

AI-ASSISTANT CODING

ASSIGNMENT-5.5

B.HARSHAVARDHAN

2303A52T02

Lab 5: Ethical Foundations – Responsible AI Coding Practices

Lab Objectives:

- To explore the ethical risks associated with AI-generated
- 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 AI tools.
- 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 transparency in 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)
- Optimized approach

Prompt:

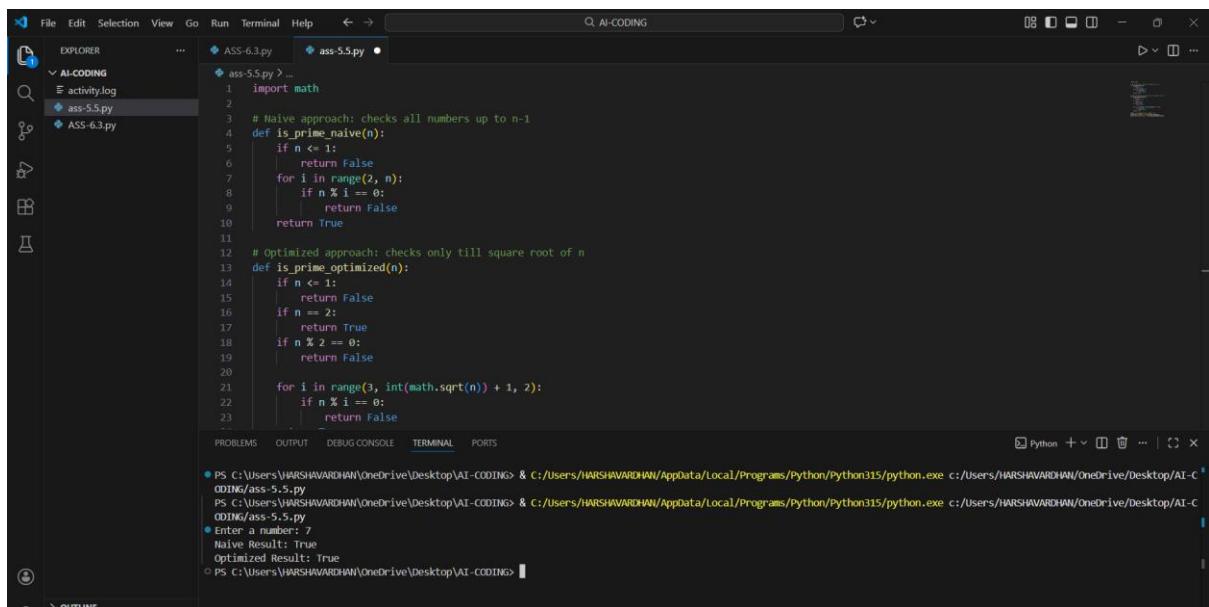
“Generate Python code for two prime-checking methods and explain how the optimized version improves performance.”

Expected Output:

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

Prompt Used

“Generate Python code for two methods to check whether a number is prime: one using a naive approach and another using an optimized approach. Explain how the optimized version improves performance and include time complexity analysis.”



