AI ASSISTED CODING

LAB 5.5: Ethical Foundations – Responsible AI Coding Practices

LAB ASSIGNMENT 5.5

T.SAI SATHWIK

2303A51221

B04

QUESTION 1:

Task (Transparency in Algorithm Optimization)

Use AI to generate **two solutions** for checking prime numbers:

- Naive approach (basic)
- Optimized approach

PROMPT

Generate Python code for two prime-checking methods and explain how the optimized version improves performance.

CODE

```
# Naive approach to check prime number
def is_prime_naive(n):
    if n <= 1:
        return False
    for i in range(2, n):
        if n % i == 0:
            return False
    return True

# Optimized approach to check prime number

import math
def is_prime_optimized(n):
    if n <= 1:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False

    for i in range(3, int(math.sqrt(n)) + 1, 2):
        if n % i == 0:
            return False
    return True
num = 29
print("Naive Method:", is_prime_naive(num))
print("Optimized Method:", is_prime_optimized(num))
```

OUTPUT

```
Naive Method: True
Optimized Method: True
PS D:\3-2 SEM\AI ASSISTED CODING - 1221>
```

EXPLANATION

The naive method checks all numbers from 2 to n−1, resulting in O(n) time complexity.
The optimized method checks divisors only up to √n and skips even numbers, reducing complexity to O(√n).

This improves efficiency while keeping the logic transparent and understandable, supporting ethical and explainable AI coding

QUESTION 2

Task (Transparency in Recursive Algorithms)

Use AI to generate a recursive function to calculate Fibonacci numbers with clear explanations.

PROMPT

Generate a recursive Fibonacci function with clear comments explaining recursion, base cases, and recursive calls.

CODE

```python
def fibonacci(n):
    # Base case: if n is 0 or 1, return n
    if n <= 1:
        return n
    # Recursive call
    return fibonacci(n - 1) + fibonacci(n - 2)

num = 12
print("Fibonacci of", num, "is:", fibonacci(num))
```

OUTPUT

```
PS D:\3-2 SEM\AI ASSISTED CODING - 1221>
Fibonacci of 12 is: 144
PS D:\3-2 SEM\AI ASSISTED CODING - 1221>
```

EXPLANATION

Recursion works by calling the function within itself.

- Base cases stop infinite recursion (n = 0 or 1)
- Recursive calls break the problem into smaller subproblems

Each call waits for the previous calls to return values, ensuring correct execution and transparency.

QUESTION 3

Task (Transparency in Error Handling)

Generate a Python program that reads a file and processes data with proper error handling.

PROMPT

Generate code with proper error handling and clear explanations for each exception.

CODE

```
try:
    file = open("data.txt", "r")
    content = file.read()
    print("File Content:")
    print(content)
    file.close()

except FileNotFoundError:
    print("Error: File not found.")

except PermissionError:
    print("Error: Permission denied while accessing the file.")

except Exception as e:
    print("Unexpected error:", e)
```

OUTPUT

```
PS D:\3-2 SEM\AI ASSISTED CODING - 1221> python
Error: File not found.
PS D:\3-2 SEM\AI ASSISTED CODING - 1221>
```

EXPLANATION

The program uses:

- try block for risky code
- except blocks to handle specific errors
- A general exception handler for unexpected issues

This ensures reliable execution and makes error behavior transparent and understandable.

QUESTION 4

Task (Security in User Authentication)

Generate a Python-based login system and analyze its security.

PROMPT

Generate a Python login system and improve it using secure password handling.

CODE

```python
import hashlib

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

stored_username = "admin"
stored_password = hash_password("Admin@123")

username = input("Enter username: ")
password = input("Enter password: ")

if username == stored_username and hash_password(password) == stored_password:
    print("Login successful")
else:
    print("Invalid credentials")
```

OUTPUT

```
PS D:\3-2 SEM\AI ASSISTED CODING - 1221> python -u
Enter username: SAI SATHWIK
Enter password: 122203
Invalid credentials
PS D:\3-2 SEM\AI ASSISTED CODING - 1221> python -u
Enter username: admin
Enter password: Admin@123
Login successful
PS D:\3-2 SEM\AI ASSISTED CODING - 1221>
```

EXPLANATION

Plain-text passwords are insecure.
This program improves security by:

- Hashing passwords using SHA-256
- Comparing hashed values instead of raw passwords
- Validating user input

This follows ethical AI practices for secure authentication.

QUESTION 5

Task (Privacy in Data Logging)

Generate a Python script that logs user activity while protecting privacy.

PROMPT

Generate a logging script and ensure privacy-aware data handling.

CODE

```python
import logging

logging.basicConfig(
    filename="activity.log",
    level=logging.INFO,
    format="%(asctime)s - User logged in"
)

def log_user_activity():
    logging.info("User activity recorded")

log_user_activity()
print("Activity logged securely")
```

OUTPUT

```
PS D:\3-2 SEM\AI ASSISTED CODING - 1221>
Activity logged securely
PS D:\3-2 SEM\AI ASSISTED CODING - 1221>
```

EXPLANATION

The program avoids storing:

- **IP addresses**
- **Personal identifiers**

Only minimal required information is logged.
This reduces privacy risks and follows responsible AI data-handling principles.