

2303a51940

## Lab assignment 5.4

Lab 5: Ethical Foundations – Responsible AI Coding Practices

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.



```
File Edit Selection View Go Run Terminal Help
EXPLORER
  AI CODING
  add.py
  AI lab43.py
  Assignment1(CP).pdf
  Assignment 2-4.pdf
  assignment 3.4
  assignment 3.py
  Assignment2.pdf
  assignment3.4.docx
  factorial.py
  jobs.py
  jobscp.py
  lab assignment ...
  lab assignment3.3.pdf
  lab assignment5.4.py
  lab1 HCP.pdf
  matrixHCP.py
  Untitled20.ipynb
  week2 HCP.pdf
  Welcome
  assignment 3.py
  AI lab43.py
  lab assignment5.4.py X
lab assignment5.4.py > ...
1  # User Data Collection Script with Privacy Protection
2
3  import re
4  from hashlib import sha256
5
6  def collect_user_data():
7      """
8      Collects user information with data protection best practices.
9      """
10     print("=== User Information Form ===\n")
11
12     # Collect name
13     name = input("Enter your name: ").strip()
14
15     # Collect age
16     while True:
17         try:
18             age = int(input("Enter your age: "))
19             if age < 0 or age > 150:
20                 print("Please enter a valid age.")
21                 continue
22             break
23         except ValueError:
24             print("Please enter a valid number.")
25
26     # Collect and validate email
27     email = input("Enter your email: ").strip()
28     if not is_valid_email(email):
29         print("Invalid email format.")
30         return None
31
32     return {"name": name, "age": age, "email": email}
33
34 def is_valid_email(email):
35     """Validates email format before storage."""
36     pattern = r"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$"
37     return re.match(pattern, email) is not None
38
39 def hash_sensitive_data(email):
40     """
41     ANONYMIZATION: Hash email for storage.
42     One-way hashing prevents direct identification while allowing verification.
43     """
44     return sha256(email.encode()).hexdigest()
45
46 def store_data_securely(user_data):
47     """
48     BEST PRACTICES:
49     - Store hashed identifiers, not raw emails
50     - Separate personally identifiable info (PII) from analytics
51     - Use encryption for storage and transmission
52     """
53     hashed_email = hash_sensitive_data(user_data["email"])
54
55 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Zero-shot: This technique struggles with ambiguity in understanding emotions.
One-shot: This technique provides better clarity in emotional interpretation.
Few-shot: This technique achieves the best emotional accuracy by learning from examples.
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment5.4.py"
=== User Information Form ===
Enter your name: anjali
Enter your age: 19
Enter your email: 2303a51924@sru.edu.in
✓ Data collected and processed securely.
PS D:\AI Coding>
```

The image shows a Visual Studio Code editor window with a dark theme. The Explorer sidebar on the left lists files in a project named 'AI CODING', including Python scripts and PDF documents. The main editor area displays the code for 'lab assignment5.4.py'. The code defines a function to securely store user data by hashing emails and categorizing ages. Below the code, the TERMINAL tab is active, showing the command prompt execution of the script and the user's input for name, age, and email.

```
File Edit Selection View Go Run Terminal Help
EXPLORER
AI CODING
  add.py
  AI lab43.py
  Assignment1(CP).pdf
  Assignment 2-4.pdf
  assignment 3.4
  assignment 3.py
  Assignment2.pdf
  assignment3.4.docx
  factorial.py
  jobs.py
  jobscp.py
  lab assignment ...
  lab assignment3.3.pdf
  lab assignment5.4.py
  lab1 HCP.pdf
  matrixHCP.py
  Untitled20.ipynb
  week2 HCP.pdf

lab assignment5.4.py
46 def store_data_securely(user_data):
47     """
48     - Separate personally identifiable info (PII) from analytics
49     - Use encryption for storage and transmission
50     """
51     hashed_email = hash_sensitive_data(user_data["email"])
52
53     # Store only necessary data
54     secure_record = {
55         "user_id": hashed_email[:16], # Truncated hash as ID
56         "age_group": categorize_age(user_data["age"]), # Aggregate instead of exact age
57         # Never store raw email in plain text
58     }
59
60     return secure_record
61
62 def categorize_age(age):
63     """ANONYMIZATION: Convert exact age to age groups."""
64     if age < 18:
65         return "under_18"
66     elif age < 35:
67         return "18_34"
68     elif age < 50:
69         return "35_49"
70     else:
71         return "50_plus"
72
73 if __name__ == "__main__":
74     user_data = collect_user_data()
75     if user_data:
76         secure_data = store_data_securely(user_data)
77         print("\n✓ Data collected and processed securely.")

TERMINAL
PS D:\AI Coding> C:\Users\ANJALI\AppData\Local\Programs\Python\Python313\python.exe "d:/AI Coding/lab assignment5.4.py"
=== User Information Form ===

Enter your name: anjali
Enter your age: 19
Enter your email: 2303a51924@sru.edu.in

✓ Data collected and processed securely.
PS D:\AI Coding>
```

