

AI Assisted Coding

Assignment-8.5

2303A51962

Srihan

Batch-24

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.

The screenshot shows a Python script named `Assignment-8.5.py` in a code editor. The code defines a function `is_valid_username` that checks if a username is valid based on length (between 5 and 15 characters) and character type (must start with a alphanumeric character and can only contain alphanumeric characters or underscores). It also includes a series of assert statements to verify the function's correctness. Below the code editor is a terminal window showing the execution of the script and its successful completion.

```
Assignment-8.5.py X
Friday.py > Assignment-8.5.py > ...

207
208     def is_valid_username(username):
209         if len(username) < 5 or len(username) > 15:
210             return False
211         if not username[0].isalnum():
212             return False
213         for char in username:
214             if not char.isalnum() and char != '_':
215                 return False
216         return True
217
# Test cases for the is_valid_username function
218 assert is_valid_username("user123") == True, "Test case 1 failed"
219 assert is_valid_username("luser") == True, "Test case 2 failed"
220 assert is_valid_username("user_name") == True, "Test case 3 failed"
221 assert is_valid_username("us") == False, "Test case 4 failed"
222 print(["All test cases passed!"])

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

● PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding> & "C:/Program Files/Python312/python.exe" nt-8.5.py
All test cases passed!
○ PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding>
```

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:

- If input is an integer, classify as "Even" or "Odd".
- If input is 0, return "Zero".
- 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

The screenshot shows a code editor window with a Python script named `Friday.py`. The code defines a function `classify_value` that returns "Negative" for negative numbers, "Zero" for 0, "Even" for even numbers, and "Odd" for odd numbers. It also contains a block of test cases using `assert` statements. The terminal below the editor shows the execution of the script and the resulting error message, indicating a `TypeError` due to a comparison between a string and an integer.

```
223  
224     def classify_value(x):  
225         if x < 0:  
226             return "Negative"  
227         elif x == 0:  
228             return "Zero"  
229         elif x%2==0:  
230             return "Even"  
231         else:  
232             return "Odd"  
233     # Test cases for the classify_value function  
234     assert classify_value(8) == "Even"  
235     assert classify_value(-3) == "Negative"  
236     assert classify_value(0) == "Zero"  
237     assert classify_value("abc") == "Invalid Input"  
238     print("All test cases passed!")
```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

```
# PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted_ Coding> & "C:/Program Files/Python312/python.exe" "c:/users/Ganne/onedrive/Desktop/Assignment-8.5.py"  
assert classify_value("abc") == "Invalid Input"  
~~~~~  
File "c:/users/Ganne/onedrive/Desktop/AI Assisted_ Coding/Friday.py", line 225, in classify_value  
    if x < 0:  
    ^~~~~~  
TypeError: '<' not supported between instances of 'str' and 'int'  
PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted_ Coding>
```

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:

- Ignore case, spaces, and punctuation.
- 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

The screenshot shows a code editor with Python code for testing palindromes. The code defines a function `is_palindrome` that takes a string and returns True if it is a palindrome. It includes test cases for various strings, including punctuation and spaces. The terminal output shows the code running and printing "All test cases passed!"

```
240
241     def is_palindrome(text):
242         cleaned_text = ''.join(char.lower() for char in text if char.isalnum())
243         return cleaned_text == cleaned_text[::-1]
244     # Test cases for the is_palindrome function
245     assert is_palindrome("A man, a plan, a canal panama") == True, "Test case 1 failed"
246     assert is_palindrome("Hello") == False, "Test case 2 failed"
247     assert is_palindrome("Hi") == False, "Test case 3 failed"
248     print("All test cases passed!")
```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding> & "C:/Program Files/Python312/python.exe" "c:/Users/Ganne/OneDrive/Desktop/AI_Assisted_Coding/test_palindrome.py"

All test cases passed!

PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding>

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.

```
Assignment-8.5.py X
Friday.py > Assignment-8.5.py > BankAccount > deposit

251 class BankAccount:
252     def __init__(self, account_number, balance=0):
253         self.account_number = account_number
254         self.balance = balance
255
256     def deposit(self, amount):
257         if amount > 0:
258             self.balance += amount
259             return True
260         return False
261
262     def withdraw(self, amount):
263         if 0 < amount <= self.balance:
264             self.balance -= amount
265             return True
266         return False
267     def get_balance(self):
268         return self.balance
269
# Test cases for the BankAccount class
270 acc = BankAccount(1000)
271 acc.deposit(500)
272 assert acc.get_balance() == 1500, "Deposit test failed"
273 acc.withdraw(1000)
274 assert acc.get_balance() == 500, "Withdraw test failed"
275 print("All test cases passed!")

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS
PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding> & "C:/Program Files/Python312/python.exe" "c:/Users/Ganne/OneDrive/Desktop\Assignment-8.5.py"
PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding> & "C:/Program Files/Python312/python.exe" "c:/Users/Ganne/OneDrive/Desktop\Assignment-8.5.py"
IndentationError: unindent does not match any outer indentation level
PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding> & "C:/Program Files/Python312/python.exe" "c:/Users/Ganne/OneDrive/Desktop\Assignment-8.5.py"
Traceback (most recent call last):
  File "c:/Users/Ganne/OneDrive/Desktop/AI Assisted Coding\Friday.py\Assignment-8.5.py", line 272, in <module>
    assert acc.get_balance() == 1500, "Deposit test failed"
    ~~~~~
AssertionError: Deposit test failed
PS C:\Users\Ganne\OneDrive\Desktop\AI Assisted Coding>
```

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:

- Must contain @ and .
- Must not start or end with special characters.
- 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.

```
276
277     def validate_email(email):
278         if '@' not in email or '.' not in email:
279             return False
280         at_index = email.index('@')
281         dot_index = email.rindex('.')
282         if at_index < 1 or dot_index < at_index + 2 or dot_index >= len(email) - 1:
283             return False
284         return True
285
# Test cases for the validate_email function
286 assert validate_email("user@example.com") == True, "Test case 1 failed"
287 assert validate_email("userexample.com") == False, "Test case 2 failed"
288 assert validate_email("user@.com") == False, "Test case 3 failed"
289 print("All test cases passed!")
```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

● PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding> & "C:/Program Files/Python312/python.exe" "c:/Users,Ganne/OneDrive/Desktop/Ai_Assisted_Coding/test_email.py"
All test cases passed!
○ PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding>