

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.

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The left sidebar displays a file tree under 'EXPLORER' with various files and folders, including 'AI CODING', 'Assignment1(CP).pdf', 'Assignment2-4.pdf', 'assignment 3.4', 'Assignment2.pdf', 'assignment3.4.docx', 'factorial.py', 'jobs.py', 'jobsCP.py', 'lab assignment ...', 'lab assignment33.pdf', and 'lab assignment5.4.py'. The main editor area contains a Python script named 'lab assignment5.4.py'. The script is designed to collect user information with privacy protection. It includes functions for collecting name, age, and email, validating the email format, hashing sensitive data using SHA256, and storing data securely. A detailed comment block at the bottom explains the best practices for data storage and hashing. The terminal tab at the bottom shows the command 'PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment5.4.py"' and the output of the script's execution, which prompts for user input and processes the data.

```
# User Data Collection Script with Privacy Protection
import re
from hashlib import sha256

def collect_user_data():
    """
    Collects user information with data protection best practices.
    """

    print("== User Information Form ==\n")

    # Collect name
    name = input("Enter your name: ").strip()

    # Collect age
    while True:
        try:
            age = int(input("Enter your age: "))
            if age < 0 or age > 150:
                print("Please enter a valid age.")
                continue
            break
        except ValueError:
            print("Please enter a valid number.")

    # Collect and validate email
    email = input("Enter your email: ").strip()
    if not is_valid_email(email):
        print("Invalid email format.")
        return None

    return {"name": name, "age": age, "email": email}

def is_valid_email(email):
    """
    Validates email format before storage.
    """
    pattern = r"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$"
    return re.match(pattern, email) is not None

def hash_sensitive_data(email):
    """
    ANONYMIZATION: Hash email for storage.
    One-way hashing prevents direct identification while allowing verification.
    """
    return sha256(email.encode()).hexdigest()

def store_data_securely(user_data):
    """
    BEST PRACTICES:
    - Store hashed identifiers, not raw emails
    - Separate personally identifiable info (PII) from analytics
    - Use encryption for storage and transmission
    """
    hashed_email = hash_sensitive_data(user_data["email"])

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 screenshot shows a code editor interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** Back, Forward.
- Explorer:** Shows a file tree under "AI CODING".
- Code Editor:** Displays the content of "lab assignment5.4.py".
- Bottom Bar:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS.
- Terminal:** Shows command-line output related to the Python script execution.

```
def store_data_securely(user_data):
    """ Separate personally identifiable info (PII) from analytics
    - Use encryption for storage and transmission
    """
    hashed_email = hash_sensitive_data(user_data["email"])

    # Store only necessary data
    secure_record = {
        "user_id": hashed_email[:16], # Truncated hash as ID
        "age_group": categorize_age(user_data["age"]), # Aggregate instead of exact age
        # Never store raw email in plain text
    }

    return secure_record

def categorize_age(age):
    """ANONYMIZATION: Convert exact age to age groups."""
    if age < 18:
        return "under_18"
    elif age < 35:
        return "18_34"
    elif age < 50:
        return "35_49"
    else:
        return "50_plus"

if __name__ == "__main__":
    user_data = collect_user_data()
    if user_data:
        secure_data = store_data_securely(user_data)
        print("\n Data collected and processed securely.")
```

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.

