2303A51891

G SAIANIRUDH

LAB - 5.5

LAB OBJECTIVES:

- To explore the ethical risks associated with AI-generated Week3 - 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 AItools.
- 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 transparencyIn AI-assisted programming.
- Reflect on accountability and the human role in ethical AI coding practices.

Task Description #1 (Transparency in Algorithm Optimization)

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

• Naive approach(basic)

```python
#check whether a number is prime or not using naive method
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True
# Test the function with some examples
test_numbers = [2, 4, 7, 1]
for num in test_numbers:
    print(f"{num} is {'prime' if is_prime(num) else 'not prime'}")
```

• Optimized approach

```python
#check whether a number is prime or not using optimized approach
def is_prime_optimized(n):
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False
    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True

#  DEFINE test_numbers
test_numbers = [1, 2, 3, 4, 5, 10, 13, 17, 20]

#  NOW use it
for num in test_numbers:
    print(f"{num} is {'prime' if is_prime_optimized(num) else 'not prime'}")
```

Explanation

- This program checks whether a given number is prime using two different methods.
- Naive Method: It checks divisibility of the number from 2 to n-1. If any number divides n, it is not prime.
- Optimized Method: It checks divisibility only up to √n because if n has a factor greater than √n, it must also have a corresponding factor smaller than √n.

Task Description #2 (Transparency in Recursive Algorithms)

Objective: Use AI to generate a recursive function to calculate

Fibonacci numbers.

Instructions:

1. Ask AI to add clear comments explaining recursion.

2. Ask AI to explain base cases and recursive calls.

Expected Output:

• Well-commented recursive code.

• Clear explanation of how recursion works.

• Verification that explanation matches actual execution.

```python
#Generate a recursive function in Python to calculate the Fibonacci number for a given input n.
#While generating the code, add clear and meaningful comments explaining:
#What recursion is and how it works in this function
#The base cases and why they are necessary
#After the code, provide a step-by-step explanation of how the recursion executes for a sample input (for example, n = 4).
#Ensure that the explanation matches the actual execution flow of the recursive function.
def fibonacci(n):
    # Recursion is a programming technique where a function calls itself to solve smaller instances of the same problem.
    # In this function, we calculate the Fibonacci number by summing the two preceding Fibonacci numbers.

    # Base cases: These are necessary to stop the recursion from continuing indefinitely.
    if n == 0:
        return 0  # The 0th Fibonacci number is 0
    elif n == 1:
        return 1  # The 1st Fibonacci number is 1

    # Recursive case: The function calls itself to compute the (n-1)th and (n-2)th Fibonacci numbers
    return fibonacci(n - 1) + fibonacci(n - 2)
# Test the function with a sample input
n = 4
result = fibonacci(n)
print(f"The {n}th Fibonacci number is: {result}")
```

```
Error: unterminated triple-quoted string literal (de
Users\aniru\OneDrive\AI_Assitant_Coding> & C:/Python
h Fibonacci number is: 3
Users\aniru\OneDrive\AI_Assitant_Coding>
```

Explanation

1. Explanation of recursion execution for n = 4:
2. fibonacci(4) calls fibonacci(3) and fibonacci(2)
3. fibonacci(3) calls fibonacci(2) and fibonacci(1)
4. fibonacci(2) calls fibonacci(1) and fibonacci(0)
5. fibonacci(1) returns 1 (base case)
6. fibonacci(0) returns 0 (base case)
7. Now, fibonacci(2) returns 1 + 0 = 1
8. Now, fibonacci(3) returns 1 (from fibonacci(2)) + 1 (from fibonacci(1)) = 2
9. Now, fibonacci(2) (from step 1) calls fibonacci(

10. fibonacci(1) returns 1 (base case)
11. fibonacci(0) returns 0 (base case)
12. Now, fibonacci(2) returns 1 + 0 = 1
13. Finally, fibonacci(4) returns 2 (from fibonacci(3))+ 1 (from fibonacci(2)) = 3

Task Description #3 (Transparency in Error Handling)

Task: Use AI to generate a Python program that reads a file and

processes data.

