ASSIGNMENT 5.4

2303A51343

Batch 10

Task-1

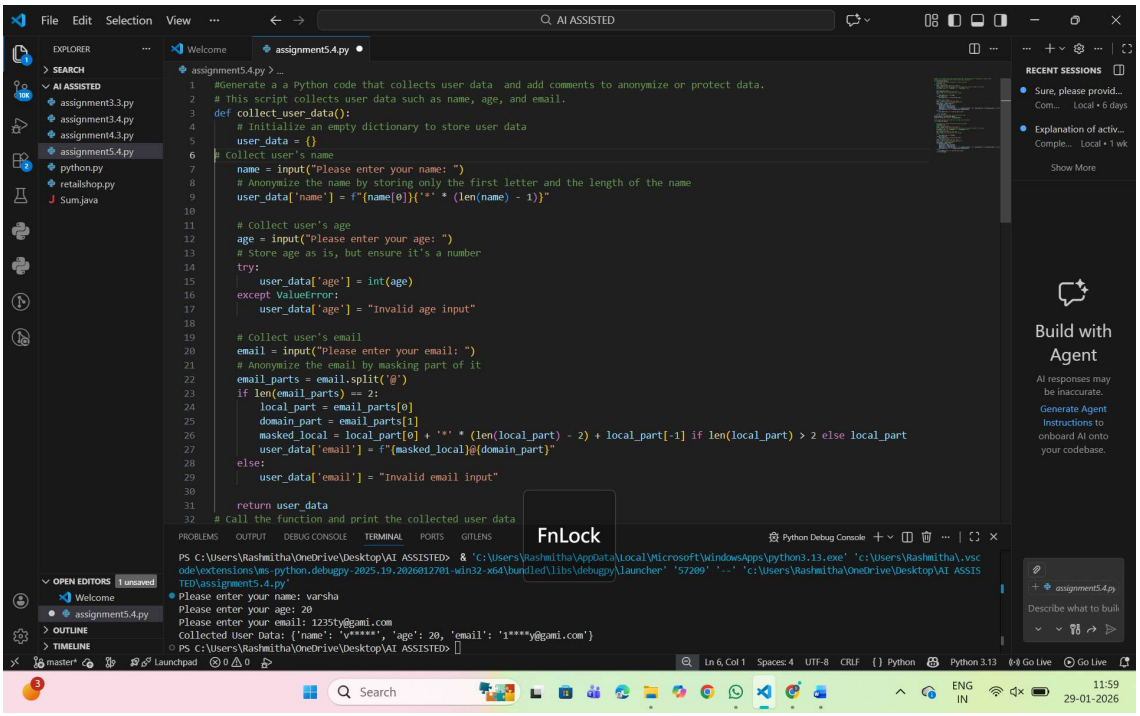
Prompt : Generate a a Python code that collects user data and add comments to anonymize or protect data. This script collects user data such as name, age, and email.

