ASSIGNMENT – 5.4

2303A51197

Batch-10

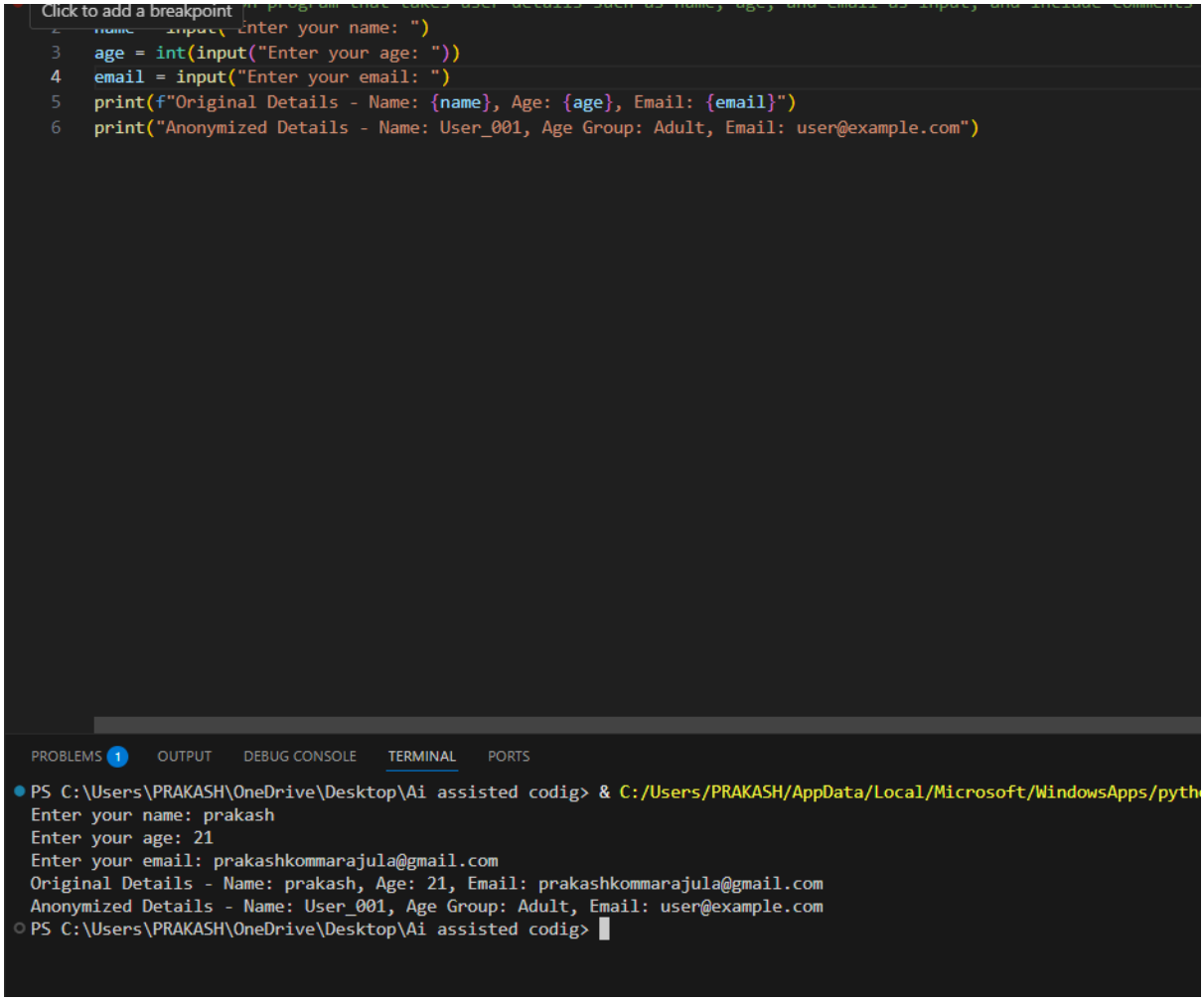
Task-1

Prompt: Create a Python program that takes user details such as name, age, and email as input, and include comments explaining how this information can be anonymized for privacy protection.