```

File Edit Selection View Go Run Terminal Help < > AI Coding
EXPLORER lab assignment54.py lab assignments54.py
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        'worse', 'disgusting', 'disappointing', 'useless', 'pathetic'
25    ]
26
27    # Normalize and Clean text
28    cleaned_text = text.lower()
29    cleaned_text = re.sub(r'[^a-zA-Z]', ' ', 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:
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
63    # Example usage
64    if __name__ == "__main__":
65        test_texts = [
66            "This product is amazing and wonderful!",
67            "I hate this, it's terrible and awful",
68            "It's okay"
69        ]
70
71        for text in test_texts:
72            result = sentiment_analysis(text)
73            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\Apdata\Local\Programs\Python\Python313\python.exe "d:/AI Coding/lab assignment54.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 []

```

File Edit Selection View Go Run Terminal Help < > AI Coding
EXPLORER lab assignment54.py lab assignments54.py
1 def sentiment_analysis(text, bias_mitigation=True):
2
3     if total_sentiment_words == 0:
4         sentiment_score = 0.0
5     else:
5         sentiment_score = (positive_count - negative_count) / total_sentiment_words
5
6     # Bias mitigation: reduce score extremism for short texts
7     if bias_mitigation and len(words) < 10:
8         sentiment_score *= 0.7 # Reduce confidence for limited data
8
9     # Determine sentiment label
10    if sentiment_score > 0:
11        label = "Positive"
12    elif sentiment_score < -0.1:
13        label = "Negative"
14    else:
15        label = "Neutral"
16
17    return {
18        "score": round(sentiment_score, 3),
19        "label": label,
20        "confidence": "low" if bias_mitigation and len(words) < 10 else "high"
21    }
22
23
24    # Example usage
25    if __name__ == "__main__":
26        test_texts = [
27            "This product is amazing and wonderful!",
28            "I hate this, it's terrible and awful",
29            "It's okay"
30        ]
31
32        for text in test_texts:
33            result = sentiment_analysis(text)
34            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\Apdata\Local\Programs\Python\Python313\python.exe "d:/AI Coding/lab assignment54.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.

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows files like `assignment 3.py`, `AI lab5.py`, and `lab assignment5.py`.
- Code Editor:** Displays the `lab assignment5.py` file, which contains Python code for an ethical product recommender system.
- Terminal:** Shows the command `AI Coding`.
- Bottom Bar:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS.
- Bottom Panel:** Shows recommendations for three products:
 - 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)

```
File Edit Selection View Go Run Terminal Help < > Q AI Coding

EXPLORER Welcome assignment3.py lab assignment54.py
AI CODING assignment54.py
add.py
AI lab43.py
Assignment 1C.pdf
Assignment 2A.pdf
Assignment 3A.pdf
Assignment3A.docx
factorial.py
jobs.py
jobs.py
lab assignment ...
lab assignment3.pdf
lab assignments...
lab1 HCP.pdf
matrixP.py
Untitled0.ipynb
weeek HCP.pdf

assignment3.py
lab assignment54.py
class EthicalProductRecommender:
    def get_recommendations(self, user_id: str, num_recommendations: int = 5) -> List[Dict]:
        recommendations = []
        seen_products = set(user_products)

        for product_id, product_info in self.product_database.items():
            if product_id in seen_products:
                continue

            # Prioritize products with good ethical ratings
            ethical_score = product_info.get('ethical_rating', 0.5)
            category_match = category_counts.get(product_info.get('Category'), 0)

            score = (category_match * 0.6) + (ethical_score * 0.4)

            recommendations.append({
                'product_id': product_id,
                'name': product_info.get('name'),
                'score': score,
                'reason': f"Similar to your interests in {product_info.get('category')} with ethical rating {ethical_score}",
                'ethical_rating': ethical_score,
                'price': product_info.get('price')
            })

        # Sort by score and return top recommendations
        recommendations.sort(key=lambda x: x['score'], reverse=True)
        return recommendations[:num_recommendations]

    def print_recommendations_with_transparency(self, user_id: str):
        """Display recommendations with full transparency"""
        recommendations = self.get_recommendations(user_id)

        print(f"\n==== Recommendations for User {user_id} ===")
        print("Recommendation Criteria: 70% Category Match + 30% Ethical Rating")
        print("-" * 40)

        for i, rec in enumerate(recommendations, 1):
            print(f"\n{i}. {rec['name']} (ID: {rec['product_id']})")
            print(f" Price: ${rec['price']}")
            print(f" Ethical Rating: {rec['ethical_rating']}/1.0")
            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: $59
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:\VAI Coding [1]
```

The screenshot shows a code editor interface with multiple tabs open. The main tab displays Python code for a 'EthicalProductRecommender' class. The code includes methods for printing recommendations with transparency and adding products to a recommender. It also demonstrates how to add user history and get recommendations for a specific user ('user_001'). The code uses print statements to output product details like name, price, and ethical rating.