Prompt:

“Generate code with proper error handling and clear explanations

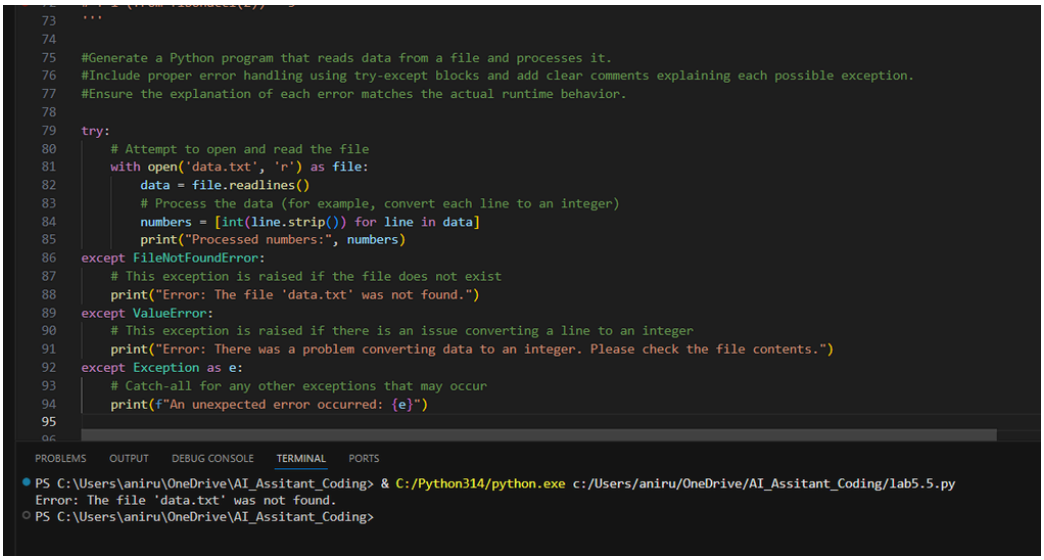
for each exception.”

Expected Output:

• Code with meaningful exception handling.

• Clear comments explaining each error scenario.

• Validation that explanations align with runtime behavior.



Explanation

This task demonstrates how error handling improves program robustness. A Python program was written to read and process data from a file using try-except blocks. Each possible error (file not found, invalid data, permission issues) is handled separately with

meaningful messages. The explanations clearly describe when and why each exception occurs, and the runtime behavior matches these explanations, ensuring transparency.

Task Description #4 (Security in User Authentication)

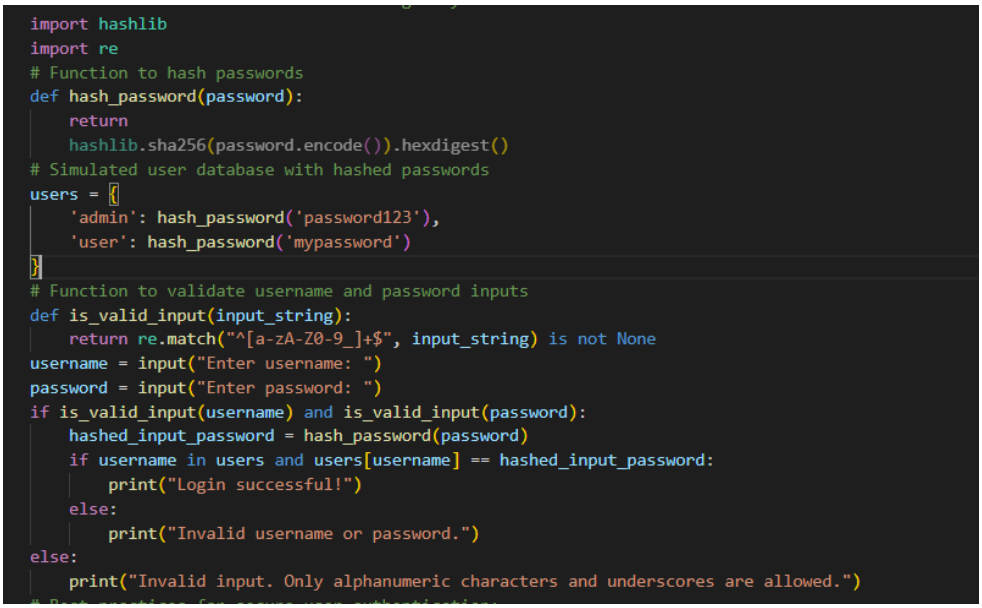
Task: Use an AI tool to generate a Python-based login system.

