AI ASSISTED CODING

LAB-5.1

M.ABHISHEK

2303A51762

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:

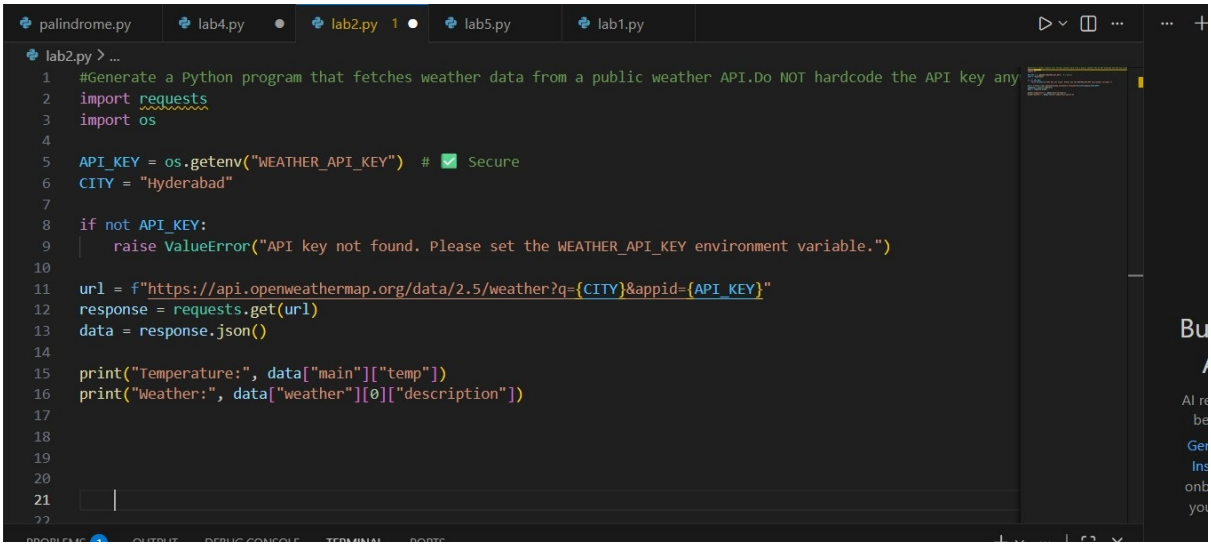
- Original AI code (check if keys are hardcoded).
- Secure version using enviro**nment variables**.

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.

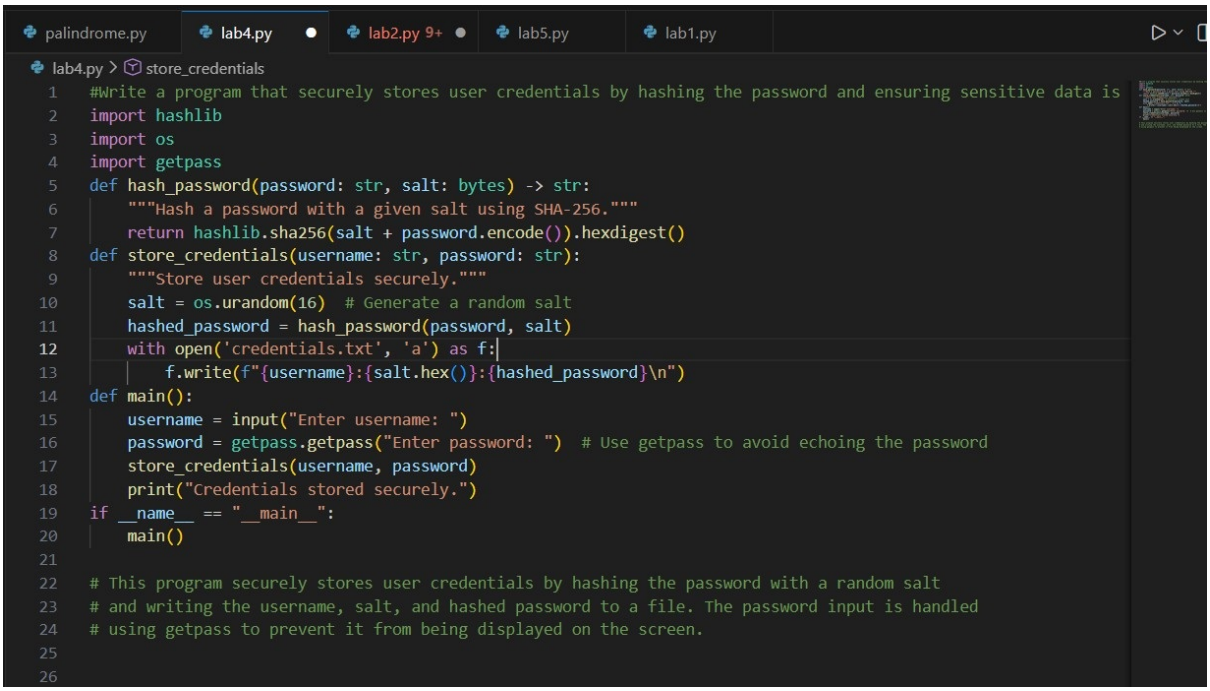
Expected Output:

- 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.

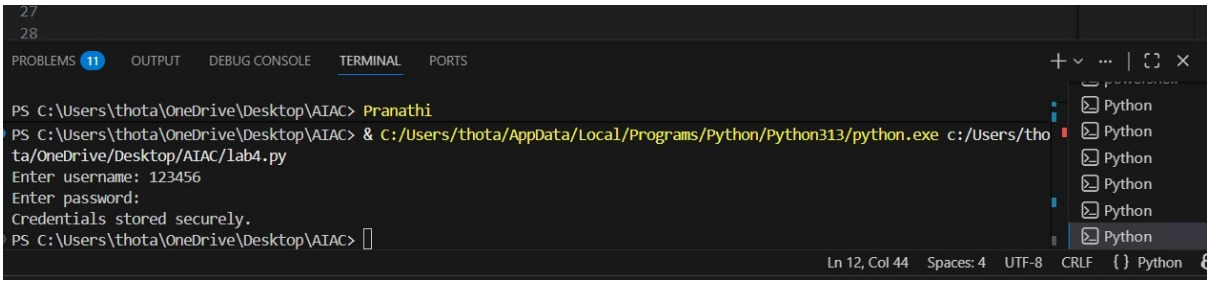
CODE:



```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:



```
PROBLEMS 11  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\thota\OneDrive\Desktop\AIAC> Pranathi
PS C:\Users\thota\OneDrive\Desktop\AIAC> & C:/Users/thota/AppData/Local/Programs/Python/Python313/python.exe c:/Users/thota/OneDrive/Desktop/AIAC/lab4.py
Enter username: 123456
Enter password:
Credentials stored securely.