```
File Edit Selection View Go Run Terminal Help ← → Q AI Coding

EXPLORER
AI CODING
+ Welcome assignment3.py AI lab3.py lab assignment5.4.py
lab assignment5.4.py ...
5     class EthicalProductRecommender:
6         def print_recommendations_with_transparency(self, user_id: str):
7             print("---- Recommendation for User (%s) ----") % user_id
8             print("Recommendation Criteria: 70% Category Match + 30% Ethical Rating")
9             print("-" * 60)
10            ...
11            ...
12            ...
13            ...
14            ...
15            ...
16            ...
17            ...
18            ...
19            ...
20            ...
21            ...
22            ...
23            ...
24            ...
25            ...
26            ...
27            ...
28            ...
29            ...
30            ...
31            ...
32            ...
33            ...
34            ...
35            ...
36            ...
37            ...
38            ...
39            ...
40            ...
41            ...
42            ...
43            ...
44            ...
45            ...
46            ...
47            ...
48            ...
49            ...
50            ...
51            ...
52            ...
53            ...
54            ...
55            ...
56            ...
57            ...
58            ...
59            ...
60            ...
61            ...
62            ...
63            ...
64            ...
65            ...
66            ...
67            ...
68            ...
69            ...
70            ...
71            ...
72            ...
73            ...
74            ...
75            ...
76            ...
77            ...
78            ...
79            ...
80            ...
81            ...
82            ...
83            ...
84            ...
85            ...
86            ...
87            ...
88            ...
89            ...
90            ...
91            ...
92            ...
93            ...
94            ...
95            ...
96            ...
97            ...
98            ...
99            ...
100           ...
101           ...
102           ...

# Example usage
if __name__ == "__main__":
    recommender = EthicalProductRecommender()

    # Add sample products
    products = [
        {'name': 'Ecological Pro', 'category': 'electronics', 'price': 999, 'ethical_rating': 0.0},
        'laptop_a': {'name': 'Budget Laptop', 'category': 'electronics', 'price': 499, 'ethical_rating': 0.6},
        'keyboard_a': {'name': 'Wireless Keyboard', 'category': 'electronics', 'price': 79, 'ethical_rating': 0.85},
        'book_a': {'name': 'Python Programming', 'category': 'books', 'price': 45, 'ethical_rating': 0.95},
        'book_b': {'name': 'AI Ethics Guide', 'category': 'BOOKS', 'price': 55, 'ethical_rating': 0.92},
        ...
    ]

    recommender.add_products(products)

    # Add user history
    recommender.add_user_history('user_001', ['laptop_a', 'book_a'])

    # Get and display recommendations
    recommender.print_recommendations_with_transparency('user_001')

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_a)
Price: $499
Ethical Rating: 0.6/1.0
Why: Similar to your interests in electronics with ethical rating 0.6
PS D:\AI Coding
```

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 Microsoft Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows files like `ass 4.4.pdf`, `ass 4.4.py`, `check_lead_year.py`, `collect_user_data_minmax.py`, `collect_user_data.py`, `lab 4.3.py`, `lab 4.3.pdf`, `lab 4.3x.py`, `lab assignment 3.4.pdf`, `lab assignment 3.4x.py`, `lab assignment 1.4.pdf`, `lab assignment 1.4x.py`, `lab assignment 2.4.pdf`, `lab assignment 2.4x.py`, `log.py`, and `README.md`.
- Code Editor:** The main editor window displays Python code for logging. It includes imports for `logging`, `re`, and `logging.handlers`, along with a `RotatingFileHandler` from `functools`. A class `SensitiveDataFilter` is defined to filter sensitive information from logs. The code uses regular expressions (`re`) to match and replace patterns in log messages.
- Terminal:** The bottom terminal shows command-line output related to file paths and error messages.
- GitHub Integration:** A sidebar on the right shows a GitHub pull request titled "Add secure logging functionality". The PR summary states: "This adds secure logging functionality to the application. It ensures that log data does not store sensitive or personal information." The PR has 4 reviews, 0 comments, and 1 merge commit. It was created by "tjw" and updated 1 day ago.
- Bottom Status Bar:** Shows the current file is `ass 4.4.py`, the line number is 141, the column is 109, and the status bar also includes icons for Python, Git, and other extensions.