The screenshot shows a Visual Studio Code (VS Code) interface. The left sidebar has an 'EXPLORER' view showing files: 'activity.log', 'ass-5.5.py', and 'ASS-6.3.py'. The main editor area displays Python code for prime number checking. The code defines two functions: 'is_prime_naive(n)' which checks all numbers up to n-1, and 'is_prime_optimized(n)' which checks only up to the square root of n. The terminal at the bottom shows the execution of 'ass-5.5.py' and the output of two prime checks: one naive and one optimized, both returning True for the input 7.

```
ASS-6.3.py ass-5.5.py
ass-5.5.py > ...
1 import math
2
3 # Naive approach: checks all numbers up to n-1
4 def is_prime_naive(n):
5     if n <= 1:
6         return False
7     for i in range(2, n):
8         if n % i == 0:
9             return False
10    return True
11
12 # Optimized approach: checks only till square root of n
13 def is_prime_optimized(n):
14     if n <= 1:
15         return False
16     if n == 2:
17         return True
18     if n % 2 == 0:
19         return False
20
21     for i in range(3, int(math.sqrt(n)) + 1, 2):
22         if n % i == 0:
23             return False
PS C:\Users\HARSHAVARDHAN\OneDrive\Desktop\AI-CODING> & C:/Users/HARSHAVARDHAN/AppData/Local/Programs/Python/Python315/python.exe c:/Users/HARSHAVARDHAN/OneDrive/Desktop/AI-CODING/ass-5.5.py
PS C:\Users\HARSHAVARDHAN\OneDrive\Desktop\AI-CODING> & C:/Users/HARSHAVARDHAN/AppData/Local/Programs/Python/Python315/python.exe c:/Users/HARSHAVARDHAN/OneDrive/Desktop/AI-CODING/ass-5.5.py
● Enter a number: 7
Naive Result: True
Optimized Result: True
PS C:\Users\HARSHAVARDHAN\OneDrive\Desktop\AI-CODING>
```

Justification / Ethical Analysis

- The naive method checks all numbers up to n , which is inefficient.
- The optimized method checks divisibility only up to \sqrt{n} , reducing unnecessary operations.
- Transparent explanation of time complexity ($O(n)$ vs $O(\sqrt{n})$) helps developers understand performance trade-offs.
- This task highlights **explainability and optimization transparency**, which are important when using AI-generated algorithms

Task Description #2 (Transparency in Recursive Algorithms)

Objective: Use AI to generate a recursive function to calculate Fibonacci numbers.

Instructions:

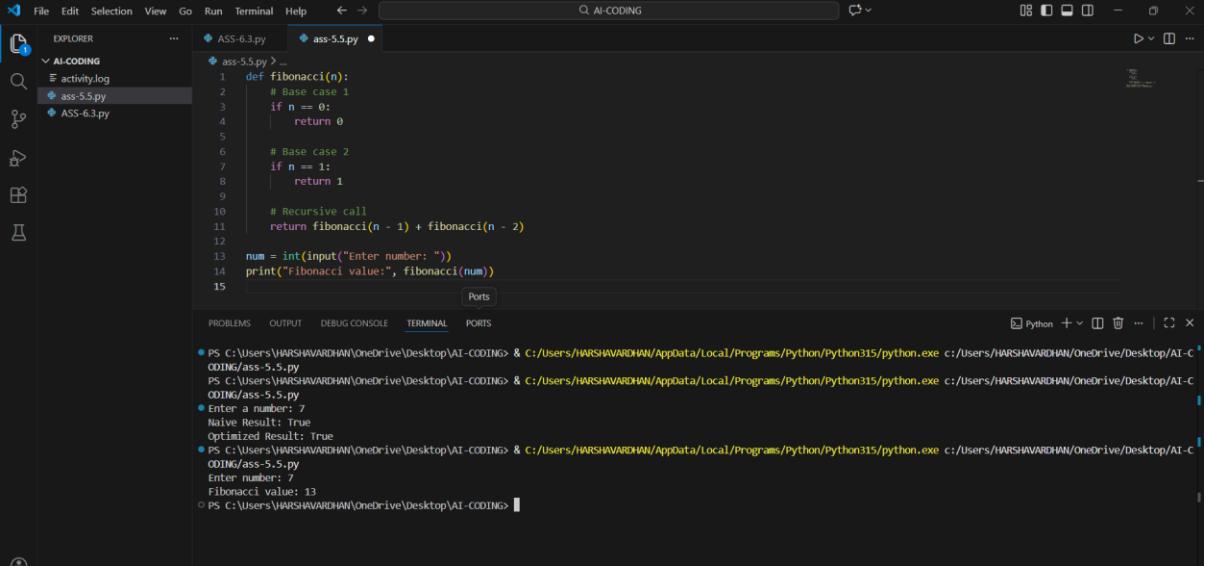
1. Ask AI to add clear comments explaining recursion.
2. Ask AI to explain base cases and recursive calls.