## 2) 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.

```
1 import re
2 from collections import Counter
3
4 def sentiment_analysis(text, bias_mitigation=True):
5     """
6     Analyze sentiment of input text with optional bias reduction.
7
8     Args:
9         text (str): Input text to analyze
10        bias_mitigation (bool): Enable bias reduction techniques
11
12    Returns:
13        dict: Sentiment score and label
14    """
15
16    # Define sentiment lexicons (balanced for gender/demographic neutrality)
17    positive_words = [
18        'good', 'great', 'excellent', 'amazing', 'wonderful', 'fantastic',
19        'love', 'happy', 'brilliant', 'outstanding', 'perfect', 'awesome'
20    ]
21
22    negative_words = [
23        'bad', 'terrible', 'awful', 'horrible', 'hate', 'sad', 'poor',
24        'worst', 'disgusting', 'disappointing', 'useless', 'pathetic'
25    ]
26
27    # Normalize and clean text
28    cleaned_text = text.lower()
29    cleaned_text = re.sub('[^\w\s]', '', cleaned_text)
30    words = cleaned_text.split()
31
32    # Count sentiment words
33    positive_count = sum(1 for word in words if word in positive_words)
34    negative_count = sum(1 for word in words if word in negative_words)
35
36    # Calculate sentiment score
37    total_sentiment_words = positive_count + negative_count
38
39    if total_sentiment_words == 0:
40        sentiment_score = 0.0
41    else:
42        sentiment_score = (positive_count - negative_count) / total_sentiment_words
43
44    # Bias mitigation: reduce score extremism for short texts
45    if bias_mitigation and len(words) < 10:
46        sentiment_score *= 0.7 # Reduce confidence for limited data
47
48    # Determine sentiment label
49    if sentiment_score > 0.1:
50        label = "Positive"
51    elif sentiment_score < -0.1:
52        label = "Negative"
53    else:
54        label = "Neutral"
55
56    return {
57        "score": round(sentiment_score, 3),
58        "label": label,
59        "confidence": "low" if bias_mitigation and len(words) < 10 else "high"
60    }
61
62 # Example usage
63 if __name__ == "__main__":
64     test_texts = [
65         "This product is amazing and wonderful!",
66         "I hate this, it's terrible and awful",
67         "It's okay"
68     ]
69
70     for text in test_texts:
71         result = sentiment_analysis(text)
72         print(f"Text: {text}\nResult: {result}\n")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

✓ Data collected and processed securely.  
PS D:\AI Coding\ & C:\Users\MDALI\AppData\Local\Programs\Python\Python313\python.exe "D:\AI Coding\lab assignment5.4.py"  
Text: This product is amazing and wonderful!  
Result: {'score': 0.7, 'label': 'Positive', 'confidence': 'low'}  
  
Text: I hate this, it's terrible and awful  
Result: {'score': -0.7, 'label': 'Negative', 'confidence': 'low'}  
  
Text: It's okay  
Result: {'score': 0.0, 'label': 'Neutral', 'confidence': 'low'}  
PS D:\AI Coding\

