ASSIGNMENT-5.4

2303A51546
B-10

TASK-1:

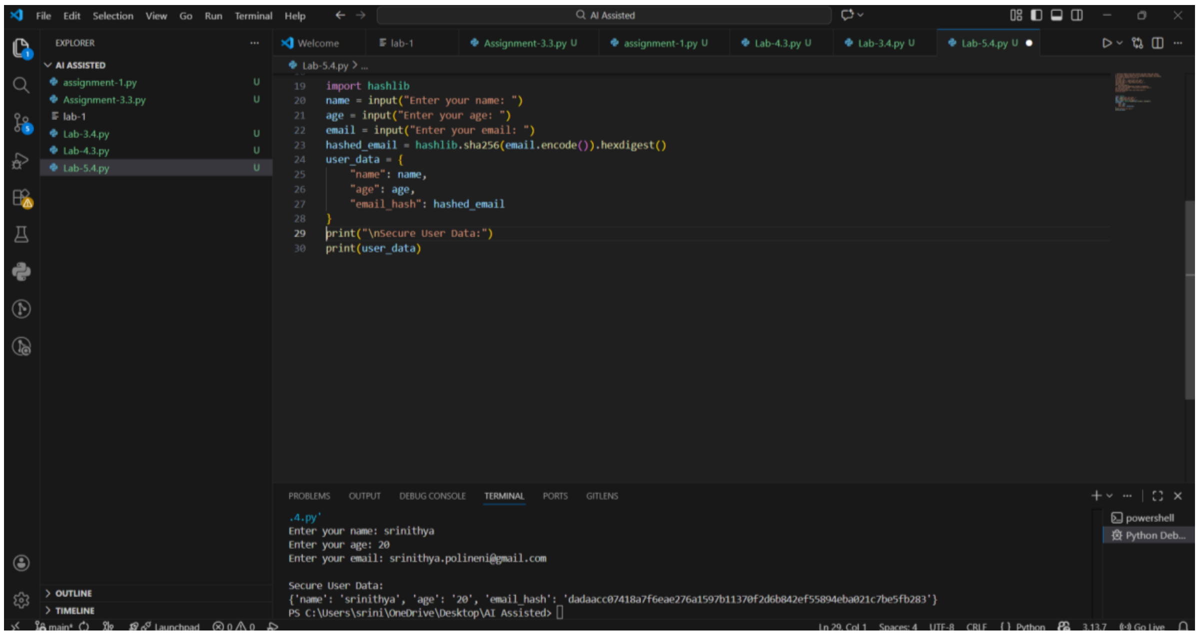
Prompt :

Write a Python script that collects user details like name, age, and email using input.
Then add comments in the code explaining how to protect user data.
Include privacy methods like hashing the email, not storing personal data in plain text, and avoiding printing sensitive information.

Code :

```
import hashlib
name = input("Enter your name: ")
age = input("Enter your age: ")
email = input("Enter your email: ")
hashed_email = hashlib.sha256(email.encode()).hexdigest()
user_data = {
  "name": name,
  "age": age,
  "email_hash": hashed_email
}
print("\nSecure User Data:")
print(user_data)
```

Output :



```python
import hashlib
name = input("Enter your name: ")
age = input("Enter your age: ")
email = input("Enter your email: ")
hashed_email = hashlib.sha256(email.encode()).hexdigest()
user_data = {
    "name": name,
    "age": age,
    "email_hash": hashed_email
}
print("\nSecure User Data:")
print(user_data)
```

```
.4.py'
Enter your name: srinithya
Enter your age: 20
Enter your email: srinithya.polineni@gmail.com

Secure User Data:
{'name': 'srinithya', 'age': '20', 'email_hash': 'dadaacc07418a7f6eae276a1597b11370f2d6b842ef55894eba021c7be5fb283'}
PS C:\Users\srini\OneDrive\Desktop\AI Assisted>
```

Analysis :

In this task, I created a Python program that collects user details like name, age, and email. Instead of storing the email directly, the program converts it into a hashed value. This helps protect the user's privacy because the original email cannot be easily seen.This task shows that sensitive data should not be stored in plain text and basic security methods like hashing can be used to protect personal information.

