ASSIGNMENT-8.5

NAME:U.VIGNESH

H.NO-2303A51964

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:

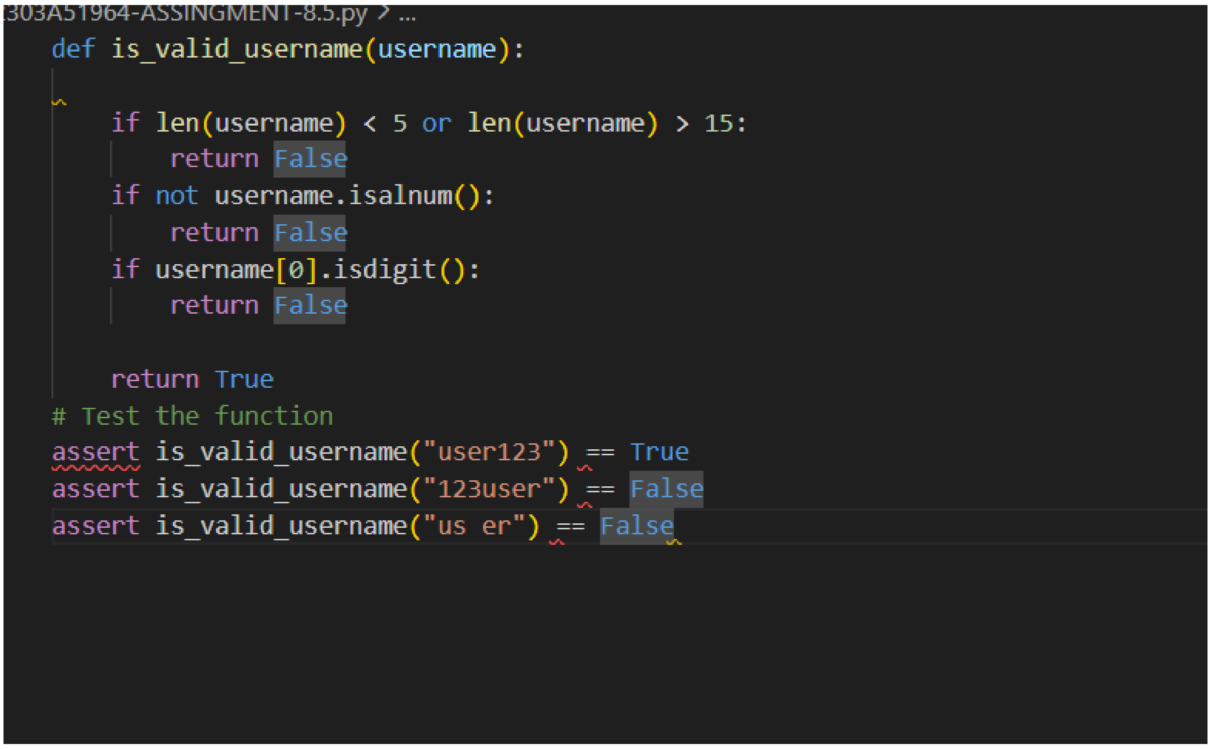
assert is_valid_username("User123") == True

assert is_valid_username("12User") == False

assert is_valid_username("Us er") == False

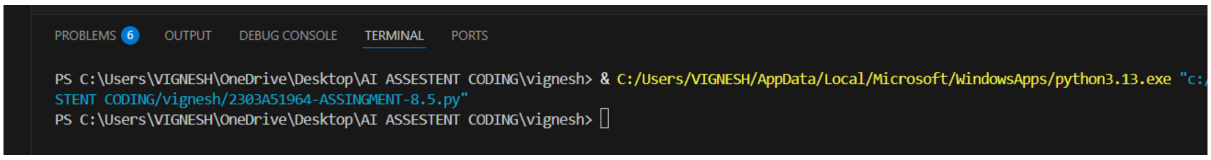
Expected Output #1:

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



```
303A51964-ASSINGMENT-8.5.py > ...
def is_valid_username(username):

    if len(username) < 5 or len(username) > 15:
        return False
    if not username.isalnum():
        return False
    if username[0].isdigit():
        return False

    return True
# Test the function
assert is_valid_username("user123") == True
assert is_valid_username("123user") == False
assert is_valid_username("us er") == False
```