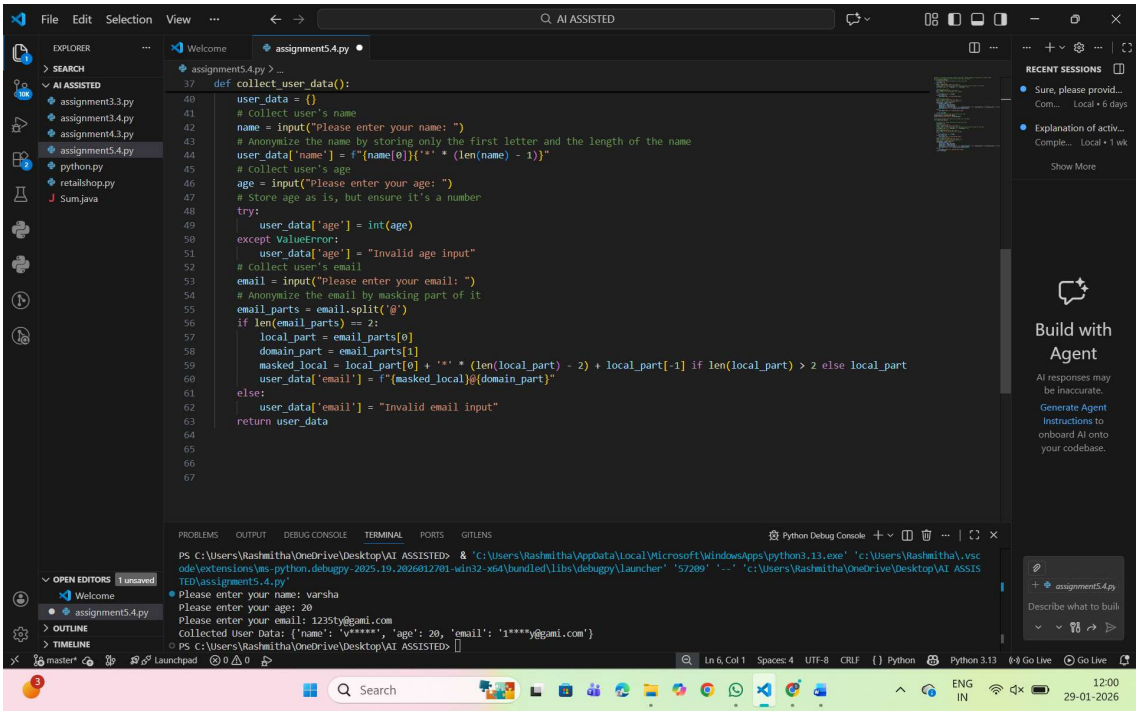
Code:

```
def collect_user_data():
    user_data = {}
    name = input("Please enter your name: ")
    user_data['name'] = f"{name[0]}{'*' * (len(name) - 1)}"
    age = input("Please enter your age: ")
    try:
        user_data['age'] = int(age)
    except ValueError:
        user_data['age'] = "Invalid age input"
    email = input("Please enter your email: ")
    # Anonymize the email by masking part of it
    email_parts = email.split('@')
    if len(email_parts) == 2:
        local_part = email_parts[0]
        domain_part = email_parts[1]
        masked_local = local_part[0] + '*' * (len(local_part) - 2) + local_part[-1] if len(local_part)
> 2 else local_part
        user_data['email'] = f"{masked_local}@{domain_part}"
    else:
        user_data['email'] = "Invalid email input"
```

```
    return user_data
collected_data = collect_user_data()
print("Collected User Data:", collected_data)
def collect_user_data():
     user_data = {}
    name = input("Please enter your name: ")
    # Anonymize the name by storing only the first letter and the length of the name
    user_data['name'] = f"{name[0]}{'*' * (len(name) - 1)}"
    age = input("Please enter your age: ")
    # Store age as is, but ensure it's a number
    try:
        user_data['age'] = int(age)
    except ValueError:
        user_data['age'] = "Invalid age input"
    email = input("Please enter your email: ")
    # Anonymize the email by masking part of it
    email_parts = email.split('@')
    if len(email_parts) == 2:
        local_part = email_parts[0]
        domain_part = email_parts[1]
        masked_local = local_part[0] + '*' * (len(local_part) - 2) + local_part[-1] if len(local_part)
> 2 else local_part
        user_data['email'] = f"{masked_local}@{domain_part}"
    else:
        user_data['email'] = "Invalid email input"
    return user_data
```

Output:





Analysis: The program first asks the user to enter personal details like name, age, and email using input(). These values are stored in variables so they can be processed later. The anonymize data() function replaces real data with dummy values to protect privacy. This shows how personal data can be hidden or masked before sharing or storing it

