LAB – 5.1

AI-Assisted Coding

Name – Rounak Raj

Hall-Ticket – 2303A54043

Lab 5: Ethical Foundations – Responsible AI Coding Practices

Week 3 – Monday

Task Description #1: Privacy in API Usage

Objective

To ensure API keys are not exposed in AI-generated code and to follow secure coding practices.

AI-Generated Code (Insecure Version)

```
import requests

API_KEY = "my_secret_api_key"
city = "London"
url = f"https://api.weatherapi.com/v1/current.json?key={API_KEY}&q={city}"

response = requests.get(url)
print(response.json())
```

Privacy Risk Identified

- API key is **hardcoded**, making it visible in source code.
- If shared on GitHub or logs, the key can be stolen and misused.

Secure Version Using Environment Variables

```
import os
import requests

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

url = f"https://api.weatherapi.com/v1/current.json?key={API_KEY}&q={city}"
response = requests.get(url)

if response.status_code == 200:
    print(response.json())
else:
    print("Error fetching weather data")
```

Ethical Reflection

Using environment variables protects sensitive credentials and demonstrates responsible handling of user and service provider data.

Task Description #2: Privacy & Security in File Handling

AI-Generated Code (Insecure Version)

```
name = input("Enter name: ")
email = input("Enter email: ")
password = input("Enter password: ")

with open("users.txt", "a") as file:
    file.write(f"{name}, {email}, {password}\n")
```

Privacy Risks Identified

- Passwords stored in **plain text**
- Vulnerable to data breaches
- Violates basic data protection principles

Revised Secure Version (Password Hashing)

```
import hashlib
```

```
name = input("Enter name: ")
email = input("Enter email: ")
password = input("Enter password: ")

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

with open("users.txt", "a") as file:
    file.write(f"{name}, {email}, {hashed_password}\n")
```

Ethical Reflection

Passwords must never be stored in plain text. Hashing ensures user privacy and aligns with responsible AI-assisted coding standards.

Task Description #3: Transparency in Algorithm Design

Armstrong Number Checking Function (With Explanation)

```
def is_armstrong(number):
    """
    Checks whether a number is an Armstrong number.
    """
    digits = str(number)           # Convert number to string
    power = len(digits)            # Number of digits
    total = 0

    for digit in digits:
        total += int(digit) ** power  # Raise digit to power and add

    return total == number
```

Line-by-Line Explanation

- Converts the number to digits
- Counts the number of digits
- Raises each digit to the power of digit count
- Compares the sum with the original number

Evaluation

The explanation matches the functionality and improves transparency, making the logic easy to understand.

Task Description #4: Transparency in Algorithm Comparison

Bubble Sort Implementation

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

    pivot = arr[len(arr) // 2]
    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)
```

Comparative Explanation

Feature	Bubble Sort	Quick Sort
Approach	Repeated swapping	Divide-and-conquer
Time Complexity	$O(n^2)$	O(n log n) average
Efficiency	Slow	Fast
Use Case	Small datasets	Large datasets

Ethical Reflection

Transparent explanations help developers choose algorithms responsibly based on efficiency and context.

Task Description #5: Transparency in AI Recommendations

Explainable Recommendation System

```
def recommend_products(user_interest, products):
    recommendations = []

    for product, category in products.items():
        if category == user_interest:
            recommendations.append(
                f"Recommended {product} because it matches your interest in
{category}."
            )

    return recommendations
```

Sample Input

```
products = {
    "Python Course": "Programming",
    "Data Science Bootcamp": "Programming",
    "Graphic Design Masterclass": "Design"
}

print(recommend_products("Programming", products))
```

Output Explanation

- Recommendations include **clear reasons**
- Users understand *why* items were suggested

Evaluation

The explanations are simple, transparent, and help build user trust in AI-driven recommendations.

Overall Ethical Reflection

This lab demonstrates that AI-generated code must always be:

- **Reviewed by humans**
- **Secured against privacy risks**
- **Transparent and explainable**
- **Ethically accountable**

AI assists development, but responsibility always lies with the developer