ACS 567 Software Project Management

HW7

Github Repository Link (for code):

https://github.com/manchalakavya/SPM-HW-7

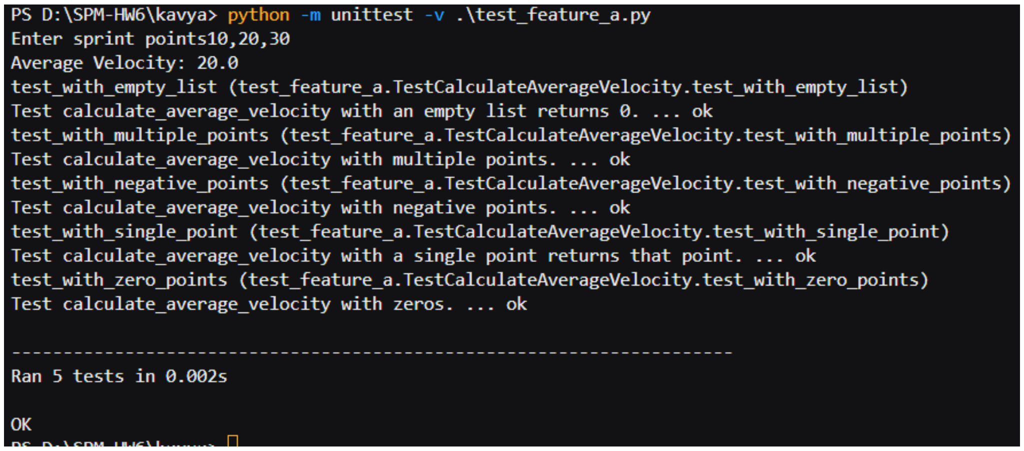
Execution of the codes are mentioned in Readme.md

Task 1: Create a unit test for a method that is used in Feature A (this should handle all possible paths).

test_feature_a.py

Screenshots :

Feature A:



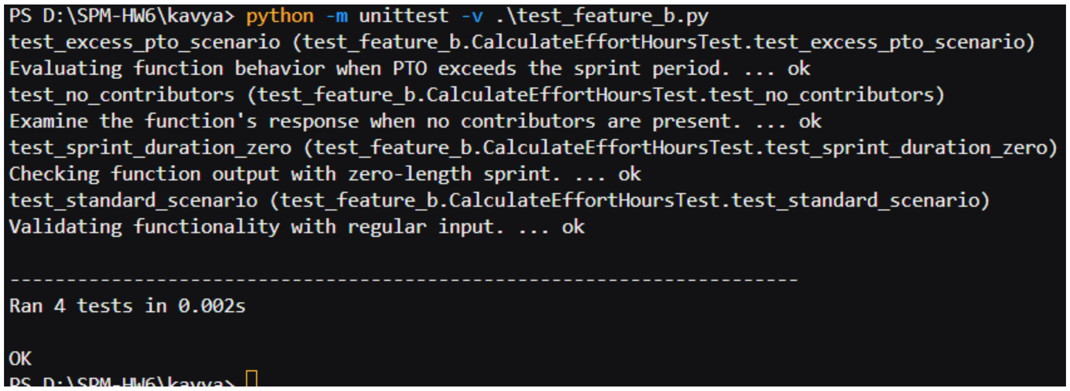
```
PS D:\SPM-HW6\kavya> python -m unittest -v .\test_feature_a.py
Enter sprint points10,20,30
Average Velocity: 20.0
test_with_empty_list (test_feature_a.TestCalculateAverageVelocity.test_with_empty_list)
Test calculate_average_velocity with an empty list returns 0. ... ok
test_with_multiple_points (test_feature_a.TestCalculateAverageVelocity.test_with_multiple_points)
Test calculate_average_velocity with multiple points. ... ok
test_with_negative_points (test_feature_a.TestCalculateAverageVelocity.test_with_negative_points)
Test calculate_average_velocity with negative points. ... ok
test_with_single_point (test_feature_a.TestCalculateAverageVelocity.test_with_single_point)
Test calculate_average_velocity with a single point returns that point. ... ok
test_with_zero_points (test_feature_a.TestCalculateAverageVelocity.test_with_zero_points)
Test calculate_average_velocity with zeros. ... ok

----------------------------------------------------------------------
Ran 5 tests in 0.002s

OK
```

Task 2 : Create a unit test for a method that is used in Feature b (this should handle all possible paths).

test_feature_b.py

Screenshots :

Feature B:



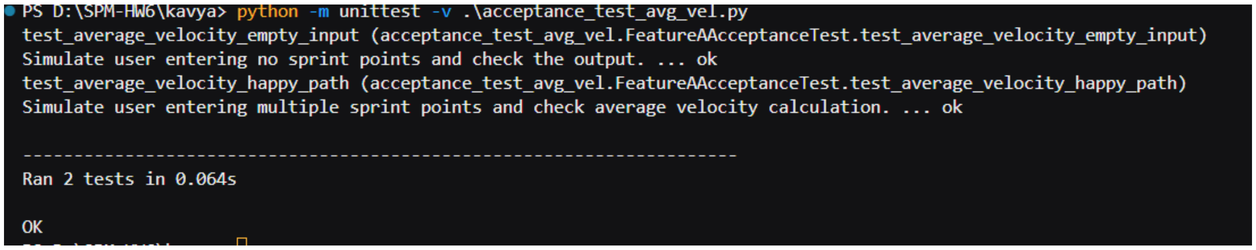
```
PS D:\SPM-HW6\kavya> python -m unittest -v .\test_feature_b.py
test_excess_pto_scenario (test_feature_b.CalculateEffortHoursTest.test_excess_pto_scenario)
Evaluating function behavior when PTO exceeds the sprint period. ... ok
test_no_contributors (test_feature_b.CalculateEffortHoursTest.test_no_contributors)
Examine the function's response when no contributors are present. ... ok
test_sprint_duration_zero (test_feature_b.CalculateEffortHoursTest.test_sprint_duration_zero)
Checking function output with zero-length sprint. ... ok
test_standard_scenario (test_feature_b.CalculateEffortHoursTest.test_standard_scenario)
Validating functionality with regular input. ... ok

----------------------------------------------------------------------
Ran 4 tests in 0.002s

OK
```

Task 3: Write an acceptance test for Feature A (this could be happy path, or unhappy path…but must exercise more than a single method). Format of acceptance test is up to you

acceptance_test_avg_vel.py

Code is for Automated Acceptance Testing

Scrnshot:



```
PS D:\SPM-HW6\kavya> python -m unittest -v .\acceptance_test_avg_vel.py
test_average_velocity_empty_input (acceptance_test_avg_vel.FeatureAAcceptanceTest.test_average_velocity_empty_input)
Simulate user entering no sprint points and check the output. ... ok
test_average_velocity_happy_path (acceptance_test_avg_vel.FeatureAAcceptanceTest.test_average_velocity_happy_path)
Simulate user entering multiple sprint points and check average velocity calculation. ... ok

----------------------------------------------------------------------
Ran 2 tests in 0.064s

OK
```

Manual Acceptance Testing for Average Velocity Calculation (Feature A)

Test Environment Setup

Application Under Test: Avg_Vel.py

Pre-requisites: Python 3.x installed on the testing environment.

Execution Method: Running a Python script that prompts for sprint points and outputs the average velocity.

Test Scenarios

Scenario 1: Happy Path (Valid Sprint Points)

Objective: Verify that the script accurately calculates the average velocity with a valid series of sprint points.

Input: A series of sprint points, e.g., "10, 20, 30".

Steps:

- Start the script.
- Enter the sprint points as prompted: "10,20,30".
- Expected Outcome: The script calculates and displays "Average Velocity: 20.0".
- Verification: Manually confirm that the output matches the expected result.

Scenario 2: Unhappy Path (Empty Input)

Objective: Ensure the script gracefully handles an empty input scenario.

Input: No sprint points (simply press Enter at the prompt).

Steps:

- Run the script.
- Press Enter without typing any sprint points when prompted for input.
- Expected Outcome: The script displays "Average Velocity: 0" indicating no sprint points were entered.
- Verification: Check that the script's output accurately reflects the scenario of no sprint points being provided.

Scenario 3: Handling Non-numeric Input

Objective: Test the script's response to non-numeric input among sprint points.

Input: A mix of numeric and non-numeric inputs, e.g., "10, abc, 30".

