Assignment 5.5

Name – Varsha Sri Chukka

Roll- 2303A52494

Batch – 50

Lab 5: Ethical Foundations – Responsible AI Coding Practices

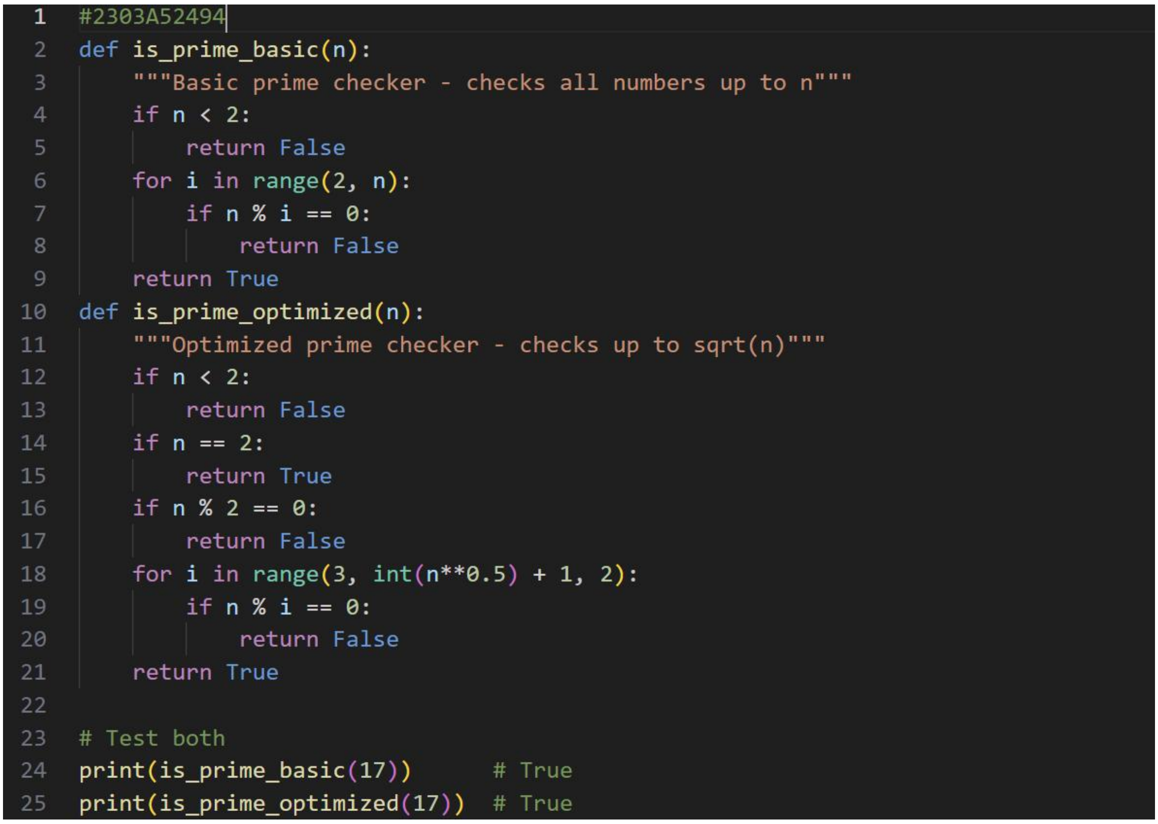
Task Description #1 (Transparency in Algorithm Optimization)

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

- Naive approach(basic)
- Optimized approach

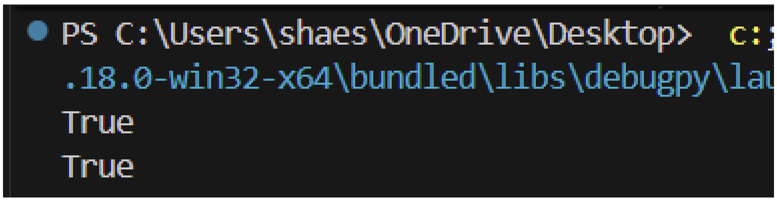
Prompt:

“Generate Python code for two prime-checking methods and explain how the optimized version improves performance.” **CODE:**



```python
#2303A52494
def is_prime_basic(n):
    """Basic prime checker - checks all numbers up to n"""
    if n < 2:
        return False
    for i in range(2, n):
        if n % i == 0:
            return False
    return True
def is_prime_optimized(n):
    """Optimized prime checker - checks up to sqrt(n)"""
    if n < 2:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False
    for i in range(3, int(n**0.5) + 1, 2):
        if n % i == 0:
            return False
    return True

# Test both
print(is_prime_basic(17))      # True
print(is_prime_optimized(17))  # True
```

Output :



```
PS C:\Users\shaes\OneDrive\Desktop>  c:
.18.0-win32-x64\bundled\libs\debugpy\lau
True
True
```

Explanation : The naive method checks all numbers from 2 to n−1, which takes more time for large numbers (O(n)).