OUTPUT:



```
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING\vignesh> & C:/Users/VIGNESH/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/
STENT CODING/vignesh/2303A51964-ASSINGMENT-8.5.py"
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING\vignesh> 
```

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"

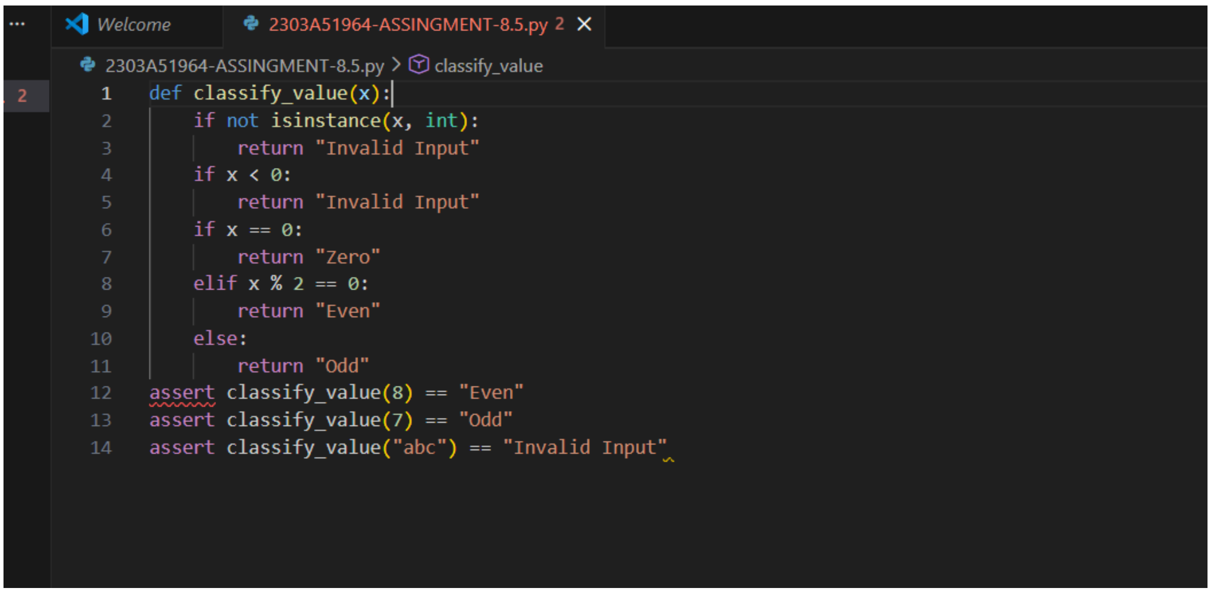
assert classify_value("abc") == "Invalid Input"

Expected Output #2:

• Function correctly classifying values and passing all test

cases.

CODE:


Welcome
2303A51964-ASSINGMENT-8.5.py 2
2303A51964-ASSINGMENT-8.5.py > classify_value

```python
def classify_value(x):
    if not isinstance(x, int):
        return "Invalid Input"
    if x < 0:
        return "Invalid Input"
    if x == 0:
        return "Zero"
    elif x % 2 == 0:
        return "Even"
    else:
        return "Odd"
assert classify_value(8) == "Even"
assert classify_value(7) == "Odd"
assert classify_value("abc") == "Invalid Input"
```