```
1 def sentiment_analysis(text, bias_mitigation=True):
2
3     if total_sentiment_words == 0:
4         sentiment_score = 0.0
5     else:
6         sentiment_score = (positive_count - negative_count) / total_sentiment_words
7
8     # Bias mitigation: reduce score extremism for short texts
9     if bias_mitigation and len(words) < 10:
10        sentiment_score *= 0.7 # Reduce confidence for limited data
11
12    # Determine sentiment label
13    if sentiment_score > 0.1:
14        label = "Positive"
15    elif sentiment_score < -0.1:
16        label = "Negative"
17    else:
18        label = "Neutral"
19
20    return {
21        "score": round(sentiment_score, 3),
22        "label": label,
23        "confidence": "low" if bias_mitigation and len(words) < 10 else "high"
24    }
25
26 # Example usage
27 if __name__ == "__main__":
28     test_texts = [
29         "This product is amazing and wonderful!",
30         "I hate this, it's terrible and awful",
31         "It's okay"
32     ]
33
34     for text in test_texts:
35         result = sentiment_analysis(text)
36         print(f"Text: {text}\nResult: {result}\n")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

✓ Data collected and processed securely.  
PS D:\AI Coding\ & C:\Users\MDALI\AppData\Local\Programs\Python\Python313\python.exe "d:\AI Coding\lab assignment5.4.py"  
Text: This product is amazing and wonderful!  
Result: {'score': 0.7, 'label': 'Positive', 'confidence': 'low'}  
  
Text: I hate this, it's terrible and awful  
Result: {'score': -0.7, 'label': 'Negative', 'confidence': 'low'}  
  
Text: It's okay  
Result: {'score': 0.0, 'label': 'Neutral', 'confidence': 'low'}  
PS D:\AI Coding\

