

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 :

The screenshot shows a Visual Studio Code (VS Code) interface. The left sidebar has icons for File Explorer, Search, and others. The main area shows an 'EXPLORER' view with files like 'AI ASSISTED', 'assignment-1.py', 'Assignment-3.3.py', 'lab-1', 'Lab-3.4.py', 'Lab-4.3.py', and 'Lab-5.4.py'. The 'Lab-5.4.py' file is open in the center, displaying the following Python code:

```
19 import hashlib
20 name = input("Enter your name: ")
21 age = input("Enter your age: ")
22 email = input("Enter your email: ")
23 hashed_email = hashlib.sha256(email.encode()).hexdigest()
24 user_data = [
25     "name": name,
26     "age": age,
27     "email_hash": hashed_email
28 ]
29 print("\nSecure User Data:")
30 print(user_data)
```

Below the code editor, the 'TERMINAL' tab is active, showing the output of running the script:

```
.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': 'dadaacc07418a7f6eae276a1597b11379f2dd8b42ef55894eba021c7be5fb283'}]
```

The status bar at the bottom indicates the code is 31.3% complete.

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"
```

```

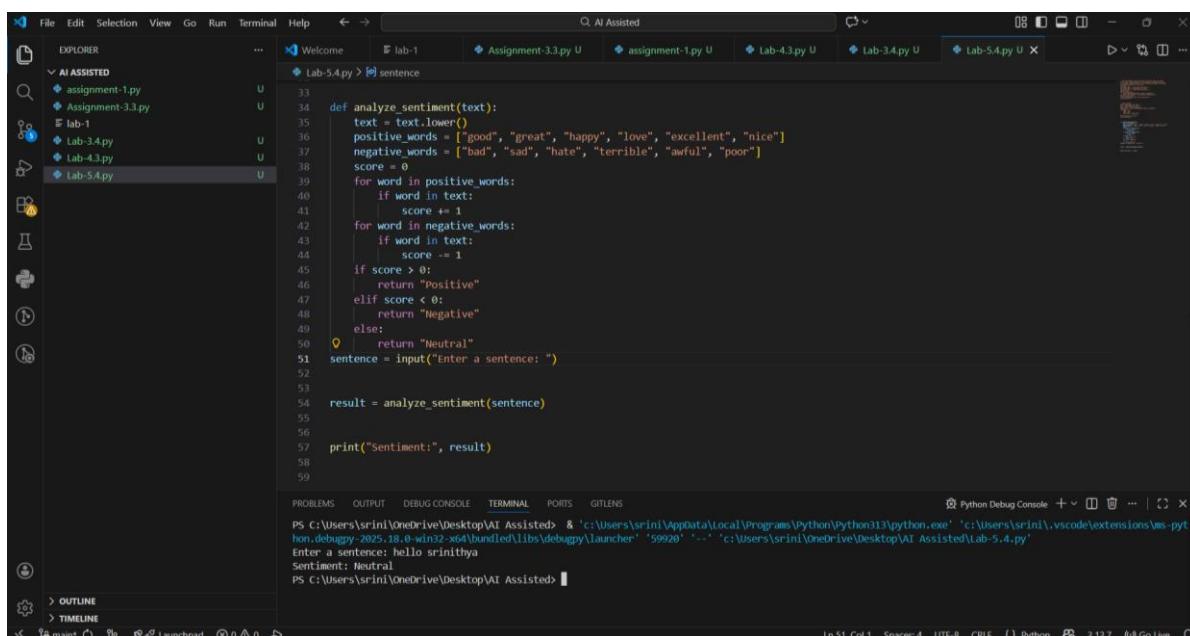
elif score < 0:
    return "Negative"
else:
    return "Neutral"
sentence = input("Enter a sentence: ")

result = analyze_sentiment(sentence)

print("Sentiment:", result)

```

Output :



The screenshot shows the Microsoft Visual Studio Code interface. The code editor displays the Python script for sentiment analysis. The terminal at the bottom shows the execution of the script and its output.

