AI ASSISTED CODING

LAB ASS-5.1

J.Varshini

2303A51758

BATCH-11

Task Description #1 (Privacy in API Usage)

Task: Use an AI tool to generate a Python program that connects to a weather API.

Prompt:

"Generate code to fetch weather data securely without exposing API keys in the code."

Expected Output:

- Original AI code (check if keys are hardcoded).
- Secure version using enviro**nment variables**.

PROMPT:

#Generate a Python program that fetches weather data from a public weather API.Do NOT hardcode the API key anywhere in the source code.The API key MUST be read only from an environment variable using the os module.If the API key is missing,the program should raise an error.Do not include placeholder API keys in the code.

CODE:

Task Description #2 (Privacy & Security in File Handling)

Task: Use an AI tool to generate a Python script that stores user data

(name, email, password) in a file.

Analyze: Check if the AI stores sensitive data in plain text or without encryption.

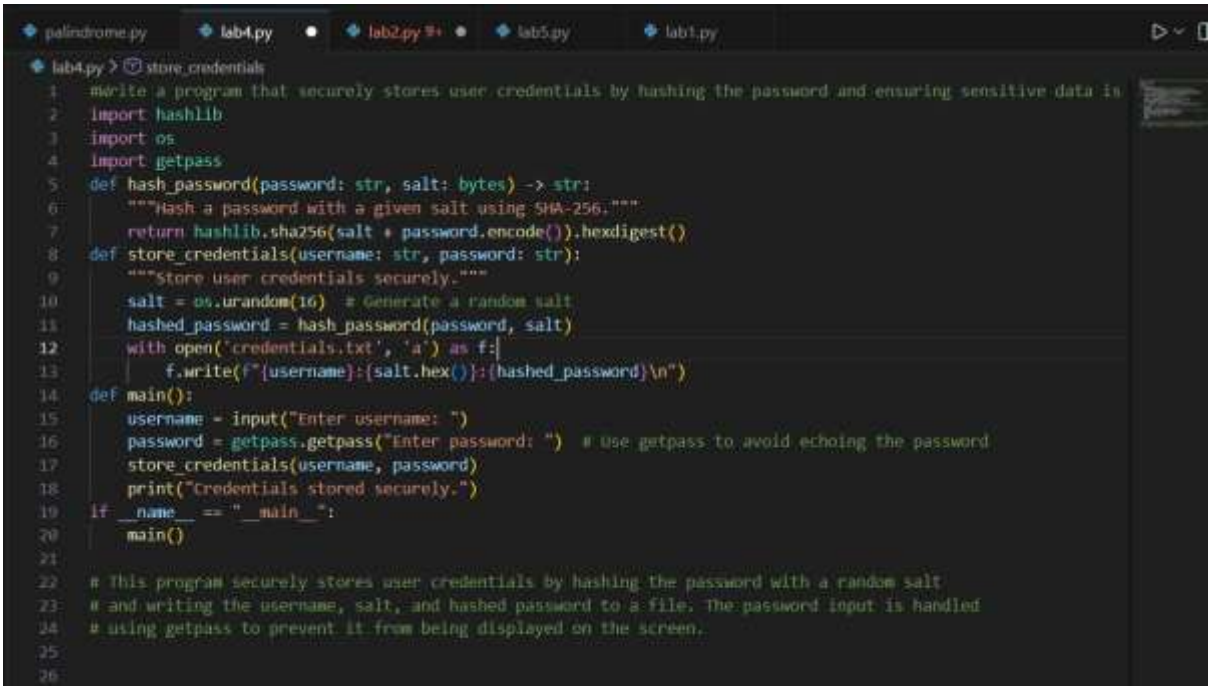
Expected Output:

- Identified privacy risks.
- Revised version with encrypted password storage (e.g., hashing).

PROMPT:

#Write a program that securely stores user credentials by hashing the password and ensuring sensitive data is not stored in plain text.

