

**AI Assisted Coding**

**Lab\_Assignment\_5.4**

**A.Vasantha Shoba Rani**

**2303A51395**

**Batch – 06**

### **Task 1: Secure User Data Collection**

#### **Prompt to Copilot**

**Generate a Python script to collect user name, age, and email. Add comments explaining how to anonymize or protect user data.**

---

#### **Python Code**

```
import hashlib

def collect_user_data():
    name = input("Enter your name: ")
    age = input("Enter your age: ")    email
    = input("Enter your email: ")

    # Hashing email to protect sensitive information
    hashed_email = hashlib.sha256(email.encode()).hexdigest()

    user_data = {
        "name": name,
        "age": age,
        "email_hash": hashed_email # Store hashed email, not plain email
    }

    return user_data
```

```
data = collect_user_data()  
print("User data stored securely.")
```

---

## Code Explanation

- The program collects basic user details.
  - `hashlib.sha256()` is used to hash the email for privacy.
  - Hashed data cannot be easily reversed, protecting user identity.
  - Sensitive information is not stored in plain text.
  - This follows ethical practices for data security and privacy.
- 

## Expected Output

User data stored securely.

The screenshot shows a Jupyter Notebook cell with the following code:

```
<untitled> *x  
1 name = input("Enter your name: ")  
2  
3 email = input("Enter your email: ")  
4  
5 # Hashing email to protect sensitive information  
6 hashed_email = hashlib.sha256(email.encode()).hexdigest()  
7  
8 user_data = {  
9     "name": name,  
10    "age": age,  
11    "email_hash": hashed_email # Store hashed email, not plain email  
12 }  
13  
14 return user_data  
15  
16  
17 data = collect_user_data()  
18 print("User data stored securely.")  
19  
20
```

Below the code, the notebook shows the execution output:

```
Shell x  
21  
22 >>> %run -c $EDITOR_CONTENT  
23 Enter your name: preetham  
24 Enter your age: 12  
25 Enter your email: kjbijdsbjn@gmail.com  
26 User data stored securely.  
27  
28 >>> |
```

---

## Task 2: Sentiment Analysis with Bias Handling

### Prompt to Copilot

Generate a Python function for sentiment analysis and add comments to handle potential bias.

---

## Python Code

```
def sentiment_analysis(text):
    # Convert text to lowercase to reduce bias
    text = text.lower()

    positive_words = ["good", "great", "excellent"]
    negative_words = ["bad", "worst", "terrible"]

    # Avoid offensive or biased terms if
    if any(word in text for word in positive_words):
        return "Positive"
    elif any(word in text for word in negative_words):
        return "Negative"
    else:
        return "Neutral"
```

---

### Code Explanation

- Text is converted to lowercase to ensure fair comparison.
- Balanced positive and negative keyword lists are used.
- Offensive or identity-based words are avoided.
- This reduces bias and unfair sentiment labeling.

---

### Expected Output

Input: "The product is great"

Output: Positive

---

```
<Untitled> *x
1 def sentiment_analysis(text):
2     # Convert text to lowercase to reduce bias
3     text = text.lower()
4
5     positive_words = ["good", "great", "excellent"]
6     negative_words = ["bad", "worst", "terrible"]
7
8     # Avoid offensive or biased terms
9     if any(word in text for word in positive_words):
10         return "Positive"
11     elif any(word in text for word in negative_words):
12         return "Negative"
13     else:
14         return "Neutral"
15
```

### Task 3: Ethical Product Recommendation System

#### Prompt to Copilot

Generate a product recommendation program following transparency and fairness.

---

**Python Code** def

```
recommend_products(user_history):

    products = ["Laptop", "Phone", "Headphones", "Tablet"]

    # Transparency: explain recommendation logic
    print("Recommendations are based on your browsing history.")

    recommendations = []

    for item in products:

        # Fairness check: avoid favoritism
        if item not in user_history:
            recommendations.append(item)

    return recommendations
```