PS C:\Users\thota\OneDrive\Desktop\AIAC>
```

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

Instructions:

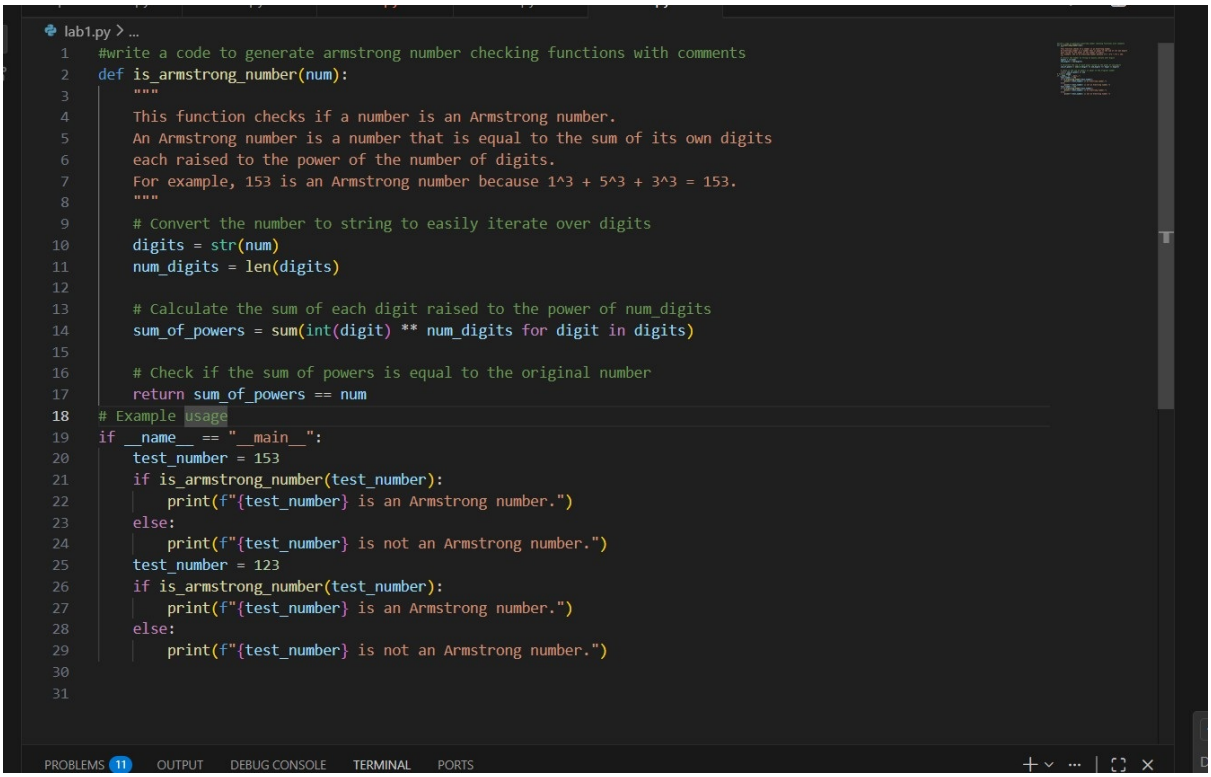
1. Ask AI to explain the code line-by-line.
2. Compare the explanation with code functionality.

Expected Output:

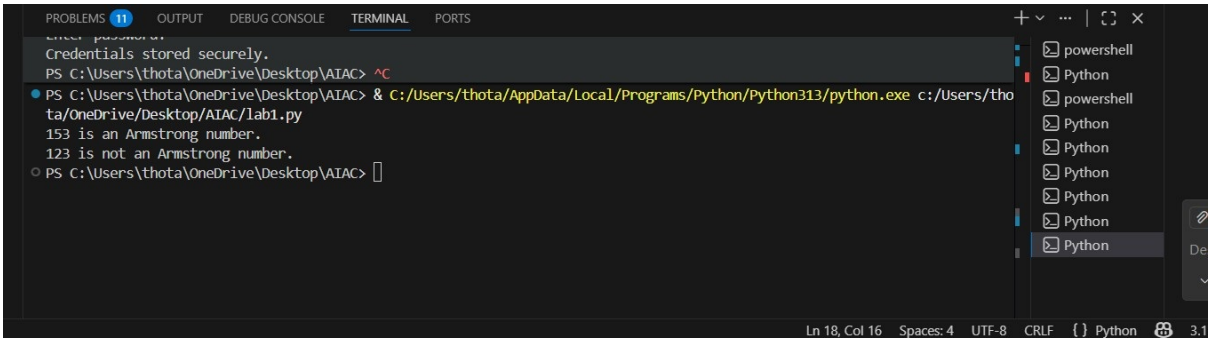
- Transparent, commented code.
- Correct, easy-to-understand explanation

PROMPT:

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



OUTPUT:



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:

```python
#Generate Python code for QuickSort and BubbleSort, and include comments explaining step-by-step how each works and where they differ
def bubble_sort(arr):
    """
    Bubble Sort algorithm sorts an array by repeatedly stepping through the list,
    comparing adjacent elements and swapping them if they are in the wrong order.
    The process is repeated until the list is sorted.
    """
    n = len(arr)
    # Traverse through all array elements
    for i in range(n):
        # Last i elements are already sorted, no need to check them
        for j in range(0, n-i-1):
            # Compare adjacent elements
            if arr[j] > arr[j+1]:
                # Swap if they are in the wrong order
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr
def quick_sort(arr):
    """
    Quick Sort algorithm sorts an array by selecting a 'pivot' element and partitioning the other elements into two sub-arrays according to whether they are less than or greater

    The sub-arrays are then sorted recursively.
    """

    if len(arr) <= 1:
        return arr
    else:
        # Choose the pivot element (here we choose the last element)
        pivot = arr[-1]
        # Create sub-arrays for elements less than and greater than the pivot
        less_than_pivot = [x for x in arr[:-1] if x <= pivot]
        greater_than_pivot = [x for x in arr[:-1] if x > pivot]
        # Recursively apply quick_sort to the sub-arrays and combine with the pivot
        return quick_sort(less_than_pivot) + [pivot] + quick_sort(greater_than_pivot)
# Example usage:
if __name__ == "__main__":
    sample_array = [64, 34, 25, 12, 22, 11, 90]
    print("Original array:", sample_array)

    # Using Bubble Sort
    sorted_array_bubble = bubble_sort(sample_array.copy())
    print("Sorted array using Bubble Sort:", sorted_array_bubble)

    # Using Quick Sort
    sorted_array_quick = quick_sort(sample_array.copy())
    print("Sorted array using Quick Sort:", sorted_array_quick)
```

OUTPUT:

```
PROBLEMS 11  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\thota\OneDrive\Desktop\AIAC> & C:/Users/thota/AppData/Local/Programs/Python/Python313/python.exe c:/Users/tho
...
PS C:\Users\thota\OneDrive\Desktop\AIAC> & C:/Users/thota/AppData/Local/Programs/Python/Python313/python.exe c:/Users/tho
ta/OneDrive/Desktop/AIAC/palindrome.py
Original array: [64, 34, 25, 12, 22, 11, 90]
Sorted array using Bubble Sort: [11, 12, 22, 25, 34, 64, 90]
Sorted array using Quick Sort: [11, 12, 22, 25, 34, 64, 90]
PS C:\Users\thota\OneDrive\Desktop\AIAC>
```

powershell
Python
powershell
Python
Python
Python
Python
Python
Python

Task Description #5 (Transparency in AI Recommendations)

Task: Use AI to create a product recommendation system.

Prompt:

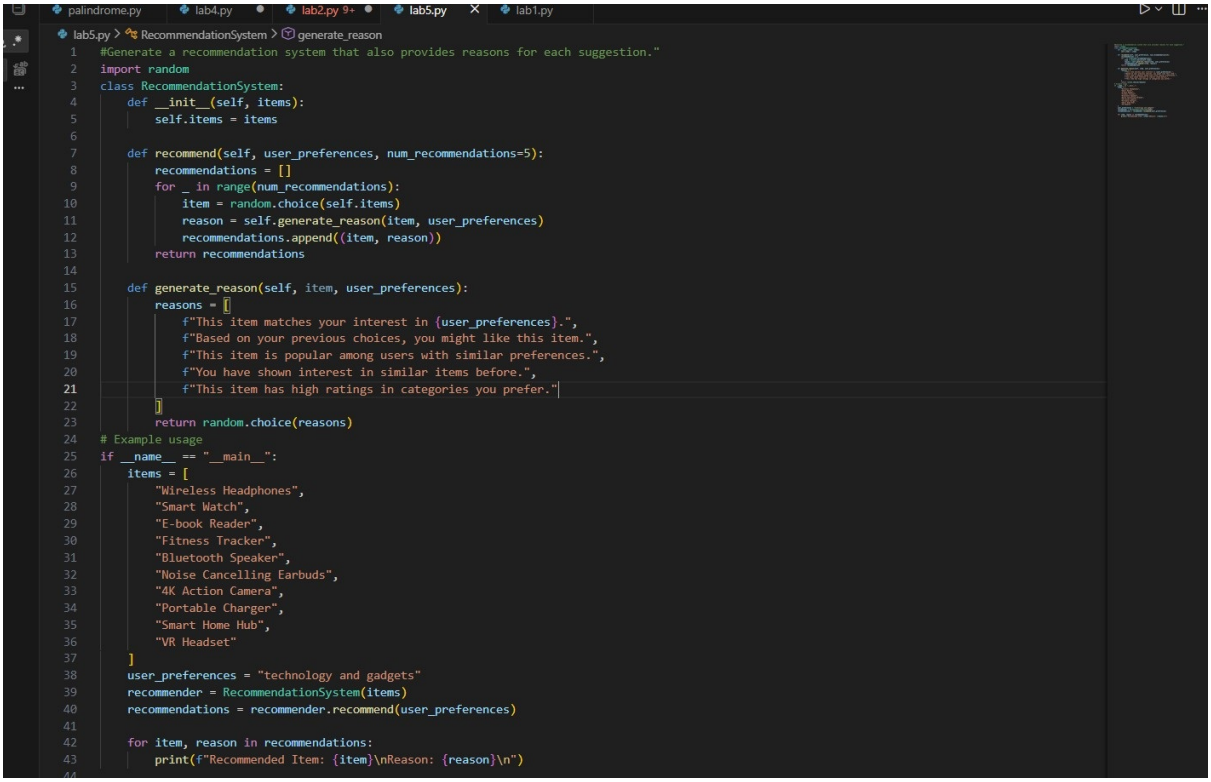
"Generate a recommendation system that also provides reasons for each suggestion."

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:**



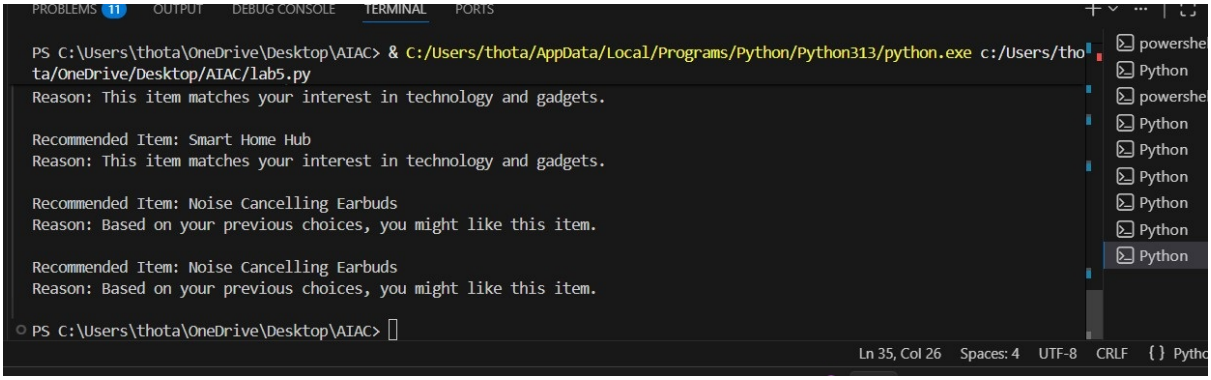
```python
#Generate a recommendation system that also provides reasons for each suggestion."
import random
class RecommendationSystem:
    def __init__(self, items):
        self.items = items

    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:


PROBLEMS 11
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\Users\thota\OneDrive\Desktop\AIAC> & C:/Users/thota/AppData/Local/Programs/Python/Python313/python.exe c:/Users/thota/OneDrive/Desktop/AIAC/lab5.py
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.
PS C:\Users\thota\OneDrive\Desktop\AIAC>
powershe
Python
powershe
Python
Python
Python
Python
Python
Python
Ln 35, Col 26
Spaces: 4
UTF-8
CRLF
{ } Pytho