Steps:

- Execute the script.
- Enter "10,abc,30" when prompted for sprint points.
- Expected Outcome: The script either displays an error message indicating invalid input or excludes non-numeric inputs and calculates the average based on numeric values only, depending on the designed behavior.
- Verification: Confirm the script behaves as expected in the presence of non-numeric inputs, according to the specified handling method.

Execution Instructions

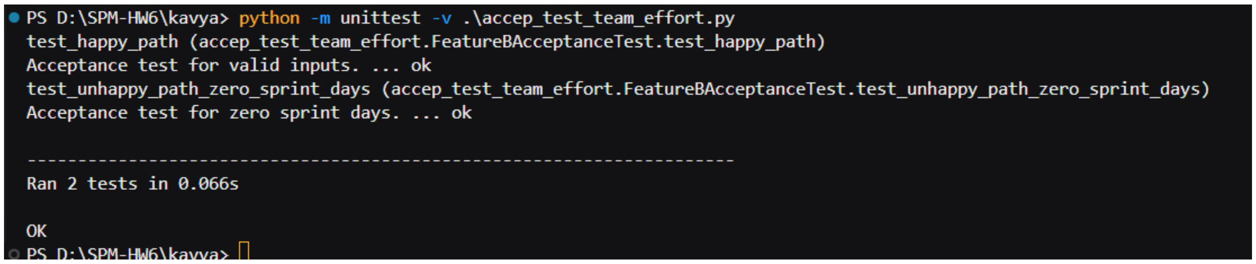
- Navigate to the directory containing the script in a command-line interface.
- Type python Avg_Vel.py and press Enter.
- Follow the on-screen prompts to enter test data for each scenario.

Task 4: Write an acceptance test for Feature B (this could be happy path, or unhappy path…but must exercise more than a single method). Format of acceptance test is up to you

accept_test_team_effort.py

Code is for Automated Acceptance Testing

Scrnshot:



```
PS D:\SPM-HW6\kavya> python -m unittest -v .\accep_test_team_effort.py
test_happy_path (accep_test_team_effort.FeatureBAcceptanceTest.test_happy_path)
Acceptance test for valid inputs. ... ok
test_unhappy_path_zero_sprint_days (accep_test_team_effort.FeatureBAcceptanceTest.test_unhappy_path_zero_sprint_days)
Acceptance test for zero sprint days. ... ok

----------------------------------------------------------------------
Ran 2 tests in 0.066s

OK
PS D:\SPM-HW6\kavya>
```

Manual Acceptance Testing for Team Effort Calculation Feature (Feature B)

Test Environment Setup

Application Under Test: Team Effort Calculation Script (Team_Effort.py)

Pre-requisites: Python 3.x installed on the testing environment.

Required Data: Team member details including names, planned time off (PTO) hours, ceremony hours, and available hours per day.

Test Scenarios

Scenario 1: Happy Path (Valid Team Data)

Objective: Verify the script accurately calculates total and individual effort hours with valid team data.

Input:

Sprint Days: 10

Team Members:

Leo: PTO hours = 2, Ceremony hours = 2, Available hours per day = 8

Riya: PTO hours = 4, Ceremony hours = 3, Available hours per day = 8

Steps:

- Run the script.
- Enter the inputs as prompted by the script.

Expected Outcome:

- The script outputs individual effort hours for Leo and Riya correctly.
- The script calculates and displays the total available effort hours for the team.

Verification: Manually compare the script's output with calculated values.

Scenario 2: Unhappy Path (Zero Sprint Days)

Objective: Ensure the script handles a scenario of zero sprint days gracefully.

Input:

Sprint Days: 0

Team Members: At least one team member with any set of PTO, ceremony, and available hours.

Steps:

- Run the script.
- Enter 0 for sprint days and proceed to enter any team member data as prompted.

Expected Outcome: The script reports zero total available effort hours for the team, regardless of the team member data provided.

Verification: Check if the script's output matches the expected outcome.

Scenario 3: No Team Members

Objective: Test the script's behavior when no team members are entered.

Input:

Sprint Days: Any positive number

Team Members: None

Steps:

- Run the script.
- Enter sprint days and 0 for the number of team members when prompted.

Expected Outcome: The script displays that the total available effort hours for the team are 0.

Verification: Ensure the script does not crash and correctly outputs the expected message.

Execution Instructions

- Open a command line interface or terminal in the directory containing the script.
- Execute the script using the command: python team_effort.py.
- Follow on-screen prompts to enter test data for each scenario.