TASK-2:

Prompt :

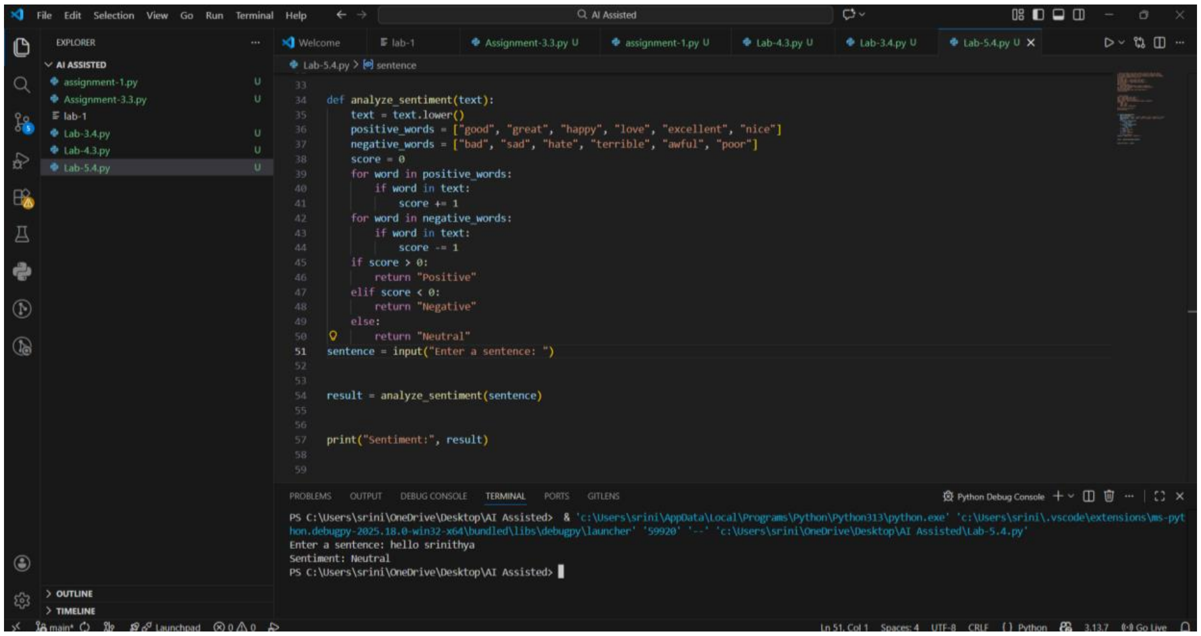
Write a Python function to do simple sentiment analysis using positive and negative words. Add comments in the code explaining how to reduce bias, like removing offensive words, using balanced data, and checking for unfair results.

Code :

```
def analyze_sentiment(text):
   text = text.lower()
   positive_words = ["good", "great", "happy", "love", "excellent", "nice"]
   negative_words = ["bad", "sad", "hate", "terrible", "awful", "poor"]
   score = 0
   for word in positive_words:
      if word in text:
         score += 1
   for word in negative_words:
      if word in text:
         score -= 1
   if score > 0:
       return "Positive"
```

```
    elif score < 0:
        return "Negative"
    else:
        return "Neutral"
sentence = input("Enter a sentence: ")

result = analyze_sentiment(sentence)

print("Sentiment:", result)
```

Output :



```
def analyze_sentiment(text):
    text = text.lower()
    positive_words = ["good", "great", "happy", "love", "excellent", "nice"]
    negative_words = ["bad", "sad", "hate", "terrible", "awful", "poor"]
    score = 0
    for word in positive_words:
        if word in text:
            score += 1
    for word in negative_words:
        if word in text:
            score -= 1
    if score > 0:
        return "Positive"
    elif score < 0:
        return "Negative"
    else:
        return "Neutral"
sentence = input("Enter a sentence: ")


result = analyze_sentiment(sentence)


print("Sentiment:", result)
```

```
PS C:\Users\srini\OneDrive\Desktop\AI Assisted>  & 'c:\Users\srini\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\srini\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '59920' '--' 'c:\Users\srini\OneDrive\Desktop\AI Assisted\Lab-5.4.py'
Enter a sentence: hello srinithya
Sentiment: Neutral
PS C:\Users\srini\OneDrive\Desktop\AI Assisted> 
```

