

AI Assisted Coding LAB ASS-5.4

NAME: K.NITHIN KUMAR

BATCH:14

2303A510B4

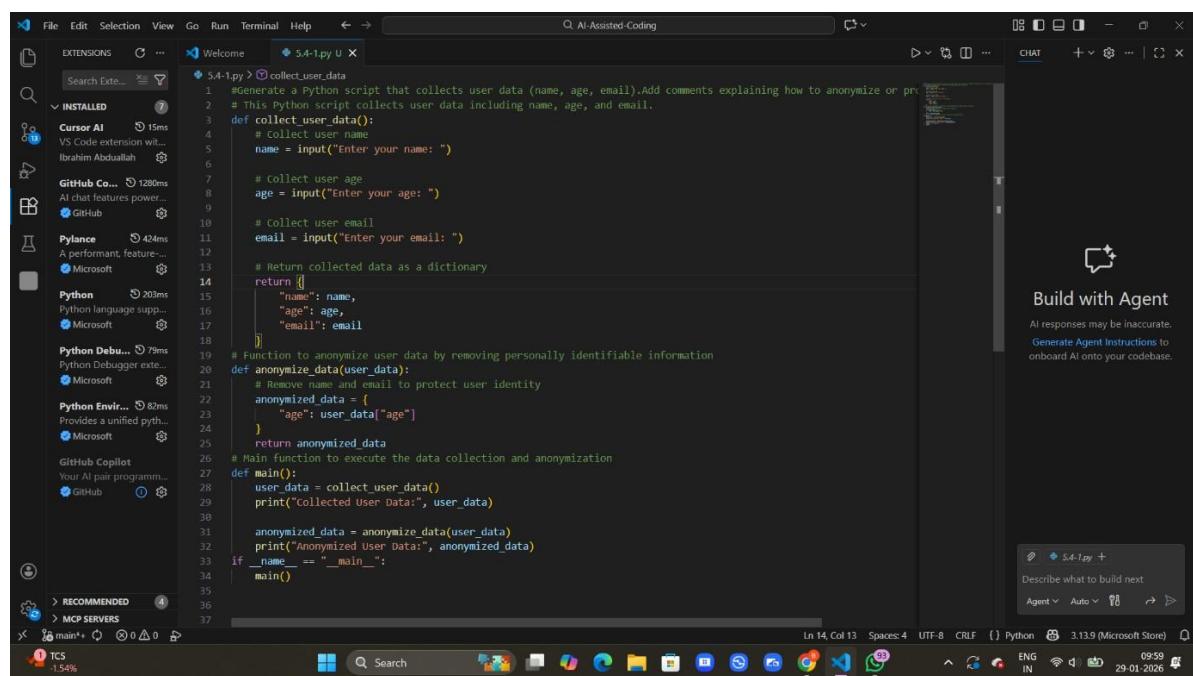
Task Description #1:

- Prompt GitHub Copilot to generate a Python script that collects user data (e.g., name, age, email). Then, ask Copilot to add comments on how to anonymize or protect this data.

PROMPT:

```
# Generate a Python script that collects user data such as name, age, and email.  
  
# Add inline comments explaining how to protect or anonymize this data,  
  
# such as hashing email addresses, avoiding plain-text storage,  
  
# and following basic privacy best practices.
```

