

AI Assisted Coding LAB ASS-5.4

NAME: CH. VENUGOPAL

BATCH:13

2303A51844

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.

The screenshot shows a code editor with multiple tabs open. The main tab contains a Python script named `user_data_collection.py`. This script defines a class `UserDataCollection` with methods for collecting user data and displaying it in various formats (original, anonymized, masked). It also includes a `print_recommendations` method and a `main` function that runs the collection script. A `__main__.py` file is also present, which contains a `main` function that calls the `main` function from `user_data_collection.py`.

On the right side of the editor, there is a sidebar with several sections:

- Code Features:** Lists features like "Server Side Collection", "Validations", "Email and Mail with regex patterns", and "Prevent injection attacks".
- Data Protection Functions:** Lists functions such as `hash_hexdigest`, `hash_hexdigest_hexdigest`, and `hash_hexdigest_hexdigest_hexdigest`.
- Storage Functions:** Lists functions for "Temporary Database", "Delete Data", and "Drop Table".
- Storage Database Dimension:** Lists database types: "MySQL", "PostgreSQL", "Oracle", "MongoDB", and "Redis".
- Every Function and Section:** Provides a detailed explanation of the code's purpose and security practices.

OUTPUT:

The screenshot shows the Microsoft Copilot IDE interface. In the terminal window, a user enters their full name, age, and email address. The script then processes this data into four categories: ORIGINAL DATA, ANONYMIZED DATA, HASHED DATA, and MINIMAL DATA. The right sidebar contains a detailed guide on data collection and anonymization, including sections on age ranges, storage options (plain text, hashed data, anonymized data, encrypted storage), and data protection best practices (HTTPS, encryption, access control, GDPR/CCPA). The code editor shows a file named `user_data_protection.py`.

```

PS C:\Users\chunc_yhjtd63\OneDrive\Documents\CP LAB ASS & C:/Users/chunc_yhjtd63> python user_data_protection.py

Enter your full name: VENUGOPAL
✓ Name accepted: VENUGOPAL
Enter your age: 19
✓ Age accepted: 19
Enter your email address: venugopalchunchu13@gmail.com
✓ Email accepted: v*****@gmail.com

-----
ORIGINAL DATA (Sensitive - Should not be logged/displayed in production)
-----
Name: VENUGOPAL
Age: 19
Email: venugopalchunchu13@gmail.com

-----
ANONYMIZED DATA (Safe for logging and analysis)
-----
User ID: 4E30C06D
Age Range: 10-19 (Adult)
Email (Masked): v*****@gmail.com

-----
HASHED DATA (For database storage and duplicate detection)
-----
Email Hash (SHA-256): 61e434e2a4f8ba288f5fb1d980dec6b...
Email Hash (Salted): 0628df7938e1088443b99fb7fbfb7ea...
Full hash stored in database, not visible to users

-----
MINIMAL DATA (Only what's necessary)
-----
```

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.

```

# Sentiment words
POSITIVE = ['good', 'great', 'love', 'excellent', 'happy', 'wonderful']
NEGATIVE = ['bad', 'terrible', 'hate', 'awful', 'sad', 'horrible']

# Stop words
STOP_WORDS = {'itself', 'ourselves', 'herself'}
GENERED_WORDS = ('businessman', 'professional', 'housewife', 'housekeeper')

# Sentiment words
def clean_text(text):
    """Remove stop words and normalize text."""
    text = text.lower()
    # Remove offensive terms
    words = text.split()
    words = [w for w in words if w not in STOP_WORDS]
    words = ' '.join(words)
    # Replace gendered words with neutral ones
    for gendered, neutral in GENERED_WORDS.items():
        text = text.replace(gendered, neutral)
    return text

# NLTK sentiment
def analyze_sentiment(text):
    """Analyze sentiment returns (label, confidence)"""
    pos = len([word for word in words if word in POSITIVE])
    neg = len([word for word in words if word in NEGATIVE])
    words = set(cleaned.split())
    # Count positive and negative words
    pos = len(words & POSITIVE)
    neg = len(words & NEGATIVE)

    # Determine sentiment
    if pos > neg:
        return 'POSITIVE', pos / (pos + neg) if (pos + neg) > 0 else 0.5
    elif neg > pos:
        return 'NEGATIVE', neg / (pos + neg) if (pos + neg) > 0 else 0.5
    else:
        return 'NEUTRAL', 0.5

# Test it
if __name__ == "__main__":
    texts = [
        "This product is absolutely great!",
        "I hate this terrible thing.",
        "The professional did excellent work."
    ]

    print("SENTIMENT ANALYSIS")
    for text in texts:
        print(f"Text: {text}")
        print(f"Sentiment: {analyze_sentiment(text)}")
        print("-----")