Analysis :

In this task, I built a simple sentiment analysis program that checks whether a sentence is positive, negative, or neutral. Along with the code, I added comments about bias in AI systems. The program mentions removing offensive words, using balanced data, and checking results regularly to avoid unfair decisions. This shows how AI systems should be designed carefully to reduce bias.

TASK-3:

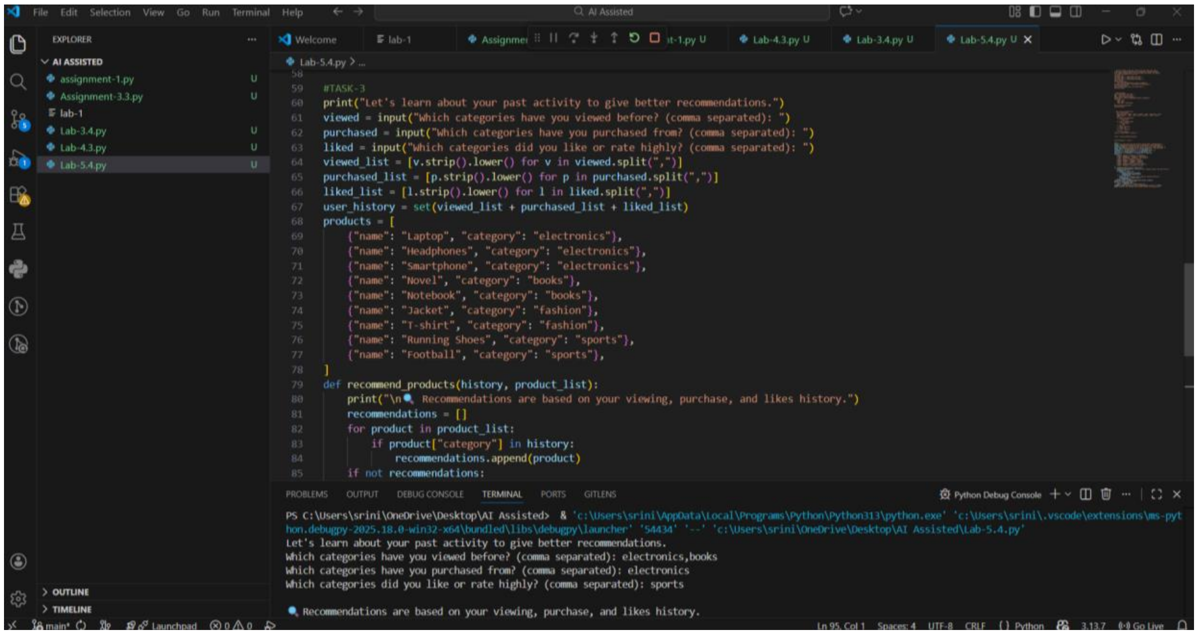
Prompt :

Create a Python program that recommends products based on user history (viewed, purchased, liked categories).
Add comments to explain transparency, fairness (not favoring only one category), and include a feedback option for users.

