Score using TDD and Unit Testing

Student Name: Victor Dinakar Kankipati Student ID: S343722

Introduction

This project aims to develop a scrabble word scoring system based on certain point valued for each letter and the time taken by the user to input the word. Test Driven Development approach was used to develop the project. The programming language used is Python and the testing framework used for TDD and Unit Testing is Pytest.

Python was used to develop this project because of ease of use, popularity, availability of libraries, ease of string manipulation (as scrabble scoring is primarily string based). Pytest was used for testing because of its simplicity and ease of getting started with writing unit test cases and TDD.

Some of the main requirements are mentioned in the following table:

Requirement	Function
Scrabble scoring for a word	• Compute score for a word based on specific points for each letter in the alphabet. • Upper and lower case letters should have same value
Validating word	• Check if input word given by user contains only alphabets • Check if input word is a valid dictionary word • Check if input word is of required length • Prompt user with right feedback if invalid word is submitted
Scale scoring for time taken	• Scale score given for a valid word according to time taken to input word. Lesser the time, higher the score

Process

The process followed for development of each unit(function) was to start off with writing test cases for that unit. Initially, the function would have empty/hardcoded logic and would fail the test case(s). The function is then developed by writing the relevant code for it and is tested against the test cases. When the function passes all the test cases written for it, additional test cases are written (mostly to ensure it works for special/corner cases) to verify its correctness. Thus, each unit is developed to pass the test cases and unit tests are used to confirm correctness.

3 functions were developed using TDD:

- validate_word(word, word_length) : Used to validate if the string submitted by the user is a valid scrabble word based on the given requirements.
- score_word(word) : Used to score a valid scrabble word submitted by the user using the points system given in the requirements.
- score_word_timed(score, time_left): Used to scale the score using the time remaining (counter starts with 15 seconds left)

The screenshots of the code during the TDD and respective unit tests are given below for each of the above mentioned units. The screenshots showcase the journey of code of each unit.

validate_word(word, word_length)

Code:

```
def validate_word(word, word_length):
    return True
```

Test case:

```
def test_special_chars_1():
    assert validate_word('abc!', 4) == False
```

Test Result:

```
lenovo$ sudo pytest
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-6.2.5, py-1.10.0, pluggy-1.0.0
rootdir: /home/coding/scrabble
collected 1 item

test_scrabble.py F                                                       [100%]

=================================== FAILURES ===================================
_____________________________ test_special_chars_1 _____________________________

    def test_special_chars_1():
>       assert validate_word('abc!', 4) == False
E       AssertionError: assert True == False
E        +  where True = validate_word('abc!', 4)

test_scrabble.py:4: AssertionError
=========================== short test summary info ============================
FAILED test_scrabble.py::test_special_chars_1 - AssertionError: assert True ...
============================== 1 failed in 0.03s ===============================
```

Code:

```
def validate_word(word, word_length):
  if(len(word) != word_length):
    return False
  return word.isalpha()
```

Test Result:

```
lenovo$ sudo pytest
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-6.2.5, py-1.10.0, pluggy-1.0.0
rootdir: /home/coding/scrabble
collected 1 item

test_scrabble.py .                                                       [100%]

============================== 1 passed in 0.04s ===============================
```

Test Case:

```
def test_special_chars_1():
    assert validate_word('abc!', 4) == False

def test_special_chars_space():
    assert validate_word('a  c', 4) == False
```

Test Result:

```
lenovo$ sudo pytest
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-6.2.5, py-1.10.0, pluggy-1.0.0
rootdir: /home/coding/scrabble
collected 2 items

test_scrabble.py ..                                                      [100%]

============================== 2 passed in 0.04s ===============================
lenovo$
```

Test Case:

```
def test_special_chars_1():
    assert validate_word('abc!', 4) == False

def test_special_chars_space():
    assert validate_word('a  c', 4) == False

def test_dictionary():
    assert validate_word('abc', 3) == False
```

Test Result:

```
lenovo$ sudo pytest
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-6.2.5, py-1.10.0, pluggy-1.0.0
rootdir: /home/coding/scrabble
collected 3 items

test_scrabble.py ..F                                                     [100%]

=================================== FAILURES ===================================
_____________________________ test_dictionary ______________________________

    def test_dictionary():
>       assert validate_word('abc', 3) == False
E       AssertionError: assert True == False
E        +  where True = validate_word('abc', 3)

test_scrabble.py:10: AssertionError
=========================== short test summary info ============================
FAILED test_scrabble.py::test_dictionary - AssertionError: assert True == False
========================= 1 failed, 2 passed in 0.04s ==========================
```

