AI ASSISTED CODING

ASSIGNMENT-5.5

NAME:D.PAVAN

ROLL NO: 2403A52L07

BATCH-50

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.”

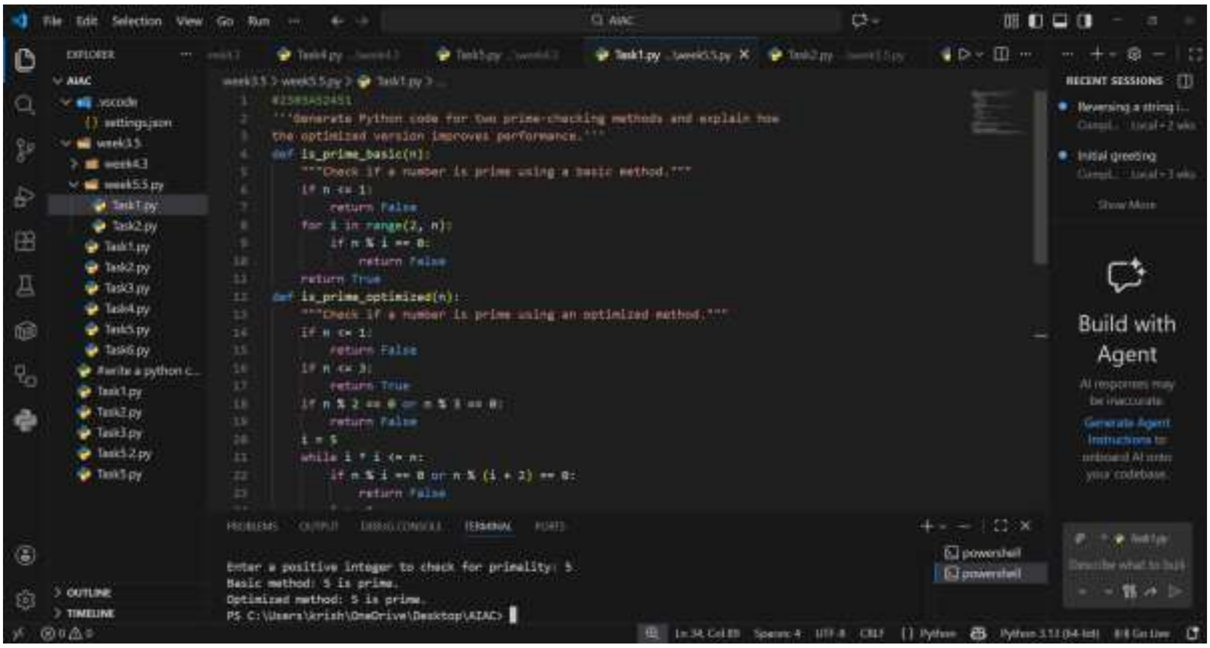
Expected Output:

- **Code for both methods.**
- **Transparent explanation of time complexity.**
- **Comparison highlighting efficiency improvements.**

PROMPT:

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

OUTPUT:



EXPLANATION:

The naive prime-checking method tests divisibility from 2 up to n-1, which is inefficient with time complexity O(n). The optimized method reduces checks by stopping at squareroot of n and skipping even numbers, improving efficiency to O(squareroot(n).This comparison highlights how algorithmic optimization saves computation time.

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:

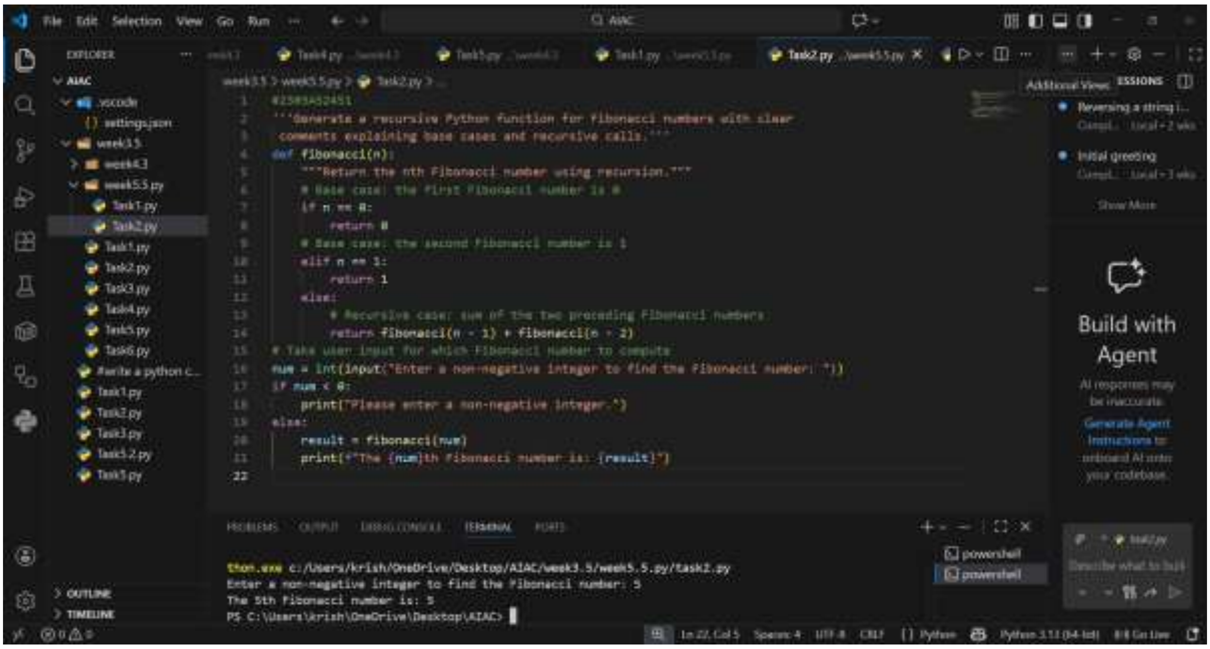
- **Well-commented recursive code.**
- **Clear explanation of how recursion works.**

- **Verification that explanation matches actual execution.**

PROMPT:

"Generate a recursive Python function for Fibonacci numbers with clear comments explaining base cases and recursive calls."

OUTPUT:



EXPLANATION:

Recursion works by breaking a problem into smaller subproblems until reaching base cases. The base cases f(0)=0 and f(1)=1 stop infinite calls, while recursive calls build the solution step by step. Clear comments make the recursive flow transparent and easier to trace.

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.”

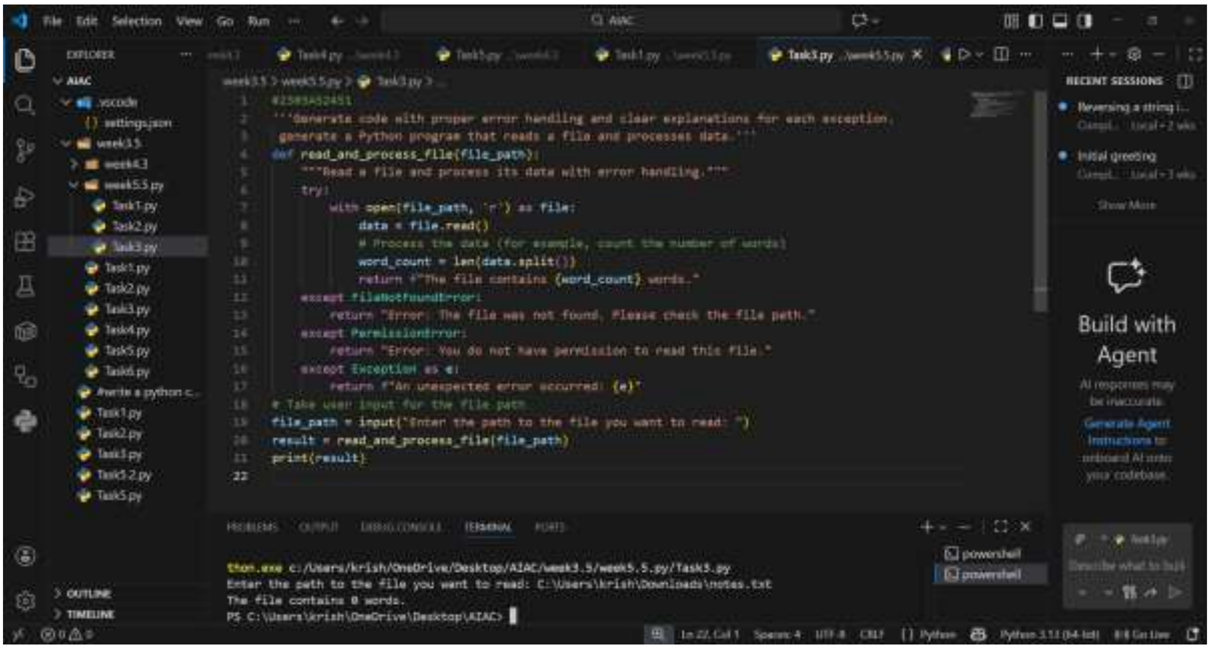
Expected Output:

- **Code with meaningful exception handling.**
- **Clear comments explaining each error scenario.**
- **Validation that explanations align with runtime behavior.**

PROMPT:

“Generate code with proper error handling and clear explanations for each exception. Generate Python program that reads a file and processes data.”

OUTPUT:



EXPLANATION:

The program uses try-except blocks to handle errors such as missing files, permission issues, or invalid paths. Each exception is explained clearly, ensuring the program does not crash unexpectedly. Transparent error handling improves reliability and user experience.

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.

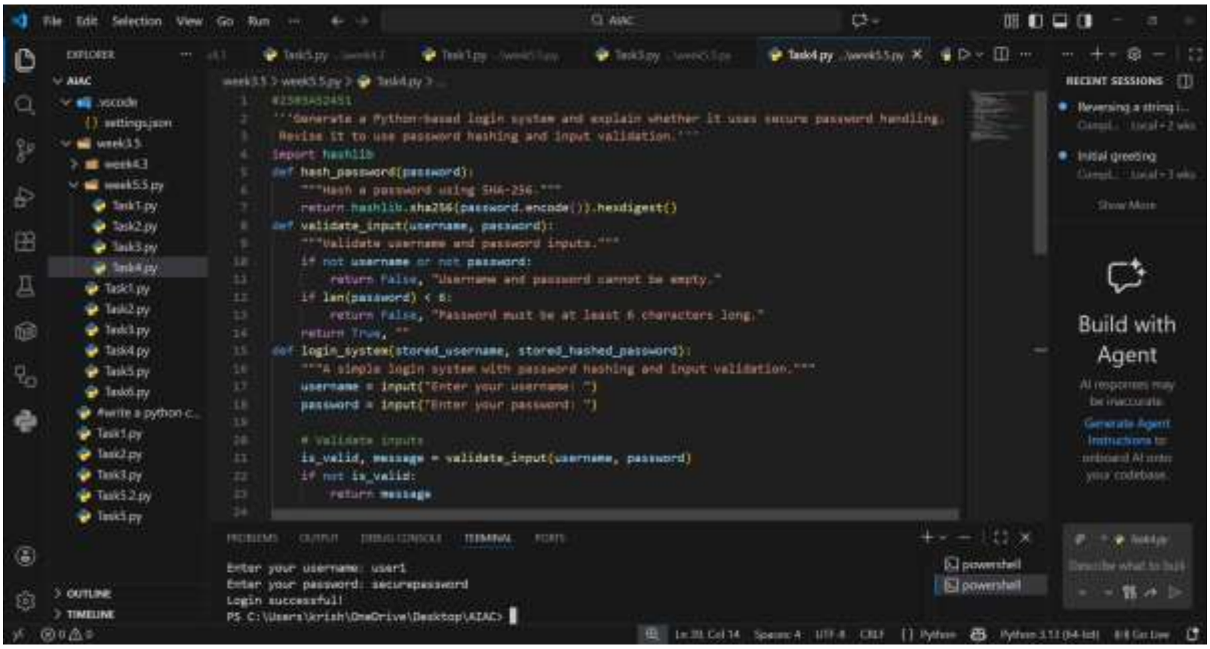
Expected Output:

- **Identification of security flaws (plain-text passwords, weak validation).**
- **Revised version using password hashing and input validation.**
- **Short note on best practices for secure authentication**

PROMPT:

“Generate a Python-based login system and explain whether it uses secure password handling.”