Code :

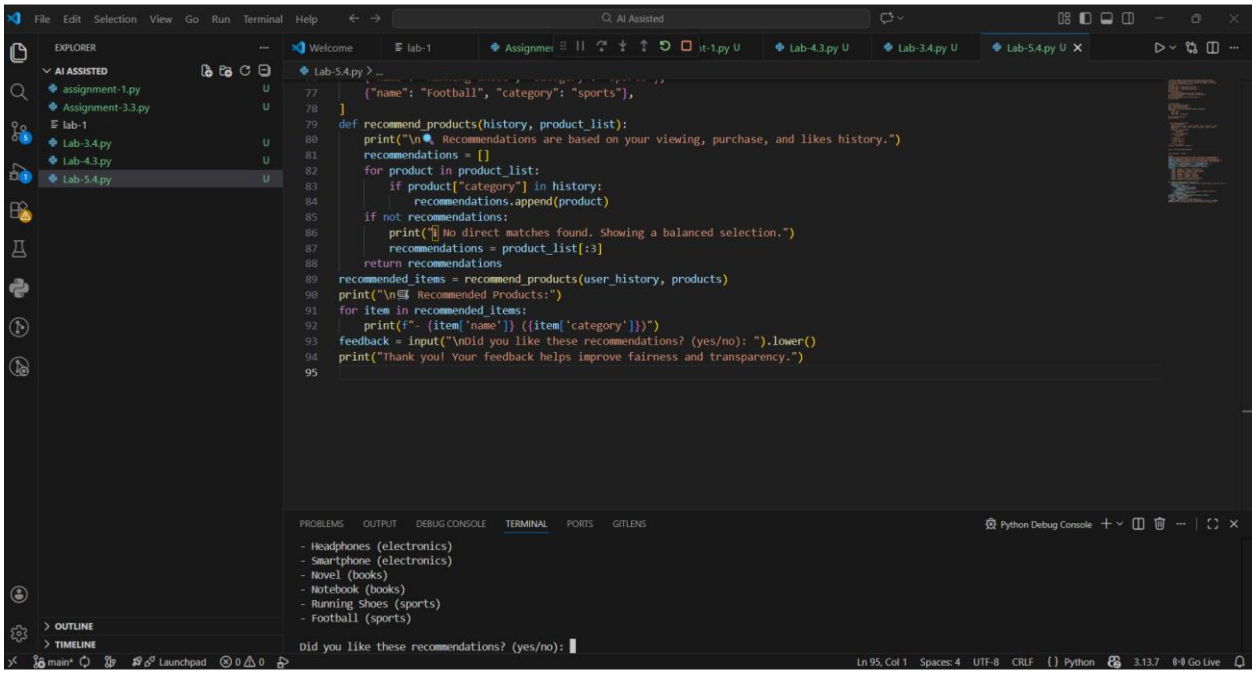
```
print("Let's learn about your past activity to give better recommendations.")
viewed = input("Which categories have you viewed before? (comma separated): ")
purchased = input("Which categories have you purchased from? (comma separated): ")
liked = input("Which categories did you like or rate highly? (comma separated): ")
viewed_list = [v.strip().lower() for v in viewed.split(",")]
purchased_list = [p.strip().lower() for p in purchased.split(",")]
liked_list = [l.strip().lower() for l in liked.split(",")]
user_history = set(viewed_list + purchased_list + liked_list)
products = [
   {"name": "Laptop", "category": "electronics"},
   {"name": "Headphones", "category": "electronics"},
   {"name": "Smartphone", "category": "electronics"},
   {"name": "Novel", "category": "books"},
   {"name": "Notebook", "category": "books"},
   {"name": "Jacket", "category": "fashion"},
   {"name": "T-shirt", "category": "fashion"},
   {"name": "Running Shoes", "category": "sports"},
   {"name": "Football", "category": "sports"},
]
def recommend_products(history, product_list):
    print("\n Recommendations are based on your viewing, purchase, and likes history.")
   recommendations = []
   for product in product_list:
      if product["category"] in history:
         recommendations.append(product)
    if not recommendations:
      print("ℹ️ No direct matches found. Showing a balanced selection.")
        recommendations = product_list[:3]
    return recommendations
recommended_items = recommend_products(user_history, products)
print("\n Recommended Products:")
for item in recommended_items:
    print(f"- {item['name']} ({item['category']})")
feedback = input("\n Did you like these recommendations? (yes/no): ").lower()
print("Thank you! Your feedback helps improve fairness and transparency.")'''
```