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") ==

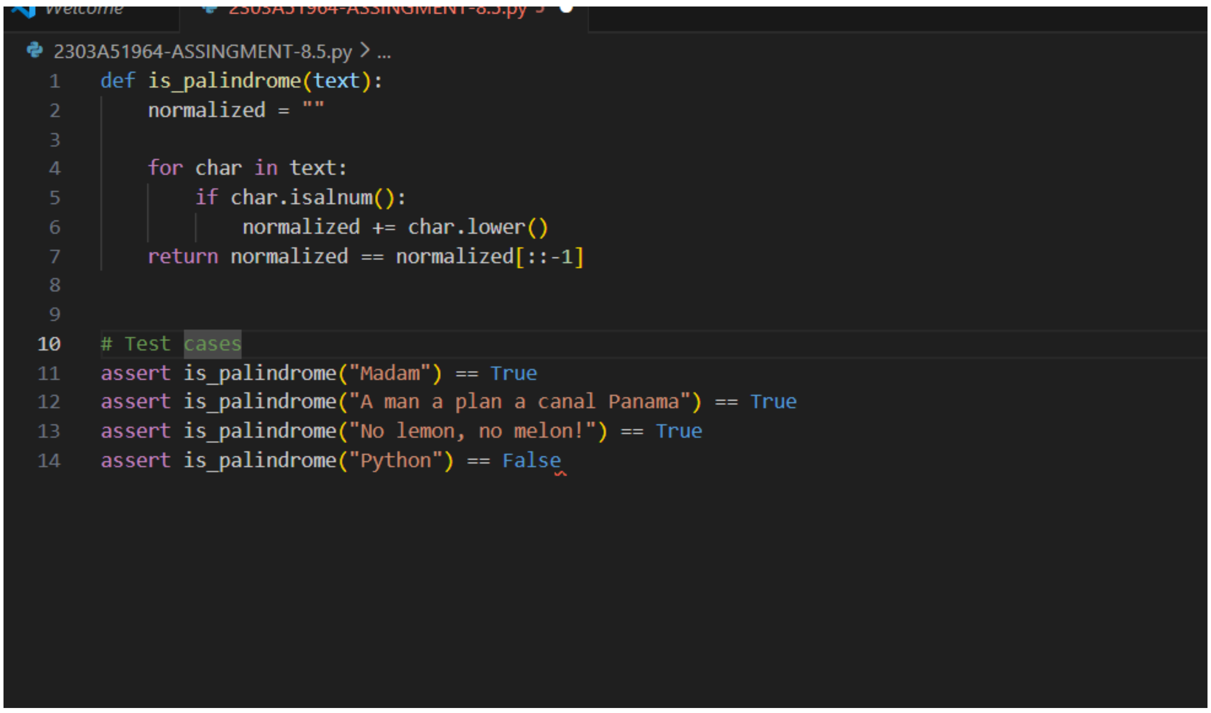
True

assert is_palindrome("Python") == False

Expected Output #3:

• Function correctly identifying palindromes and passing all

AI-generated tests.



```python
def is_palindrome(text):
    normalized = ""

    for char in text:
        if char.isalnum():
            normalized += char.lower()
    return normalized == normalized[::-1]


# Test cases
assert is_palindrome("Madam") == True
assert is_palindrome("A man a plan a canal Panama") == True
assert is_palindrome("No lemon, no melon!") == True
assert is_palindrome("Python") == False
```

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

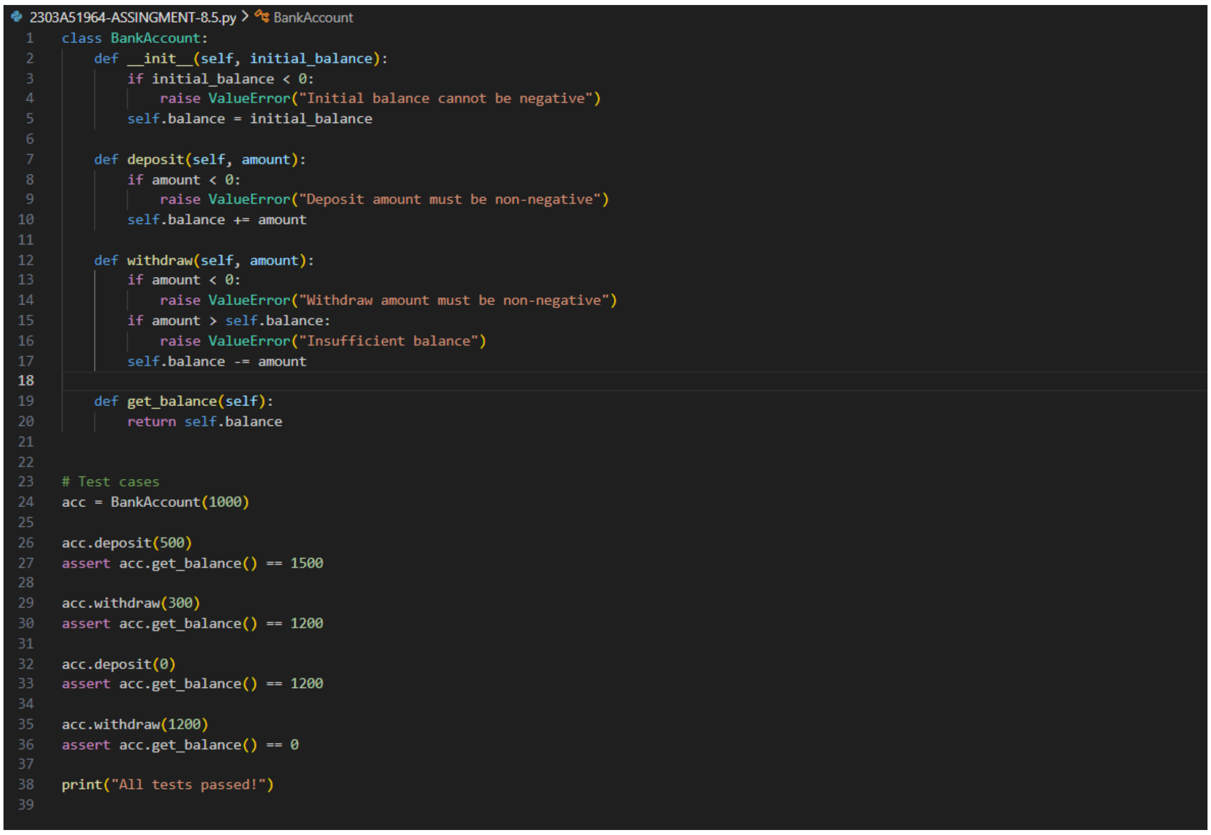
acc.withdraw(300)

assert acc.get_balance() == 1200

Expected Output #4:

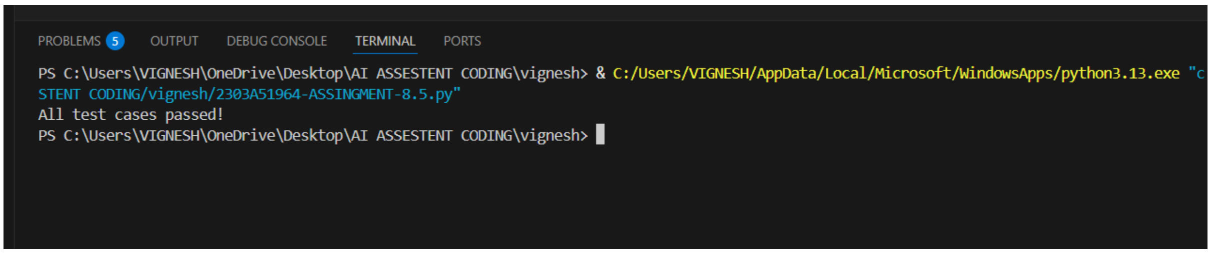
• Fully functional class that passes all AI-generated assertions.

CODE:



2303A51964-ASSINGMENT-8.5.py > BankAccount

```python
class BankAccount:
    def __init__(self, initial_balance):
        if initial_balance < 0:
            raise ValueError("Initial balance cannot be negative")
        self.balance = initial_balance

    def deposit(self, amount):
        if amount < 0:
            raise ValueError("Deposit amount must be non-negative")
        self.balance += amount

    def withdraw(self, amount):
        if amount < 0:
            raise ValueError("Withdraw amount must be non-negative")
        if amount > self.balance:
            raise ValueError("Insufficient balance")
        self.balance -= amount

    def get_balance(self):
        return self.balance


# Test cases
acc = BankAccount(1000)

acc.deposit(500)
assert acc.get_balance() == 1500

acc.withdraw(300)
assert acc.get_balance() == 1200

acc.deposit(0)
assert acc.get_balance() == 1200

acc.withdraw(1200)
assert acc.get_balance() == 0

print("All tests passed!")
```



```
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING\vignesh> & C:/Users/VIGNESH/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c
STENT CODING/vignesh/2303A51964-ASSINGMENT-8.5.py"
All test cases passed!
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING\vignesh> 
```