```

File Edit Selection View Go Run Terminal Help ← → ⌘ AI Assisted
EXPLORER lab-1 Assignment-3.3.py assignment-1.py Lab-3.4.py Lab-4.3.py Lab-5.4.py
AI ASSISTED
assignment-1.py U Assignment-3.3.py U assignment-1.py U Lab-3.4.py U Lab-5.4.py U
Lab-1 Lab-3.4.py U Lab-4.3.py U Lab-5.4.py U
Lab-3.4.py U Lab-4.3.py U Lab-5.4.py U
Lab-5.4.py U
Lab-5.4.py > [e] sentence
33
34 def analyze_sentiment(text):
35     text = text.lower()
36     positive_words = ["good", "great", "happy", "love", "excellent", "nice"]
37     negative_words = ["bad", "sad", "hate", "terrible", "awful", "poor"]
38     score = 0
39     for word in positive_words:
40         if word in text:
41             score += 1
42     for word in negative_words:
43         if word in text:
44             score -= 1
45     if score > 0:
46         return "Positive"
47     elif score < 0:
48         return "Negative"
49     else:
50         return "Neutral"
51 sentence = input("Enter a sentence: ")
52
53
54 result = analyze_sentiment(sentence)
55
56
57 print("Sentiment:", result)
58
59
Python Debug Console + × ⌘ ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
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
hon.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

```

The status bar at the bottom indicates the file is 51 lines long, has 4 spaces per tab, and is using Python 3.13.2. The Go Live button is also visible.

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("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 :

The screenshot shows the Visual Studio Code interface with the AI Assisted extension active. The Explorer sidebar on the left lists files in the 'AI ASSISTED' folder. The main editor area contains Python code for recommendation logic. The Python Debug Console at the bottom shows the execution of the script and user interaction.

```

File Edit Selection View Go Run Terminal Help ← → ⌘ AI Assisted
EXPLORER AI ASSISTED
assignment-1.py U
Assignment-3.3.py U
lab-1 U
Lab-3.4.py U
Lab-4.3.py U
Lab-5.4.py U
Welcome Lab-1 Assignment-1.py U Lab-3.4.py U Lab-5.4.py U
58 #TASK 3
59
60 print("Let's learn about your past activity to give better recommendations.")
61 viewed = input("Which categories have you viewed before? (comma separated): ")
62 purchased = input("Which categories have you purchased from? (comma separated): ")
63 liked = input("Which categories did you like or rate highly? (comma separated): ")
64 viewed_list = [v.strip().lower() for v in viewed.split(",")]
65 purchased_list = [p.strip().lower() for p in purchased.split(",")]
66 liked_list = [l.strip().lower() for l in liked.split(",")]
67 user_history = set(viewed_list + purchased_list + liked_list)
68 products = [
69     {"name": "Laptop", "category": "electronics"},
70     {"name": "Headphones", "category": "electronics"},
71     {"name": "Smartphone", "category": "electronics"},
72     {"name": "Novel", "category": "books"},
73     {"name": "Notebook", "category": "books"},
74     {"name": "Jacket", "category": "fashion"},
75     {"name": "T-shirt", "category": "fashion"},
76     {"name": "Running Shoes", "category": "sports"},
77     {"name": "Football", "category": "sports"},
78 ]
79 def recommend_products(history, product_list):
80     print("\n Recommendations are based on your viewing, purchase, and likes history.")
81     recommendations = []
82     for product in product_list:
83         if product["category"] in history:
84             recommendations.append(product)
85     if not recommendations:
86
PS C:\Users\sriniv\OneDrive\Desktop\AI Assisted & c:\Users\srini\AppData\Local\Programs\Python\Python313\python.exe 'c:\Users\srini\vscode\extensions\ai-assisted\hon.debugpy-2025.18.0-win32-x64\bundle\libs\debug\launcher' '5d434' '--' '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.

```