Output :



```
#TASK-3
print("Let's learn about your past activity to give better recommendations.")
viewed = input("Which categories have you viewed before? (comma separated): ")
purchased = input("Which categories have you purchased from? (comma separated): ")
liked = input("Which categories did you like or rate highly? (comma separated): ")
viewed_list = [v.strip().lower() for v in viewed.split(",")]
purchased_list = [p.strip().lower() for p in purchased.split(",")]
liked_list = [l.strip().lower() for l in liked.split(",")]
user_history = set(viewed_list + purchased_list + liked_list)
products = [
    {"name": "Laptop", "category": "electronics"},
    {"name": "Headphones", "category": "electronics"},
    {"name": "Smartphone", "category": "electronics"},
    {"name": "Novel", "category": "books"},
    {"name": "Notebook", "category": "books"},
    {"name": "Jacket", "category": "fashion"},
    {"name": "T-shirt", "category": "fashion"},
    {"name": "Running Shoes", "category": "sports"},
    {"name": "Football", "category": "sports"},
]
def recommend_products(history, product_list):
    print("\n🔍 Recommendations are based on your viewing, purchase, and likes history.")
    recommendations = []
    for product in product_list:
        if product["category"] in history:
            recommendations.append(product)
    if not recommendations:
```

```
PS C:\Users\srini\OneDrive\Desktop\AI Assisted>  & 'c:\Users\srini\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\srini\.vscode\extensions\ms-pyt
hon.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '54434' '--' 'c:\Users\srini\OneDrive\Desktop\AI Assisted\Lab-5.4.py'
Let's learn about your past activity to give better recommendations.
Which categories have you viewed before? (comma separated): electronics,books
Which categories have you purchased from? (comma separated): electronics
Which categories did you like or rate highly? (comma separated): sports

🔍 Recommendations are based on your viewing, purchase, and likes history.
```



```python
    {"name": "Football", "category": "sports"},
]
def recommend_products(history, product_list):
    print("\n🔍 Recommendations are based on your viewing, purchase, and likes history.")
    recommendations = []
    for product in product_list:
        if product["category"] in history:
            recommendations.append(product)
    if not recommendations:
        print("ℹ No direct matches found. Showing a balanced selection.")
        recommendations = product_list[:3]
    return recommendations
recommended_items = recommend_products(user_history, products)
print("\n🛒 Recommended Products:")
for item in recommended_items:
    print(f"- {item['name']} ({item['category']})")
feedback = input("\nDid you like these recommendations? (yes/no): ").lower()
print("Thank you! Your feedback helps improve fairness and transparency.")
```

```
- Headphones (electronics)
- Smartphone (electronics)
- Novel (books)
- Notebook (books)
- Running Shoes (sports)
- Football (sports)

Did you like these recommendations? (yes/no): 
```

Analysis :

In this task, I created a product recommendation system based on user interests. The system suggests products from categories the user has viewed or liked.The program also explains how recommendations are made (transparency), avoids suggesting only one category (fairness), and asks for user feedback. This shows how recommendation systems can be made more ethical and user-friendly.

TASK-4:

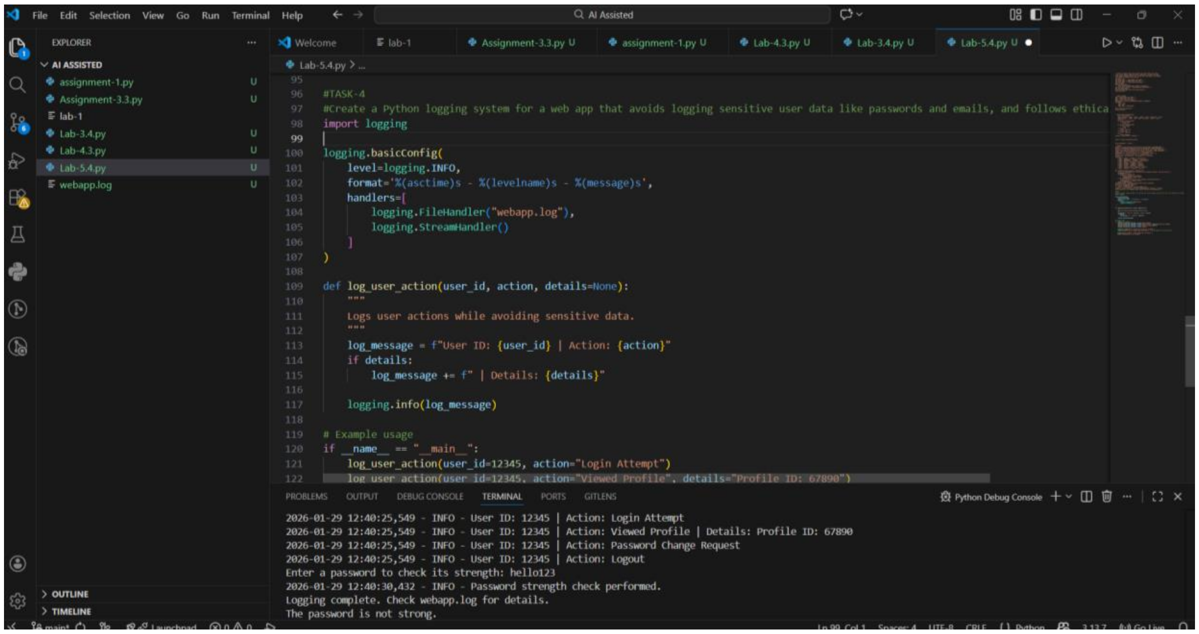
Prompt :

Generate Python logging code for a web app that records user actions.
Make sure the logs do NOT store sensitive data like passwords or emails.
Add comments about safe and ethical logging practices.

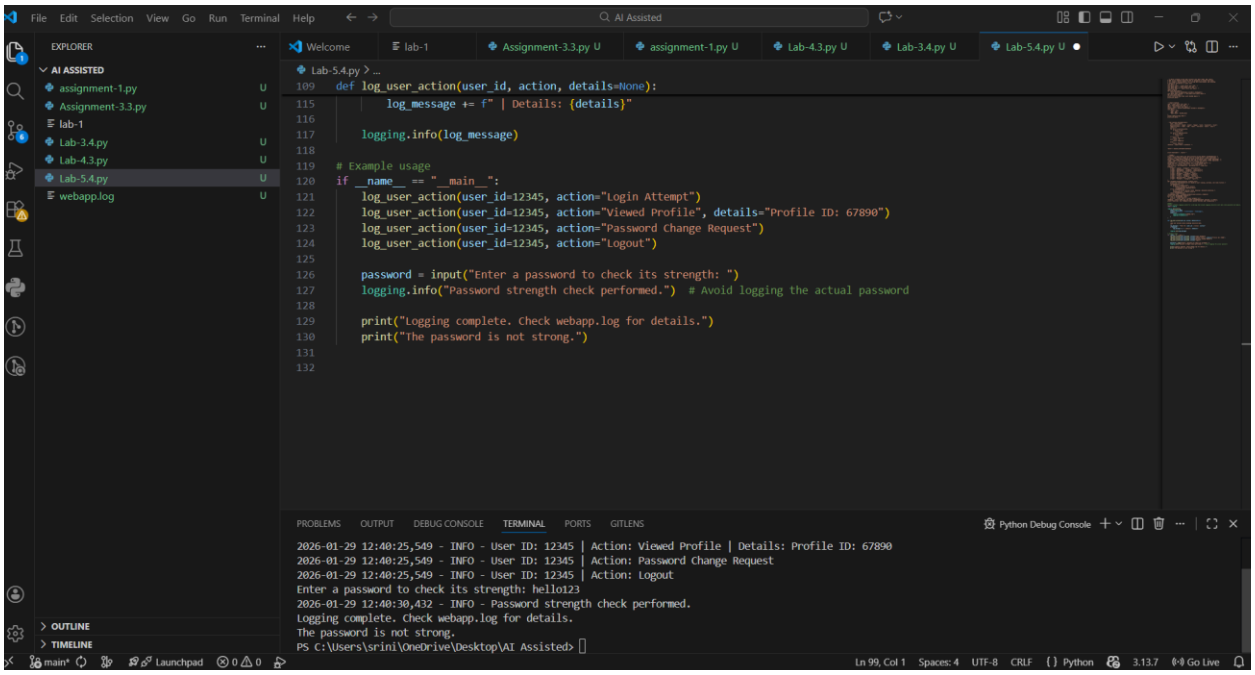
Code :

```python
import logging

logging.basicConfig(
   level=logging.INFO,
   format='%(asctime)s - %(levelname)s - %(message)s',
   handlers=[
      logging.FileHandler("webapp.log"),
      logging.StreamHandler()
   ]
)

def log_user_action(user_id, action, details=None):
    Logs user actions while avoiding sensitive data.

   log_message = f"User ID: {user_id} | Action: {action}"
   if details:
      log_message += f" | Details: {details}"

   logging.info(log_message)


if __name__ == "__main__":
   log_user_action(user_id=12345, action="Login Attempt")
   log_user_action(user_id=12345, action="Viewed Profile", details="Profile ID: 67890")
   log_user_action(user_id=12345, action="Password Change Request")
   log_user_action(user_id=12345, action="Logout")

   password = input("Enter a password to check its strength: ")
   logging.info("Password strength check performed.")  # Avoid logging the actual
password

   print("Logging complete. Check webapp.log for details.")
   print("The password is not strong.")
```

