NAME:Boppidi Sanjana Reddy

ROLL NO:2303A51455

BATCH NO:07

LAB 5 – Ethical Foundations: Responsible AI Coding Practices

Task 1 – Privacy in API Usage

Objective

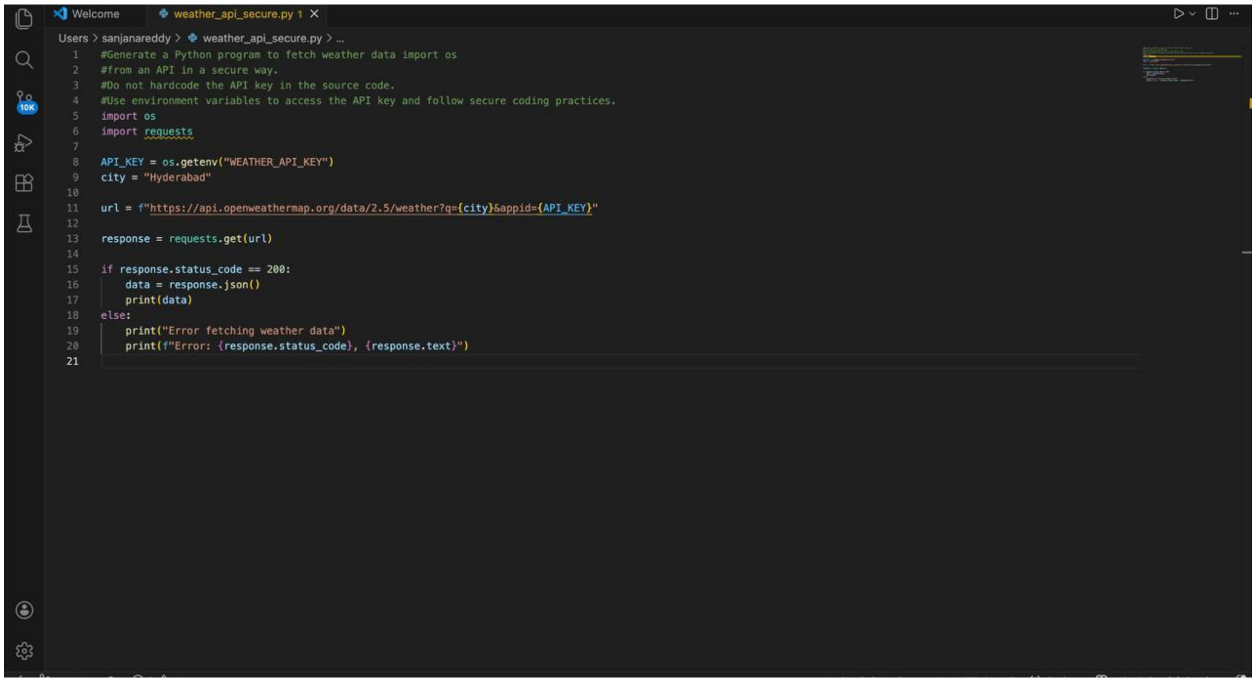
To fetch weather data using an API without exposing the API key in the source code.

Prompt Used

Generate a Python program to fetch weather data from an API in a secure way.

Do not hardcode the API key in the source code.

Use environment variables to access the API key.



```python
#Generate a Python program to fetch weather data import os
#from an API in a secure way.
#Do not hardcode the API key in the source code.
#Use environment variables to access the API key and follow secure coding practices.
import os
import requests

API_KEY = os.getenv("WEATHER_API_KEY")
city = "Hyderabad"

url = f"https://api.openweathermap.org/data/2.5/weather?q={city}&appid={API_KEY}"

response = requests.get(url)

if response.status_code == 200:
    data = response.json()
    print(data)
else:
    print("Error fetching weather data")
    print(f"Error: {response.status_code}, {response.text}")
```

Sample Output:-

Weather data fetched successfully

Ethical Explanation

The API key is not hardcoded in the program and is accessed using an environment variable.

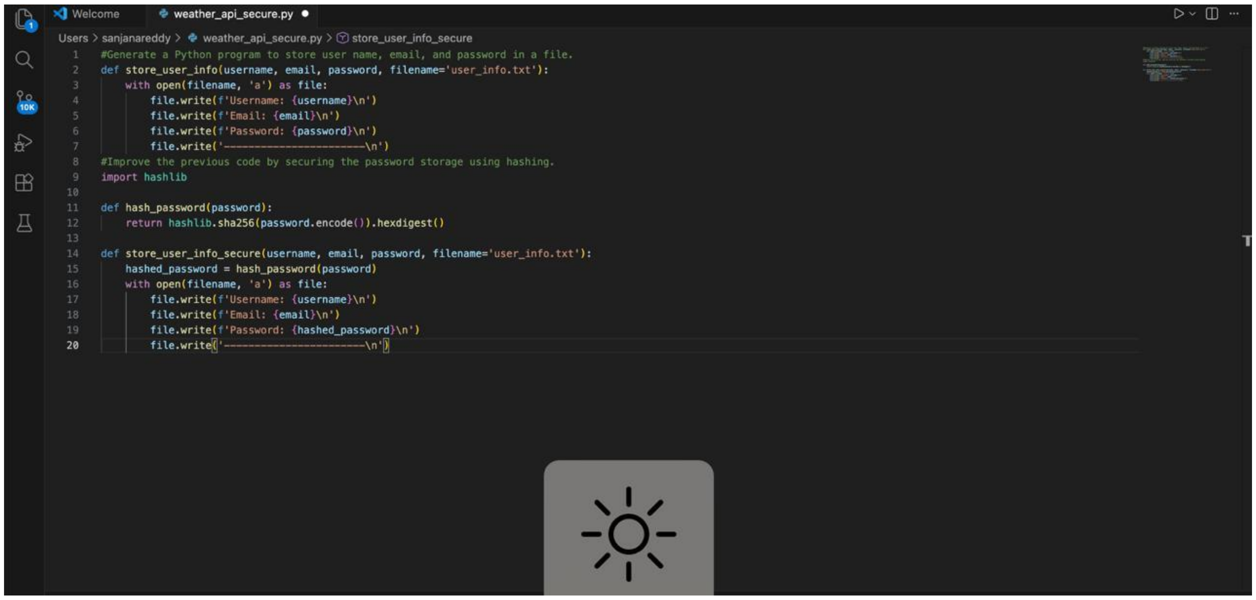
This prevents sensitive information from being exposed if the code is shared publicly.

Using environment variables follows secure and ethical AI coding practices.

Task 2 – Privacy & Security in File Handling

Objective

To analyze how AI-generated code handles sensitive user data and improve its security.


Welcome
weather_api_secure.py
Users > sanjanareddy > weather_api_secure.py > store_user_info_secure

```python
#Generate a Python program to store user name, email, and password in a file.
def store_user_info(username, email, password, filename='user_info.txt'):
    with open(filename, 'a') as file:
        file.write(f'Username: {username}\n')
        file.write(f'Email: {email}\n')
        file.write(f'Password: {password}\n')
        file.write('-----------------------\n')
#Improve the previous code by securing the password storage using hashing.
import hashlib

def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()

def store_user_info_secure(username, email, password, filename='user_info.txt'):
    hashed_password = hash_password(password)
    with open(filename, 'a') as file:
        file.write(f'Username: {username}\n')
        file.write(f'Email: {email}\n')
        file.write(f'Password: {hashed_password}\n')
        file.write('-----------------------\n')
```

sample output:-

User details are stored securely in the file.

File Content:

Username: Ravi

Email: ravi@gmail.com

Password: <hashed_password_value>

Privacy Risk Identified

The password is stored in plain text.

If the file is accessed by unauthorized users, sensitive information can be exposed.

Ethical Explanation

Hashing converts the password into a non-readable format before storage.

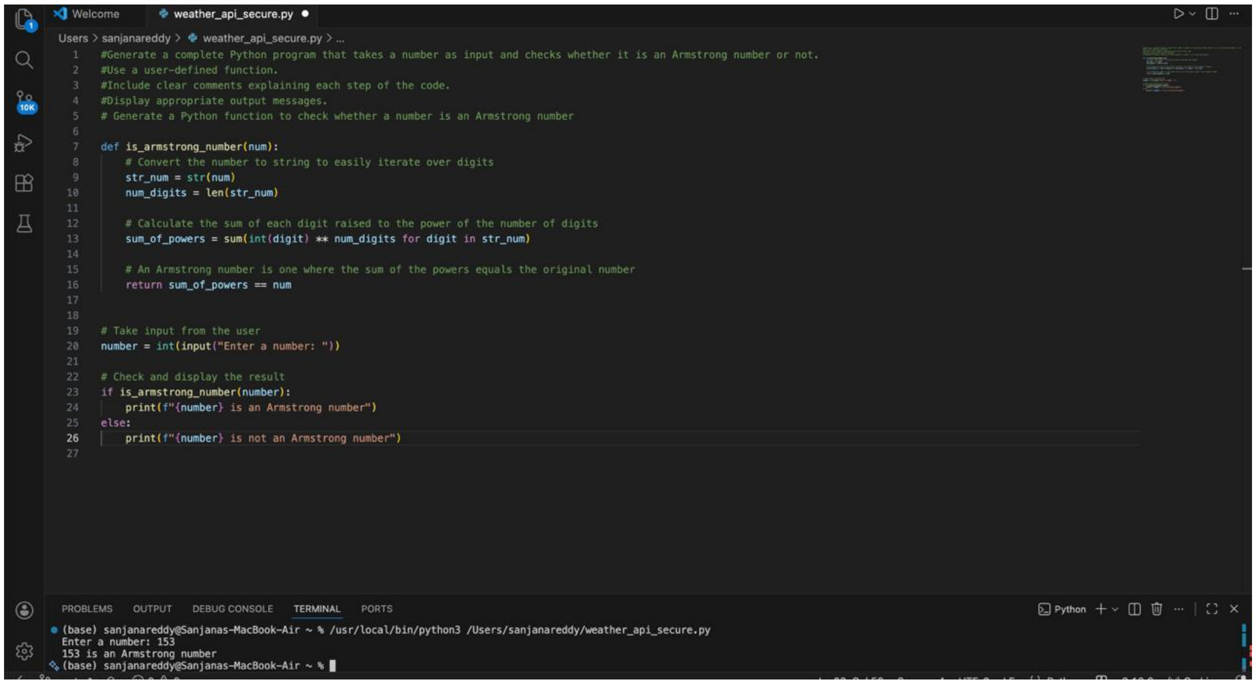
This protects user credentials even if the file is compromised.

It demonstrates responsible handling of sensitive data.

Task 3 – Transparency in Algorithm Design (Armstrong Number)

Objective

To generate an algorithm with clear logic and transparent explanation.



```python
#Generate a complete Python program that takes a number as input and checks whether it is an Armstrong number or not.
#Use a user-defined function.
#Include clear comments explaining each step of the code.
#Display appropriate output messages.
# Generate a Python function to check whether a number is an Armstrong number

def is_armstrong_number(num):
    # Convert the number to string to easily iterate over digits
    str_num = str(num)
    num_digits = len(str_num)

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

    # An Armstrong number is one where the sum of the powers equals the original number
    return sum_of_powers == num


# Take input from the user
number = int(input("Enter a number: "))

# Check and display the result
if is_armstrong_number(number):
    print(f"{number} is an Armstrong number")
else:
    print(f"{number} is not an Armstrong number")
```

```
(base) sanjanareddy@Sanjanas-MacBook-Air ~ % /usr/local/bin/python3 /Users/sanjanareddy/weather_api_secure.py
Enter a number: 153
153 is an Armstrong number
(base) sanjanareddy@Sanjanas-MacBook-Air ~ %
```

AI Explanation

The program converts the number into a string to count its digits.

Each digit is raised to the power of the total number of digits.

The results are added and compared with the original number.

If they match, the number is an Armstrong number.

Transparency Analysis

The explanation clearly matches the code logic.

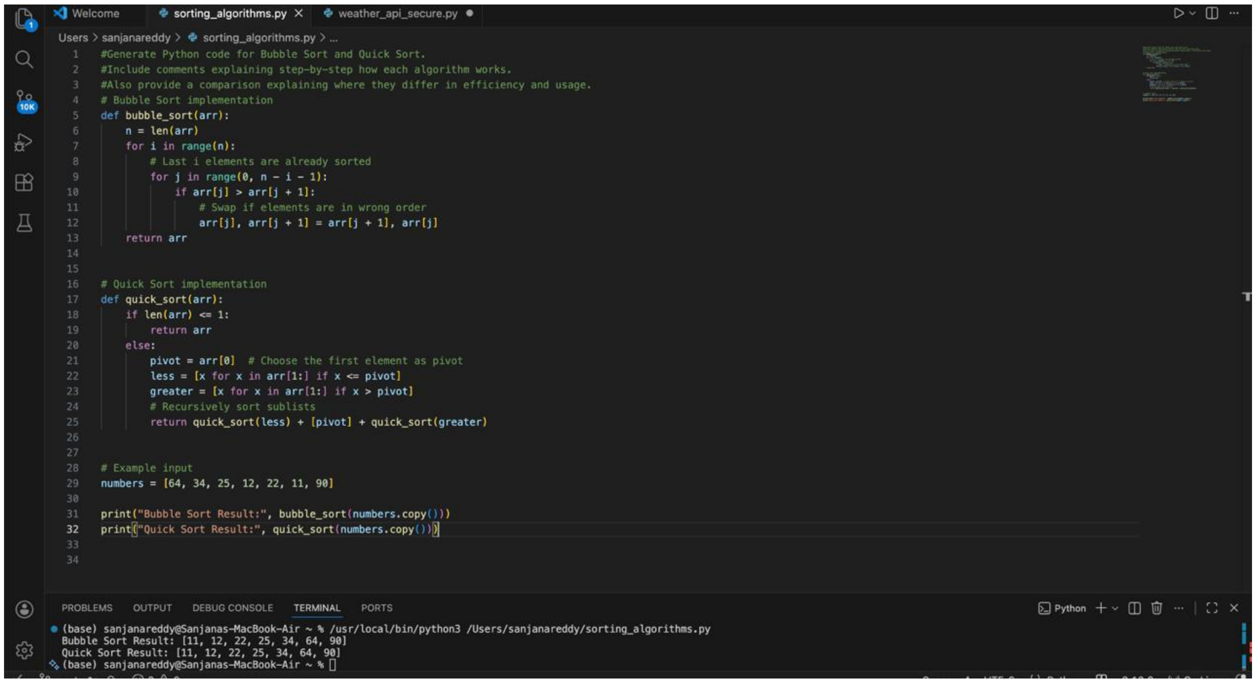
Each step is easy to understand and transparent.

This improves trust in AI-generated code.

Task 4 – Transparency in Algorithm Comparison

Objective:

To compare two sorting algorithms generated by AI and understand their working and efficiency.



```python
#Generate Python code for Bubble Sort and Quick Sort.
#Include comments explaining step-by-step how each algorithm works.
#Also provide a comparison explaining where they differ in efficiency and usage.
# Bubble Sort implementation
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        # Last i elements are already sorted
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                # Swap if elements are in wrong order
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr


# Quick Sort implementation
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    else:
        pivot = arr[0]  # Choose the first element as pivot
        less = [x for x in arr[1:] if x <= pivot]
        greater = [x for x in arr[1:] if x > pivot]
        # Recursively sort sublists
        return quick_sort(less) + [pivot] + quick_sort(greater)


# Example input
numbers = [64, 34, 25, 12, 22, 11, 90]

print("Bubble Sort Result:", bubble_sort(numbers.copy()))
print("Quick Sort Result:", quick_sort(numbers.copy()))
```

```
(base) sanjanareddy@Sanjanas-MacBook-Air ~ % /usr/local/bin/python3 /Users/sanjanareddy/sorting_algorithms.py
Bubble Sort Result: [11, 12, 22, 25, 34, 64, 90]
Quick Sort Result: [11, 12, 22, 25, 34, 64, 90]
(base) sanjanareddy@Sanjanas-MacBook-Air ~ %
```

AI Explanation

Bubble Sort repeatedly compares adjacent elements and swaps them if they are in the wrong order.

This process continues until the list is completely sorted.

Quick Sort selects a pivot element and divides the list into smaller sublists.

These sublists are sorted recursively.

Comparison of Algorithms

Bubble Sort is simple and easy to understand but inefficient for large datasets.

It has a time complexity of $O(n^2)$.