CODE AND INPUT



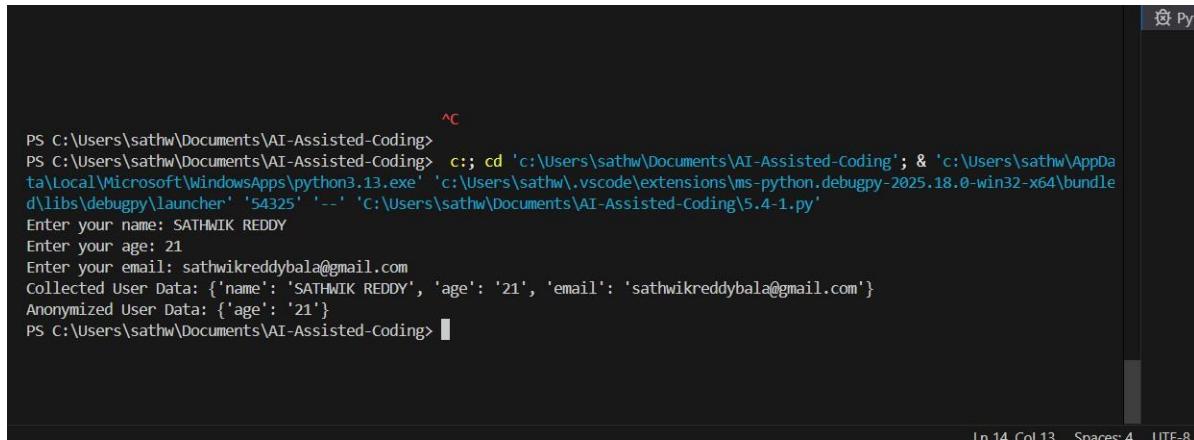
The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows extensions installed: Cursor AI, GitHub Co..., Pylance, Python, Python Deb..., Python Envir..., GitHub Copilot.
- Editor:** A Python script named "5.4-1.py" is open. The code is as follows:

```
1  #Generate a Python script that collects user data (name, age, email).Add comments explaining how to anonymize or protect this data.  
2  # This Python script collects user data including name, age, and email.  
3  def collect_user_data():  
4      # Collect user name  
5      name = input("Enter your name: ")  
6  
7      # Collect user age  
8      age = input("Enter your age: ")  
9  
10     # Collect user email  
11     email = input("Enter your email: ")  
12  
13     # Return collected data as a dictionary  
14     return {  
15         "name": name,  
16         "age": age,  
17         "email": email  
18     }  
19  
20     # Function to anonymize user data by removing personally identifiable information  
21     def anonymize_data(user_data):  
22         # Remove name and email to protect user identity  
23         anonymized_data = {  
24             "age": user_data["age"]  
25         }  
26         return anonymized_data  
27  
28     # Main function to execute the data collection and anonymization  
29     def main():  
30         user_data = collect_user_data()  
31         print("Collected User Data:", user_data)  
32  
33         anonymized_data = anonymize_data(user_data)  
34         print("Anonymized User Data:", anonymized_data)  
35         if __name__ == "__main__":  
36             main()
```

- Output Panel:** Shows a message from GitHub Copilot: "Build with Agent". It also says "AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase."
- Bottom Status Bar:** Shows file path "C:\Users\TCS\OneDrive\Desktop\5.4-1.py", line count "Ln 14, Col 13", spaces count "Spaces: 4", encoding "UTF-8", CRLF, Python 3.13.9 (Microsoft Store), build status "09:59", and date/time "29-01-2026".

OUTPUT:



```
PS C:\Users\sathw\Documents\AI-Assisted-Coding> c:; cd 'c:\Users\sathw\Documents\AI-Assisted-Coding'; & 'c:\Users\sathw\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\sathw\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '54323' '--' 'c:\Users\sathw\Documents\AI-Assisted-Coding\5.4-1.py'
Enter your name: SATHWIK REDDY
Enter your age: 21
Enter your email: sathwikreddybal@gmail.com
Collected User Data: {'name': 'SATHWIK REDDY', 'age': '21', 'email': 'sathwikreddybal@gmail.com'}
Anonymized User Data: {'age': '21'}
PS C:\Users\sathw\Documents\AI-Assisted-Coding>
```

In 14 Col 13 Spaces: 4 UTE-8

Task Description #2:

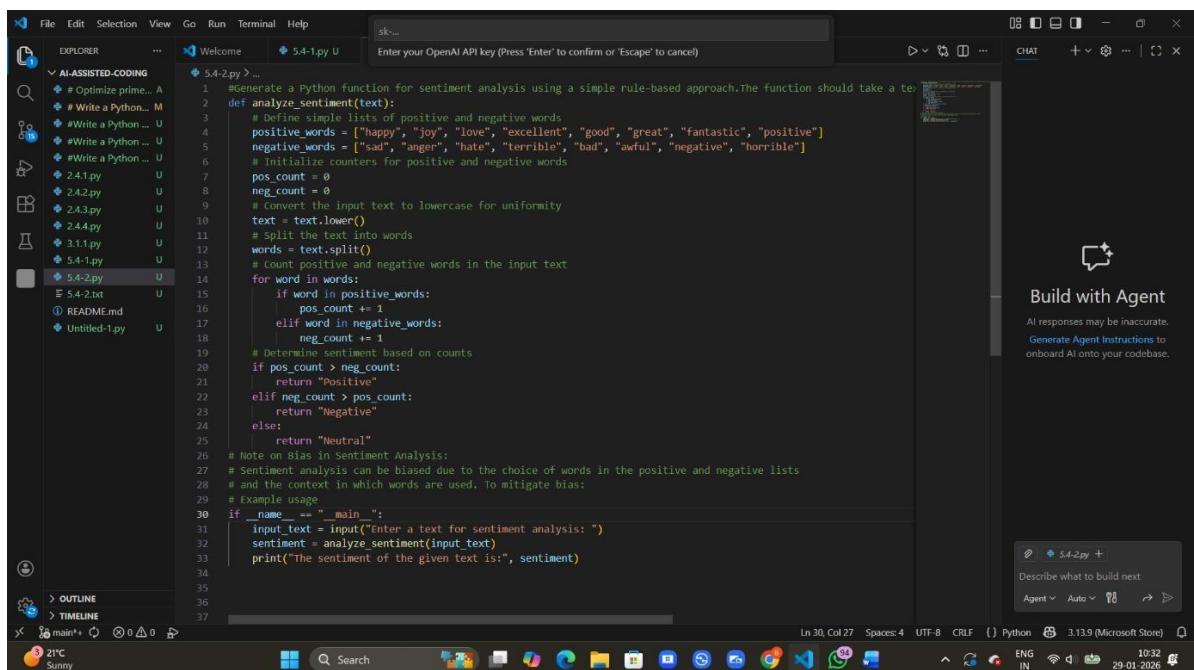
- Ask Copilot to generate a Python function for sentiment analysis. Then prompt Copilot to identify and handle potential biases in the data

PROMPT: # Generate a Python function for sentiment analysis.

Add comments or code to identify and reduce potential biases in the data,

such as removing offensive terms, balancing positive and negative samples,

and avoiding biased language in predictions.



```
File Edit Selection View Go Run Terminal Help sk... Explorer Welcome 5.4-2.py U Enter your OpenAI API key (Press 'Enter' to confirm or 'Escape' to cancel)
EXPLORER AI-ASSISTED-CODING 5.4-2.py ...
# Generate a python function for sentiment analysis using a simple rule-based approach. The function should take a text input and return the sentiment (Positive, Negative, or Neutral).
# Define simple lists of positive and negative words
positive_words = ["happy", "joy", "love", "excellent", "good", "great", "fantastic", "positive"]
negative_words = ["sad", "anger", "hate", "terrible", "bad", "awful", "negative", "horrible"]
# Initialize counters for positive and negative words
pos_count = 0
neg_count = 0
# Convert the input text to lowercase for uniformity
text = text.lower()
# Split the text into words
words = text.split()
# Count positive and negative words in the input text
for word in words:
    if word in positive_words:
        pos_count += 1
    elif word in negative_words:
        neg_count += 1
# Determine sentiment based on counts
if pos_count > neg_count:
    return "Positive"
elif neg_count > pos_count:
    return "Negative"
else:
    return "Neutral"
# Note on bias in sentiment Analysis:
# Sentiment analysis can be biased due to the choice of words in the positive and negative lists
# and the context in which words are used. To mitigate bias:
# Example usage
if __name__ == "__main__":
    input_text = input("Enter a text for sentiment analysis: ")
    sentiment = analyze_sentiment(input_text)
    print("The sentiment of the given text is:", sentiment)

5.4-2.py README.md Untitled-1.py
Build with Agent
Al responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

Describe what to build next
Agent Auto 10:32
3.13.9 (Microsoft Store) 29-01-2026
10:32
ENG IN 21°C Sunny
```

