ASSIGNMENT-5.1

G.Srinidhi

2303A53023

B-46

Lab 5: Ethical Foundations – Responsible AI Coding Practices

Lab Objectives:

- To explore the ethical risks associated with AI-generated code.
- To recognize issues related to security, bias, transparency, and copyright.
- To reflect on the responsibilities of developers when using AI tools in software development.
- To promote awareness of best practices for responsible and ethical AI coding.

Lab Outcomes (LOs):

After completing this lab, students will be able to:

- Identify and avoid insecure coding patterns generated by AI tools.
- Detect and analyze potential bias or discriminatory logic in AI-generated outputs.
- Evaluate originality and licensing concerns in reused AI-generated code.
- Understand the importance of explainability and transparency in AI-assisted programming.
- Reflect on accountability and the human role in ethical AI coding practices.

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

CODE:

```
import os
import requests
from dotenv import load_dotenv
load_dotenv()
def get_weather_data(city):
    api_key=os.getenv('WEATHER_API_KEY')
    if not api_key:
        raise ValueError("API key not found. Please set the WEATHER_API_KEY environment variable.")
    base_url="http://api.openweathermap.org/data/2.5/weather"
    params={'q':city,'appid':api_key,'units':'metric'}
    response=requests.get(base_url,params=params)
    if response.status_code==200:
        return response.json()
    else:
```

```
    response.raise_for_status()
if    name      == "  main":
    city = input("Enter city name: ")
    try:
        weather_data=get_weather_data(city)
        print(f"Weather in {city}: {weather_data['weather'][0]
                ['description']}, Temperature:{weather_data['main']
                        ['temp']}°C")
     except Exception as e:
         print(f"Error fetching weather data: {e}")
```

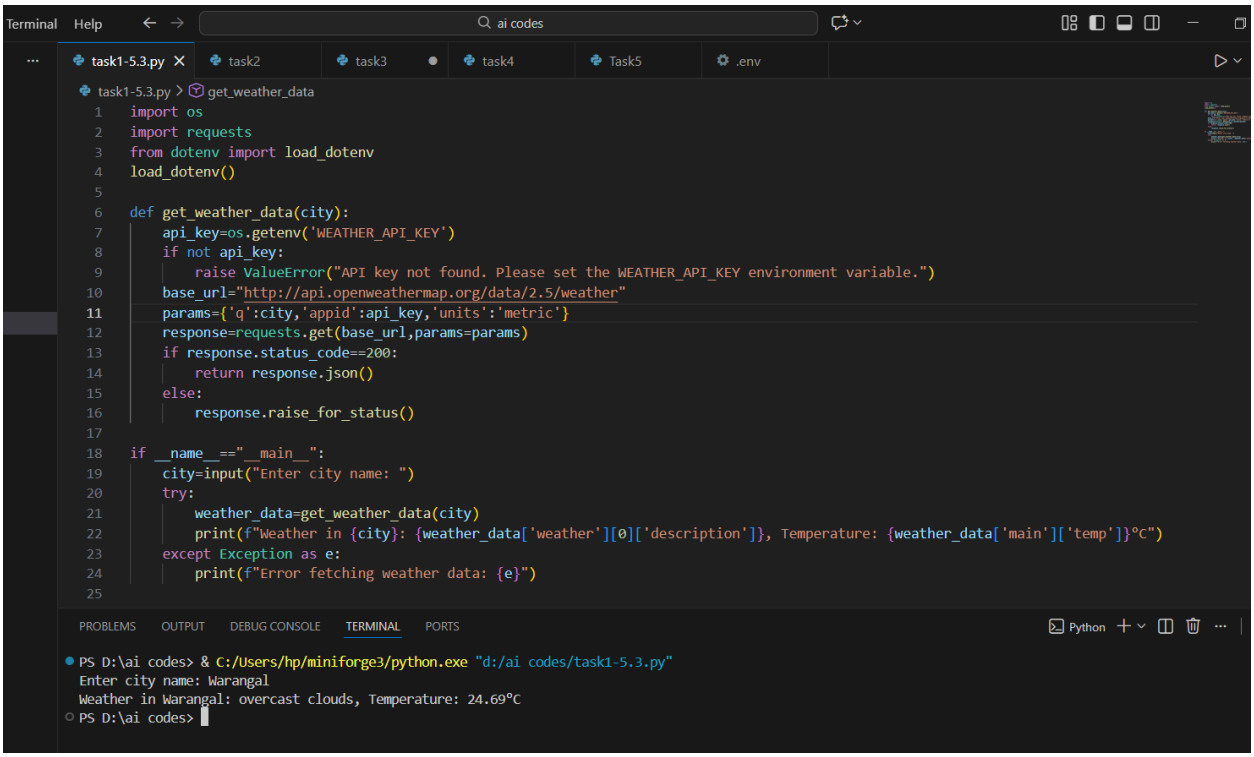
OUTPUT:

Enter city name: warangal

Weather in warangal: clear sky, Temperature: 24.69C

Expected Output:

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



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:

dentified privacy risks.

Revised version with encrypted password storage (e.g., hashing).

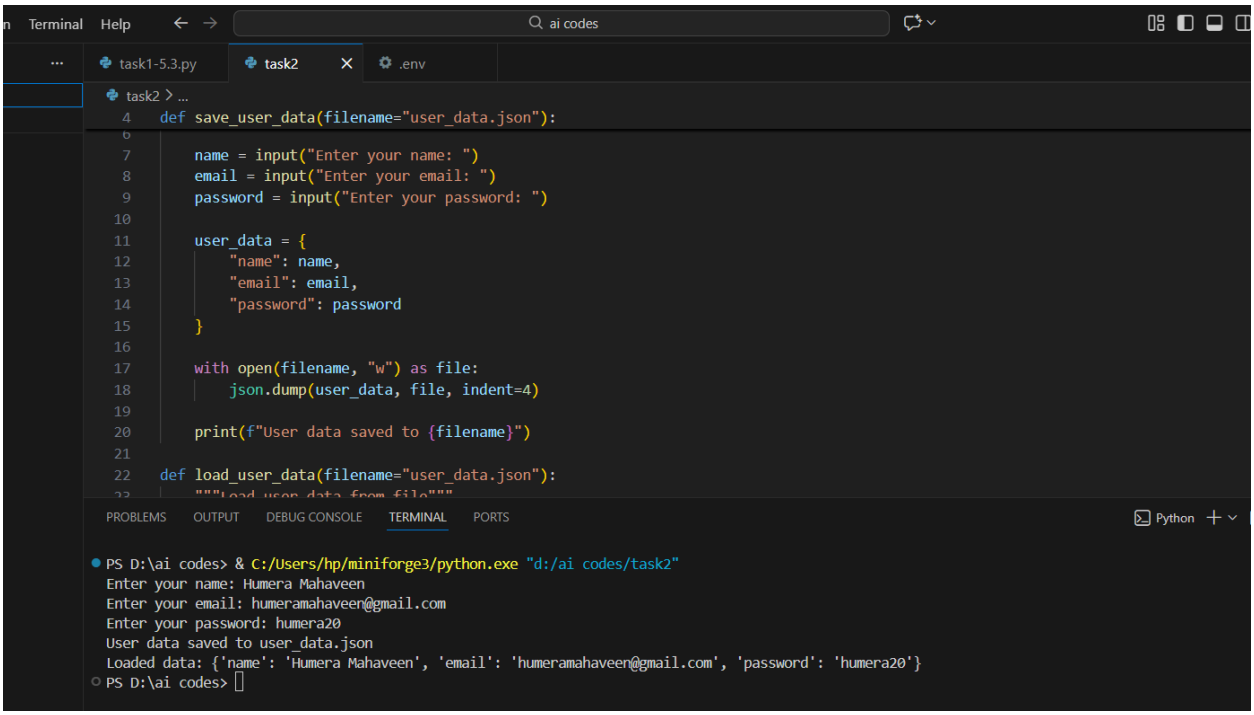
PROMPT:

#Generate a Python script to store user name, email, and password in a file

CODE:

```
import os
import json
def save_user_data(filename="user_data.json"):
    name = input("Enter your name: ")
    email = input("Enter your email: ")
    password = input("Enter your password: ")
    user_data = {
        "name": name,
        "email": email,  “password": password}
 with open(filename, "w") as file:
        json.dump(user_data, file, indent=4)
print(f"User data saved to {filename}")
def load_user_data(filename="user_data.json"):
 if not os.path.exists(filename):
        print(f"File {filename} not found")
        return None
 with open(filename, "r") as file:
```

```
        user_data = json.load(file)
    return user_data
if __name__ == "__main__":
    save_user_data()
    data = load_user_data()
    print("Loaded data:", data)
```



Task Description #3 (Transparency in Algorithm Design)

Objective: Use AI to generate an Armstrong number checking function with comments and explanations.

Instructions:

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

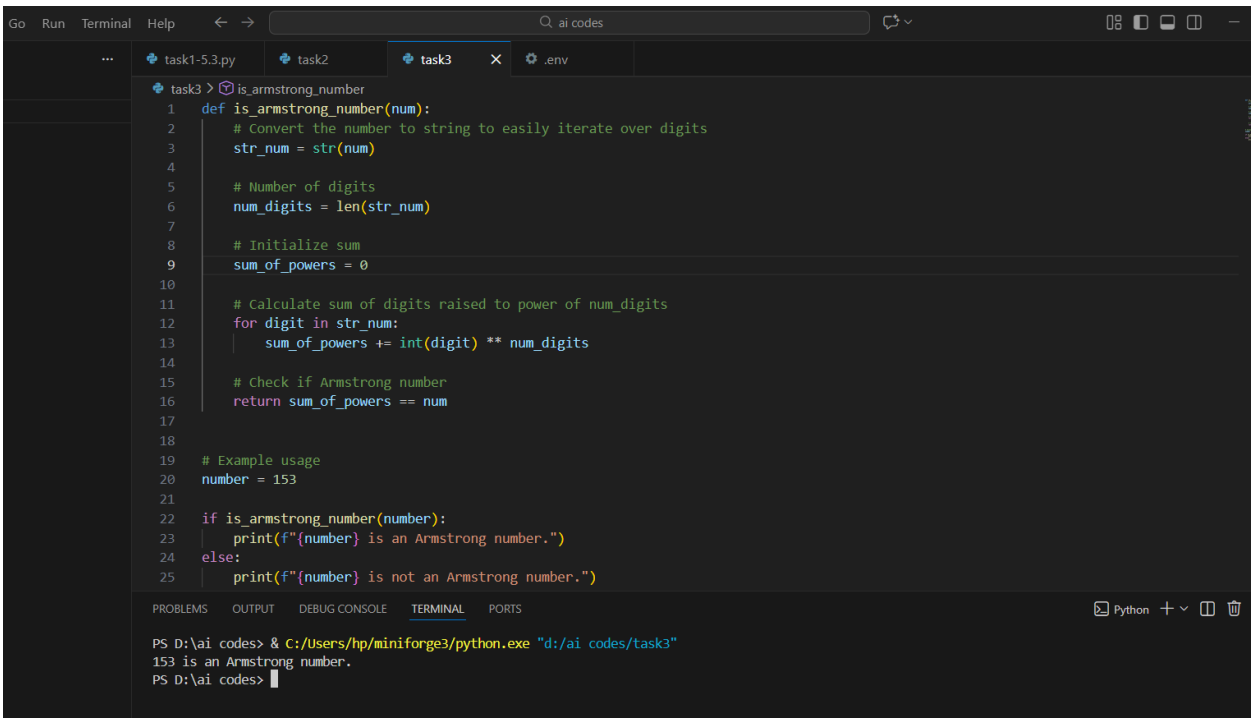
Expected Output:

Transparent, commented code.

Correct, easy-to-understand explanation.

PROMPT:

#Generate a Python function to check if a number is an Armstrong number and explain the code line by line.



Task Description #4 (Transparency in Algorithm Comparison)

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

Expected Output:

Code for both algorithms.

Transparent, comparative explanation of their logic and efficiency.