CODE:



```python
#Write a program that securely stores user credentials by hashing the password and ensuring sensitive data is
import hashlib
import os
import getpass
def hash_password(password: str, salt: bytes) -> str:
    """Hash a password with a given salt using SHA-256."""
    return hashlib.sha256(salt + password.encode()).hexdigest()
def store_credentials(username: str, password: str):
    """Store user credentials securely."""
    salt = os.urandom(16)  # Generate a random salt
    hashed_password = hash_password(password, salt)
    with open('credentials.txt', 'a') as f:
        f.write(f"{username}:{salt.hex()}:{hashed_password}\n")
def main():
    username = input("Enter username: ")
    password = getpass.getpass("Enter password: ")  # Use getpass to avoid echoing the password
    store_credentials(username, password)
    print("Credentials stored securely.")
if __name__ == "__main__":
    main()

# This program securely stores user credentials by hashing the password with a random salt
# and writing the username, salt, and hashed password to a file. The password input is handled
# using getpass to prevent it from being displayed on the screen.
```

OUTPUT:

```
Enter username: 123456
Enter password:
Credentials stored securely.
```

Task Description #3 (Transparency in Algorithm Design) Objective: Use AI to generate an Armstrong number checking function with comments and explanations.

Instructions:

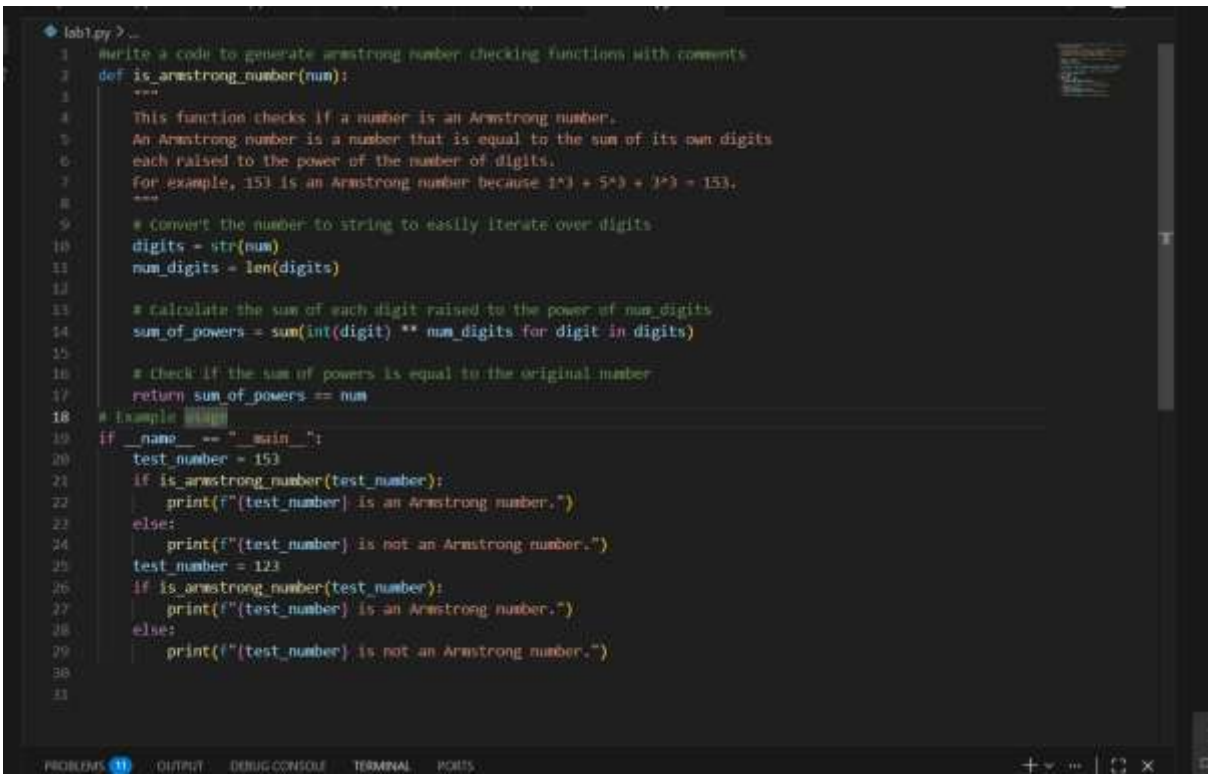
1. Ask AI to explain the code line-by-line.

2. Compare the explanation with code functionality.

Expected Output:

- Transparent, commented code.
- Correct, easy-to-understand explanation **PROMPT:**

#write a code to generate armstrong number checking functions with comments **CODE:**



```
lab1.py > ...
#write a code to generate armstrong number checking functions with comments
def is_armstrong_number(num):
    """
    This function checks if a number is an Armstrong number.
    An Armstrong number is a number that is equal to the sum of its own digits
    each raised to the power of the number of digits.
    For example, 153 is an Armstrong number because 1^3 + 5^3 + 3^3 = 153.
    """
    # Convert the number to string to easily iterate over digits
    digits = str(num)
    num_digits = len(digits)

    # Calculate the sum of each digit raised to the power of num_digits
    sum_of_powers = sum(int(digit) ** num_digits for digit in digits)

    # Check if the sum of powers is equal to the original number
    return sum_of_powers == num
# Example usage
if __name__ == "__main__":
    test_number = 153
    if is_armstrong_number(test_number):
        print(f"{test_number} is an Armstrong number.")
    else:
        print(f"{test_number} is not an Armstrong number.")
    test_number = 123
    if is_armstrong_number(test_number):
        print(f"{test_number} is an Armstrong number.")
    else:
        print(f"{test_number} is not an Armstrong number.")


PROBLEMS 11  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
```

OUTPUT:

```
153 is an Armstrong number.
123 is not an Armstrong number.
Python
```

Task Description #4 (Transparency in Algorithm Comparison)

Task: Use AI to implement two sorting algorithms (e.g., QuickSort and BubbleSort).

Prompt:

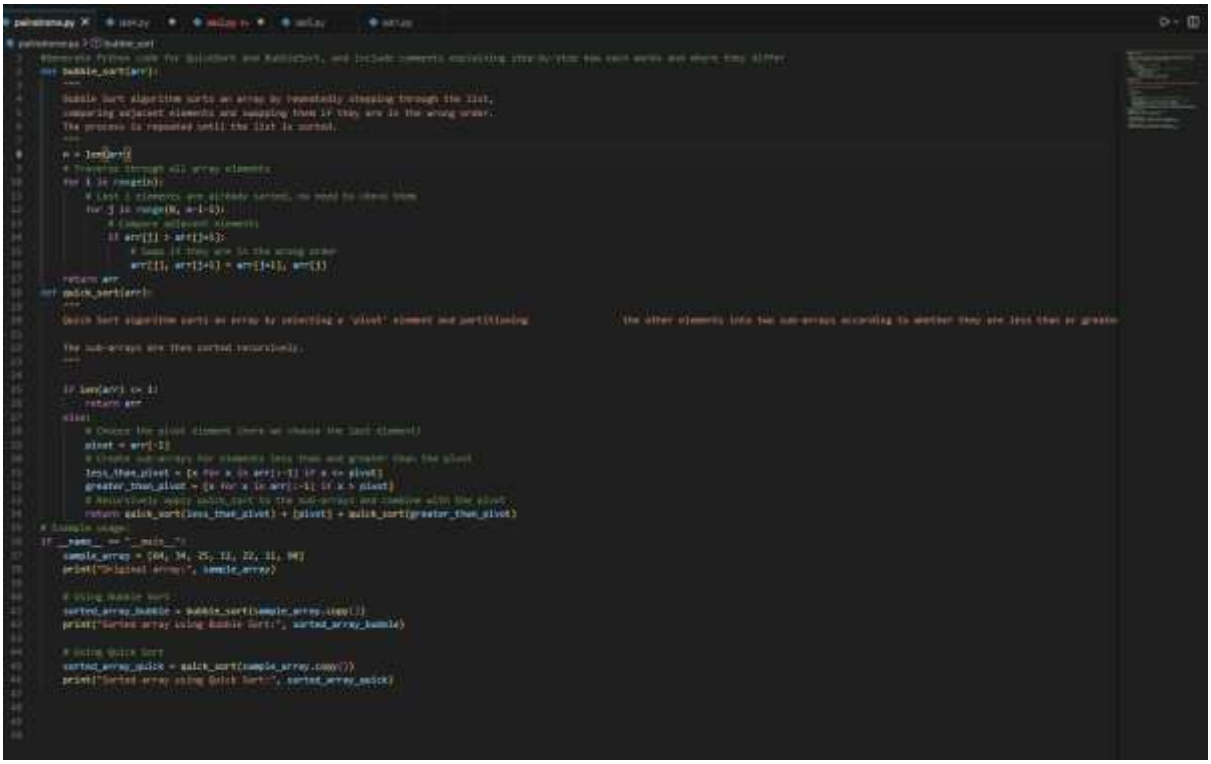
"Generate Python code for QuickSort and BubbleSort, and include comments explaining step-by-step how each works and where they differ." Expected Output:

- Code for both algorithms.
- Transparent, comparative explanation of their logic and efficiency.

PROMPT:

#Generate Python code for QuickSort and BubbleSort, and include comments explaining step-by-step how each works and where they differ.

CODE:



OUTPUT:

Task Description #5 (Transparency in AI Recommendations)

Task: Use AI to create a product recommendation system.

Prompt:

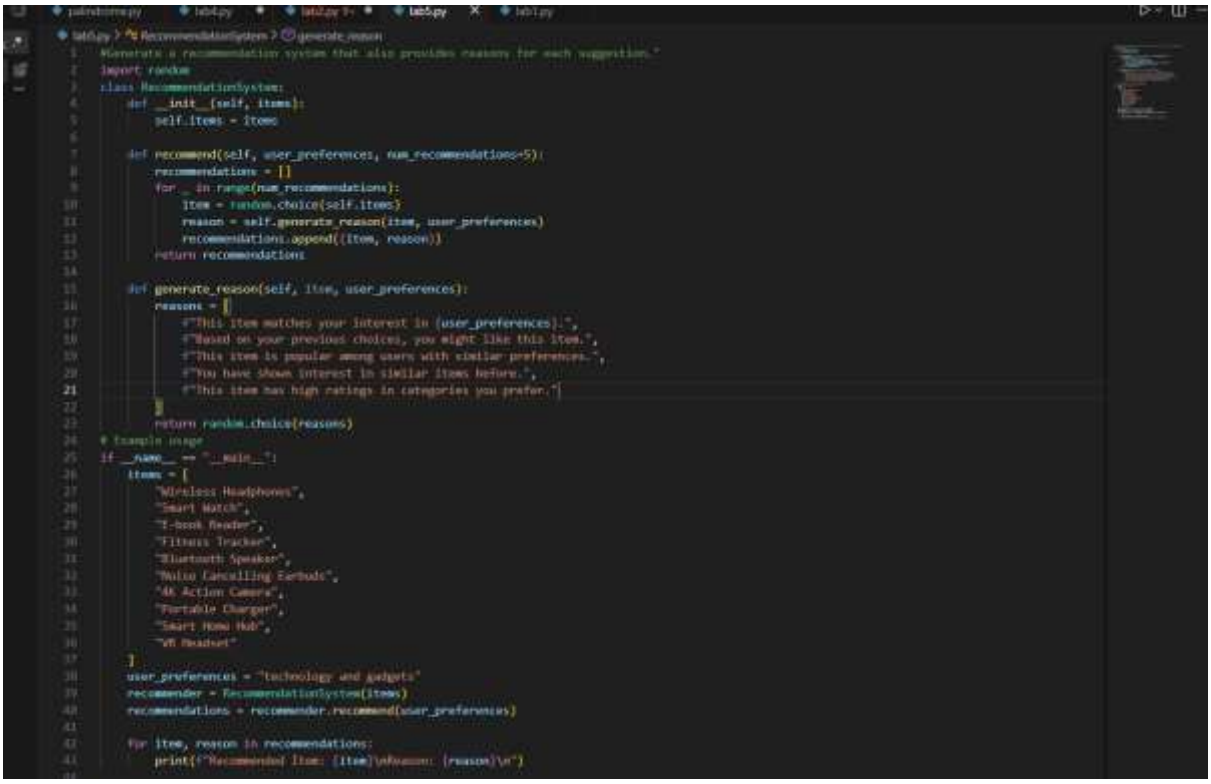
"Generate a recommendation system that also provides reasons for each suggestion."