Code:

```
def validate_word(word, word_length):
  if(len(word) != word_length):
    return False
  d = enchant.Dict("en_US")
  return word.isalpha() and d.check(word)
```

Test Result:

```
lenovo$ sudo pytest
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-6.2.5, py-1.10.0, pluggy-1.0.0
rootdir: /home/coding/scrabble
collected 3 items

test_scrabble.py ...                                                     [100%]

============================== 3 passed in 0.09s ===============================
lenovo$
```

Test:

```
def test_special_chars_1():
    assert validate_word('abc!', 4) == False

def test_special_chars_space():
    assert validate_word('a  c', 4) == False

def test_dictionary():
    assert validate_word('abc', 3) == False

def test_dictionary_2():
    assert validate_word('car', 3) == True
```

Test Result:

```
lenovo$ sudo pytest
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-6.2.5, py-1.10.0, pluggy-1.0.0
rootdir: /home/coding/scrabble
collected 4 items

test_scrabble.py ....                                                   [100%]

============================== 4 passed in 0.13s ===============================
lenovo$
```

Test:

```
def test_special_chars_1():
    assert validate_word('abc!', 4) == False

def test_special_chars_space():
    assert validate_word('a  c', 4) == False

def test_dictionary():
    assert validate_word('abc', 3) == False

def test_dictionary_2():
    assert validate_word('car', 3) == True

def test_dictionary_3():
    assert validate_word('testing', 7) == True
```

Test Result:

```
lenovo$ sudo pytest
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-6.2.5, py-1.10.0, pluggy-1.0.0
rootdir: /home/coding/scrabble
collected 5 items

test_scrabble.py .....                                                  [100%]

============================== 5 passed in 0.17s ===============================
lenovo$
```

Code (adding feedback promp to validation):

```
def validate_word(word, word_length):
  if(len(word) != word_length):
    print(f"Input word length {len(word)} not equal to {word_length}.\nPlease enter word with correct length")
    return False

  if(word.isalpha()):
    d = enchant.Dict("en_US")
    if d.check(word):
      return True
    else:
      print("Word not in dictionary.\nPlease give a valid word ")
  else:
    print("Word contains non-alphabetic characters.\nPlease give a valid word containing only alphabets")
```

Test Result:

```
lenovo$ sudo pytest
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-6.2.5, py-1.10.0, pluggy-1.0.0
rootdir: /home/coding/scrabble
collected 5 items

test_scrabble.py FFF..                                                   [100%]

=================================== FAILURES ===================================
_____________________________ test_special_chars_1 _____________________________

    def test_special_chars_1():
>       assert validate_word('abc!', 4) == False
E       AssertionError: assert None == False
E        +  where None = validate_word('abc!', 4)

test_scrabble.py:4: AssertionError
---------------------------- Captured stdout call -----------------------------
Word contains non-alphabetic characters.
Please give a valid word containing only alphabets
_____________________________ test_special_chars_space _____________________________

    def test_special_chars_space():
>       assert validate_word('a  c', 4) == False
E       AssertionError: assert None == False
E        +  where None = validate_word('a  c', 4)

test_scrabble.py:7: AssertionError
---------------------------- Captured stdout call -----------------------------
Word contains non-alphabetic characters.
Please give a valid word containing only alphabets
_____________________________ test_dictionary _____________________________

    def test_dictionary():
>       assert validate_word('abc', 3) == False
E       AssertionError: assert None == False
E        +  where None = validate_word('abc', 3)

test_scrabble.py:10: AssertionError
---------------------------- Captured stdout call -----------------------------
Word not in dictionary.
Please give a valid word
=========================== short test summary info ============================
FAILED test_scrabble.py::test_special_chars_1 - AssertionError: assert None ...
FAILED test_scrabble.py::test_special_chars_space - AssertionError: assert N...
FAILED test_scrabble.py::test_dictionary - AssertionError: assert None == False
========================= 3 failed, 2 passed in 0.11s ==========================
```

Code (fixing bug):

```
def validate_word(word, word_length):
  if(len(word) != word_length):
    print(f"Input word length {len(word)} not equal to {word_length}.\nPlease enter word with correct length")
    return False

  if(word.isalpha()):
    d = enchant.Dict("en_US")
    if d.check(word):
      return True
    else:
      print("Word not in dictionary.\nPlease give a valid word ")
      return False
  else:
    print("Word contains non-alphabetic characters.\nPlease give a valid word containing only alphabets")
    return False
```

Test Result:

```
lenovo$ sudo pytest
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-6.2.5, py-1.10.0, pluggy-1.0.0
rootdir: /home/coding/scrabble
collected 5 items

test_scrabble.py .....                                                   [100%]

============================== 5 passed in 0.17s ===============================
lenovo$
```