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

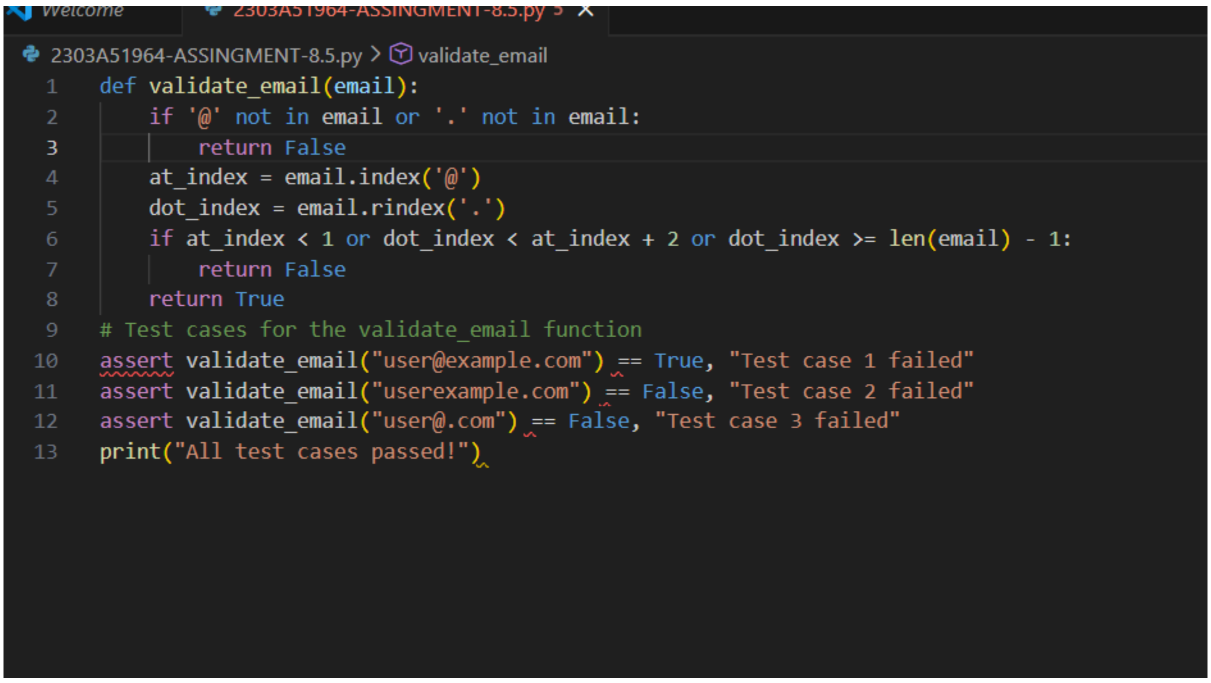
assert validate_email("userexample.com") == False

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

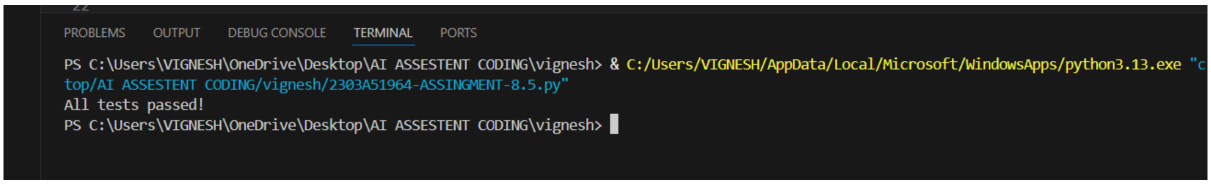
Expected Output #5:

• Email validation function passing all AI-generated test cases

and handling edge cases correctly.


Welcome
2303A51964-ASSINGMENT-8.5.py 5
2303A51964-ASSINGMENT-8.5.py > validate_email

```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, "Test case 1 failed"
assert validate_email("userexample.com") == False, "Test case 2 failed"
assert validate_email("user@.com") == False, "Test case 3 failed"
print("All test cases passed!")
```



```
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING\vignesh> & C:/Users/VIGNESH/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c
top/AI ASSESTENT CODING/vignesh/2303A51964-ASSINGMENT-8.5.py"
All tests passed!
PS C:\Users\VIGNESH\OneDrive\Desktop\AI ASSESTENT CODING\vignesh> 
```