# ASSIGNMENT 8.1

**TASK 1:**

(Password Strength Validator – Apply AI in  
Security Context)  
• Task: Apply AI to generate at least 3 assert test cases for  
is\_strong\_password(password) and implement the validator  
function.  
• Requirements:  
o Password must have at least 8 characters.  
o Must include uppercase, lowercase, digit, and special  
character.  
o Must not contain spaces.  
Example Assert Test Cases:  
assert is\_strong\_password("Abcd@123") == True  
assert is\_strong\_password("abcd123") == False  
assert is\_strong\_password("ABCD@1234") == True  
Expected Output #1:  
• Password validation logic passing all AI-generated test cases

**PROMPT:**

(Password Strength Validator – Apply AI in Security Context) • Task: Apply AI to generate at least 3 assert test cases for is\_strong\_password(password) and implement the validator function. • Requirements: o Password must have at least 8 characters. o Must include uppercase, lowercase, digit, and special character. o Must not contain spaces.Example Assert Test Cases:assert is\_strong\_password("Abcd@123") == True  
assert is\_strong\_password("abcd123") == False assert is\_strong\_password("ABCD@1234") == True Expected Output #1:  
• Password validation logic passing all AI-generated test cases

**CODE AND OUTPUT:**

A computer screen shot of a program code

Description automatically generated

****

**OBSERVAION:** In this code we observe that the Inventory class correctly manages stock by allowing items to be added, removed. It handles edge cases such as removing more items than available and checking stock for non-existent items. All assert test cases pass, confirming the class works.

**Task 2:**

(Number Classification with Loops – Apply AI for  
Edge Case Handling)  
• Task: Use AI to generate at least 3 assert test cases for a  
classify\_number(n) function. Implement using loops.  
• Requirements:  
o Classify numbers as Positive, Negative, or Zero.  
o Handle invalid inputs like strings and None.

o Include boundary conditions (-1, 0, 1).  
Example Assert Test Cases:  
assert classify\_number(10) == "Positive"  
assert classify\_number(-5) == "Negative"  
assert classify\_number(0) == "Zero"  
Expected Output #2:  
• Classification logic passing all assert tests

**PROMPT:**

(Number Classification with Loops – Apply AI for Edge Case Handling)  
• Task: Use AI to generate at least 3 assert test cases for a classify\_number(n) function. Implement using loops. • Requirements: o Classify numbers as Positive, Negative, or Zero. o Handle invalid inputs like strings and None. o Include boundary conditions (-1, 0, 1). Example Assert Test Cases: assert classify\_number(10) == "Positive"  
assert classify\_number(-5) == "Negative" assert classify\_number(0) == "Zero"  
Expected Output #2:• Classification logic passing all assert tests

**CODE AND OUTPUT:**

**A computer screen shot of a program

Description automatically generated**

**A black background with white text

Description automatically generated**

**OBSERVATION:** In this code we observe thatthe date validation function checks if the date is real and in the correct format. It gives back the date in "YYYY-MM-DD" if valid, or "Invalid Date" if not. All tests work, so the function is correct.

**TASK 3:** (Anagram Checker – Apply AI for String Analysis)  
• Task: Use AI to generate at least 3 assert test cases for  
is\_anagram(str1, str2) and implement the function.  
• Requirements:  
o Ignore case, spaces, and punctuation.  
o Handle edge cases (empty strings, identical words).  
Example Assert Test Cases:  
assert is\_anagram("listen", "silent") == True  
assert is\_anagram("hello", "world") == False  
assert is\_anagram("Dormitory", "Dirty Room") == True  
Expected Output #3:  
• Function correctly identifying anagrams and passing all AI-  
generated tests.

**PROMPT:**

(Anagram Checker – Apply AI for String Analysis) • Task: Use AI to generate at least 3 assert test cases for is\_anagram(str1, str2) and implement the function. • Requirements: o Ignore case, spaces, and punctuation. o Handle edge cases (empty strings, identical words). Example Assert Test Cases: assert is\_anagram("listen", "silent") == True assert is\_anagram("hello", "world") == False assert is\_anagram("Dormitory", "Dirty Room") == True Expected Output #3:  
• Function correctly identifying anagrams and passing all AI-generated tests.