```

File Edit Selection View Go Run Terminal Help < > AI Assisted Welcome lab-1 Assignment-1.py lab-1 Lab-3.3.py Lab-3.4.py Lab-4.3.py Lab-4.5.py Lab-5.4.py
EXPLORER
AI ASSISTED
assignment-1.py
Assignment-3.3.py
lab-1
Lab-3.4.py
Lab-4.3.py
Lab-5.4.py
TERMINAL
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
- Headphones (electronics)
- Smartphone (electronics)
- Novel (books)
- Notebook (books)
- Running Shoes (sports)
- Football (sports)
Did you like these recommendations? (yes/no): 
Ln 95, Col 1 Spaces: 4 UTF-8 CRLF Python 3.13.7 Go Live

```

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 :

```
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 :

The screenshot shows a Python code editor with several tabs open. The main tab displays a script named `Lab-5.py` containing a logging configuration and a function to log user actions. The code uses the `logging` module to log messages to a file named `webapp.log`. The `log_user_action` function takes a user ID, action, and optional details. It logs the user ID, action, and details if provided. The log message is formatted as `User ID: [user_id] | Action: [action]`. If details are provided, they are appended to the log message. The code also includes a section for example usage with a command-line argument check.

```
File Edit Selection View Go Run Terminal Help < > Q: AI Assisted

EXPLORER
AI ASSISTED
assignment-1.py
Assignment-3.3.py
lab-1
Lab-3.4.py
Lab-4.3.py
Lab-5.py
webapp.log

Welcome lab-1 Assignment-3.3.py assignment-1.py Lab-4.3.py Lab-3.4.py Lab-5.py

Lab-5.py > ...
95
96 #TASK-4
97 #Create a Python logging system for a web app that avoids logging sensitive user data like passwords and emails, and follows ethical
98 import logging
99
100 logging.basicConfig(
101     level=logging.INFO,
102     format='%(asctime)s - %(levelname)s - %(message)s',
103     handlers=[
104         logging.FileHandler("webapp.log"),
105         logging.StreamHandler()
106     ]
107 )
108
109 def log_user_action(user_id, action, details=None):
110     """
111     Logs user actions while avoiding sensitive data.
112     """
113     log_message = f"User ID: {user_id} | Action: {action}"
114     if details:
115         log_message += f" | Details: {details}"
116
117     logging.info(log_message)
118
119 # Example usage
120 if __name__ == "__main__":
121     log_user_action(user_id=12345, action="Login Attempt")
122     log_user_action(user_id=12345, action="Viewed Profile", details="Profile ID: 67890")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Python Debug Console + □ ... || x
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.

> OUTLINE
> TIMELINE
< %main( ) %< %Launchpad D 3.117 4.1.17 Full Go Up
```

The screenshot shows the Visual Studio Code interface with the 'AI Assisted' extension active. The left sidebar displays a tree view of files under 'EXPLORER', including 'assignment-1.py', 'Assignment-3.3.py', 'lab-1', 'Lab-3.4.py', 'Lab-4.3.py', 'Lab-5.4.py', and 'webapp.log'. The main editor area shows a Python script named 'Lab-5.4.py' with the following code:

```
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.)
```

The bottom status bar shows the terminal output:

```
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.
```

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 :

The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows files in the AI ASSISTED folder: assignment-1.py, Assignment-3.3.py, lab-1, Lab-3.4.py, Lab-4.3.py, Lab-5.4.py, and webapp-log.
- Code Editor:** The file Lab-5.4.py is open, displaying the provided Python code.
- Terminal:** The terminal shows the command 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.python.debug-2025.18.0-win32-x64\bundled\libs\debug\launcher' '57297' '--' 'c:\Users\srini\OneDrive\Desktop\AI Assisted\Lab-5.4.py'. It then prompts for input: Enter a number: 30, followed by Model Prediction: High.
- Status Bar:** Shows the current file is Lab-5.4.py, and other details like the current column (Col 1), spaces (Spaces: 4), and line numbers (Line 141).

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.