

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.

```
File Edit Selection View ... Q CP LAB ASS
user_data_collection.py user_data_protection.py
user_data_protection.py
1 import sys
2 import hashlib
3 import getpass
4 from datetime import datetime
5
6 # User Data Collection Script with Data Protection and Anonymization
7 # This script demonstrates how to collect user data while implementing security best practices.
8
9 # ===== DATA VALIDATION FUNCTIONS =====
10
11 # [Feature: Add Comment]
12 def validate_email(email):
13     """
14     Validate email format using regex pattern.
15     Protection Tip: Always validate input to prevent injection attacks and ensure data quality.
16     """
17     pattern = r"^[a-zA-Z0-9_-]+@[a-zA-Z0-9_-]+(\.[a-zA-Z0-9_-]+)?$"
18     return re.match(pattern, email) is not None
19
20 # [Feature: Add Comment]
21 def validate_age(age):
22     """
23     Validate age is a reasonable integer value.
24     Protection Tip: Validate and sanitize all user inputs before storing.
25     """
26     try:
27         age_int = int(age)
28         return 0 < age_int < 150
29     except ValueError:
30         return False
31
32 # [Feature: Add Comment]
33 def validate_name(name):
34     """
35     Validate name contains only alphabetic characters and spaces.
36     Protection Tip: Restrict input to expected formats to prevent malicious data.
37     """
38     return bool(re.match("^[a-zA-Z ]+$", name))
39
40 # ===== DATA ANONYMIZATION & PROTECTION FUNCTIONS =====
41
42 # [Feature: Add Comment]
43 def hash_email(email):
44     """
45     Hash email address for display purposes.
46     Protection Tip: Hashing sensitive data in logs/displays prevents exposure.
47     Example: user@example.com -> "e783c75ad6e2f8b2222777123e4a51f0"
48     """
49     parts = email.split('@')
50     hashed_local = parts[0] + " " * (len(parts[0]) - 2) + parts[1] + "@"
51     return "hashed_" + hashed_local
52
53 # [Feature: Add Comment]
54 def hash_name(name):
55     """
56     Hash user name for display purposes.
57     """
58     return hashlib.sha256(name.encode()).hexdigest()
59
60 # ===== MAIN COLLECTION SCRIPT =====
61
62 def main():
63     """Main function to run the secure user data collection script."""
64     # Collect user data with validation
65     user_data = collect_user_data_secure()
66
67     # Display different representations of the data
68     display_original_data(user_data)
69     display_anonymized_data(user_data)
70     display_hashed_data(user_data)
71
62     # Print recommendations
63     print_protection_recommendations()
64
65     print("\n" + "="*70)
66     print("Data collection completed successfully!")
67     print("\n" + "="*70)
68
69 if __name__ == "__main__":
70     main()
```

```
File Edit Selection View ... Q CP LAB ASS
user_data_collection.py user_data_protection.py
user_data_protection.py
210 def print_protection_recommendations():
211     """
212     Print security recommendations for non-password fields.
213     """
214     # 1. MASKING FOR DISPLAY
215     # - Mask sensitive fields in logs and UI displays
216     # - Only show last 4 digits of sensitive identifiers
217
218     # 2. DATA MINIMIZATION
219     # - Only collect data you actually need
220     # - Delete data when it's no longer needed (retention policies)
221
222     # 3. ACCESS CONTROL
223     # - Limit who can access user data
224     # - Implement role-based access control (RBAC)
225
226     # 4. ANONYMIZATION TECHNIQUES
227     # - Use age ranges instead of exact age
228     # - Use ID numbers instead of names
229     # - Remove identifying information when possible
230
231     # 5. REGULAR AUDITS
232     # - Audit who accesses user data and when
233     # - Monitor for unauthorized access attempts
234
235     # 6. COMPLIANCE
236     # - Follow GDPR, CCPA, or other data protection regulations
237     # - Provide users ability to access and delete their data
238
239     # 7. SECURE CODING
240     # - Validate all inputs to prevent injection attacks
241     # - Use parameterized queries to prevent SQL injection
242     # - Avoid storing sensitive data in plain text in code
243
244     for rec in recommendations:
245         print(rec)
246
247 # ===== MAIN COLLECTION SCRIPT =====
248
249 # [Feature: Add Comment]
250 def main():
251     """Main function to run the secure user data collection script."""
252     # Collect user data with validation
253     user_data = collect_user_data_secure()
254
255     # Display different representations of the data
256     display_original_data(user_data)
257     display_anonymized_data(user_data)
258     display_hashed_data(user_data)
259
260     # Print recommendations
261     print_protection_recommendations()
262
263     print("\n" + "="*70)
264     print("Data collection completed successfully!")
265     print("\n" + "="*70)
266
267 if __name__ == "__main__":
268     main()
```