Task-2

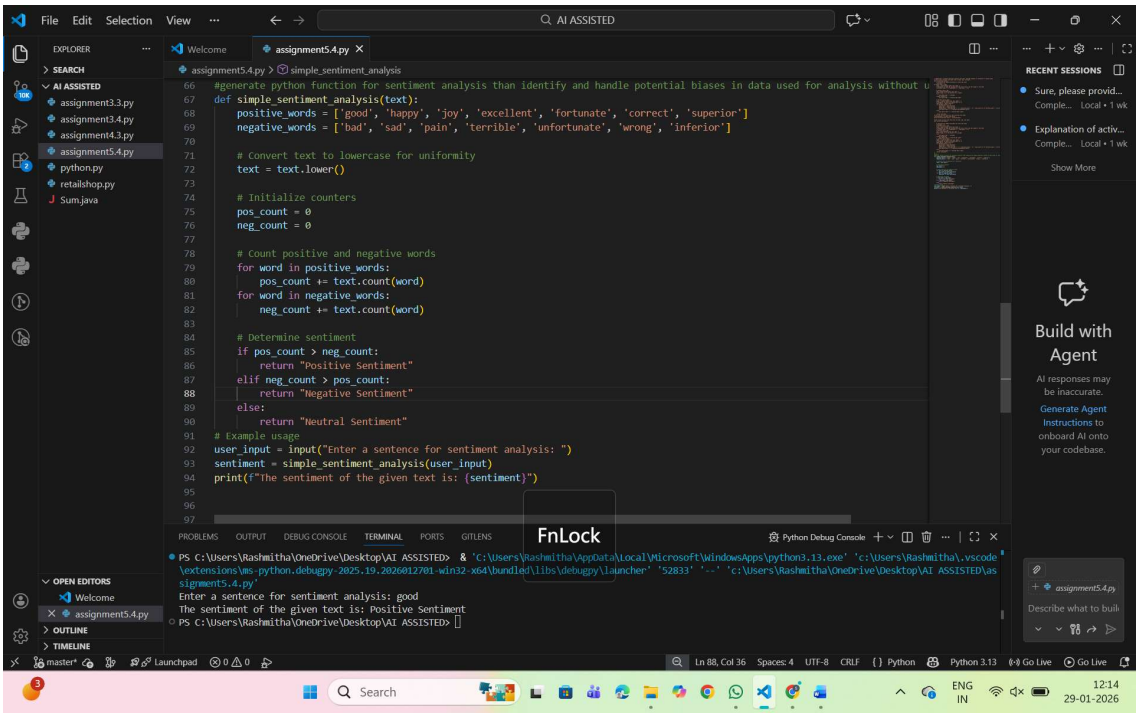
Prompt: generate python function for sentiment analysis than identify and handle potential biases in data used for analysis without using modules

Code:

generate python function for sentiment analysis than identify and handle potential biases in data used for analysis without using modules

```
def simple_sentiment_analysis(text):
    positive_words = ['good', 'happy', 'joy', 'excellent', 'fortunate', 'correct', 'superior']
    negative_words = ['bad', 'sad', 'pain', 'terrible', 'unfortunate', 'wrong', 'inferior']
    text = text.lower(
    pos_count = 0
    neg_count = 0
     for word in positive_words:
       pos_count += text.count(word)
    for word in negative_words:
       neg_count += text.count(word)
       if pos_count > neg_count:
       return "Positive Sentiment"
    elif neg_count > pos_count:
       return "Negative Sentiment"
    else:
       return "Neutral Sentiment"
user_input = input("Enter a sentence for sentiment analysis: ")
sentiment = simple_sentiment_analysis(user_input)
print(f"The sentiment of the given text is: {sentiment}")
```

Output:



Analysis: The function checks the text for positive and negative words using predefined lists. The input text is converted to lowercase to avoid case-sensitive errors. It counts how many positive and negative words are present in the sentence. Based on the count, the program decides whether the sentiment is Positive, Negative, or Neutral.

Task-3

Prompt: Generate python program to recommends products based on user history and follow ethical guidelines to avoid manipulative practices