The optimized method checks only up to the square root of the number, reducing unnecessary checks (O(√n)).

Because of this reduced range, the optimized approach runs much faster.

Hence, optimization improves performance and efficiency.

Task Description #2 (Transparency in Recursive Algorithms)

Objective: Use AI to generate a recursive function to calculate

Fibonacci numbers.

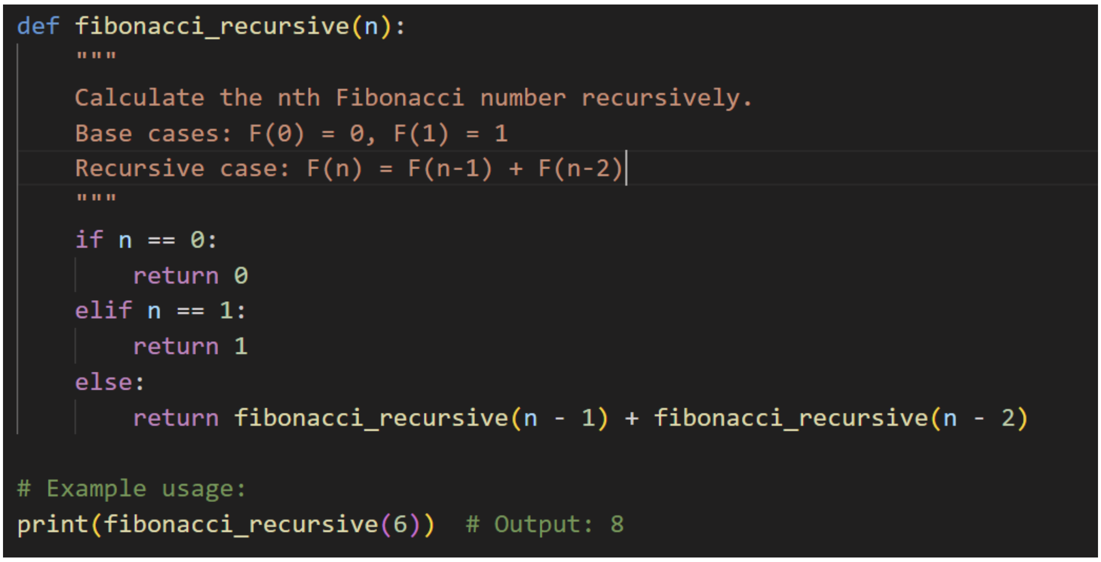
Instructions:

1. Ask AI to add clear comments explaining recursion.

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

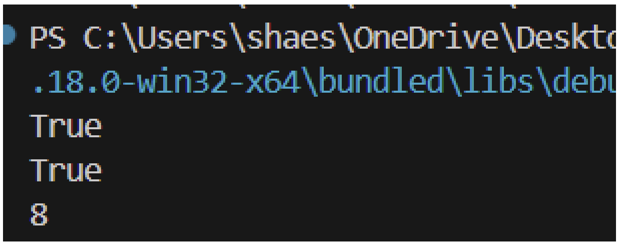
Prompt:

generate a recursive function to calculate Fibonacci numbers. add clear comments explaining recursion. explain base cases and recursive calls.



```python
def fibonacci_recursive(n):
    """
    Calculate the nth Fibonacci number recursively.
    Base cases: F(0) = 0, F(1) = 1
    Recursive case: F(n) = F(n-1) + F(n-2)
    """
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci_recursive(n - 1) + fibonacci_recursive(n - 2)

# Example usage:
print(fibonacci_recursive(6))  # Output: 8
```

OUTPUT :



```
PS C:\Users\shaes\OneDrive\Deskt
.18.0-win32-x64\bundled\libs\deb
True
True
8
```

Example:

Recursion is a technique where a function calls itself to solve a smaller part of the problem.

The base cases (n = 0 and n = 1) stop the recursion and return fixed values.