OUTPUT:

```
File Edit Selection View Go Run ... CP LAB ASS
EXPLORER
  CP LAB ASS
    CP ASS-3.py
    user_data_collection.py
    user_data_protection.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\chunc_yhjt63\OneDrive\Documents\CP LAB ASS> & C:/Users/chunc_yhjt63
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*****3@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*****3@gmail.com

-----
HASHED DATA (For database storage and duplicate detection)
-----
Email Hash (SHA-256): 61e434e2a4f8ba288f5fbd1d980dec6b...
Email Hash (Salted): 0628d7f938e1088443b99fbf7fbfb7ea...
Full hash stored in database, not visible to users

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

CHAT
USER DATA COLLECTION AND ANONY...
age to ranges (25 -> "20-29")
  ○ get_age_category() - Groups ages into demographic categories
  ○ generate_user_id() - Creates pseudonymous IDs from names
3. Storage Options Demonstration - Shows 4 approaches:
  ○ ✗ Plain text (insecure)
  ○ ✓ Hashed data (secure)
  ○ ✓ Anonymized data (moderate security)
  ○ ✓ Encrypted storage (highly secure)
4. 10 Data Protection Best Practices with explanations covering HTTPS, encryption, access control, compliance (GDPR/CCPA), and auditing
Every function and section includes inline comments explaining why the protection is needed and how it works. The script is saved at user_data_collection.py
+ user_data_protection.py
Describe what to build next
Agent Auto
```

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.

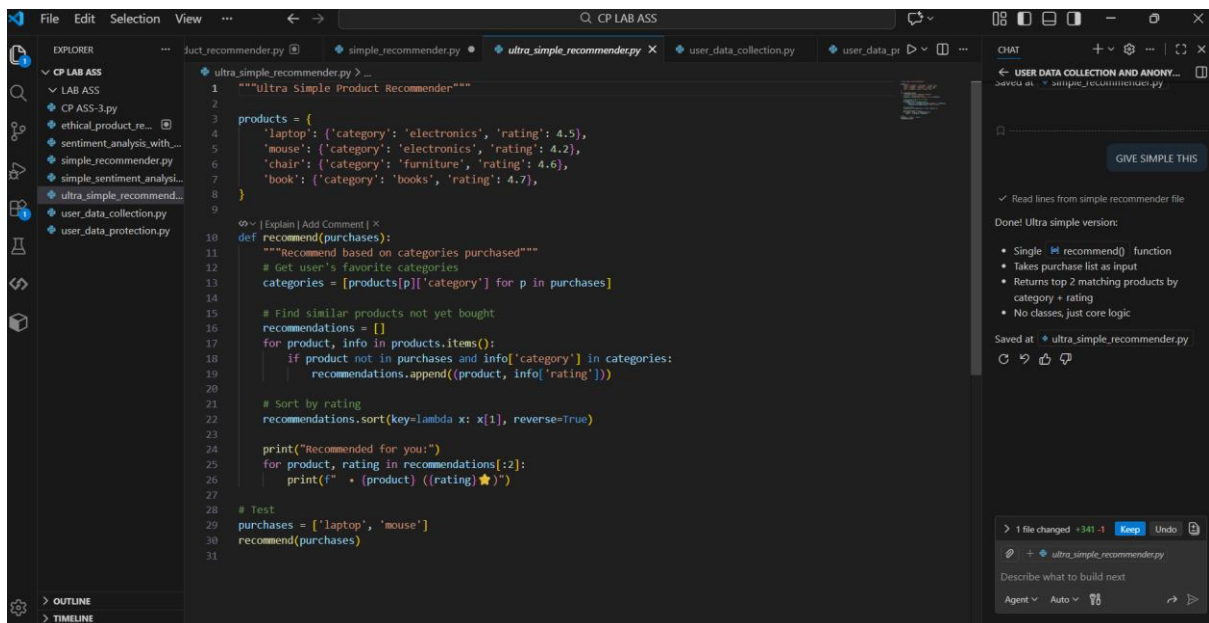
PROMPT: # Generate a Python program that recommends products based on user purchase history.

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.



```
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 def recommend(purchases):
11     """Recommend based on categories purchased"""
12     # Get user's favorite categories
13     categories = [products[p]['category'] for p in purchases]
14
15     # Find similar products not yet bought
16     recommendations = []
17     for product, info in products.items():
18         if product not in purchases and info['category'] in categories:
19             recommendations.append((product, info['rating']))
20
21     # Sort by rating
22     recommendations.sort(key=lambda x: x[1], reverse=True)
23
24     print("Recommended for you:")
25     for product, rating in recommendations[:2]:
26         print(f"  * {product} ((rating)★)")
27
28 # Test
29 purchases = ['laptop', 'mouse']
30 recommend(purchases)
```

CHAT

USER DATA COLLECTION AND ANONY...