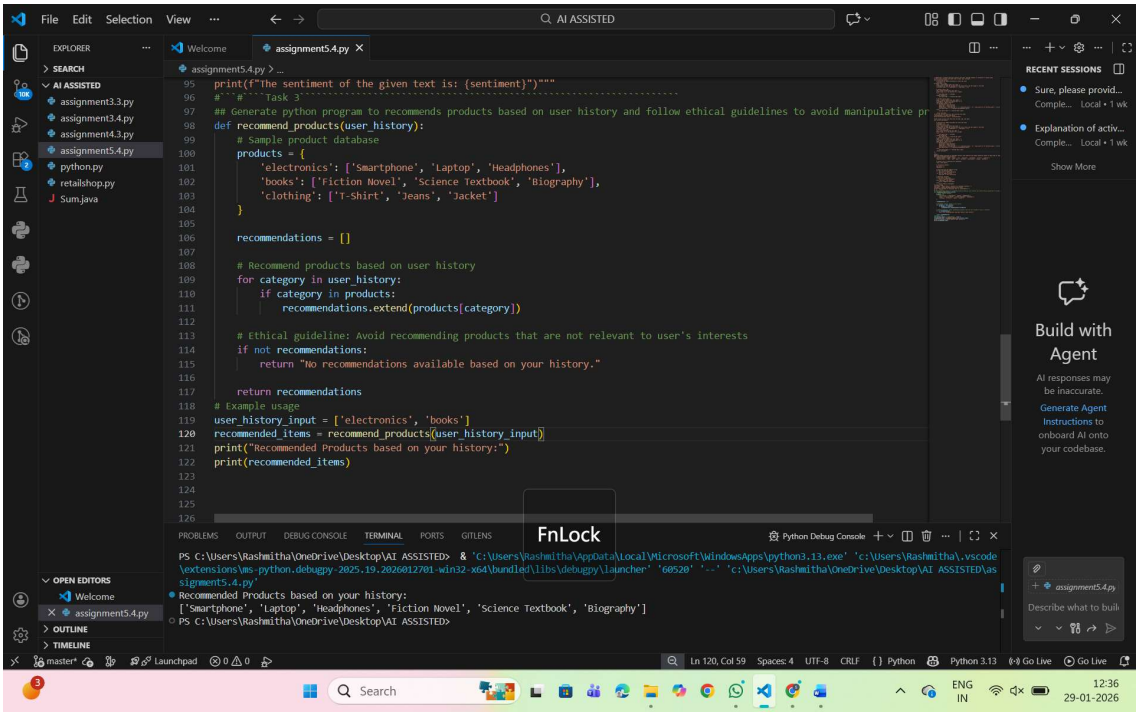
Code: def recommend_products(user_history):

```
    # Sample product database
    products = {
        'electronics': ['Smartphone', 'Laptop', 'Headphones'],
        'books': ['Fiction Novel', 'Science Textbook', 'Biography'],
        'clothing': ['T-Shirt', 'Jeans', 'Jacket']
    }

    recommendations = []
     for category in user_history:
        if category in products:
```

```
        recommendations.extend(products[category])
    if not recommendations:
        return "No recommendations available based on your history."
    return recommendations
user_history_input = ['electronics', 'books']
recommended_items = recommend_products(user_history_input)
print("Recommended Products based on your history:")
print(recommended_items)
```

Output:



Analysis:

The function checks the text for positive and negative words using predefined lists. The input text is converted to lowercase to avoid case-sensitive errors. It counts how many positive and negative words are present in the sentence. Based on the count, the program decides whether the sentiment is Positive, Negative, or Neutral.

Task-4:

Prompt: Generate python program that logging fuctionality in python web application and logs do not record senstive information

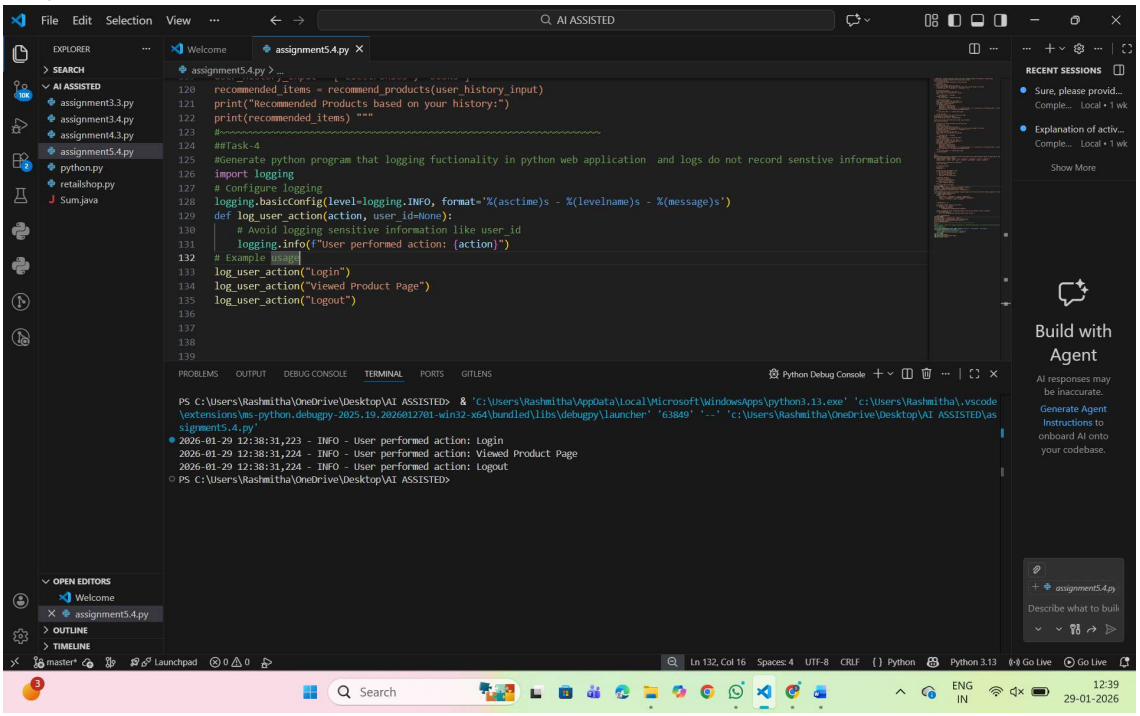
Code: import logging

Configure logging

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

```
def log_user_action(action, user_id=None):
    # Avoid logging sensitive information like user_id
    logging.info(f"User performed action: {action}")
# Example usage
log_user_action("Login")
log_user_action("Viewed Product Page")
log_user_action("Logout")
```