```

OUTPUT:

```

Text: This product is absolutely great!
Sentiment: NEUTRAL (50%)

Text: I hate this terrible thing.
Sentiment: NEGATIVE (100%)

Text: The professional did excellent work.
Sentiment: POSITIVE (100%)

Text: The professional did excellent work.
Sentiment: NEGATIVE (100%)

Text: The professional did excellent work.
Sentiment: POSITIVE (100%)

Text: The professional did excellent work.
Sentiment: POSITIVE (100%)
Text: The professional did excellent work.
Sentiment: POSITIVE (100%)
Text: This product is absolutely great!
Sentiment: NEUTRAL (50%)

Text: I hate this terrible thing.
Sentiment: NEGATIVE (100%)

Text: The professional did excellent work.
Sentiment: POSITIVE (100%)

```

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 favoritism toward only popular products,

and allow users to give feedback or opt out of recommendations.

The screenshot shows a code editor interface with several files listed in the Explorer panel on the left. The main editor area contains the following Python code:

```
1  """Ultra Simple Product Recommender"""
2
3  products = {
4      'laptop': {'category': 'electronics', 'rating': 4.5},
5      'mouse': {'category': 'electronics', 'rating': 4.2},
6      'chair': {'category': 'furniture', 'rating': 4.6},
7      'book': {'category': 'books', 'rating': 4.7},
8  }
9
10 # Explain | Add Comment | X
11 def recommend(purchases):
12     """Recommend based on categories purchased"""
13     # Get user's favorite categories
14     categories = [products[p]['category'] for p in purchases]
15
16     # Find similar products not yet bought
17     recommendations = []
18     for product, info in products.items():
19         if product not in purchases and info['category'] in categories:
20             recommendations.append((product, info['rating']))
21
22     # Sort by rating
23     recommendations.sort(key=lambda x: x[1], reverse=True)
24
25     print("Recommended for you:")
26     for product, rating in recommendations[:2]:
27         print(f"    • {product} ({rating}★)")
28
29 # Test
30 purchases = ['laptop', 'mouse']
31 recommend(purchases)
```

The right side of the interface includes a Chat window with a message "GIVE SIMPLE THIS", a status bar indicating "1 file changed +341-1", and a bottom bar with various icons.

OUTPUT:

The screenshot shows a VS Code interface. In the Explorer sidebar, there's a folder named 'CP LAB ASS' containing several files: LAB ASS, CP ASS-3.py, ethical_product_re..., sentiment_analysis_with..., simple_recommend..., simple_sentiment_analys..., ultra_simple_recommend..., user_data_collection.py, and user_data_protection.py. The 'ultra_simple_recommend...' file is open in the editor, displaying code for an 'Ultra Simple Product Recommender'. The terminal tab shows command-line output related to recommendations being disabled. The Chat sidebar on the right contains a task description for generating a Python program that recommends products based on purchase history, along with guidelines for ethical AI practices like transparency and fairness.

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.

OUTPUT:

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

- File Explorer:** Shows files like `ASS_5.4.py`, `sentiment_analysis_with_bias_mitigation.py`, `simple_secure_logging.py` (the active file), `user_data_collection.py`, and `user_d...`.
- Code Editor:** Displays the `simple_secure_logging.py` file with the following content:

```
#!/usr/bin/env python
# %%Simple Secure Logging - No Sensitive Data%%

# PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

```
x-agent/python.exe "C:/Users/chunc_yhjtd63/OneDrive/Documents/CP LAB ASS/simple_secure_logging.py"
Testing 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_yhjtd63\OneDrive\Documents\CP LAB ASS>
```
- Terminal:** Shows the command `x-agent/python.exe "C:/Users/chunc_yhjtd63/OneDrive/Documents/CP LAB ASS/simple_secure_logging.py"` and its output.
- Status Bar:** Shows the path `C:\Users\chunc_yhjtd63\OneDrive\Documents\CP LAB ASS`.
- Right Sidebar:** Contains a tip about generating logging functionality for a Python web application, ensuring logs do not store sensitive information like passwords, emails, or personal identifiers, and adding comments explaining ethical logging practices.

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.

OUTPUT:

