

LAB ASSIGNMENT – 5.1

R. Akshay (2303A51933)

BATCH 27

Task 1: Privacy in API Usage

Aim

To generate AI-assisted Python code for fetching weather data securely without exposing API keys.

AI Generated Code

```
import requests

api_key = "ABC123XYZ"

url = "https://api.weatherapi.com/data"

response = requests.get(url + "?key=" + api_key)

print(response.text)
```

Issues Identified

API key is hardcoded in the source code.
Sensitive credentials are exposed, leading to security risks.

Corrected Code

```
import os

import requests

api_key = os.getenv("WEATHER_API_KEY")

url = "https://api.weatherapi.com/data"

response = requests.get(url + "?key=" + api_key)

print(response.text)
```

Result

The API key is securely accessed using environment variables, preventing exposure of sensitive data.

Task 2: Privacy & Security in File Handling

Aim

To analyze AI-generated code that stores user data and identify privacy risks.

AI Generated Code

```
name = input("Enter name: ")
email = input("Enter email: ")
password = input("Enter password: ")
with open("users.txt", "a") as f:
    f.write(name + " " + email + " " +
password + "\n")
```

Issues Identified

Password is stored in plain text.
High risk of data breach.

Corrected Code

```
import hashlib

password = input("Enter password: ")

hashed_password = hashlib.sha256(password.encode()).hexdigest()

with open("users.txt", "a") as f:

f.write(hashed_password + "\n")
```

Result

Sensitive data is protected using password hashing, improving data security.

Task 3: Transparency in Algorithm Design

Aim

To generate an Armstrong number program with transparent logic and explanation.

AI Generated Code

```
num = int(input("Enter number: "))

temp = num
sum = 0
digits = len(str(num))

while temp > 0:

    digit = temp % 10

    sum += digit ** digits

    temp //= 10

if sum == num:
    print("Armstrong Number")
else:
    print("Not an Armstrong Number")
```

Explanation

The number of digits is calculated.
Each digit is raised to the power of digits and added.
Final sum is compared with original number.

Result

The program correctly identifies Armstrong numbers with clear logic and transparency.

Task 4: Transparency in Algorithm Comparison

Aim

To implement and compare Bubble Sort and Quick Sort algorithms.

Bubble Sort Code

```
def bubble_sort(arr):  
    for i in  
range(len(arr)):  
        for j in range(0,  
len(arr)-i-1):  
            if arr[j] > arr[j+1]:  
                arr[j], arr[j+1]  
= arr[j+1], arr[j]
```

Quick Sort Code

```
def quick_sort(arr):  
    if  
len(arr) <= 1:  
        return arr  
    pivot = arr[0]  
    left = [x for x in arr[1:] if x < pivot]  
    right = [x for x in arr[1:] if x >= pivot]  
    return quick_sort(left) + [pivot] + quick_sort(right)
```

Comparison

Bubble Sort is simple but inefficient.
Quick Sort is faster and more efficient for large data.

Result

Quick Sort outperforms Bubble Sort in time complexity and efficiency.

Task 5: Transparency in AI Recommendations

Aim

To create an explainable recommendation system using AI.

AI Generated Code

```
user_interest = "programming"
```

```
products = {
```

```
    "Laptop": "Recommended because it supports  
coding tasks",
```

```
    "Headphones": "Recommended for online  
learning",
```

```
}
```

```
for item, reason in products.items():  
  
    print(item, "-",  
reason)
```

Evaluation

Recommendations include clear reasons.
Improves transparency and user trust.

Result

The system provides understandable and explainable recommendations.