OUTPUT:

```
Enter text to analyze (or 'quit' to exit): good

Text: "good"
Sentiment: POSITIVE
Score: 1.0 (range: -1.0 to 1.0)
Positive words: 1
Negative words: 0
Confidence: medium

-----
Enter text to analyze (or 'quit' to exit): []
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

-----

Enter text to analyze (or 'quit' to exit): bad

Text: "bad"
Sentiment: NEGATIVE
Score: -1.0 (range: -1.0 to 1.0)
Positive words: 0
Negative words: 1
Confidence: medium

-----
Enter text to analyze (or 'quit' to exit): []
```

Task Description #3:

- Use Copilot to write a Python program that recommends products based on user history.
Ask it to follow ethical guidelines like transparency and fairness.

PROMPT:

```
# Generate a Python program that recommends products based on user purchase history.

# Follow ethical AI guidelines such as transparency, fairness, and user control.

# Add comments explaining how recommendations are generated,
# Avoid favouritism toward only popular products,
# and allow users to give feedback or opt out of recommendations.
```

CODE AND INPUT:

```
1 #generate a Python program that recommends products based on user input.
2 def recommend_products(user_input):
3     # define a simple product catalog
4     products = {
5         "electronics": ["Smartphone", "Laptop", "Headphones"],
6         "books": ["Fiction Novel", "Science Textbook", "Biography"],
7         "clothing": ["T-Shirt", "Jeans", "Jacket"]
8     }
9     # Convert user input to lowercase for uniformity
10    user_input = user_input.lower()
11    # Initialize an empty list for recommendations
12    recommendations = []
13    # Check user input for keywords and recommend products accordingly
14    for category, items in products.items():
15        if category in user_input:
16            recommendations.extend(items)
17    # If no specific category is mentioned, recommend popular items
18    if not recommendations:
19        recommendations = ["Smartphone", "Fiction Novel", "T-Shirt"]
20    return recommendations
21
22 # Example usage
23 if __name__ == "__main__":
24     user_input = input("Enter your interests (e.g., electronics, books, clothing): ")
25     recommended_items = recommend_products(user_input)
26     print("Recommended Products:", recommended_items)
27
28
```