For other values, the function makes recursive calls to calculate Fibonacci(n−1) and Fibonacci(n−2). This matches the Fibonacci formula and correctly produces the sequence step by step.

Task Description #3 (Transparency in Error Handling)

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

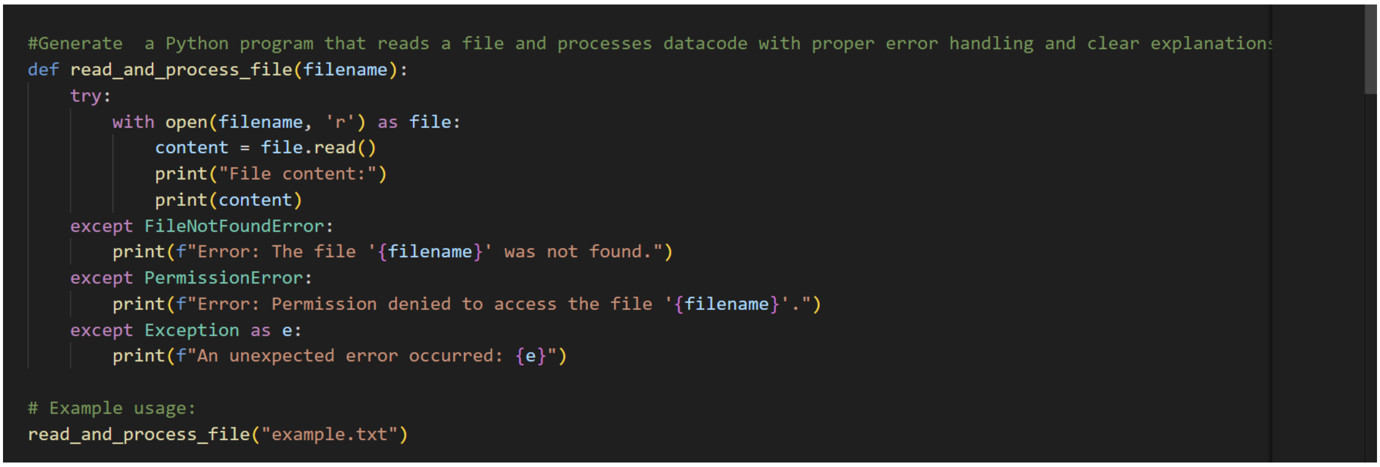
processes data.

Prompt:

"Generate code with proper error handling and clear explanations

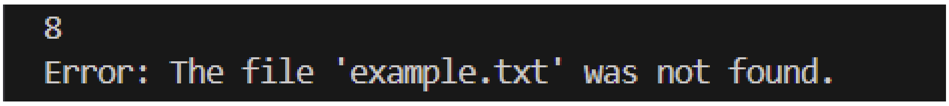
for each exception."

CODE:



```python
#Generate  a Python program that reads a file and processes datacode with proper error handling and clear explanations
def read_and_process_file(filename):
    try:
        with open(filename, 'r') as file:
            content = file.read()
            print("File content:")
            print(content)
    except FileNotFoundError:
        print(f"Error: The file '{filename}' was not found.")
    except PermissionError:
        print(f"Error: Permission denied to access the file '{filename}'.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

# Example usage:
read_and_process_file("example.txt")
```

OUTPUT:



```
8
Error: The file 'example.txt' was not found.
```

Explanation:

The try block contains code that may cause errors while reading a file.
If the file does not exist, FileNotFoundError is raised and handled.
If the program lacks permission, PermissionError is handled.
Any other unexpected issue is caught using a general Exception block.