Expected Output:

- Code with explainable recommendations.
- Evaluation of whether explanations are understandable.

PROMPT:

#Generate a recommendation system that also provides reasons for each suggestion." **CODE:**



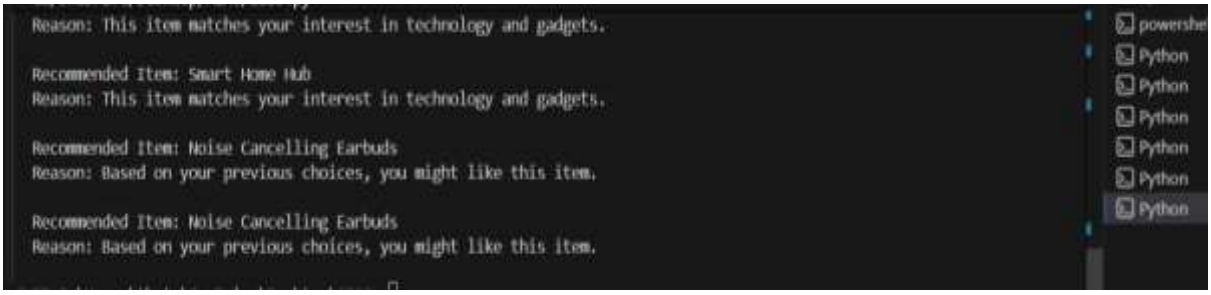
```
#Generate a recommendation system that also provides reasons for each suggestion."
import random
class RecommendationSystem:
    def __init__(self, items):
        self.items = items

    def recommend(self, user_preferences, num_recommendations=5):
        recommendations = []
        for _ in range(num_recommendations):
            item = random.choice(self.items)
            reason = self.generate_reason(item, user_preferences)
            recommendations.append((item, reason))
        return recommendations

    def generate_reason(self, item, user_preferences):
        reasons = [
            f"This item matches your interest in {user_preferences}.",
            f"Based on your previous choices, you might like this item.",
            f"This item is popular among users with similar preferences.",
            f"You have shown interest in similar items before.",
            f"This item has high ratings in categories you prefer."
        ]
        return random.choice(reasons)
# Example usage
if __name__ == "__main__":
    items = [
        "Wireless Headphones",
        "Smart Watch",
        "E-book Reader",
        "Fitness Tracker",
        "Bluetooth Speaker",
        "Noise Cancelling Earbuds",
        "4K Action Camera",
        "Portable Charger",
        "Smart Home Hub",
        "VR Headset"
    ]
    user_preferences = "technology and gadgets"
    recommender = RecommendationSystem(items)
    recommendations = recommender.recommend(user_preferences)

    for item, reason in recommendations:
        print(f"Recommended Item: {item}\nReason: {reason}\n")
```

OUTPUT:


Reason: This item matches your interest in technology and gadgets.
Recommended Item: Smart Home Hub
Reason: This item matches your interest in technology and gadgets.
Recommended Item: Noise Cancelling Earbuds
Reason: Based on your previous choices, you might like this item.
Recommended Item: Noise Cancelling Earbuds
Reason: Based on your previous choices, you might like this item.
powershe
Python
Python
Python
Python
Python
Python