---

score_word(word)

Code:

```
def score_word(word):
  return 0
```

Test:

```
def test_score_word_1():
    assert score_word('car') == 5
```

Test Result:

```
lenovo$ sudo pytest
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-6.2.5, py-1.10.0, pluggy-1.0.0
rootdir: /home/coding/scrabble
collected 6 items

test_scrabble.py .....F                                                  [100%]

=================================== FAILURES ===================================
_____________________________ test_score_word_1 _____________________________

    def test_score_word_1():
>       assert score_word('car') == 5
E       AssertionError: assert 0 == 5
E        +  where 0 = score_word('car')

test_scrabble.py:19: AssertionError
=========================== short test summary info ============================
FAILED test_scrabble.py::test_score_word_1 - AssertionError: assert 0 == 5
========================== 1 failed, 5 passed in 0.19s =========================
lenovo$
```

Code:

```
def score_word(word):
  if not validate_word(word):
    raise ValueError
  point_total = 0
  for letter in word:
    point_total += points_dic[letter]
  return point_total
```

Test Result:

```
lenovo$ sudo pytest
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-6.2.5, py-1.10.0, pluggy-1.0.0
rootdir: /home/coding/scrabble
collected 6 items

test_scrabble.py ......                                                  [100%]

============================== 6 passed in 0.23s ===============================
lenovo$
```

Test:

```
def test_score_word_1():
    assert score_word('car') == 5

def test_score_word_2():
    assert score_word('market') == 12

def test_score_word_3():
    assert score_word('cabbage') == 14

def test_score_word_4():
    assert score_word('zebra') == 16

def test_score_word_5():
    assert score_word('justice') == 16

def test_score_word_6():
    assert score_word('quiz') == 22

def test_score_word_7():
    assert score_word('formula') == 12
```

Test Result:

```
lenovo$ sudo pytest
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-6.2.5, py-1.10.0, pluggy-1.0.0
rootdir: /home/coding/scrabble
collected 12 items

test_scrabble.py ............                                            [100%]

============================== 12 passed in 0.48s ==============================
lenovo$
```

---

score_word_timed(score, time_left):

Code:

```
def score_word_timed(score, time_left):
  return 0
```

Test:

```
def test_timed_score_1():
    assert score_word_timed(10, 12) == 8
```

Test Result:

```
lenovo$ sudo pytest
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-6.2.5, py-1.10.0, pluggy-1.0.0
rootdir: /home/coding/scrabble
collected 13 items

test_scrabble.py ............F                                          [100%]

=================================== FAILURES ===================================
_____________________________ test_timed_score_1 ______________________________

    def test_timed_score_1():
>       assert score_word_timed(10, 12) == 8
E       assert 0 == 8
E        +  where 0 = score_word_timed(10, 12)

test_scrabble.py:40: AssertionError
=========================== short test summary info ============================
FAILED test_scrabble.py::test_timed_score_1 - assert 0 == 8
========================= 1 failed, 12 passed in 0.48s =========================
lenovo$
```

Code:

```
def score_word_timed(score, time_left):
  return int(score*(time_left/15))
```

Test Result:

```
lenovo$ sudo pytest
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-6.2.5, py-1.10.0, pluggy-1.0.0
rootdir: /home/coding/scrabble
collected 13 items

test_scrabble.py .............                                            [100%]

============================== 13 passed in 0.53s ==============================
lenovo$
```

Test:

```
def test_timed_score_1():
    assert score_word_timed(10, 12) == 8

def test_timed_score_2():
    assert score_word_timed(10, 0) == 0

def test_timed_score_3():
    assert score_word_timed(20, 5) == 6
```

Test Result:

```
lenovo$ sudo pytest
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-6.2.5, py-1.10.0, pluggy-1.0.0
rootdir: /home/coding/scrabble
collected 15 items

test_scrabble.py ...............                                         [100%]

============================== 15 passed in 0.51s ==============================
lenovo$
```

Conclusion

Test Driven Development was found to be beneficial in developing the code with different inputs and scenarios in mind. Writing test cases before writing code has led to developing code that is less prone to missing out on some edge cases. Also, writing test cases helped in finding bugs quickly and resolving them (as shown in one of the screenshots). It is learnt that writing good test cases is key for TDD to be effective. TDD's ability in finding bugs quickly might lead to increasing the pace of development as bugs are found right away when the code is in nascent stages.

Overall, TDD helps in delivering a robust product and increases the pace of development if the test cases are designed smartly.

Github link: https://github.com/victor100k/Scrabble-Score.git