

2303a51504

## 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

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
56
57 Zero-shot: This technique struggles with ambiguity in understanding emotions.
58 One-shot: This technique provides better clarity in emotional interpretation.
59 Few-shot: This technique achieves the best emotional accuracy by learning from examples.
60 PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment5.4.py"
61 === User Information Form ===
62
63 Enter your name: anjali
64 Enter your age: 19
65 Enter your email: 2303a51924@sru.edu.in
66
67 ✓ Data collected and processed securely.
68 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 under 'AI CODING', including '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' (selected), 'lab1 HCP.pdf', 'matrixHCP.py', 'Untitled20.ipynb', and 'week2 HCP.pdf'. The main editor displays the code for 'lab assignment5.4.py'. The code defines a function 'store\_data\_securely' to securely store user data by hashing sensitive information and categorizing age. It also includes a main block that collects user data and prints a confirmation message. The bottom panel shows the 'TERMINAL' tab with the command prompt 'PS D:\AI Coding> C:\Users\ANJALI\AppData\Local\Programs\Python\Python313\python.exe "d:\AI Coding\lab assignment5.4.py"' and the program's output, which prompts the user for their name, age, and email, and then prints a confirmation message.

```
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):
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     # Store only necessary data
56     secure_record = {
57         "user_id": hashed_email[:16], # Truncated hash as ID
58         "age_group": categorize_age(user_data["age"]), # Aggregate instead of exact age
59         # Never store raw email in plain text
60     }
61
62     return secure_record
63
64 def categorize_age(age):
65     """ANONYMIZATION: Convert exact age to age groups."""
66     if age < 18:
67         return "under_18"
68     elif age < 35:
69         return "18_34"
70     elif age < 50:
71         return "35_49"
72     else:
73         return "50_plus"
74
75 if __name__ == "__main__":
76     user_data = collect_user_data()
77     if user_data:
78         secure_data = store_data_securely(user_data)
79         print("\n✓ Data collected and processed securely.")

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>
```

## 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.

The screenshot shows a VS Code editor with a Python file named `lab assignment5.4.py`. The script defines a `sentiment_analysis` function that takes `text` and `bias_mitigation` as arguments. It uses a `Counter` to count words and a list of `positive_words` and `negative_words` to calculate a sentiment score. The score is then normalized and rounded. The script also includes a `main` function that tests the `sentiment_analysis` function with various inputs.

```
1 import sys
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 = total_sentiment_words / 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")
```

The screenshot shows a VS Code editor with a Python file named `lab assignment5.4.py`. The script defines a `sentiment_analysis` function that takes `text` and `bias_mitigation` as arguments. It uses a `Counter` to count words and a list of `positive_words` and `negative_words` to calculate a sentiment score. The score is then normalized and rounded. The script also includes a `main` function that tests the `sentiment_analysis` function with various inputs.

```
1 import sys
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 = total_sentiment_words / 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")
```

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
64 # Sort by score and return top recommendations
65 recommendations.sort(key=lambda x: x['score'], reverse=True)
66 return recommendations[:num_recommendations]
67
68 def print_recommendations_with_transparency(self, user_id: str):
69     """Display recommendations with full transparency"""
70     recommendations = self.get_recommendations(user_id)
71
72     print(f"===== Recommendations for User {user_id} =====")
73     print(f"Recommendation Criteria: 70% Category Match + 30% Ethical Rating")
74     print(f"--- * 60")
75
76     for i, rec in enumerate(recommendations, 1):
77         print(f"{i}. {rec['name']} (ID: {rec['product_id']})")
78         print(f"Price: ${rec['price']}")
79         print(f"Ethical Rating: {rec['ethical_rating']}/1.0")
80         print(f"Why: {rec['reason']}")
```

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 >
```

The screenshot shows a VS Code editor with a Python file named `lab_assignment5.py`. The script defines a class `EthicalProductRecommender` with methods for printing recommendations with transparency. It includes sample products like 'laptop\_a', 'laptop\_b', 'keyboard\_a', 'book\_a', and 'book\_b' with their respective prices and ethical ratings. The script also includes a main function that initializes the recommender, adds products, adds user history for 'user\_001', and prints recommendations with transparency for 'user\_001'.

```
1 class EthicalProductRecommender:
2     def print_recommendations_with_transparency(self, user_id: str):
3
4         print(f"--- Recommendations for user {user_id} ---")
5         print(f"Recommendation Criteria: 70% Category Match + 30% Ethical Rating")
6         print(f"{'-' * 60}")
7
8         for i, rec in enumerate(recommendations, 1):
9             print(f"[{i}] {rec['name']} (ID: {rec['product_id']})")
10            print(f"    Price: ${rec['price']}")
11            print(f"    Ethical Rating: {rec['ethical_rating']/1.0}")
12            print(f"    Why: {rec['reason']}")
13
14 # Example usage
15 if __name__ == "__main__":
16     recommender = EthicalProductRecommender()
17
18     # Add sample products
19     products = [
20         {'product_id': 'laptop_a', 'name': 'EcoLaptop Pro', 'category': 'electronics', 'price': 999, 'ethical_rating': 0.9},
21         {'product_id': 'laptop_b', 'name': 'Budget Laptop', 'category': 'electronics', 'price': 499, 'ethical_rating': 0.6},
22         {'product_id': 'keyboard_a', 'name': 'Wireless Keyboard', 'category': 'electronics', 'price': 79, 'ethical_rating': 0.85},
23         {'product_id': 'book_a', 'name': 'Python Programming', 'category': 'books', 'price': 49, 'ethical_rating': 0.95},
24         {'product_id': 'book_b', 'name': 'AI Ethics Guide', 'category': 'books', 'price': 35, 'ethical_rating': 0.92},
25     ]
26
27     recommender.add_products(products)
28
29     # Add user history
30     recommender.add_user_history('user_001', ['laptop_a', 'book_a'])
31
32     # Get and display recommendations
33     recommender.print_recommendations_with_transparency('user_001')
```

The output in the terminal shows the following recommendations for 'user\_001':

```
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
```

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

The screenshot shows a VS Code editor with a Python file named `app.py`. The script implements a secure logging system using the `logging` module. It defines a `filter_sensitive_data` function that filters out sensitive information from log messages. It also defines a `filter` function that uses a list of patterns to filter out sensitive data. The script then sets up a logger with a file handler and a secure logger with a rotating file handler.

```
1 import logging
2 import re
3 from logging.handlers import RotatingFileHandler
4 from functools import wraps
5
6 # Configure secure logging
7 class SensitiveDataFilter(logging.Filter):
8     def filter(self, record):
9         """Filter to remove sensitive information from logs."""
10
11         # Patterns to redact
12         PATTERNS = [
13             'password',
14             'token',
15             'email',
16             'card',
17             'phone',
18             'bearer_token',
19             'authorization'
20         ]
21
22         def filter(self, record):
23             """Filter log records to remove sensitive data."""
24             message = record.getMessage()
25
26             for pattern_name, pattern in self.PATTERNS.items():
27                 message = re.sub(pattern, f"[REDACTED-{pattern_name.upper()}]", message)
28
29             record.msg = message
30             record.args = ()
31             return True
32
33 def setup_logger(name, log_file='app.log', level=logging.INFO):
34     """
35     Set up a secure logger with rotating file handler.
36
37     Args:
38         name: Logger name
39         log_file: Path to log file
40         level: Logging level
41
42     Returns:
43         Configured logger instance
44     """
45     logger = logging.getLogger(name)
46     logger.setLevel(level)
47
48     # Add sensitive data filter
49     filter = SensitiveDataFilter()
50     logger.addFilter(filter)
```

The output in the terminal shows the following error:

```
raise ValueError("Invalid group reference '%s' at position %s")
re.PatternError: Invalid group reference 1 at position 1
```