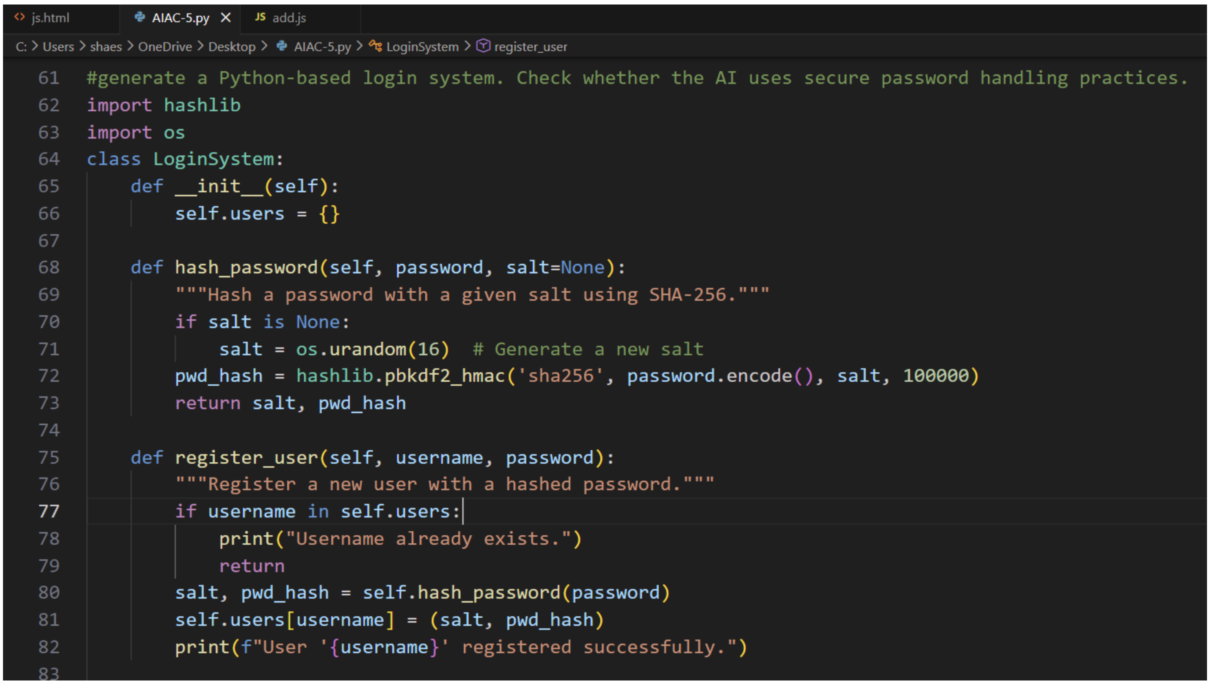
Task Description #4 (Security in User Authentication)

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

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

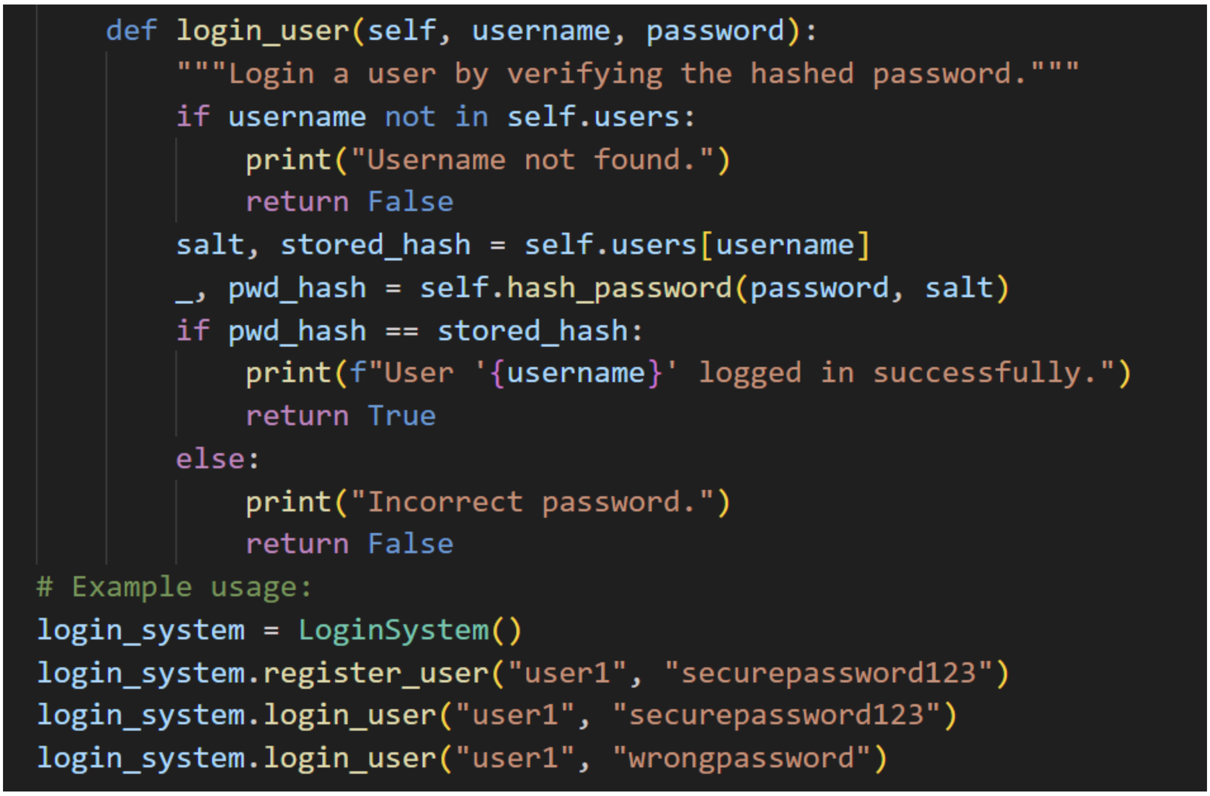
Prompt: generate a Python-based login system. Check whether the AI uses secure password handling practices.