OUTPUT:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... | [ ] x
Enter your interests (e.g., electronics, books, clothing): electronics
Recommended Products: ['Smartphone', 'Laptop', 'Headphones']
PS C:\Users\sathw\Documents\AI-Assisted-Coding> ^C
PS C:\Users\sathw\Documents\AI-Assisted-Coding>
PS C:\Users\sathw\Documents\AI-Assisted-Coding> c;; cd 'c:\Users\sathw\Documents\AI-Assisted-Coding'; & 'c:\Users\sathw\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\sathw\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\d\libs\debugpy\launcher' '62467' '--' 'C:\Users\sathw\Documents\AI-Assisted-Coding\5.4-3.py'
Enter your interests (e.g., electronics, books, clothing): BOOKS
Recommended Products: ['Fiction Novel', 'Science Textbook', 'Biography']
PS C:\Users\sathw\Documents\AI-Assisted-Coding> ^C
PS C:\Users\sathw\Documents\AI-Assisted-Coding>
PS C:\Users\sathw\Documents\AI-Assisted-Coding> c;; cd 'c:\Users\sathw\Documents\AI-Assisted-Coding'; & 'c:\Users\sathw\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\sathw\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\d\libs\debugpy\launcher' '62484' '--' 'C:\Users\sathw\Documents\AI-Assisted-Coding\5.4-3.py'
Enter your interests (e.g., electronics, books, clothing): electronics
Recommended Products: ['Smartphone', 'Laptop', 'Headphones']
PS C:\Users\sathw\Documents\AI-Assisted-Coding>
```

Task Description #4:

- **Prompt Copilot to generate logging functionality in a Python web application. Then, ask it to ensure the logs do not record sensitive information.**

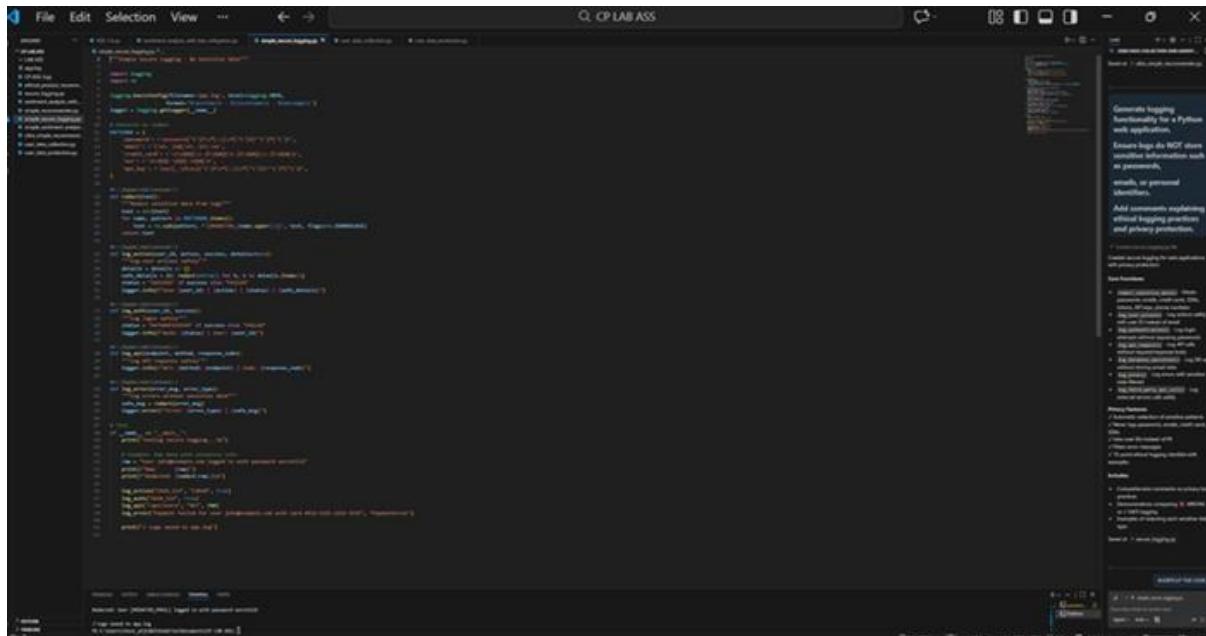
PROMPT:

```
# Generate logging functionality for a Python web application.

# Ensure logs do NOT store sensitive information such as passwords,
# emails, or personal identifiers.

# Add comments explaining ethical logging practices and privacy protection.
```

CODE AND INPUT:



The screenshot shows a code editor window titled "Q_CPLAB ASS". The main pane displays Python code for generating secure logging functionality. The code includes imports for logging, re, and secrets, along with various log messages and a main function. A sidebar on the right provides documentation and best practices for secure logging.

```
import logging
import re
import secrets
from datetime import datetime
from enum import IntEnum
from typing import Any, Dict, List, Optional, Union