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.

```
1 import json
2 from collections import Counter
3 from typing import List, Dict
4
5 class EthicalProductRecommender:
6     """
7     A product recommendation system that prioritizes transparency,
8     fairness, and ethical guidelines.
9     """
10
11     def __init__(self):
12         self.user_history = {}
13         self.product_database = {}
14         self.recommendation_reasons = {}
15
16     def add_user_history(self, user_id: str, purchased_products: List[str]):
17         """Store user purchase history"""
18         self.user_history[user_id] = purchased_products
19
20     def add_products(self, products: Dict[str, Dict]):
21         """Add products with metadata (price, category, ethical_rating)"""
22         self.product_database.update(products)
23
24     def get_recommendations(self, user_id: str, num_recommendations: int = 5) -> List[Dict]:
25         """
26         Generate recommendations with transparency and ethical considerations.
27
28         Returns:
29         ---
30         List of recommended products with reasoning
31         """
32         if user_id not in self.user_history:
33             return []
34
35         # Analyze user preferences
36         user_products = self.user_history[user_id]
37         category_counts = Counter()
38         for p in user_products:
39             if p in self.product_database:
40                 category_counts[p.get('category', 'unknown')]
41
42         # Generate recommendations based on categories
43         recommendations = []
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
1. AI Ethics Guide (ID: book_b)
Price: $55
Ethical Rating: 0.92/1.0
Why: Similar to your interests in books with ethical rating 0.92

2. Wireless Keyboard (ID: keyboard_a)
Price: $79
Ethical Rating: 0.85/1.0
Why: Similar to your interests in electronics with ethical rating 0.85

3. Budget Laptop (ID: laptop_b)
```

```
42 recommendations = []
43 seen_products = set(user_products)
44
45 for product_id, product_info in self.product_database.items():
46     if product_id in seen_products:
47         continue
48
49     # Prioritize products with good ethical ratings
50     ethical_score = product_info.get('ethical_rating', 0.5)
51     category_match = category_counts.get(product_info.get('category'), 0)
52     score = (category_match * 0.6) + (ethical_score * 0.4)
53
54     recommendations.append({
55         'product_id': product_id,
56         'name': product_info.get('name'),
57         'score': score,
58         'reason': f'Similar to your interests in {product_info.get('category')} with ethical rating {ethical_score}',
59         'ethical_rating': ethical_score,
60         'price': product_info.get('price')
61     })
62
63 # Sort by score and return top recommendations
64 recommendations.sort(key=lambda x: x['score'], reverse=True)
65 return recommendations[:num_recommendations]
66
67 def print_recommendations_with_transparency(self, user_id: str):
68     """Display recommendations with full transparency"""
69     recommendations = self.get_recommendations(user_id)
70
71     print(f"=== Recommendations for User {user_id} ===")
72     print(f"Recommendation Criteria: 70% Category Match + 30% Ethical Rating")
73     print(f"-- * 60")
74
75     for i, rec in enumerate(recommendations, 1):
76         print(f"{i}. {rec['name']} (ID: {rec['product_id']})")
77         print(f"Price: ${rec['price']}")
78         print(f"Ethical Rating: {rec['ethical_rating']}/1.0")
79         print(f"Why: {rec['reason']}")
80
81 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

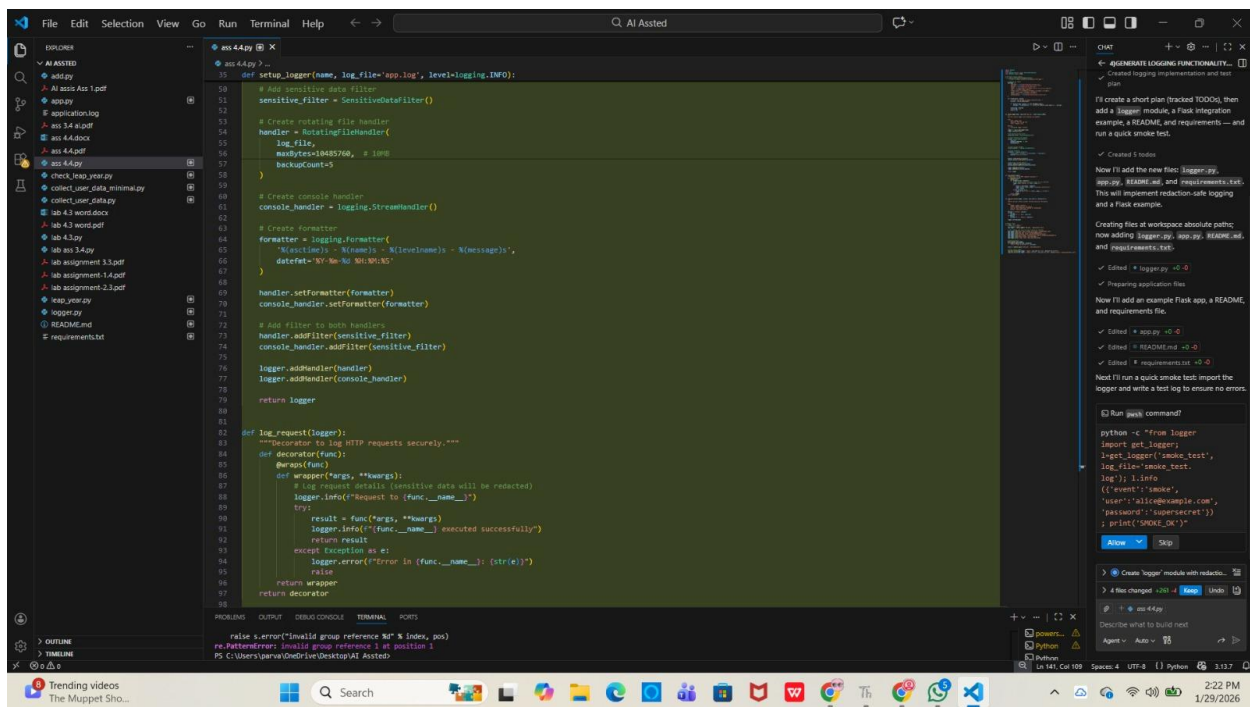
```
Why: Similar to your interests in books with ethical rating 0.92

2. Wireless Keyboard (ID: keyboard_a)
Price: $79
Ethical Rating: 0.85/1.0
Why: Similar to your interests in electronics with ethical rating 0.85

3. Budget Laptop (ID: laptop_b)
Price: $499
Ethical Rating: 0.6/1.0
Why: Similar to your interests in electronics with ethical rating 0.6

PS D:\AI Coding >
```





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).