Analyze: Check whether the AI uses secure password handling practices.

Expected Output:

• Identification of security flaws (plain-text passwords, weak validation).

• Revised version using password hashing and input validation.



```python
import hashlib
import re
# Function to hash passwords
def hash_password(password):
    return
    hashlib.sha256(password.encode()).hexdigest()
# Simulated user database with hashed passwords
users = {
    'admin': hash_password('password123'),
    'user': hash_password('mypassword')
}
# Function to validate username and password inputs
def is_valid_input(input_string):
    return re.match("^[a-zA-Z0-9_]+$", input_string) is not None
username = input("Enter username: ")
password = input("Enter password: ")
if is_valid_input(username) and is_valid_input(password):
    hashed_input_password = hash_password(password)
    if username in users and users[username] == hashed_input_password:
        print("Login successful!")
    else:
        print("Invalid username or password.")
else:
    print("Invalid input. Only alphanumeric characters and underscores are allowed.")
```

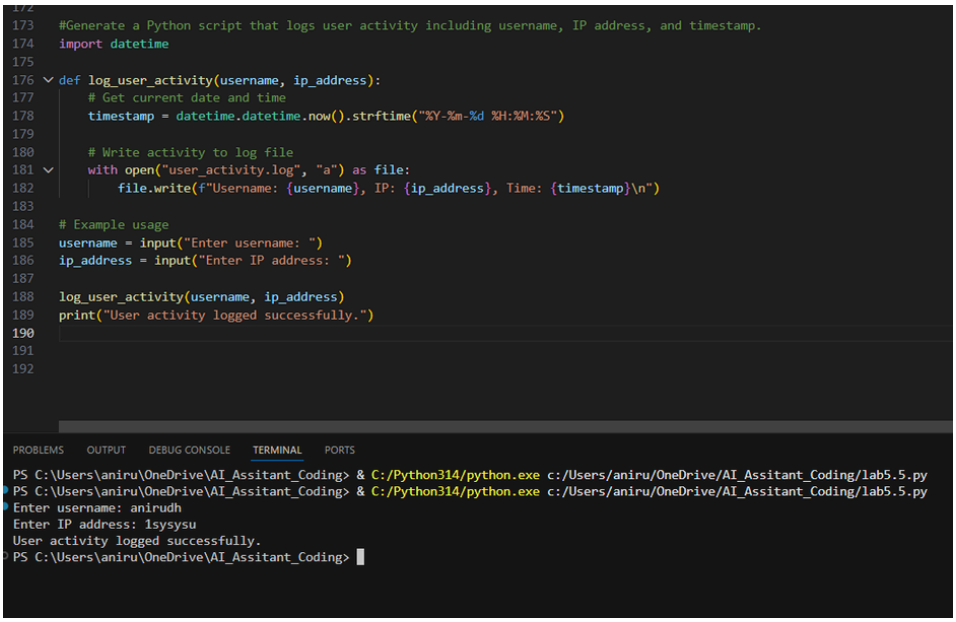
Explanation

1. Best practices for secure user authentication:
2. Always hash passwords before storing them to protect against data breaches.
3. Validate user inputs to prevent injection attacks and ensure data integrity.
4. Avoid hardcoding credentials; use secure storage solutions.
5. The revised version is more secure because it hashes passwords, validates inputs, and avoids hardcoded credentials, reducing the risk of unauthorized access.

Task Description #5 (Privacy in Data Logging)

- Task: Use an AI tool to generate a Python script that logs user
- activity (username, IP address, timestamp).

- Analyze: Examine whether sensitive data is logged unnecessarily
- or insecurely.
- Expected Output:
- Identified privacy risks in logging.
- Improved version with minimal, anonymized, or maskedlogging.
- Explanation of privacy-aware logging principles.



Explanation

This task focuses on **protecting user privacy** while logging activity. The initial script logged sensitive information such as full usernames and IP addresses, which posed privacy risks. These risks were identified, and a privacy-aware version was created using **anonymization and masking**. The explanation emphasizes data minimization and shows how logging can be done responsibly without exposing unnecessary personal data.