Output:



Analysis: The program uses Python's logging feature to record user actions. It logs only general actions like login or logout, not private data. Sensitive details such as user ID or passwords are intentionally avoided. This improves system monitoring while maintaining user privacy and security.

Task-5

Prompt: Generate python program that machine learning model than add documentation on how to use the model like explainability ,accuracy limkits **.**

Code: def simple_ml_model(data):

    # A simple placeholder function for a machine learning model

    # In a real scenario, this would involve training a model on the provided data

    model_accuracy = 0.85 # Example accuracy

return model_accuracy

#

This function represents a simple machine learning model.

It takes input data and returns an accuracy score.

Explainability: The model is a placeholder and does not provide detailed explanations.

"""

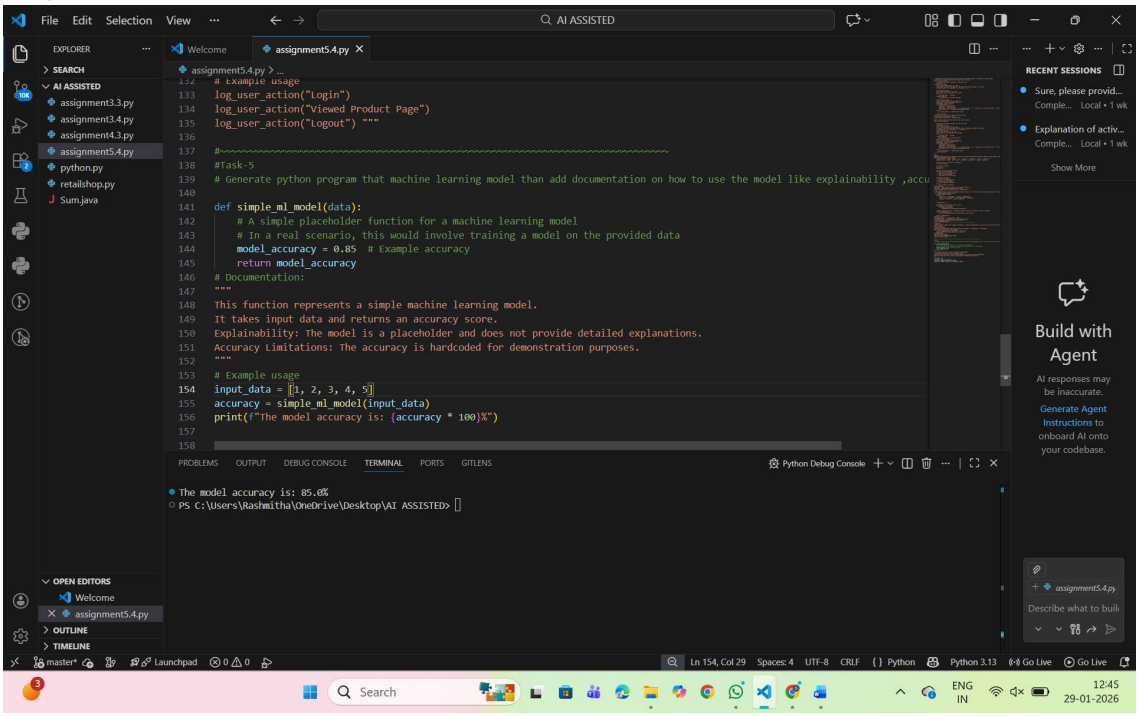
Example usage

input_data = [1, 2, 3, 4, 5]

accuracy = simple_ml_model(input_data)

print(f"The model accuracy is: {accuracy * 100}%")

Output:



Analysis: It represents a basic machine learning model using a placeholder.It returns a fixed accuracy value for demonstration purposes. Comments explain that the model does not show real predictions or explanaϴons. Documentaionclearly mentions limitations in accuracy and explainability.