GIVE SIMPLE THIS

✓ Read lines from simple recommender file

Done! Ultra simple version:

- Single `recommend()` function
- Takes purchase list as input
- Returns top 2 matching products by category + rating
- No classes, just core logic

Saved at `ultra_simple_recommender.py`

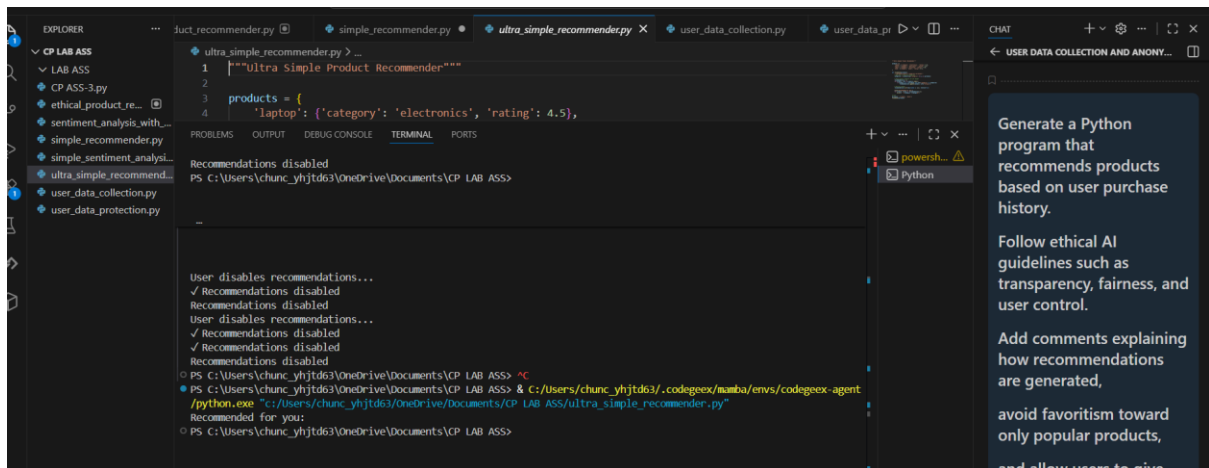
> 1 file changed +341 -1 Keep Undo

+ ultra_simple_recommender.py

Describe what to build next

Agent Auto

OUTPUT:



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

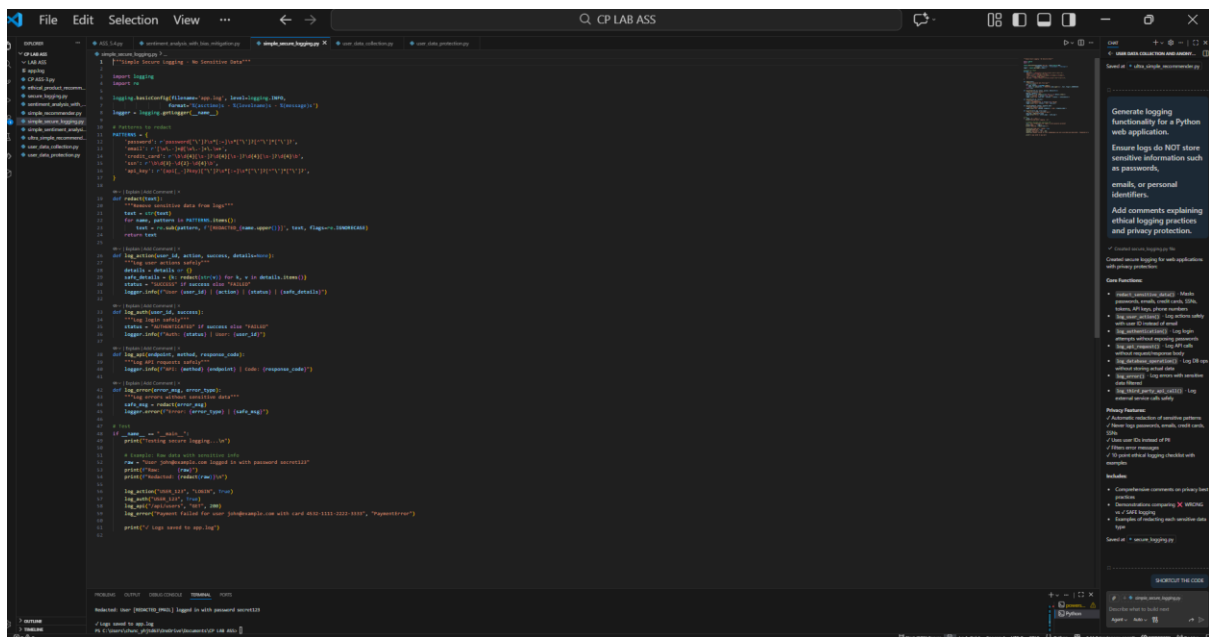
information.

