ASSIGNMENT -6.3

2303a51343

b-10

TASK-1

Prompt: Create a Python Student class with name, roll_number, and branch.

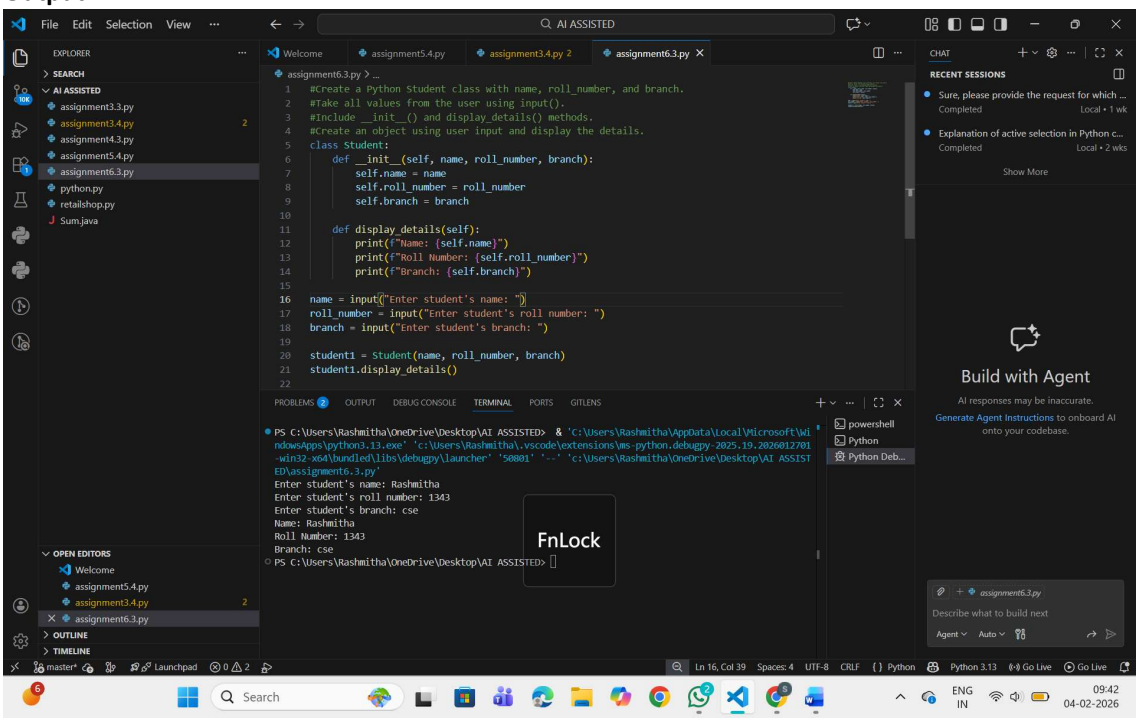
Take all values from the user using input() Include __init__() and display_details() methods.

Create an object using user input and display the details

Code: class Student:

```
    def __init__(self, name, roll_number, branch):
        self.name = name
        self.roll_number = roll_number
        self.branch = branch
    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Roll Number: {self.roll_number}")
        print(f"Branch: {self.branch}")
name = input("Enter student's name: ")
roll_number = input("Enter student's roll number: ")
branch = input("Enter student's branch: ")
student1 = Student(name, roll_number, branch)
student1.display_details()
```

Output:



Analysis:

- AI-generated code correctly defines a Student class with a constructor to initialize name, roll number, and branch.
- The display_details() method clearly prints the student information, making the output easy to understand.
- Overall, the code is simple, readable, and follows proper object-oriented programming

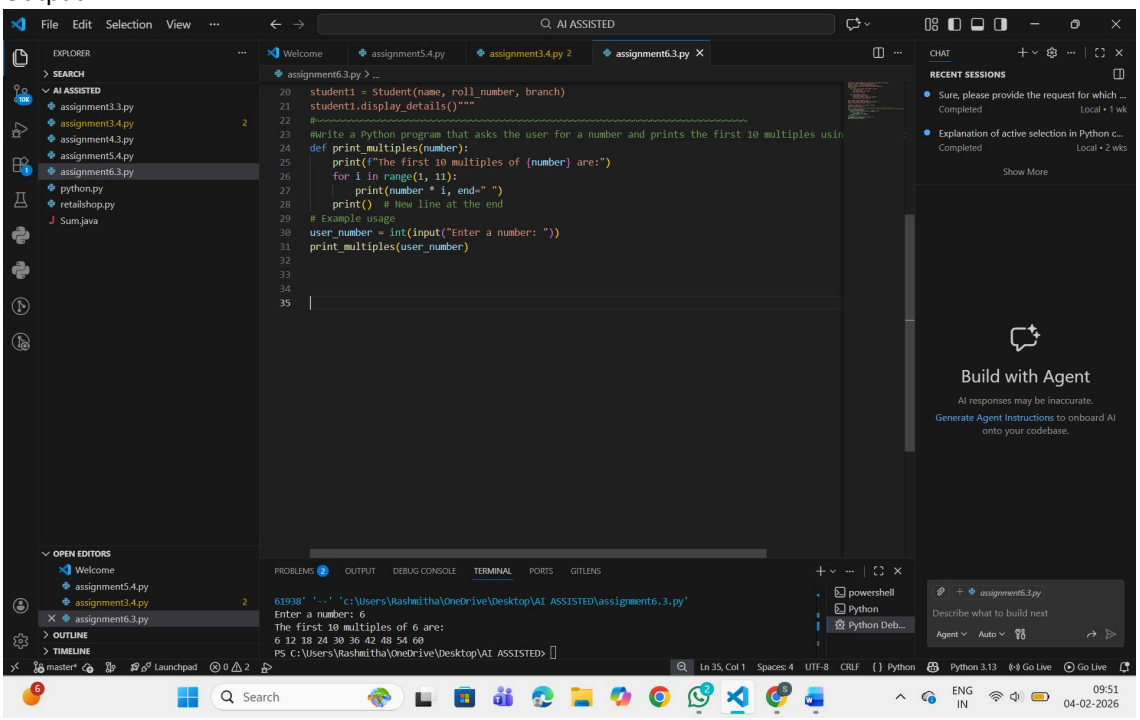
TASK-2

Prompt: Write a Python program that asks the user for a number and prints the first 10 multiples using a for loop

Code:

```
def print_multiples(number):
    print(f"The first 10 multiples of {number} are:")
    for i in range(1, 11):
        print(number * i, end=" ")
    print()  # New line at the end

# Example usage
user_number = int(input("Enter a number: "))
print_multiples(user_number)
```

Output:



Analysis:

The AI-generated code correctly uses a loop to print the first 10 multiples of a given number.

The loop logic is clear and easy to understand, Both the for loop and while loop versions work properly and produce the expected

TASK-3:

Prompt: Write a Python program that takes age as user input and classifies it as child, teenager, adult, or senior using if-elif-else**.**

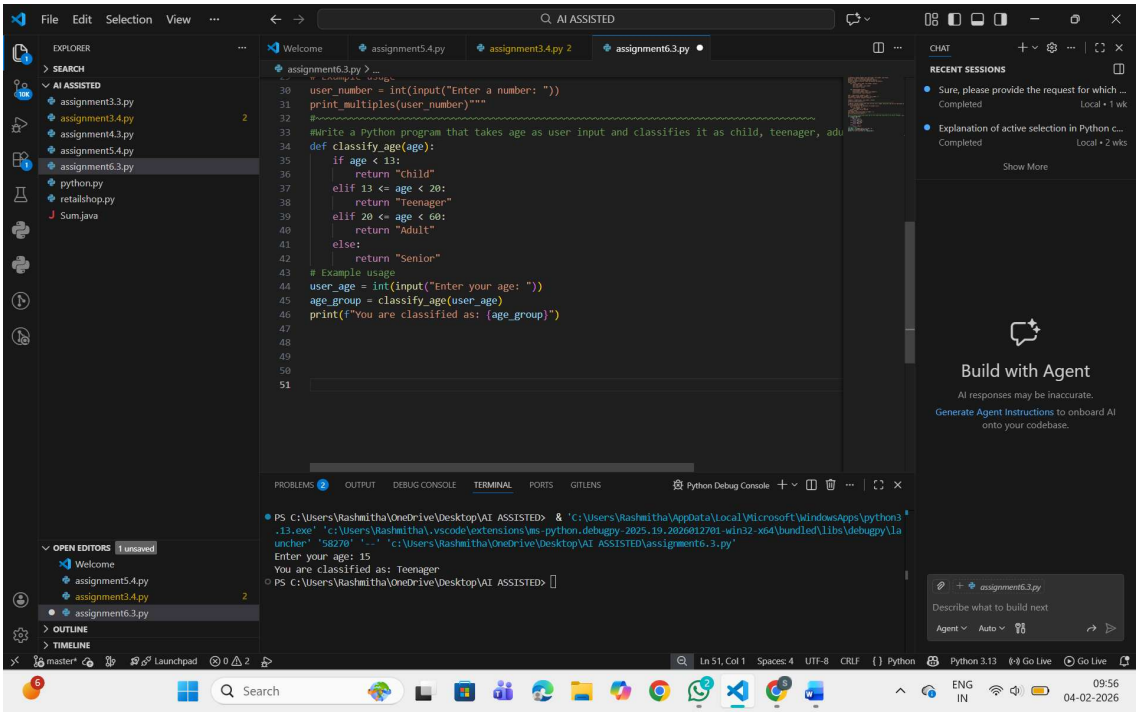
Code: def classify_age(age):

```
    if age < 13:
        return "Child"
    elif 13 <= age < 20:
        return "Teenager"
    elif 20 <= age < 60:
        return "Adult"
    else:
        return "Senior"
# Example usage
user_age = int(input("Enter your age: "))
```

age_group = classify_age(user_age)

print(f"You are classified as: {age_group}")

Output:



Analysis: The AI-generated code uses simple if-elif-else conditions to check the age entered by the user. Each condition clearly matches an age group such as child, teenager, adult, or senior. The code is easy to read and gives correct results for different age values.

TASK-4

Prompt: Write a Python program that takes n as user input and calculates the sum of the first n natural numbers using a for loop.

Code: def sum_of_natural_numbers(n):

total = 0

for i in range(1, n + 1):

total += i

return total

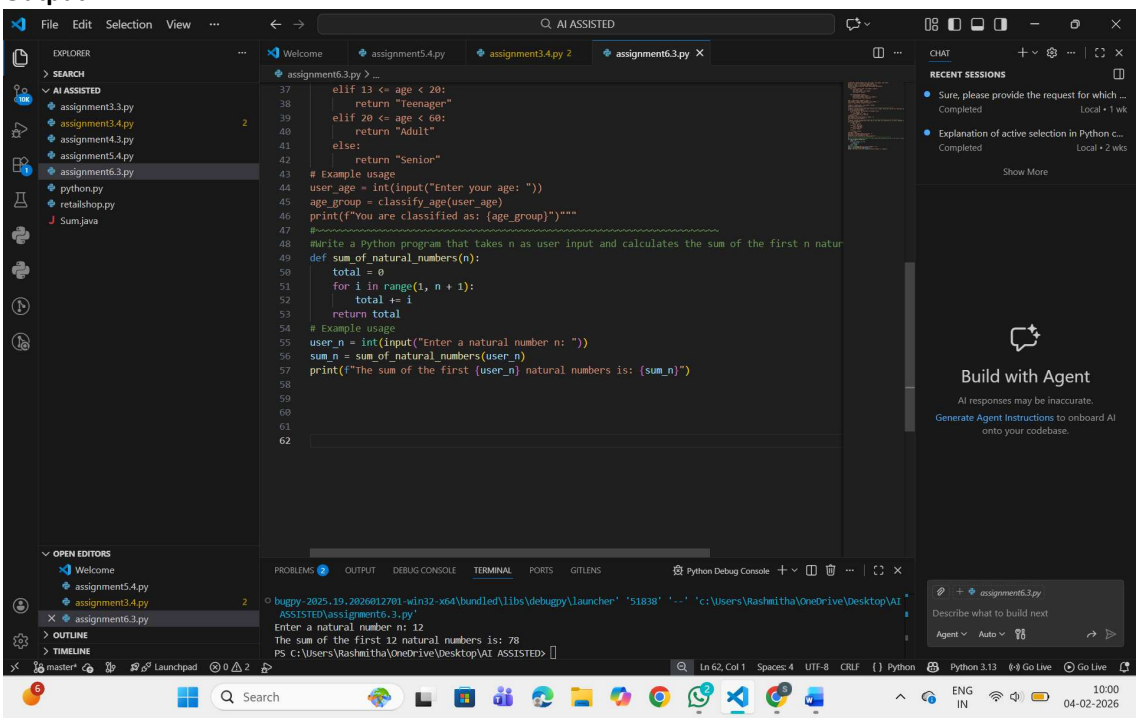
Example usage

user_n = int(input("Enter a natural number n: "))

sum_n = sum_of_natural_numbers(user_n)

print(f"The sum of the first {user_n} natural numbers is: {sum_n}")

Output:



Analysis: The AI-generated code correctly calculates the sum of the first n natural numbers using a loop. The logic is simple and easy to understand, whether it uses a for loop, a while loop, or a formula. The program gives the correct result for any positive number entered by the user.

TASK-5

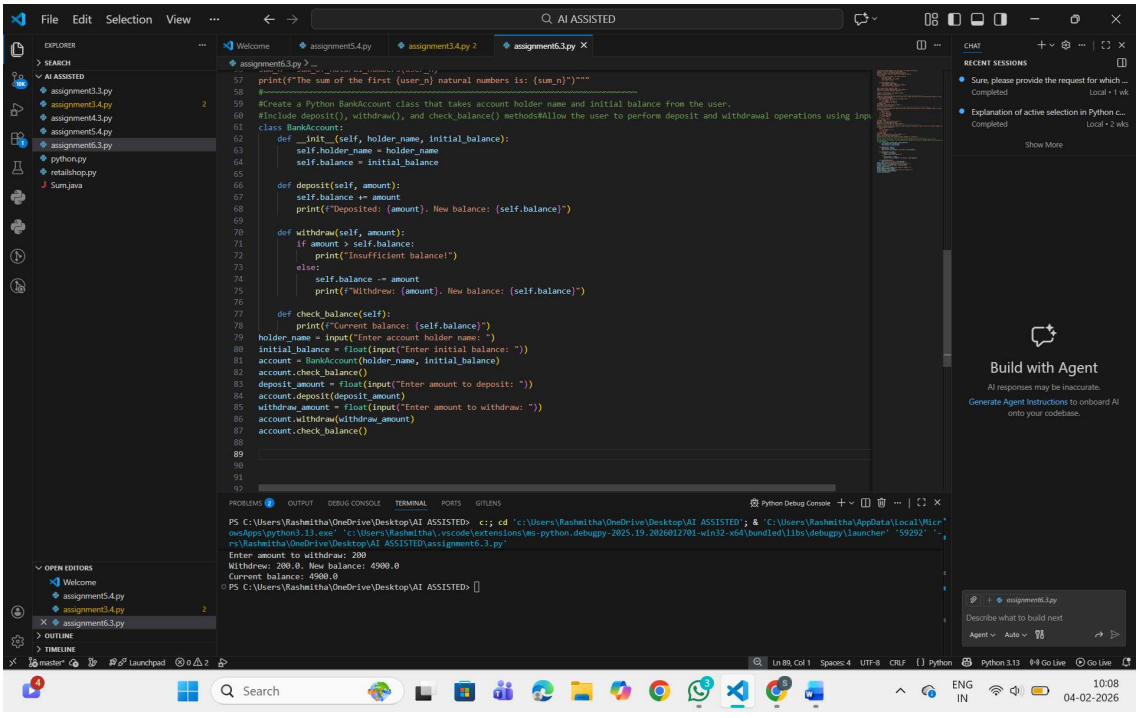
Prompt: Create a Python Bankaccount class that takes account holder name and initial balance from the user.Include deposit(), withdraw(), and check_balance() methods Allow the user to perform deposit and withdrawal operations using input().

Code: class BankAccount:

```
    def __init__(self, holder_name, initial_balance):
        self.holder_name = holder_name
        self.balance = initial_balance


    def deposit(self, amount):
        self.balance += amount
        print(f"Deposited: {amount}. New balance: {self.balance}")


    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient balance!"
```

```
        else:
            self.balance -= amount
            print(f"Withdrew: {amount}. New balance: {self.balance}")


    def check_balance(self):
        print(f"Current balance: {self.balance}")
holder_name = input("Enter account holder name: ")
initial_balance = float(input("Enter initial balance: "))
account = BankAccount(holder_name, initial_balance)
account.check_balance()
deposit_amount = float(input("Enter amount to deposit: "))
account.deposit(deposit_amount)
withdraw_amount = float(input("Enter amount to withdraw: "))
account.withdraw(withdraw_amount)
account.check_balance()
```

Output:



Analysis: Your Bankaccount code is functional using classes and methods for deposit, withdrawal, and balance checks.
It handles basic operations and prevents overdrafts but lacks input validation and only allows one transaction of each type.