Output :


```python
#TASK-4
#Create a Python logging system for a web app that avoids logging sensitive user data like passwords and emails, and follows ethica
import logging

logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s',
    handlers=[
        logging.FileHandler("webapp.log"),
        logging.StreamHandler()
    ]
)

def log_user_action(user_id, action, details=None):
    """
    Logs user actions while avoiding sensitive data.
    """
    log_message = f"User ID: {user_id} | Action: {action}"
    if details:
        log_message += f" | Details: {details}"

    logging.info(log_message)

# Example usage
if __name__ == "__main__":
    log_user_action(user_id=12345, action="Login Attempt")
    log_user_action(user_id=12345, action="Viewed Profile", details="Profile ID: 67890")
```

```
2026-01-29 12:40:25,549 - INFO - User ID: 12345 | Action: Login Attempt
2026-01-29 12:40:25,549 - INFO - User ID: 12345 | Action: Viewed Profile | Details: Profile ID: 67890
2026-01-29 12:40:25,549 - INFO - User ID: 12345 | Action: Password Change Request
2026-01-29 12:40:25,549 - INFO - User ID: 12345 | Action: Logout
Enter a password to check its strength: hello123
2026-01-29 12:40:30,432 - INFO - Password strength check performed.
Logging complete. Check webapp.log for details.
The password is not strong.
```


```python
def log_user_action(user_id, action, details=None):
            log_message += f" | Details: {details}"

        logging.info(log_message)

# Example usage
if __name__ == "__main__":
    log_user_action(user_id=12345, action="Login Attempt")
    log_user_action(user_id=12345, action="Viewed Profile", details="Profile ID: 67890")
    log_user_action(user_id=12345, action="Password Change Request")
    log_user_action(user_id=12345, action="Logout")

    password = input("Enter a password to check its strength: ")
    logging.info("Password strength check performed.")  # Avoid logging the actual password

    print("Logging complete. Check webapp.log for details.")
    print("The password is not strong.")
```

```
2026-01-29 12:40:25,549 - INFO - User ID: 12345 | Action: Viewed Profile | Details: Profile ID: 67890
2026-01-29 12:40:25,549 - INFO - User ID: 12345 | Action: Password Change Request
2026-01-29 12:40:25,549 - INFO - User ID: 12345 | Action: Logout
Enter a password to check its strength: hello123
2026-01-29 12:40:30,432 - INFO - Password strength check performed.
Logging complete. Check webapp.log for details.
The password is not strong.
PS C:\Users\srini\OneDrive\Desktop\AI Assisted>
```

Analysis :

In this task, I developed a logging system that records user actions in an application. The important part is that the system does not log sensitive data like passwords or email addresses.The code includes comments explaining safe logging practices. This shows the importance of protecting user privacy while still keeping useful system logs.