**CODE AND OUTPUT:**

**A computer screen shot of a program code

Description automatically generated**

**A black screen with white text

Description automatically generated**

**OBSERVATION**: In this code We can see that the code is checking if the word is an anagram or not

**Task 4**: (Inventory Class – Apply AI to Simulate Real-  
World Inventory System)  
• Task: Ask AI to generate at least 3 assert-based tests for an  
Inventory class with stock management.  
• Methods:  
o add\_item(name, quantity)  
o remove\_item(name, quantity)  
o get\_stock(name)  
Example Assert Test Cases:  
inv = Inventory()  
inv.add\_item("Pen", 10)  
assert inv.get\_stock("Pen") == 10  
inv.remove\_item("Pen", 5)  
assert inv.get\_stock("Pen") == 5  
inv.add\_item("Book", 3)  
assert inv.get\_stock("Book") == 3  
Expected Output #4:  
• Fully functional class passing all assertions.

Task Description #5 (Date Validation & Formatting – Apply AI for  
Data Validation)  
• Task: Use AI to generate at least 3 assert test cases for  
validate\_and\_format\_date(date\_str) to check and convert dates.  
• Requirements:  
o Validate "MM/DD/YYYY" format.  
o Handle invalid dates.  
o Convert valid dates to "YYYY-MM-DD".  
Example Assert Test Cases:  
assert validate\_and\_format\_date("10/15/2023") == "2023-10-15"  
assert validate\_and\_format\_date("02/30/2023") == "Invalid Date"  
assert validate\_and\_format\_date("01/01/2024") == "2024-01-01"

**PROMPT:**

Task Description #4 (Inventory Class – Apply AI to Simulate Real-  
World Inventory System)• Task: Ask AI to generate at least 3 assert-based tests for an  
Inventory class with stock management.• Methods:o add\_item(name, quantity)o remove\_item(name, quantity)o get\_stock(name)  
Example Assert Test Cases: inv = Inventory() inv.add\_item("Pen", 10) assert inv.get\_stock("Pen") == 10 inv.remove\_item("Pen", 5)  
assert inv.get\_stock("Pen") == 5 inv.add\_item("Book", 3) assert inv.get\_stock("Book") == 3 Expected Output #4:  
• Fully functional class passing all assertions. Task Description #5 (Date Validation & Formatting – Apply AI for Data Validation) • Task: Use AI to generate at least 3 assert test cases for validate\_and\_format\_date(date\_str) to check and convert dates. • Requirements: o Validate "MM/DD/YYYY" format. o Handle invalid dates. o Convert valid dates to "YYYY-MM-DD". Example Assert Test Cases: assert validate\_and\_format\_date("10/15/2023") == "2023-10-15" assert validate\_and\_format\_date("02/30/2023") == "Invalid Date" assert validate\_and\_format\_date("01/01/2024") == "2024-01-01"

**CODE:**

**A screen shot of a computer program

Description automatically generatedA screen shot of a computer program

Description automatically generatedA screen shot of a computer program

Description automatically generated**

**OUTPUT:**

**A screenshot of a computer program

Description automatically generated**

**OBSERVATION:** In this code we observe that adding new items and updating quantities. Removing items partially and attempting to remove more than available stock. The Inventory class works correctly for adding, removing, and checking stock items. The test cases showed that it can handle normal cases and edge cases, such as removing more items than available or checking items that are not in stock.

Expected Output #5:  
• Function passes all AI-generated assertions and handles edge  
cases.  
 Deliverables (For All Tasks)  
1. AI-generated prompts for code and test case generation.  
2. At least 3 assert test cases for each task.  
3. AI-generated initial code and execution screenshots.  
4. Analysis of whether code passes all tests.  
5. Improved final version with inline comments and explanation.  
6. Compiled report (Word/PDF) with prompts, test cases, assertions,  
code, and output.

**PROMPT:**

• Function passes all AI-generated assertions and handles edgecases. Deliverables (For All Tasks) 1. AI-generated prompts for code and test case generation. 2. At least 3 assert test cases for each task. 3. AI-generated initial code and execution screenshots.  
4. Analysis of whether code passes all tests. 5. Improved final version with inline comments and explanation. 6. Compiled report (Word/PDF) with prompts, test cases, assertions, code, and output.

**CODE:**

A screen shot of a computer code

Description automatically generated

**OUTPUT:**

A screenshot of a computer

Description automatically generated

**OBSERVATION**: In this code we can observe that the code is checking for the date that is inputted is correct or not.