PROMPT: # Generate logging functionality for a Python web application.

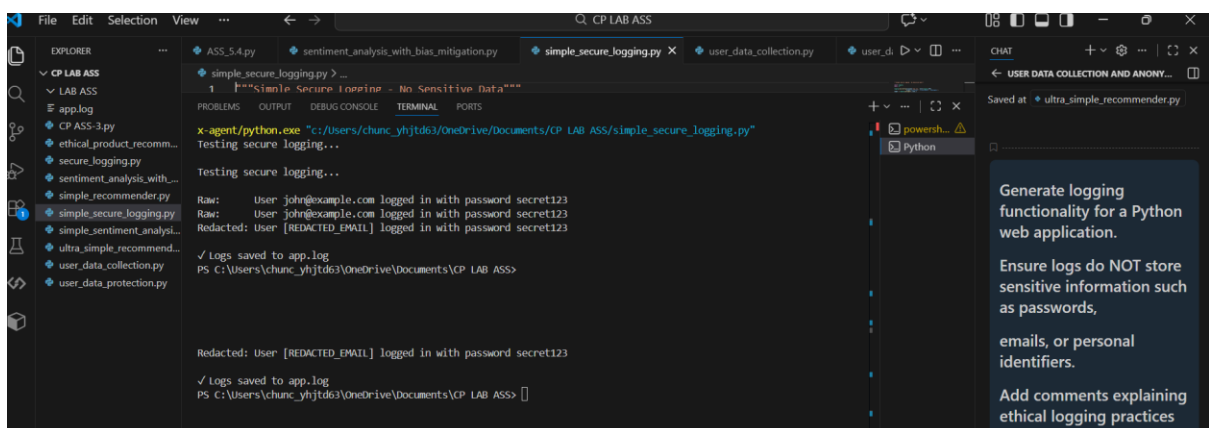
Ensure logs do NOT store sensitive information such as passwords,

emails, or personal identifiers.

Add comments explaining ethical logging practices and privacy protection.



OUTPUT:



Task Description #5:

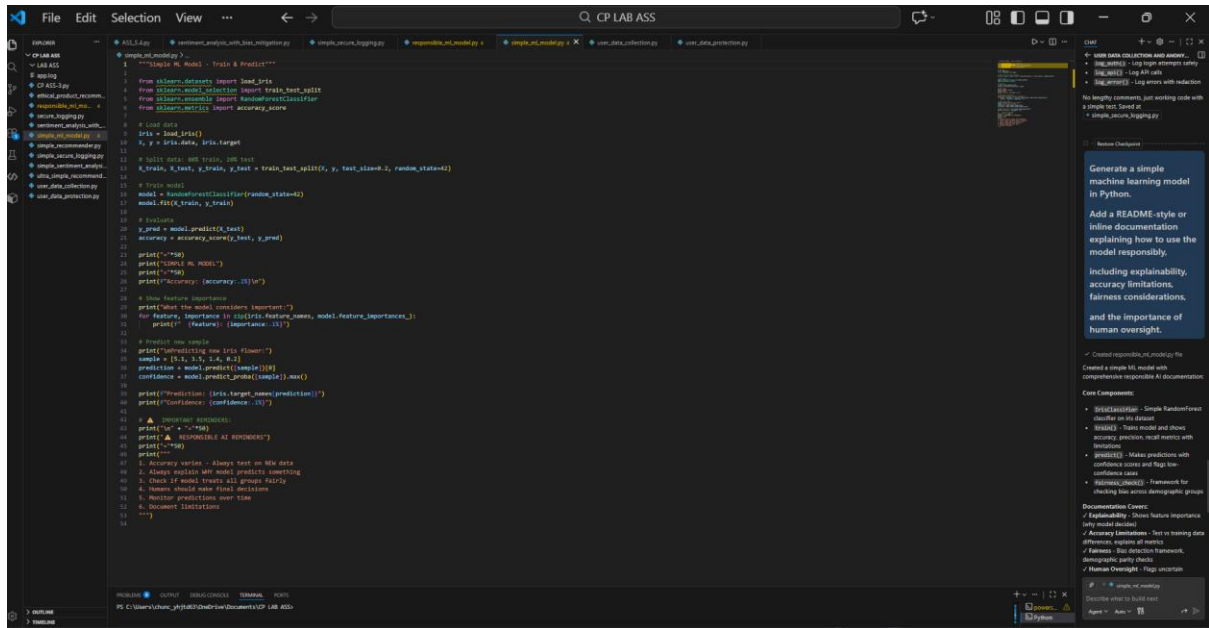
- Ask Copilot to generate a machine learning model. Then, prompt

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:

