

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.”

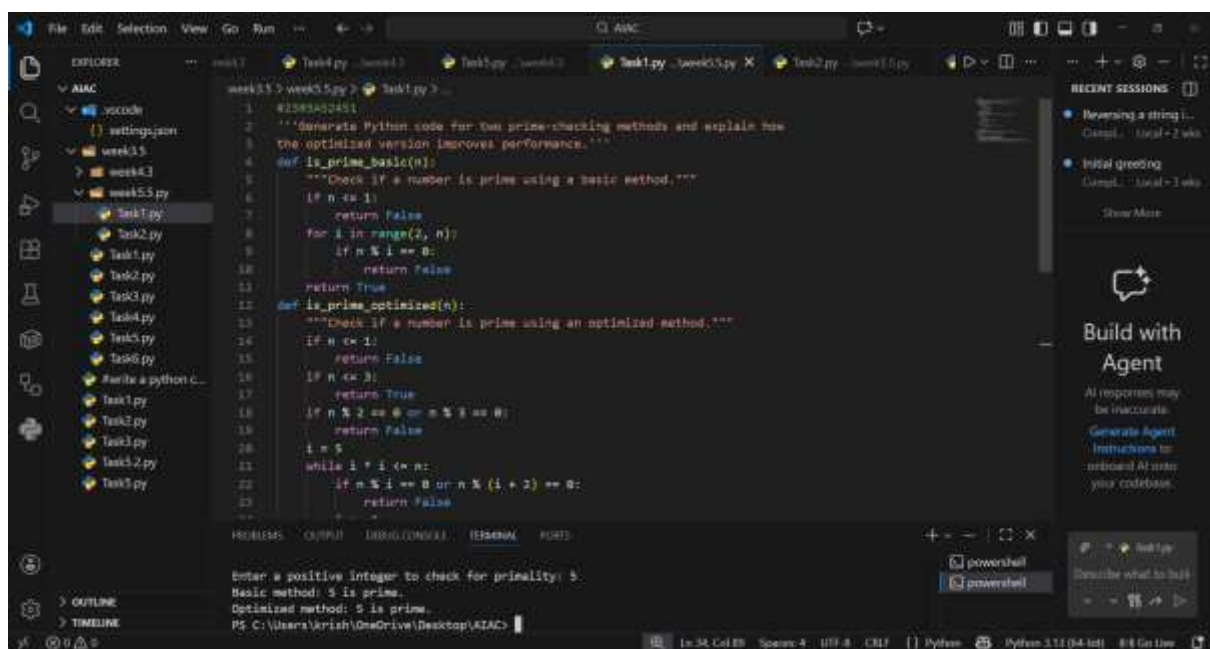
Expected Output:

- **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:



```
1 #2583452451
2 """Generate Python code for two prime-checking methods and explain how
3 the optimized version improves performance."""
4 def is_prime_basic(n):
5     """Check if a number is prime using a basic method."""
6     if n <= 1:
7         return False
8     for i in range(2, n):
9         if n % i == 0:
10            return False
11    return True
12 def is_prime_optimized(n):
13     """Check if a number is prime using an optimized method."""
14     if n <= 1:
15         return False
16     if n <= 3:
17         return True
18     if n % 2 == 0 or n % 3 == 0:
19         return False
20     i = 5
21     while i * i <= n:
22         if n % i == 0 or n % (i + 2) == 0:
23             return False
24         i += 6
```

Enter a positive integer to check for primality: 5
Basic method: 5 is prime.
Optimized method: 5 is prime.
PS C:\Users\krish\OneDrive\Desktop\AZAC>

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 $O(\text{squareroot}(n))$. 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:

```

1 #2585452451
2 """Generate a recursive Python function for fibonacci numbers with clear
3 comments explaining base cases and recursive calls."""
4 def fibonacci(n):
5     """Return the nth Fibonacci number using recursion."""
6     # Base case: the first Fibonacci number is 0
7     if n == 0:
8         return 0
9     # Base case: the second Fibonacci number is 1
10    elif n == 1:
11        return 1
12    else:
13        # Recursive case: sum of the two preceding Fibonacci numbers
14        return fibonacci(n - 1) + fibonacci(n - 2)
15    # Take user input for which Fibonacci number to compute
16    num = int(input("Enter a non-negative integer to find the Fibonacci number: "))
17    if num < 0:
18        print("Please enter a non-negative integer.")
19    else:
20        result = fibonacci(num)
21        print(f"The {num}th Fibonacci number is: {result}")
22

```

Terminal Output:

```

Python.exe c:/Users/krish/OneDrive/Desktop/AIAC/week3.5/week5.5.py/task2.py
Enter a non-negative integer to find the Fibonacci number: 5
The 5th Fibonacci number is: 5
PS C:\Users\krish\OneDrive\Desktop\AIAC>

```

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.”

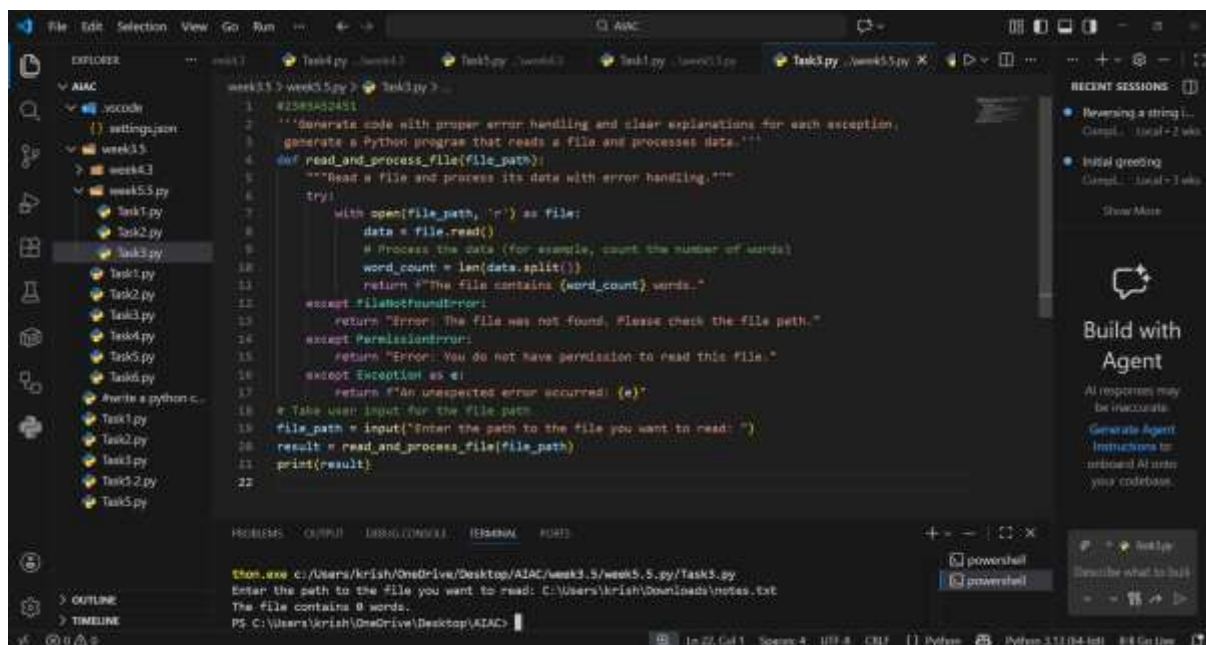
Expected Output:

- 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:



The screenshot shows a Visual Studio Code editor with a Python script named `Task3.py` open. The script is designed to read a file and process its data, with comprehensive error handling. The code includes comments explaining the purpose of each section and the specific exceptions being handled. The script uses a `try-except` block to catch `FileNotFoundError`, `PermissionError`, and a general `Exception`. The output window at the bottom shows the execution of the script, which prompts the user for a file path and displays the word count of the file.

```
1 #2585452451
2 """generate code with proper error handling and clear explanations for each exception,
3 generate a Python program that reads a file and processes data."""
4 def read_and_process_file(file_path):
5     """Read a file and process its data with error handling."""
6     try:
7         with open(file_path, 'r') as file:
8             data = file.read()
9             # Process the data (for example, count the number of words)
10            word_count = len(data.split())
11            return f"The file contains {word_count} words."
12        except FileNotFoundError:
13            return "Error: The file was not found. Please check the file path."
14        except PermissionError:
15            return "Error: You do not have permission to read this file."
16        except Exception as e:
17            return f"An unexpected error occurred: {e}"
18    # Take user input for the file path
19    file_path = input("Enter the path to the file you want to read: ")
20    result = read_and_process_file(file_path)
21    print(result)
22
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
thon.exe c:/Users/krish/OneDrive/Desktop/AIAC/week3.5/week5.5.py/Task3.py
Enter the path to the file you want to read: C:/Users/krish/Downloads/notes.txt
The file contains 8 words.
PS C:/Users/krish/OneDrive/Desktop/AIAC>
```

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.

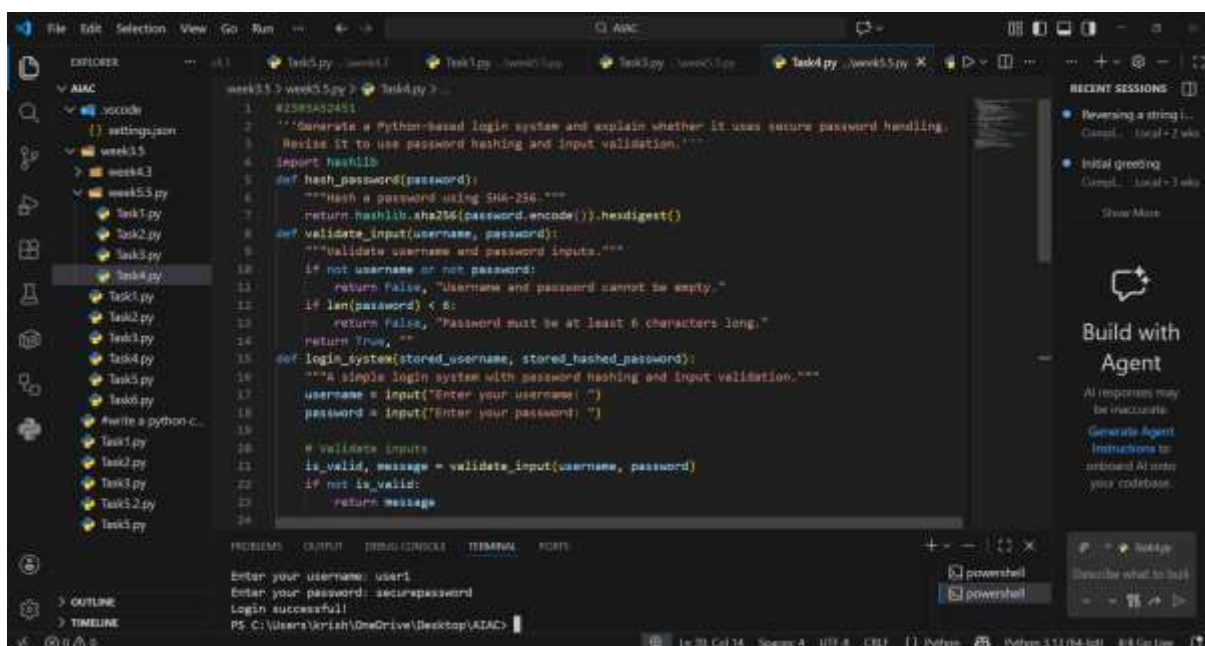
Expected Output:

- 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:



The screenshot shows a VS Code editor with a file explorer on the left displaying a project structure with files like `task1.py` through `task5.py`. The main editor window shows the code for `task4.py`, which implements a login system. The code includes comments explaining the use of `hashlib` for password hashing and input validation. The terminal at the bottom shows the script being executed, with prompts for username and password, and a successful login message.

```
1 # Generate a Python-based login system and explain whether it uses secure password handling.
2 # Revise it to use password hashing and input validation.
3 import hashlib
4
5 def hash_password(password):
6     """Hash a password using SHA-256."""
7     return hashlib.sha256(password.encode()).hexdigest()
8
9 def validate_input(username, password):
10    """Validate username and password inputs."""
11    if not username or not password:
12        return False, "Username and password cannot be empty."
13    if len(password) < 6:
14        return False, "Password must be at least 6 characters long."
15    return True, ""
16
17 def login_system(stored_username, stored_hashed_password):
18    """A simple login system with password hashing and input validation."""
19    username = input("Enter your username: ")
20    password = input("Enter your password: ")
21
22    # Validate inputs
23    is_valid, message = validate_input(username, password)
24    if not is_valid:
25        return message
26
27    # Check credentials
28    if username == stored_username and password == stored_hashed_password:
29        return "Login successful!"
30    else:
31        return "Invalid credentials!"
32
33 # Main execution
34 stored_username = "user1"
35 stored_hashed_password = hash_password("securepassword")
36
37 result = login_system(stored_username, stored_hashed_password)
38 print(result)
```

Terminal output:

```
Enter your username: user1
Enter your password: securepassword
Login successful!
PS C:\Users\Krish\OneDrive\Desktop\AI>
```

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:

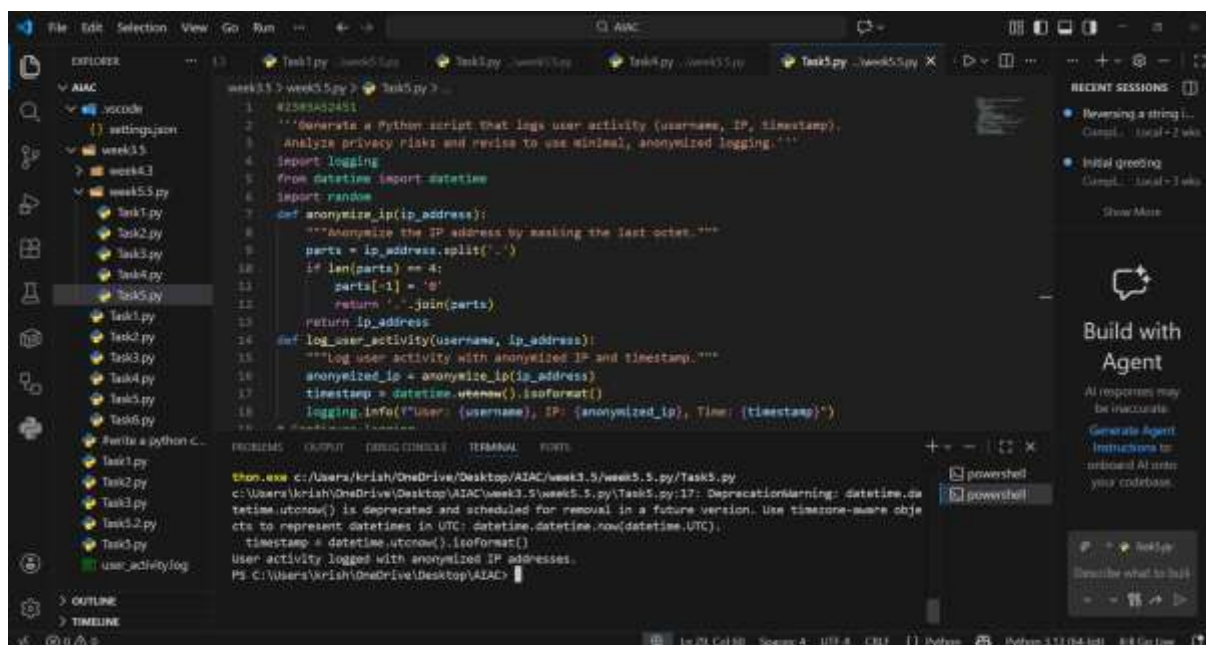
- Identified privacy risks in logging.
- Improved version with minimal, anonymized, or masked logging.
- Explanation of privacy-aware logging principles.

PROMPT:

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

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

OUTPUT:



The screenshot shows a VS Code editor window with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The code editor displays a Python script named `Task5.py` that implements a logging function with IP anonymization. The script includes comments explaining the privacy improvements. The terminal shows the execution of the script, which logs user activity with anonymized IP addresses.

```
1 #2583AS2451
2 """Generate a Python script that logs user activity (username, IP, timestamp).
3 Analyze privacy risks and revise to use minimal, anonymized logging."""
4 import logging
5 from datetime import datetime
6 import random
7
8 def anonymize_ip(ip_address):
9     """Anonymize the IP address by masking the last octet."""
10    parts = ip_address.split('.')
11    if len(parts) == 4:
12        parts[-1] = '0'
13    return '.'.join(parts)
14    return ip_address
15
16 def log_user_activity(username, ip_address):
17     """Log user activity with anonymized IP and timestamp."""
18     anonymized_ip = anonymize_ip(ip_address)
19     timestamp = datetime.utcnow().isoformat()
20     logging.info("User: {username}, IP: {anonymized_ip}, Time: {timestamp}")
21
22 if __name__ == '__main__':
23     log_user_activity('user', '192.168.1.1')
```

The terminal output shows the execution of the script:

```
python.exe c:/Users/Krish/OneDrive/Desktop/AIAC/week3.5/week5.5.py/Task5.py
c:/Users/Krish/OneDrive/Desktop/AIAC/week3.5/week5.5.py/Task5.py:17: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
  timestamp = datetime.utcnow().isoformat()
User activity logged with anonymized IP addresses.
PS C:\Users\krish\OneDrive\Desktop\AIAC>
```

EXPLANATION:

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.