```

    # Add sensitive data filter
    sensitive_filter = SensitiveDataFilter()

    # Create rotating file handler
    handler = RotatingFileHandler(
        log_file,
        maxBytes=MAXBYTES,
        backupCount=5
    )

    # Create console handler
    console_handler = Logging.StreamHandler()

    # Create formatter
    formatter = Logging.Formatter(
        '[%(asctime)s] %(name)s - %(message)s',
        datefmt='%Y-%m-%d %H:%M:%S'
    )

    handler.setFormatter(formatter)
    console_handler.setFormatter(formatter)

    # Add filter to both handlers
    handler.addFilter(sensitive_filter)
    console_handler.addFilter(sensitive_filter)

    logger.addHandler(handler)
    logger.addHandler(console_handler)

    return logger
}

def log_request(logger):
    """Decorator to log HTTP requests securely."""
    def decorator(func):
        @wraps(func)
        def wrapper(*args, **kwargs):
            # Log request details (sensitive data will be redacted)
            logger.info("Request to (%s)", func.__name__)
            try:
                result = func(*args, **kwargs)
                logger.info("%s executed successfully", func.__name__)
                return result
            except Exception as e:
                logger.error("%s: %s", func.__name__, str(e))
            raise
        return wrapper
    return decorator
}

raise RuntimeError("Invalid group reference %r % index, pos")
re.PatternError: Invalid group reference 1 at position 1
PS C:\Users\parva\Desktop\AI Assisted>

```

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).

```

    from sklearn.datasets import load_iris
    from sklearn.model_selection import train_test_split
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
    import numpy as np

    class IrisClassifier:
        """
        A simple k-Nearest Neighbors classifier for the Iris dataset.
        This class encapsulates the training and prediction logic for classifying
        iris flowers into one of three species based on their measurements.
        """

        def __init__(self, n_neighbors=3):
            """
            Initialize the classifier.

            Args:
                n_neighbors (int): Number of neighbors to consider for prediction. Default is 3.
            """
            self.n_neighbors = n_neighbors
            self.model = KNeighborsClassifier(n_neighbors=n_neighbors)
            self.X_train = None
            self.X_test = None
            self.y_train = None
            self.y_test = None
            self.iris_data = None
            self.target_names = None

        def load_data(self, test_size=0.2, random_state=42):
            """
            Load and split the Iris dataset.

            Args:
                test_size (float): Proportion of data to use for testing. Default is 0.2 (20%).
                random_state (int): Seed for reproducibility. Default is 42.
            """
            self.iris_data = load_iris()
            X = self.iris_data.data
            y = self.iris_data.target
            self.target_names = self.iris_data.target_names

            self.X_train, self.X_test, self.y_train, self.y_test = train_test_split(
                X, y, test_size=test_size, random_state=random_state
            )
            print(f"Data loaded: {len(self.X_train)} training samples, {len(self.X_test)} test samples")

        def train(self):
            ...

    raise RuntimeError("Invalid group reference %r % index, pos")
re.PatternError: Invalid group reference 1 at position 1
PS C:\Users\parva\Desktop\AI Assisted>

```

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows files in the workspace, including Python scripts like `ass_4.py`, `ass_4.ipynb`, and `ass_4.pdf`, as well as configuration files like `requirements.txt`.
- Code Editor:** Displays Python code for a `IrisClassifier` class. The code includes methods for training a KNN model, evaluating it, and predicting flower species based on features.
- Status Bar:** Shows the current file path (`C:\Users\sravanrao\Desktop\AI Assisted`), the number of changes (4), and the Python version (Python 3.8.5).
- Bottom Taskbar:** Includes icons for File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar.
- Floating Command Palette:** Opened on the right, it lists several options:
 - GENERATE LOGGING FUNCTIONALITY... (with checkboxes for 'Add standard logging implementation and test plan' and 'I'll add the new files: logger.py, app.py, README.md, and requirements.txt')
 - Created 7 todos
 - Now I'll add the new files: logger.py, app.py, README.md, and requirements.txt. This will implement redaction-safe logging and a Flask example.
 - Creating files at workspace absolute paths; now adding logger.py, app.py, README.md, and requirements.txt...
 - Edited 1 logger.py ← 0
 - Preparing application files
 - Now I'll add an example flask app, a README, and requirements file.
 - Edited 1 app.py ← 0
 - Edited 1 README.md ← 0
 - Edited 1 requirements.txt ← 0
- Bottom Right:** Shows a terminal window with the command `python -c "from logger import get_logger; logger.info('smoke test', log_file='smoke_test.log'); logger.info('({event}:{smoke}, {user}:{alice@example.com}, {password}:{supersecret})', event='smoke', user='alice@example.com', password='supersecret')"; print(SMOKED_SK)"` and its output.