Assignment-8.5

2303A510G6

G.Charanya

Batch-23

Task Description #1 (Username Validator – Apply AI in Authentication Context)

• Task: Use AI to generate at least 3 assert test cases for a function is_valid_username(username) and then implement the function using Test-Driven Development principles.

• Requirements:

o Username length must be between 5 and 15 characters.

o Must contain only alphabets and digits.

o Must not start with a digit.

o No spaces allowed.

Example Assert Test Cases:

assert is_valid_username("User123") == True

assert is_valid_username("12User") == False

assert is_valid_username("Us er") == False

Expected Output #1

: • Username validation logic successfully passing all AI-generated test cases.

```
130
131 def is_valid_username(username):
132     if len(username) < 5 or len(username) > 15:
133         return False
134     if not username[0].isalpha():
135         return False
136     for char in username:
137         if not (char.isalnum() or char == "_"):
138             return False
139     return True
140 # Test cases for the is_valid_username function
141 assert is_valid_username("User123") == True
142 assert is_valid_username("12User") == False
143 assert is_valid_username("Us er") == False
144 print("All test cases for is_valid_username passed!")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/OneDrive/Desktop/AI/Lab_8_13.py
All test cases for is_valid_username passed!
PS C:\Users\2303a\OneDrive\Desktop\AI> (C:\Users\2303a\miniconda3\Scripts\activate) ; (conda activate base)
```

Task Description #2 (Even–Odd & Type Classification – Apply AI for Robust Input Handling)

• Task: Use AI to generate at least 3 assert test cases for a function classify_value(x) and implement it using conditional logic and loops.

• Requirements:

o If input is an integer, classify as "Even" or "Odd"

. o If input is 0, return "Zero".

o If input is non-numeric, return "Invalid Input".

Example Assert Test Cases:

assert classify_value(8) == "Even"

assert classify_value(7) == "Odd"

assert classify_value("abc") == "Invalid Input"

Expected Output #2:

• Function correctly classifying values and passing all test cases

```
145
146 def classify_value(x):
147     if not isinstance(x, int):
148         return "Invalid Input"
149     if x < 0:
150         return "Negative"
151     elif x == 0:
152         return "Zero"
153     elif x % 2 == 0:
154         return "Even"
155     else:
156         return "Odd"
157 # Test cases for the classify_value function
158 assert classify_value(8) == "Even"
159 assert classify_value(7) == "Odd"
160 assert classify_value("abc") == "Invalid Input"
161 print("All test cases for classify_value passed!")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/OneDrive/Desktop/AI/Lab_8_13.py
PS C:\Users\2303a\OneDrive\Desktop\AI> (C:\Users\2303a\miniconda3\Scripts\activate) ; (conda activate base)
PS C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/OneDrive/Desktop/AI/Lab_8_13.py
All test cases for classify_value passed!
PS C:\Users\2303a\OneDrive\Desktop\AI>
```

Task Description #3 (Palindrome Checker – Apply AI for String Normalization)

• Task: Use AI to generate at least 3 assert test cases for a function is_palindrome(text) and implement the function.

• Requirements:

o Ignore case, spaces, and punctuation.

o Handle edge cases such as empty strings and single characters.

Example Assert Test Cases:

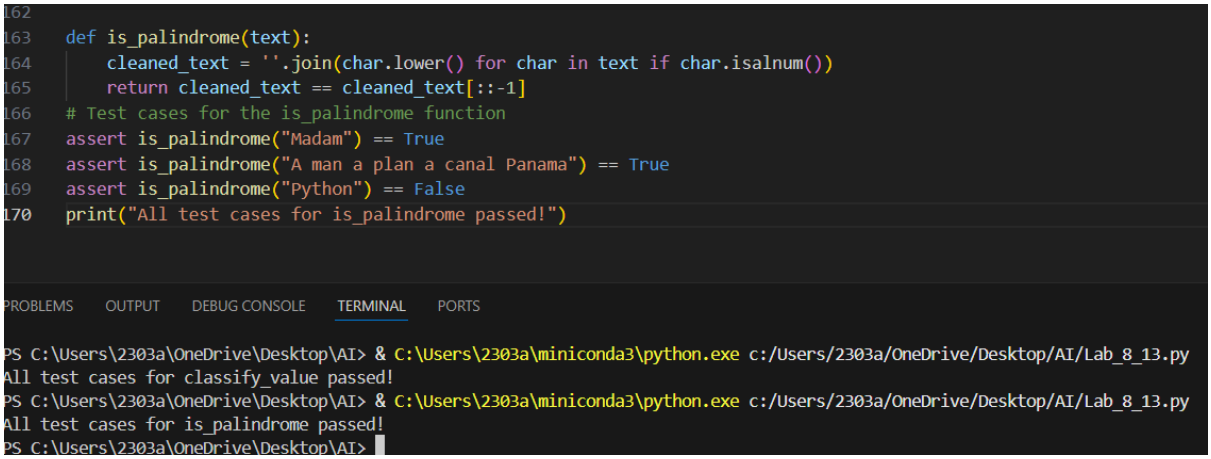
assert is_palindrome("Madam") == True

assert is_palindrome("A man a plan a canal Panama") == True

assert is_palindrome("Python") == False

Expected Output #3:

• Function correctly identifying palindromes and passing all AI-generated tests



Task Description #4 (BankAccount Class – Apply AI for Object-Oriented Test-Driven Development)

• Task: Ask AI to generate at least 3 assert-based test cases for a BankAccount class and then implement the class.

• Methods:

o deposit(amount)

o withdraw(amount)

o get_balance

Example Assert Test Cases:

acc = BankAccount(1000)

acc.deposit(500)

assert acc.get_balance() == 1500

acc.withdraw(300)

assert acc.get_balance() == 1200

Expected Output #4:

• Fully functional class that passes all AI-generated assertions.

```python
class BankAccount:
    def __init__(self, account_number, balance=0):
        self.account_number = account_number
        self.balance = balance
    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            return True
        return False
    def withdraw(self, amount):
        if 0 < amount <= self.balance:
            self.balance -= amount
            return True
        return False
    def get_balance(self):
        return self.balance
# Test cases for the BankAccount class
acc = BankAccount(12345, 1000)  # account_number, balance
acc.deposit(500)
assert acc.get_balance() == 1500
acc.withdraw(300)
assert acc.get_balance() == 1200
print("All test cases for BankAccount passed!")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/OneDrive/Desktop/AI/Lab_8_13.py
All test cases for is_palindrome passed!
PS C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/OneDrive/Desktop/AI/Lab_8_13.py
All test cases for BankAccount passed!
PS C:\Users\2303a\OneDrive\Desktop\AI>
```

Task Description #5 (Email ID Validation – Apply AI for Data Validation)

• Task: Use AI to generate at least 3 assert test cases for a function validate_email(email) and implement the function.

• Requirements:

o Must contain @ and

. o Must not start or end with special characters.

o Should handle invalid formats gracefully.

Example Assert Test Cases:

assert validate_email("user@example.com") == True

assert validate_email("userexample.com") == False

assert validate_email("@gmail.com") == False

Expected Output #5:

• Email validation function passing all AI-generated test cases and handling edge cases correctly.

```python
def validate_email(email):
    if "@" not in email or "." not in email:
        return False
    at_index = email.index("@")
    dot_index = email.rindex(".")
    if at_index < 1 or dot_index < at_index + 2 or dot_index >= len(email) - 1:
        return False
    return True
# Test cases for the validate_email function
assert validate_email("user@example.com") == True
assert validate_email("userexample.com") == False
assert validate_email("@gmail.com") == False
print("All test cases for validate_email passed!")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/OneDrive/Desktop/AI/Lab_8_13.py
All test cases for BankAccount passed!
PS C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/OneDrive/Desktop/AI/Lab_8_13.py
All test cases for validate_email passed!
PS C:\Users\2303a\OneDrive\Desktop\AI>
```