BMI Code Test Plan

Prepared by: Michael Swan

**Introduction**

The purpose of this test plan is to showcase the framework for validating the functionality and accuracy of the BMI calculator codebase. The codebase was designed to provide the users with a dependable resource for calculating their BMI based on user inputs for weight and height data while categorizing their results into the standard health categorization.

**Scope of Testing**

In scope: Input validation for negative numbers and in range values, correct conversion of height to cm and incorrect conversion of height to cm. incorrect conversion of weight to kg

Out of scope: Speed of testing, extreme input values, and decimal values

**Testing approaches**

Unit Testing: The objective of the unit testing is to validate the accuracy of the individual functions and methods within the BMI calculator. This will focus on ensuring that the valid and invalid inputs of the code are correctly handled. This will also focus on ensuring that some of the calculations are properly carried out.

Integration Testing: The objective of integration testing is to ensure that the different components of the codebase work together. This will focus on verifying that the user actions trigger the expected responses throughout the code.

**Test Schedule**

Unit Testing:

Planning phase: 1 week

Execution Phase: 1 week

Review Phase: 2 weeks

Integration Testing:

Planning Phase: 1 week

Execution Phase: 2 weeks

Review Phase: 2 weeks

**Risks & Issues**

Risk: Misinterpretation of input values leading to inaccurate results

Mitigation: Implement clear input guidelines and error messages.

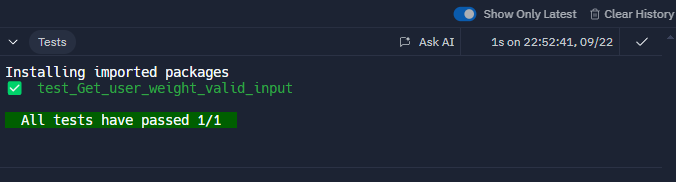
I chose to use unit testing because I felt there are a good number of values in the code that should be checked to ensure the accuracy of the code.

Test Case 1: I am testing to see if I will receive the error message for entering a negative number in the height value. I am testing this to ensure that my code can recognize errors and display the proper message. The input for this test will be -2 and the expected output should be “Height in feet must be a non-negative number”

A screenshot of a computer program

Description automatically generated

Test Case 2: I am testing to see if my code can recognize a valid input when it is entered for the user weight input. I am testing this to ensure that my code can recognize valid inputs when they are entered so that the code runs smoothly. The input for this test will be 150. The expected outcome should be test passed.



Test Case 3: I am testing to see if the conversion of height to cm is accurate. I am testing this to ensure that my program is accurately calculating the conversion of the height to centimeters. The input for this test will be 5 for the height in feet and 8 for the height in inches. The expected outcome is 172.72.

A screenshot of a computer

Description automatically generated

Test Case 4: I am testing to see if the program will recognize the inaccurate conversion of height to cm. I am testing this to ensure that the program accurately converts height to centimeters while also recognizing any errors in the conversion process. The inputs for this test will be 5 for height in feet and 8 for height in inches. The expected cm for this test will be 162.72 so the conversion will not be the same. The expected outcome should be that 162.72 does not equal 172.72 with an error message.

A screenshot of a computer program

Description automatically generated

Test Case 5: I am testing to see if the program will accurately convert the weight from pounds to kilograms. I am testing this by providing the code with an incorrect conversion. I am testing this to ensure that the code accurately calculates the conversion of weight from pounds to kilograms. My input will be 150 for weight in pounds and my input for expected kg will be 136.0776. The expected outcome should be 68.0388 does not equal 136.0776 with an error message.

A screenshot of a computer program

Description automatically generated