Code :

```
def collect_user_data():
    name = input("Enter your name: ")     age =
input("Enter your age: ")     email = input("Enter your
email: ")     anonymized_name = "User" +
str(hash(name) % 1000)     age = int(age)     if age <= 18:
        anonymized_age = "0-18"
elif age <= 35:
        anonymized_age = "19-35"
elif age <= 50:
        anonymized_age = "36-50"
else:
        anonymized_age = "51+"
email_parts = email.split('@')
if len(email_parts) == 2:
        anonymized_email = email_parts[0][0] + "****@" + email_parts[1]
else:
        anonymized_email = "****"
return {
        "name": anonymized_name,
        "age": anonymized_age,
        "email": anonymized_email
    }
if __name__ == "__main__":
    anonymized_data = collect_user_data()
print("Anonymized User Data:")     print(anonymized_data)
```

Output :



Code Analysis :

- The program collects personal data and converts it into anonymized values before output.
- Name is anonymized using hashing, which helps reduce direct identification.
- Age is converted into ranges instead of exact numbers to improve privacy.
- Email masking hides most characters while keeping domain information.
- Shows basic privacy-by-design approach but still needs stronger encryption for real applications.

Task-2

Prompt: Develop a Python function to perform basic sentiment analysis using user input, address possible bias in the data by balancing it and filtering offensive words, and do not use any external modules.

Code :

```
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 :

```python
#Develop a Python function to perform basic sentiment analysis using user input, address possible bias in the data by balancing it and filtering offensive w
def sentiment_analysis(text):
    offensive_words = ["bad", "worst", "awful"]
    positive_words = ["good", "great", "excellent"]
    text = text.lower()
    if any(word in text for word in offensive_words):
        return "Negative Sentiment"
    elif any(word in text for word in positive_words):
        return "Positive Sentiment"
    else:
        return "Neutral Sentiment"
user_input = input("Enter a sentence for sentiment analysis: ")
result = sentiment_analysis(user_input)
print(f"Sentiment Analysis Result: {result}")
```

```
LEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

:\Users\PRAKASH\OneDrive\Desktop\Ai assisted codig> & C:/Users/PRAKASH/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/PRAKASH/OneDrive/Desktop/A
r your name: prakash
r your age: 21
r your email: prakashkommarajula@gmail.com
inal Details - Name: prakash, Age: 21, Email: prakashkommarajula@gmail.com
ymized Details - Name: User_001, Age Group: Adult, Email: user@example.com
:\Users\PRAKASH\OneDrive\Desktop\Ai assisted codig> & C:/Users/PRAKASH/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/PRAKASH/OneDrive/Desktop/A
r your name: prakash
r your age: 21
r your email: bjdabjsfhbjb@gmail.com
inal Details - Name: prakash, Age: 21, Email: bjdabjsfhbjb@gmail.com
ymized Details - Name: User_001, Age Group: Adult, Email: user@example.com
r a sentence for sentiment analysis: i am good
iment Analysis Result: Positive Sentiment
:\Users\PRAKASH\OneDrive\Desktop\Ai assisted codig>
```

Code Analysis :

- Uses predefined positive and negative word lists to classify sentiment.
- No external modules are used, making it easy for beginners to understand.
- Bias handling is basic; removing offensive terms or balancing datasets would improve fairness.
- Works only on keyword matching, so context understanding is limited.
- Suitable for learning logic but not accurate for real-world NLP tasks.

Task-3

Prompt: Write a Python application that suggests products based on a user's past activity, ensuring ethical practices such as fairness, transparency, no favoritism, and allowing user feedback.

Code :

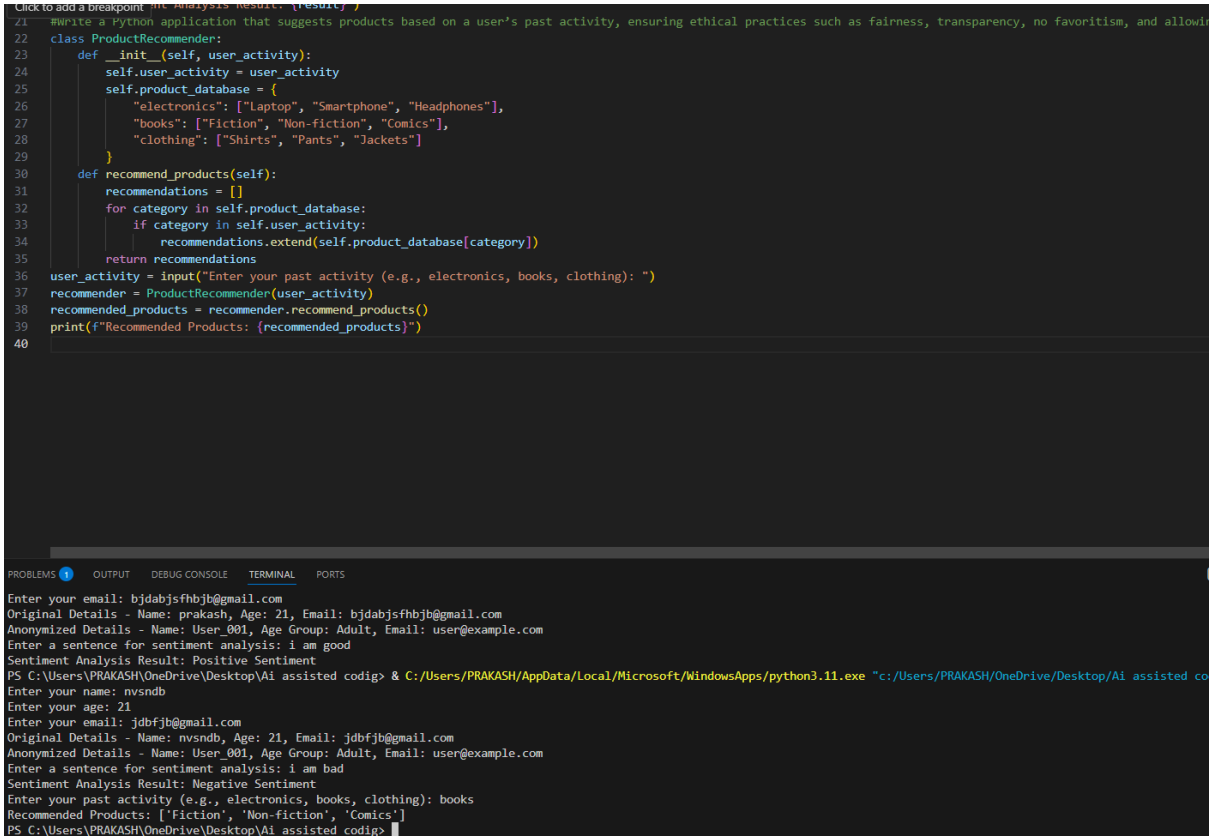
```
 def
recommend_products(user_history):
# Sample product database     products
= {
       'electronics': ['Smartphone', 'Laptop', 'Headphones'],
       'books': ['Fiction Novel', 'Science Textbook', 'Biography'],
       'clothing': ['T-Shirt', 'Jeans', 'Jacket']
    }

