AI ASSISTED CODING:

# TASK DESCRIPTION1:

# Collecting user data

def collect\_user\_data():

name = input("Enter your name: ")

age = input("Enter your age: ")

email = input("Enter your email: ")

user\_data = {

"name": name,

"age": age,

"email": email

}

return user\_data

# Example usage

data = collect\_user\_data()

print("Collected Data:", data)

# --- Data Protection & Anonymization Tips ---

# 1. Avoid storing personally identifiable information (PII) unless necessary.

# If you must store it, use encryption (e.g., AES) to protect it.

# 2. Hash sensitive fields like email using a secure hashing algorithm (e.g., SHA-256)

# if you only need to verify identity without revealing the actual data.

# 3. Consider pseudonymization: replace names with random IDs or aliases.

# 4. Limit access: ensure only authorized users or systems can access raw data.

# 5. Use secure transmission protocols (e.g., HTTPS) when sending data over networks.

# 6. Regularly audit and purge old or unused data to reduce exposure risk.

# 7. If sharing data for analytics, aggregate it (e.g., age groups instead of exact age)

# and remove direct identifiers like name and email.

# 8. Comply with data protection regulations like GDPR or India's DPDP Act

# to ensure legal and ethical handling of user data.

# TASK DESCRIPTION 2:

from textblob import TextBlob

def analyze\_sentiment(text):

"""

Analyzes the sentiment of the input text.

Returns polarity (-1 to 1) and subjectivity (0 to 1).

"""

blob = TextBlob(text)

polarity = blob.sentiment.polarity

subjectivity = blob.sentiment.subjectivity

sentiment = "Positive" if polarity > 0 else "Negative" if polarity < 0 else "Neutral"

return {

"sentiment": sentiment,

"polarity": polarity,

"subjectivity": subjectivity

}

# Example usage

sample\_text = "I love the new design! It's clean and intuitive."

result = analyze\_sentiment(sample\_text)

print("Sentiment Analysis Result:", result)

### Prompt to Copilot: Bias Identification & Mitigation

"Copilot, can you help me identify potential biases in the sentiment analysis results? Specifically:

* Are there linguistic or cultural patterns that might skew polarity scores?
* Could the model misinterpret sarcasm, slang, or regional dialects?
* How can I diversify my training data to reduce demographic or domain bias?

# TASK DESCRIPTION 3:

# Sample user history data

user\_history = {

"user\_id": "U123",

"viewed\_products": ["Wireless Mouse", "Laptop Stand", "USB-C Hub"],

"purchased\_products": ["Wireless Mouse"]

}

# Sample product catalog with categories

product\_catalog = {

"Wireless Mouse": "Electronics",

"Laptop Stand": "Accessories",

"USB-C Hub": "Electronics",

"Bluetooth Keyboard": "Electronics",

"Ergonomic Chair": "Furniture",

"Noise-Cancelling Headphones": "Electronics"

}

def recommend\_products(user\_history, catalog):

"""

Recommends products based on user's viewed and purchased history.

Ensures recommendations are relevant and avoids over-personalization.

"""

viewed = set(user\_history["viewed\_products"])

purchased = set(user\_history["purchased\_products"])

# Recommend products from the same category as viewed items

recommended = []

for product in catalog:

if product not in viewed and product not in purchased:

for item in viewed:

if catalog[product] == catalog[item]:

recommended.append(product)

break

return recommended

# Generate recommendations

recommendations = recommend\_products(user\_history, product\_catalog)

print("Recommended Products:", recommendations)

# --- Ethical Guidelines ---

# ✅ Transparency:

# Inform users how recommendations are generated (e.g., based on categories of viewed items).

# Provide an option to opt out of personalized recommendations.

# ✅ Fairness:

# Avoid reinforcing stereotypes or excluding products based on biased assumptions.

# Ensure diverse product exposure across categories and user demographics.

# ✅ Privacy:

# Do not store or share identifiable user data without consent.

# Anonymize user IDs and avoid tracking sensitive behavior.

# ✅ Accountability:

# Log recommendation logic and allow users to provide feedback or flag irrelevant suggestions.

# ✅ Inclusivity:

# Periodically audit recommendation outcomes to ensure they serve all user groups fairly.

# TASK DESCRIPTION 4:

from flask import Flask, request

import logging

app = Flask(\_\_name\_\_)

# Configure logging

logging.basicConfig(

filename='app.log',

level=logging.INFO,

format='%(asctime)s %(levelname)s: %(message)s'

)

@app.route('/submit', methods=['POST'])

def submit():

username = request.form.get('username')

email = request.form.get('email')

password = request.form.get('password') # ⚠️ Sensitive

# Log only non-sensitive metadata

logging.info(f"Form submitted by user: {username}")

# Never log passwords or sensitive fields

# logging.info(f"Email: {email}, Password: {password}") # ❌ Avoid this

return "Submission received"

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

### Prompt to Ensure Ethical Logging

"Copilot, please help me review this logging setup to ensure it follows ethical guidelines:

* Are we avoiding logging sensitive data like passwords, emails, or personal identifiers?
* Should we mask or hash any fields before logging?
* How can we rotate logs securely and limit access to log files?
* What best practices should we follow to comply with data protection laws like GDPR or DPDP?"

# TASK DESCRIPTION5:

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.pipeline import Pipeline

# Sample training data

texts = [

"I love this product!", "This is terrible.", "Absolutely fantastic!",

"Worst experience ever.", "Not bad, could be better.", "I hate it."

]

labels = ["positive", "negative", "positive", "negative", "neutral", "negative"]

# Create a pipeline for text classification

model = Pipeline([

('vectorizer', CountVectorizer()),

('classifier', MultinomialNB())

])

# Train the model

model.fit(texts, labels)

# Example prediction

sample\_input = ["I really enjoyed this!"]

prediction = model.predict(sample\_input)

print("Predicted sentiment:", prediction[0])

### Prompt to Ensure Ethical Logging

"Copilot, please help me review this logging setup to ensure it follows ethical guidelines:

* Are we avoiding logging sensitive data like passwords, emails, or personal identifiers?
* Should we mask or hash any fields before logging?
* How can we rotate logs securely and limit access to log files?
* What best practices should we follow to comply with data protection laws like GDPR or DPDP?"