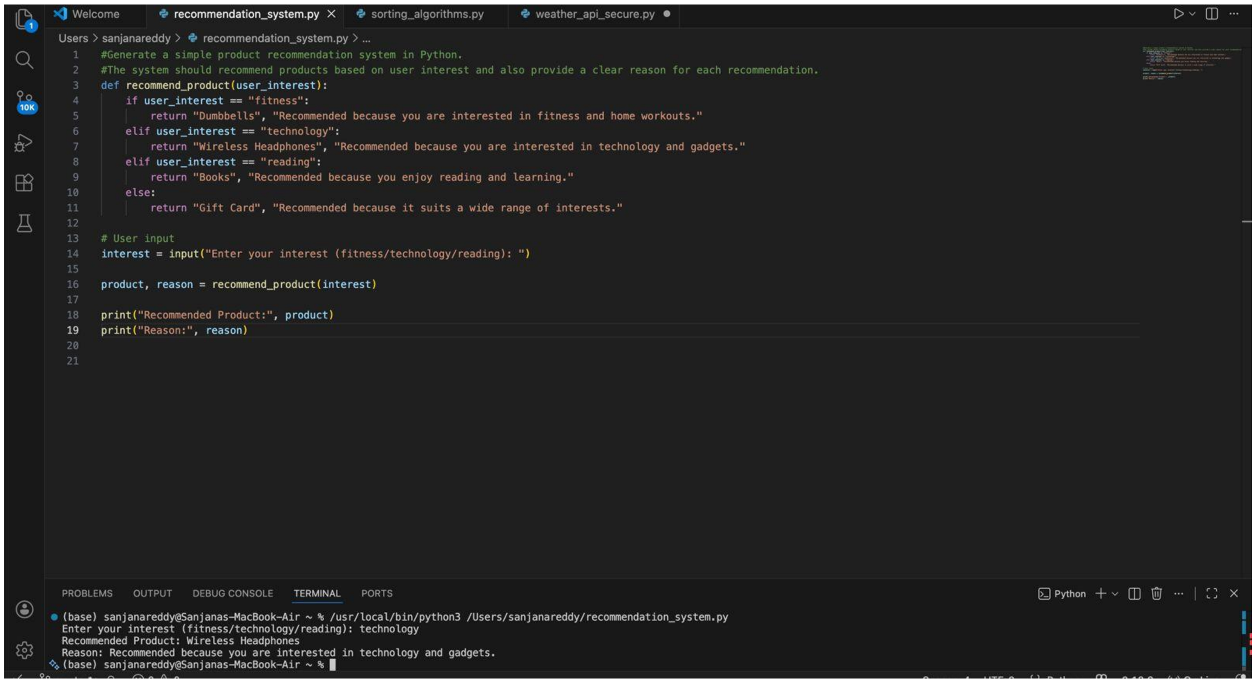
Quick Sort is faster and more efficient for large datasets with an average time complexity of $O(n \log n)$.

Quick Sort is commonly used in real-world applications.

Task 5 – Transparency in AI Recommendations

Objective

To generate an AI-based recommendation system that provides clear reasons for its suggestions.



```python
#Generate a simple product recommendation system in Python.
#The system should recommend products based on user interest and also provide a clear reason for each recommendation.
def recommend_product(user_interest):
    if user_interest == "fitness":
        return "Dumbbells", "Recommended because you are interested in fitness and home workouts."
    elif user_interest == "technology":
        return "Wireless Headphones", "Recommended because you are interested in technology and gadgets."
    elif user_interest == "reading":
        return "Books", "Recommended because you enjoy reading and learning."
    else:
        return "Gift Card", "Recommended because it suits a wide range of interests."

# User input
interest = input("Enter your interest (fitness/technology/reading): ")

product, reason = recommend_product(interest)

print("Recommended Product:", product)
print("Reason:", reason)
```

```
(base) sanjanareddy@Sanjanas-MacBook-Air ~ % /usr/local/bin/python3 /Users/sanjanareddy/recommendation_system.py
Enter your interest (fitness/technology/reading): technology
Recommended Product: Wireless Headphones
Reason: Recommended because you are interested in technology and gadgets.
(base) sanjanareddy@Sanjanas-MacBook-Air ~ %
```

Evaluation of Recommendation Transparency

The recommendation system provides clear and understandable reasons for each suggestion.

Users can easily see why a particular product is recommended.

This improves transparency and trust in AI-generated recommendations.

Explainable recommendations help users make informed decisions.