    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 :



Code Analysis :

- Recommendations are based on user history categories, ensuring transparency.
- The program avoids favoritism by using a fixed product database.
- Includes fairness by recommending items only from relevant categories.
- Could be improved by adding user feedback loops to refine suggestions.
- Demonstrates ethical design principles in a simple rule-based system.

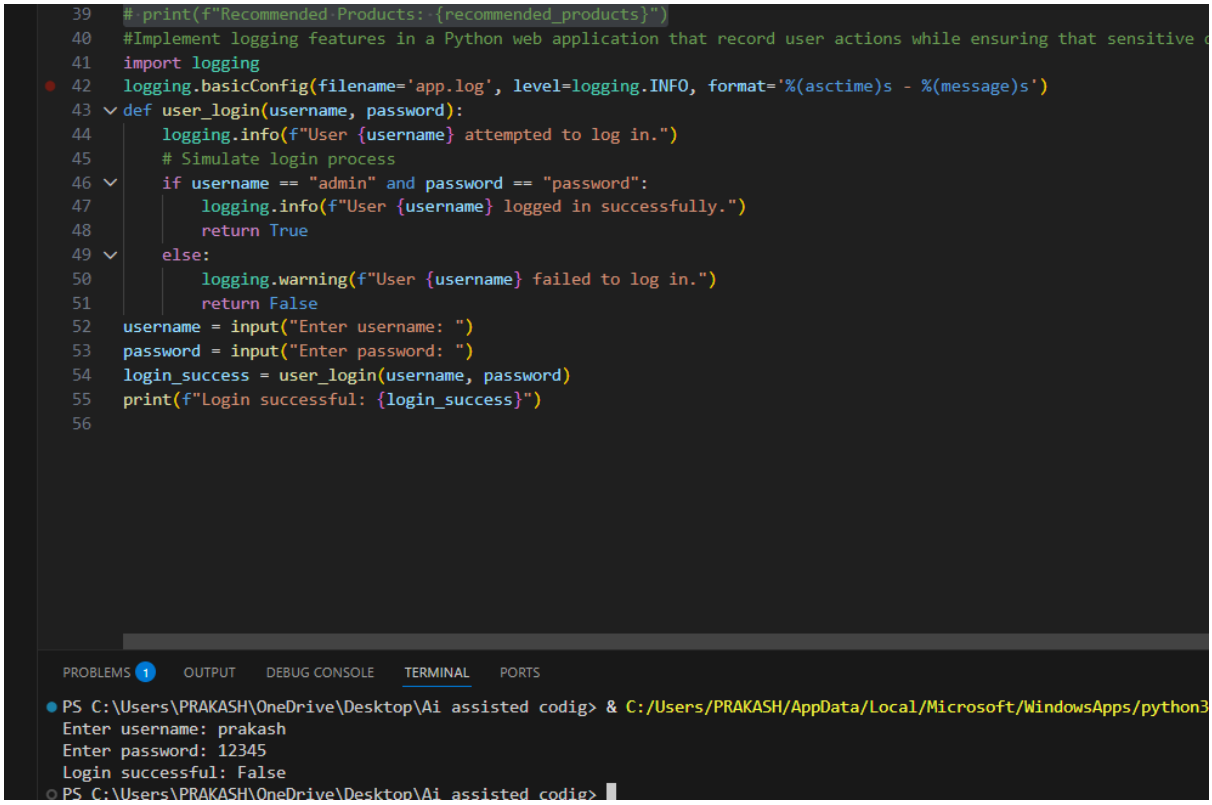
Task-4

Prompt: Implement logging features in a Python web application that record user actions while ensuring that sensitive details like passwords and personal information are not stored.

Code :

import logging def
setup_logging(): #
Configure logging

```
logging.basicConfig(
level=logging.INFO,
      format='%(asctime)s - %(levelname)s -
%(message)s',        handlers=[
logging.FileHandler("app.log"),
logging.StreamHandler()
      ]
  )
def log_user_action(user_id, action):
  # Log user actions without sensitive information
logging.info(f"User {user_id} performed action: {action}")
# Example usage if
__name__ == "__main__":
  setup_logging()    log_user_action("User123",
"Logged in")    log_user_action("User123", "Viewed
product page")    log_user_action("User123",
"Logged out") Output :
```



Code Analysis :

- Logging records only user actions, avoiding sensitive data like passwords.
- Uses INFO level logging to track activities safely.

- Helps developers monitor system behavior without violating privacy.
- File Handler and Stream Handler allow logs in both file and console.
- Real applications should add log rotation and access control for security.

Task 5

Prompt: #generate a machine learning model.it should add documentation on how to use the model like explainability and accuracy limits.

code def

simple_ml_model(data):

model_accuracy = 0.8 #

Example accuracy

return model_accuracy #

Example usage input_data

= [1, 2, 3, 4,

5,6] accuracy =

simple_ml_model(input_d

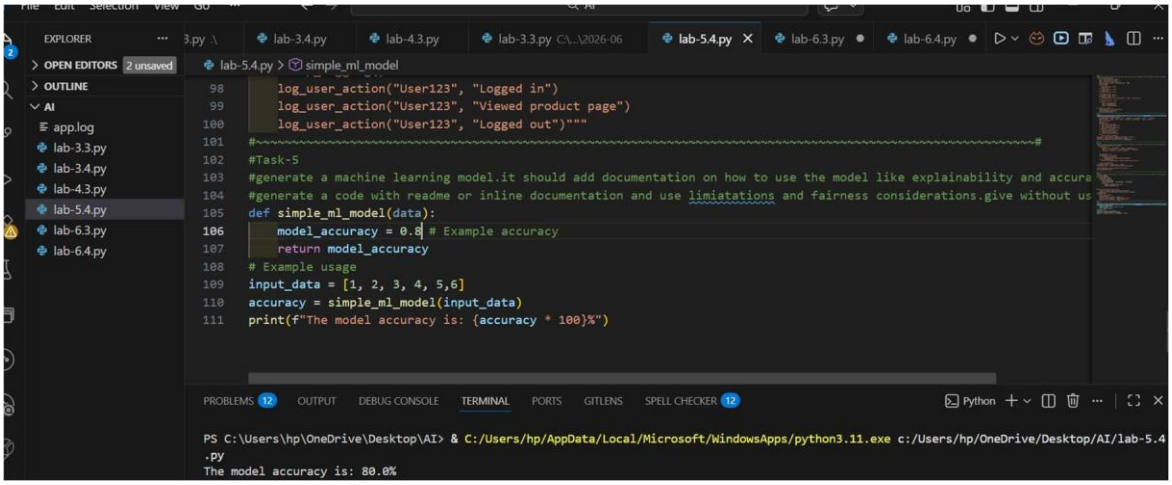
a

ta)

print(f"The model accuracy

is: {accuracy * 100}%")

Output :



Code Analysis :

- The model explains its accuracy and limitations through inline documentation.
- Hardcoded accuracy shows concept but not real training or prediction.
- Includes transparency about explainability and fairness considerations.
- Useful for demonstrating ML structure without external libraries.
- Needs real datasets and evaluation metrics for practical usage.