#Code:



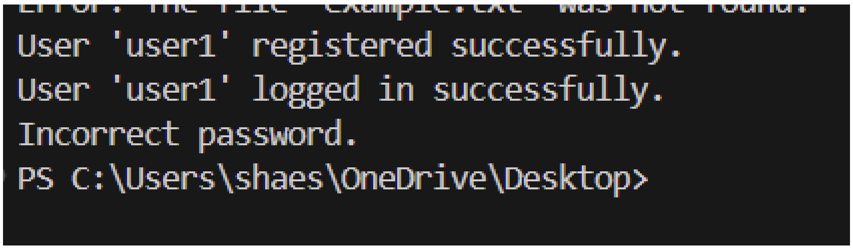
js.html
AIAC-5.py
add.js
C: > Users > shaes > OneDrive > Desktop > AIAC-5.py > LoginSystem > register_user

```python
#generate a Python-based login system. Check whether the AI uses secure password handling practices.
import hashlib
import os
class LoginSystem:
    def __init__(self):
        self.users = {}

    def hash_password(self, password, salt=None):
        """Hash a password with a given salt using SHA-256."""
        if salt is None:
            salt = os.urandom(16)  # Generate a new salt
        pwd_hash = hashlib.pbkdf2_hmac('sha256', password.encode(), salt, 100000)
        return salt, pwd_hash

    def register_user(self, username, password):
        """Register a new user with a hashed password."""
        if username in self.users:
            print("Username already exists.")
            return
        salt, pwd_hash = self.hash_password(password)
        self.users[username] = (salt, pwd_hash)
        print(f"User '{username}' registered successfully.")
```



```
    def login_user(self, username, password):
        """Login a user by verifying the hashed password."""
        if username not in self.users:
            print("Username not found.")
            return False
        salt, stored_hash = self.users[username]
        _, pwd_hash = self.hash_password(password, salt)
        if pwd_hash == stored_hash:
            print(f"User '{username}' logged in successfully.")
            return True
        else:
            print("Incorrect password.")
            return False
# Example usage:
login_system = LoginSystem()
login_system.register_user("user1", "securepassword123")
login_system.login_user("user1", "securepassword123")
login_system.login_user("user1", "wrongpassword")
```

Output:



```
Error: The file 'example.txt' was not found.
User 'user1' registered successfully.
User 'user1' logged in successfully.
Incorrect password.
PS C:\Users\shaes\OneDrive\Desktop>
```

Explanation:

Basic login systems store passwords in plain text, which is insecure.

The secure version hashes passwords using SHA-256, so real passwords are never stored.

Input validation prevents empty or invalid login attempts.

This improves security and protects user credentials.

Task Description #5 (Privacy in Data Logging)

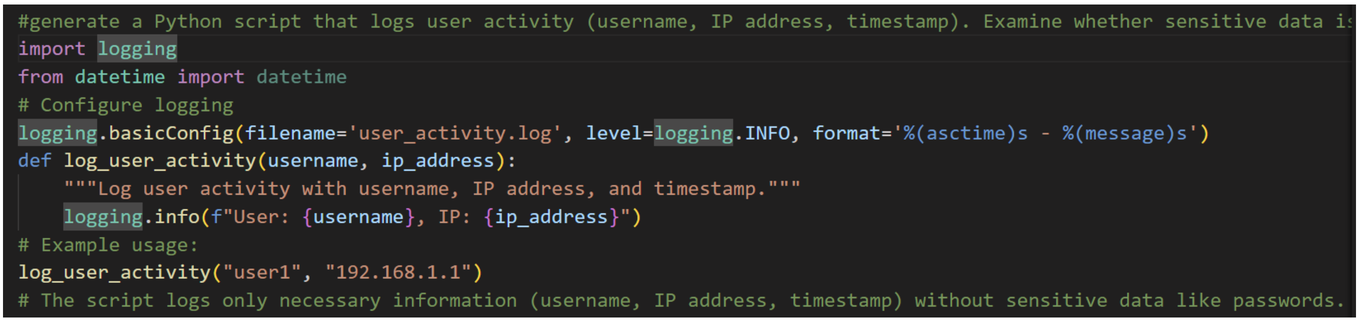
Task: Use an AI tool to generate a Python script that logs user