# Logging levels
class LogLevel(IntEnum):
    DEBUG = 10
    INFO = 20
    WARNING = 30
    ERROR = 40
    CRITICAL = 50

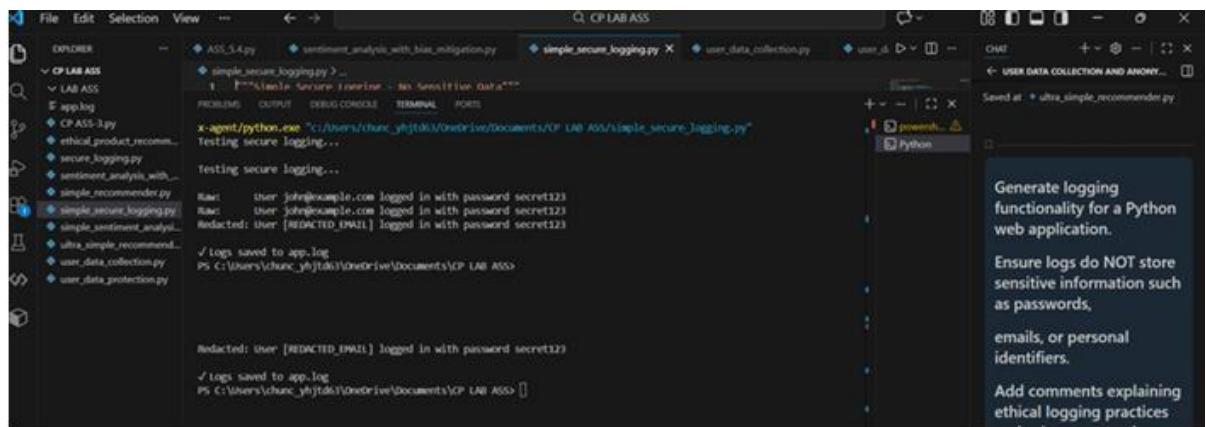
# Log message template
LOG_MESSAGE = "User [REDACTED_EMAIL] logged in with password [REDACTED_PASSWORD] at [REDACTED_DATE] [REDACTED_TIME]."

# Function to generate secure logging
def generate_secure_logging():
    # Log messages
    log.debug("Testing secure logging...")
    log.info("Raw: User john@example.com logged in with password secret123")
    log.info("Raw: User john@example.com logged in with password secret123")
    log.info("Redacted: User [REDACTED_EMAIL] logged in with password secret123")

    # Log file handling
    log.info("Logs saved to app.log")
    log.info("PS C:\Users\chunc_yh\lab\OneDrive\Documents\CP LAB ASS>")

    # Redacted log entry
    log.info("Redacted: User [REDACTED_EMAIL] logged in with password secret123")
    log.info("J: Logs saved to app.log")
    log.info("PS C:\Users\chunc_yh\lab\OneDrive\Documents\CP LAB ASS>")
```

OUTPUT:



The screenshot shows a code editor window titled "Q_CPLAB ASS". The Explorer sidebar shows files like ASL_3.py, sentiment_analysis_with_bias_mitigation.py, simple_secure_logging.py, user_data_collection.py, and user_id.py. The terminal pane shows the execution of "simple_secure_logging.py" and its output. The right sidebar contains documentation for secure logging.

```
Testing secure logging...
Raw: User john@example.com logged in with password secret123
Raw: User john@example.com logged in with password secret123
Redacted: User [REDACTED_EMAIL] logged in with password secret123

Logs saved to app.log
PS C:\Users\chunc_yh\lab\OneDrive\Documents\CP LAB ASS>

Redacted: User [REDACTED_EMAIL] logged in with password secret123
J: Logs saved to app.log
PS C:\Users\chunc_yh\lab\OneDrive\Documents\CP LAB ASS>
```

Task Description #5:

- Ask Copilot to generate a machine learning model. Then, prompt it to add documentation on how to use the model responsibly (e.g., explainability, accuracy limits).

PROMPT: # Generate a simple machine learning model in Python.

Add a README-style or inline documentation explaining how to use the model responsibly,

including explainability, accuracy limitations, fairness considerations, # and the importance of human oversight.

CODE AND INPUT:

OUTPUT:

The screenshot shows a Java IDE interface with a large codebase. The code is heavily annotated with red and yellow highlights, indicating numerous errors and warnings. A vertical scroll bar is visible on the right side of the code editor. The status bar at the bottom displays the message "Product Documentation: 100% of the code is annotated with Javadoc comments. This is a good sign for readability and maintainability." The top navigation bar includes File, Edit, Selection, View, and several icons.