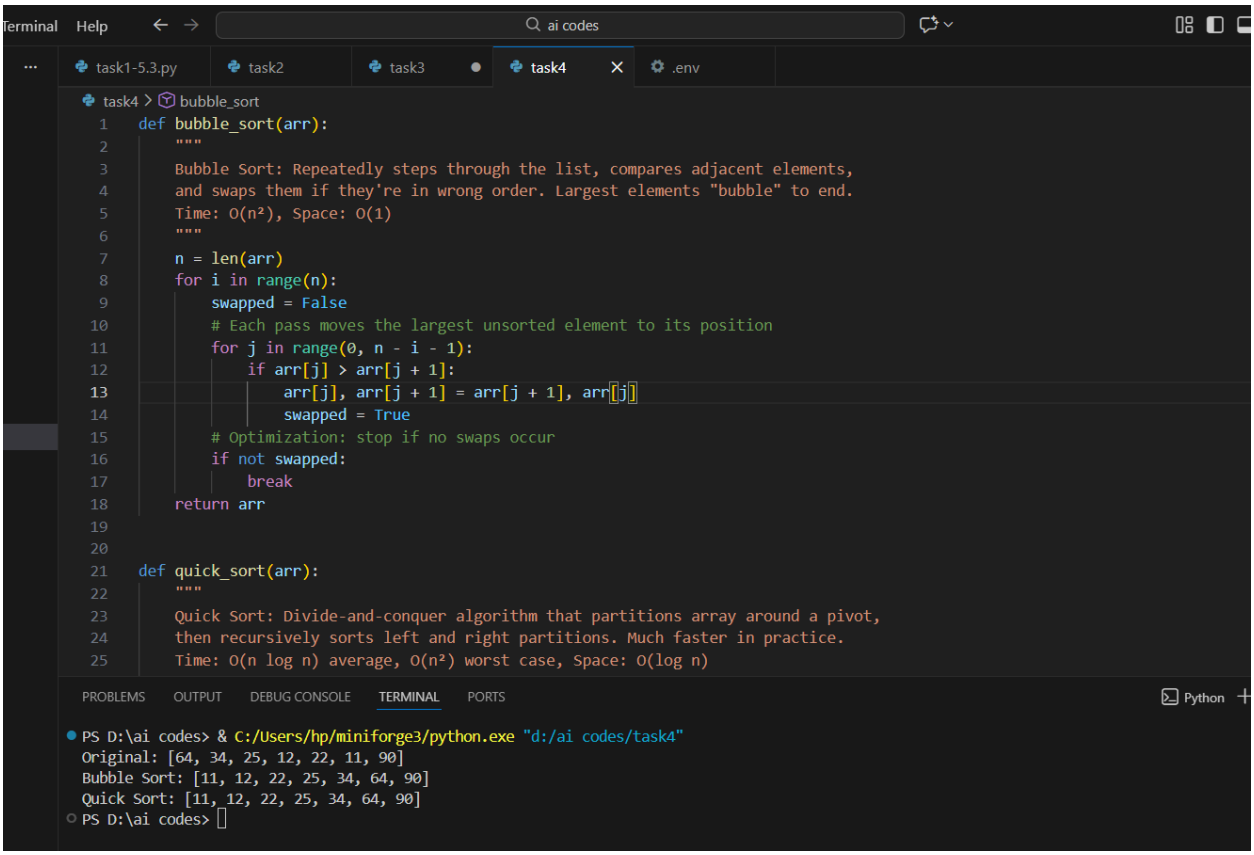
Prompt:

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

CODE:

```
def bubble_sort(arr):
    n=len(arr)
    for i in range(n):
        swapped=False
        for j in range(0,n-i-1):
            if arr[j]>arr[j+1]:
                arr[j],arr[j+1]=arr[j+1],arr[j]
                swapped=True
        if not swapped:
            Break
return arr
def quick_sort(arr):
    if len(arr)<=1:
        return arr
     pivot=arr[len(arr)//2] l
```

```
    left=[x for x in arr if x<pivot]
    middle=[x for x in arr if x==pivot]
    right=[x for x in arr if x>pivot]
     return quick_sort(left)+middle+quick_sort(right) i
if name=="main":
test_arr=[64,34,25,12,22,11,90]
   print("Original:",test_arr)
   print("Bubble Sort:",bubble_sort(test_arr.copy()))
   print("Quick Sort:",quick_sort(test_arr.copy()))
```



Task Description #5 (Transparency in AI Recommendations)

Task: Use AI to create a product recommendation system.
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:

import random

def recommend_items(user_preferences,items):

```
  recommendations=[]
  for item in items:
    score=0
    reasons=[]
    for preference in user_preferences:
      if preference in item['tags']:
        score+=1
        reasons.append(f"Matches your preference for {preference}.")
    if score>0:

recommendations.append({'item':item['name'],'score':score,'reasons':reasons})
  recommendations.sort(key=lambda x:x['score'],reverse=True)
  return recommendations
user_preferences=['action','comedy','sci-fi']
items=[
  {'name':'Movie A','tags':['action','thriller']},
  {'name':'Movie B','tags':['comedy','romance']},
  {'name':'Movie C','tags':['sci-fi','adventure']},
  {'name':'Movie D','tags':['drama','biography']}
]
```

recommendations=recommend_items(user_preferences,items)

for rec in recommendations:

print(f"Recommended: {rec['item']} (Score: {rec['score']})")

for reason in rec['reasons']:

print(f" - {reason}")