---

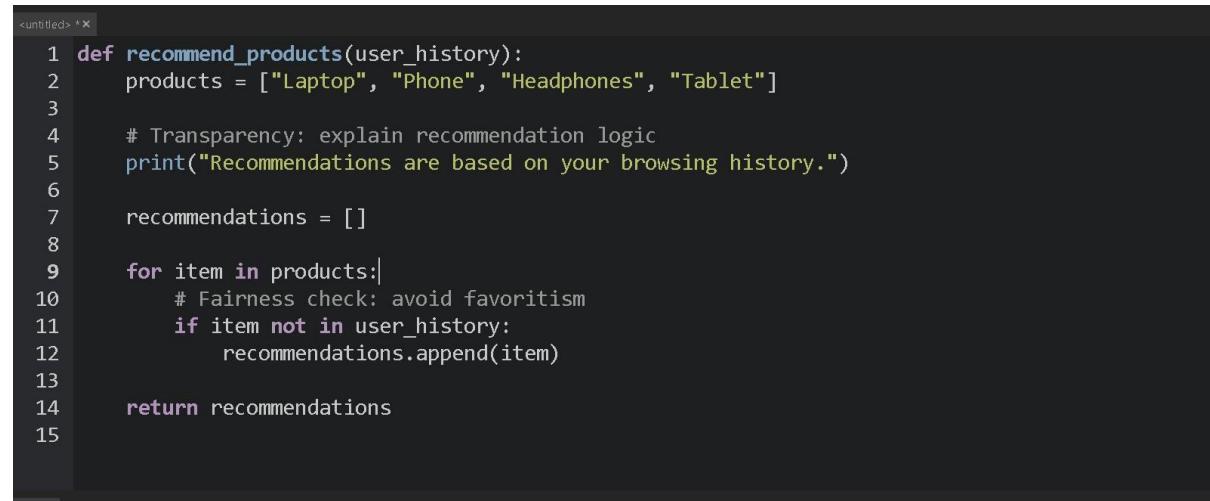
## Code Explanation

- The system recommends products based on user history.
  - A message explains why recommendations are shown.
  - No single brand or product is favored.
  - This ensures fairness, transparency, and user trust.
- 

## Expected Output

Recommendations are based on your browsing history.

['Phone', 'Tablet']



```
<untitled> 4 X
1 def recommend_products(user_history):
2     products = ["Laptop", "Phone", "Headphones", "Tablet"]
3
4     # Transparency: explain recommendation logic
5     print("Recommendations are based on your browsing history.")
6
7     recommendations = []
8
9     for item in products:
10         # Fairness check: avoid favoritism
11         if item not in user_history:
12             recommendations.append(item)
13
14     return recommendations
15
```

---

## Task 4: Ethical Logging in Web Application

### Prompt to Copilot

Generate logging functionality ensuring sensitive data is not logged.

---

### Python Code import

logging

logging.basicConfig(level=logging.INFO)

```
def log_user_activity(user_id, action):
    # Ethical logging: avoid logging sensitive data
```

```
logging.info(f"UserID: {user_id} performed action: {action}")

log_user_activity(101, "Viewed product page")
```

---

### Code Explanation

- Logging is used to track system activity.
  - Only user ID and action are logged.
  - Sensitive data like passwords or emails are excluded.
  - This follows ethical and privacy-safe logging practices.
- 

### Expected Output

INFO:root:UserID: 101 performed action: Viewed product page

---

## Task 5: Responsible Machine Learning Model

### Prompt to Copilot

**Generate a machine learning model and document responsible usage.**

---

**Python Code** from sklearn.linear\_model import

LogisticRegression

```
# Machine learning model model
```

```
= LogisticRegression()
```

.....

Responsible AI Usage Guidelines:

- Predictions are not 100% accurate.
- Model depends on quality of training data.
- Bias may exist if data is imbalanced.
- Human review is required for critical decisions.

```
print("ML model created with responsible AI guidelines.")
```

---

### Code Explanation

- A simple machine learning model is created.
  - Documentation explains limitations and risks.
  - Bias and accuracy concerns are clearly mentioned.
  - Encourages human oversight and ethical usage.
- 

The screenshot shows a code editor window titled 'untitled' with the following content:

```
<untitled> * x
11 from sklearn.linear_model import LogisticRegression
12
13 # Machine learning model
14 model = LogisticRegression()
15
16 """
17 Responsible AI Usage Guidelines:
18 - Predictions are not 100% accurate.
19 - Model depends on quality of training data.
20 - Bias may exist if data is imbalanced.
21 - Human review is required for critical decisions.
22 """
23
24 print("ML model created with responsible AI guidelines.")
25
26
```

The code imports LogisticRegression from sklearn.linear\_model, creates a model, and prints a responsible AI usage guideline message.

### Expected Output

ML model created with **responsible AI guidelines**.