activity (username, IP address, timestamp).

Analyze: Examine whether sensitive data is logged unnecessarily or insecurely.

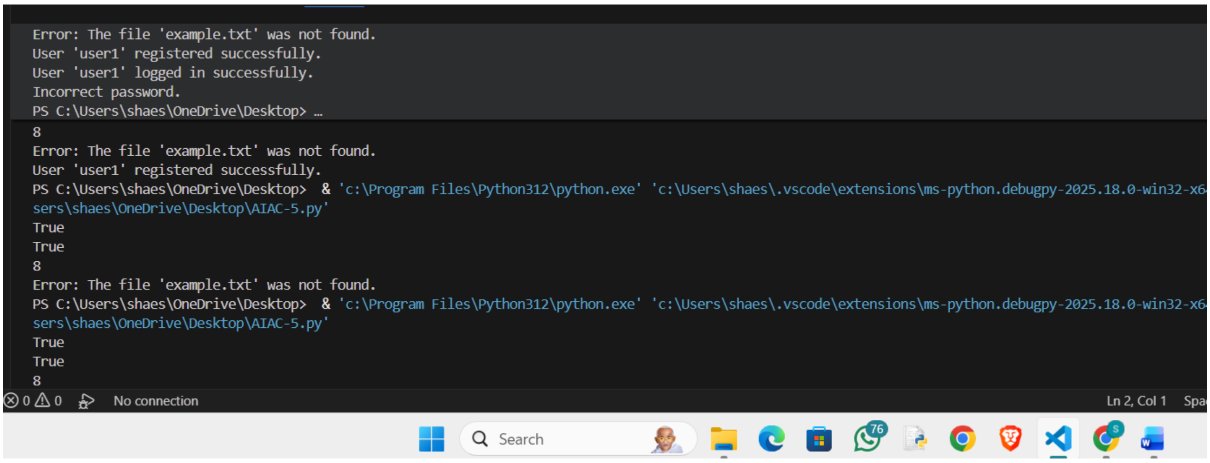
Prompt : generate a Python script that logs user activity (username, IP address, timestamp). Examine whether sensitive data is logged unnecessarily or insecurely.

CODE :



```
#generate a Python script that logs user activity (username, IP address, timestamp). Examine whether sensitive data is
import logging
from datetime import datetime
# Configure logging
logging.basicConfig(filename='user_activity.log', level=logging.INFO, format='%(asctime)s - %(message)s')
def log_user_activity(username, ip_address):
    """Log user activity with username, IP address, and timestamp."""
    logging.info(f"User: {username}, IP: {ip_address}")
# Example usage:
log_user_activity("user1", "192.168.1.1")
# The script logs only necessary information (username, IP address, timestamp) without sensitive data like passwords.
```

OUTPUT:



```
Error: The file 'example.txt' was not found.
User 'user1' registered successfully.
User 'user1' logged in successfully.
Incorrect password.
PS C:\Users\shaes\OneDrive\Desktop> …
8
Error: The file 'example.txt' was not found.
User 'user1' registered successfully.
PS C:\Users\shaes\OneDrive\Desktop>  & 'c:\Program Files\Python312\python.exe' 'c:\Users\shaes\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x6
sers\shaes\OneDrive\Desktop\AIAC-5.py'
True
True
8
Error: The file 'example.txt' was not found.
PS C:\Users\shaes\OneDrive\Desktop>  & 'c:\Program Files\Python312\python.exe' 'c:\Users\shaes\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x6
sers\shaes\OneDrive\Desktop\AIAC-5.py'
True
True
8
```

Explanation:

Logging full personal data can violate user privacy.

Privacy-aware logging records only necessary information and masks sensitive data like IP addresses.

This reduces the risk of data misuse while still allowing activity tracking.