TASK-5:

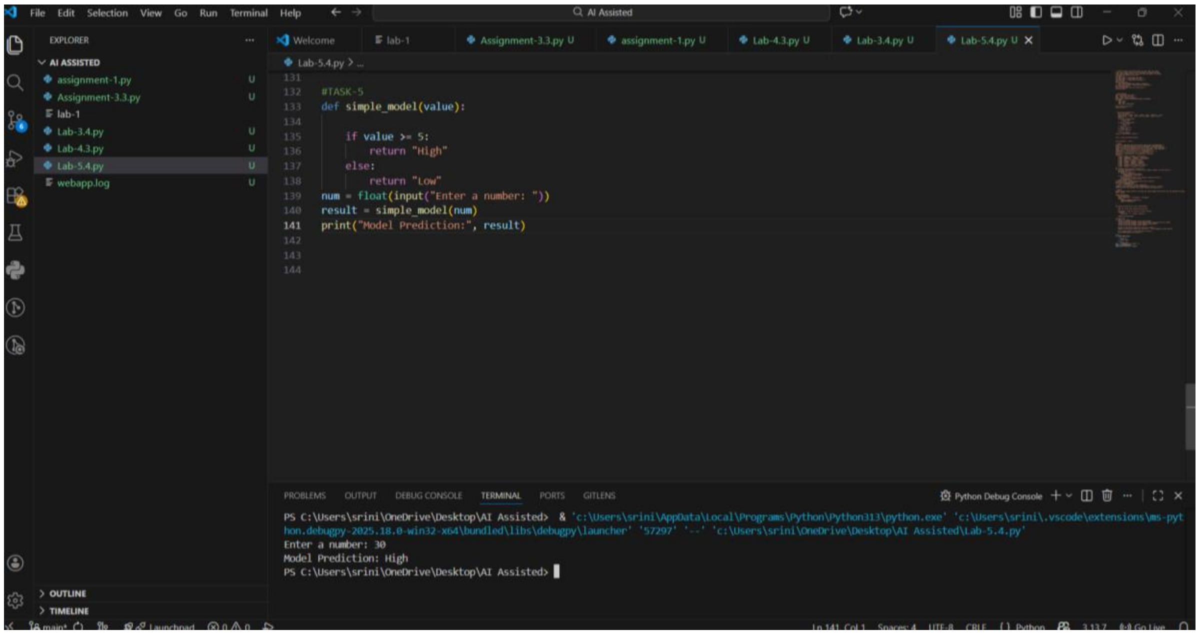
Prompt :

Create a simple Python prediction model.
Add documentation or comments explaining responsible use, model limitations, fairness, and that human review may be needed.

Code :

```
def simple_model(value):

    if value >= 5:
        return "High"
    else:
        return "Low"
num = float(input("Enter a number: "))
result = simple_model(num)
print("Model Prediction:", result)
```

Output :



```
#TASK-5
def simple_model(value):

    if value >= 5:
        return "High"
    else:
        return "Low"
num = float(input("Enter a number: "))
result = simple_model(num)
print("Model Prediction:", result)
```

```
PS C:\Users\srini\OneDrive\Desktop\AI Assisted>  & 'c:\Users\srini\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\srini\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '57297' '--' 'c:\Users\srini\OneDrive\Desktop\AI Assisted\Lab-5.4.py'
Enter a number: 30
Model Prediction: High
PS C:\Users\srini\OneDrive\Desktop\AI Assisted>
```

Analysis :

In this task, I created a simple prediction model. I also added notes about how the model should be used responsibly.The program explains that the model may not always be correct, can be biased, and should not be used alone for important decisions. It also mentions that users should understand how the model works. This shows the importance of fairness, transparency, and human supervision in AI systems.