OUTPUT:



EXPLANATION:

Plain-text password storage is insecure because it exposes sensitive data. The improved version uses hashing and input validation, which protect user credentials. Secure authentication practices emphasize the importance of safeguarding user data and preventing 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:

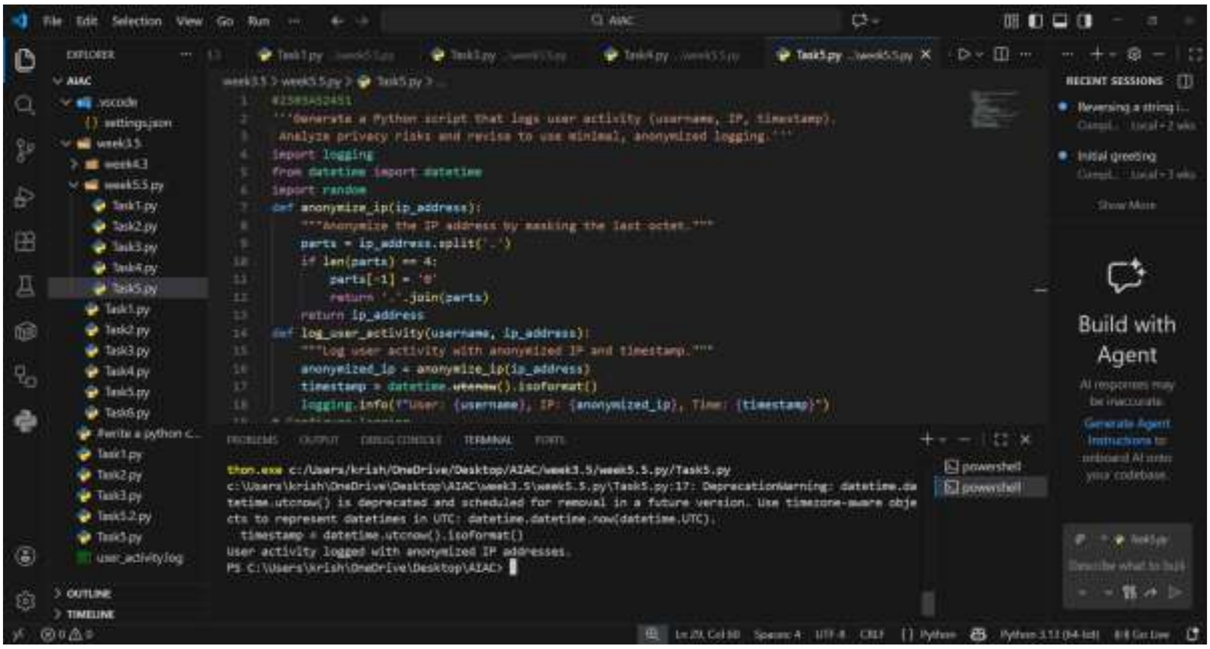
- **Identified privacy risks in logging.**
- **Improved version with minimal, anonymized, or masked logging.**
- **Explanation of privacy-aware logging principles.**

PROMPT:

"Generate a Python script that logs user activity (username, IP, timestamp).

Analyze privacy risks and revise to use minimal, anonymized logging."

OUTPUT:



EXPLANTION:

Logging sensitive data such as usernames or IP addresses can create privacy risks. The improved version uses anonymization or minimal logging to reduce exposure. Privacy-aware coding ensures only necessary information is collected, protecting user identity and maintaining ethical standards.