Expected Output:

- Well-commented recursive code.
- Clear explanation of how recursion works.
- Verification that explanation matches actual execution.

Prompt Used

“Generate a recursive Python function to calculate Fibonacci numbers. Add clear comments explaining recursion, base cases, and recursive calls. Also explain how recursion works step by step.”



The screenshot shows a code editor interface with a dark theme. In the Explorer sidebar, there are files named 'ass-5.3.py', 'activity.log', 'ass-5.5.py', and 'ASS-6.3.py'. The 'ass-5.5.py' file is open in the main editor area, displaying the following Python code:

```
def fibonacci(n):
    # Base case 1
    if n == 0:
        return 0

    # Base case 2
    if n == 1:
        return 1

    # Recursive call
    return fibonacci(n - 1) + fibonacci(n - 2)

num = int(input("Enter number: "))
print("Fibonacci value:", fibonacci(num))
```

Below the code editor is a terminal window showing the execution of the script. The terminal output is as follows:

```
PS C:\Users\HARSHAWARDHAN\OneDrive\Desktop\AI-CODING> & C:/Users/HARSHAWARDHAN/AppData/Local/Programs/Python/Python315/python.exe c:/Users/HARSHAWARDHAN/OneDrive/Desktop/AI-CODING/ass-5.5.py
PS C:\Users\HARSHAWARDHAN\OneDrive\Desktop\AI-CODING> & C:/Users/HARSHAWARDHAN/AppData/Local/Programs/Python/Python315/python.exe c:/Users/HARSHAWARDHAN/OneDrive/Desktop/AI-CODING/ass-5.5.py
● Enter a number: 7
  Naive Result: True
  Optimized Result: True
● PS C:\Users\HARSHAWARDHAN\OneDrive\Desktop\AI-CODING> & C:/Users/HARSHAWARDHAN/AppData/Local/Programs/Python/Python315/python.exe c:/Users/HARSHAWARDHAN/OneDrive/Desktop/AI-CODING/ass-5.5.py
  Enter number: 7
  Fibonacci value: 13
○ PS C:\Users\HARSHAWARDHAN\OneDrive\Desktop\AI-CODING>
```

Justification / Ethical Analysis

- Clear comments help understand how recursion breaks problems into smaller sub-problems.
- Base cases prevent infinite recursion and ensure correctness.
- Verifying that the explanation matches actual execution ensures **truthfulness and transparency**.
- This task emphasizes that AI-generated code must be **understandable and explainable**, not just correct

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 Used

“Generate a Python program that reads and processes a file using proper error handling. Include clear explanations and comments for each exception such as FileNotFoundError and PermissionError.”

```
File Edit Selection View Go Run Terminal Help ← → 🔍 AI-CODING 🛡️
```

EXPLORER ... ASS-6.3.py ass-5.5.py

```
ass-5.5.py > read_file
1 def read_file(filename):
2     try:
3         with open(filename, "r") as file:
4             print("File content:")
5             print(file.read())
6
7     except FileNotFoundError:
8         print("Error: File not found.")
9
10    except PermissionError:
11        print("Error: Permission denied.")
12
13    except Exception as e:
14        print("Unexpected error:", e)
15
16 file_name = input("Enter filename: ")
17 read_file(file_name)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\HARSHAVARDHAN\OneDrive\Desktop\AI-CODING> & C:/Users/HARSHAVARDHAN/AppData/Local/Programs/Python/Python315/python.exe c:/AI-CODING/ass-5.5.py
Enter filename: r
Error: File not found.
PS C:\Users\HARSHAVARDHAN\OneDrive\Desktop\AI-CODING>

✓ Justification / Ethical Analysis

- Meaningful exception handling prevents application crashes and unsafe behavior.
- Clear error messages improve user trust and debugging clarity.
- Validating that explanations match runtime behavior ensures **responsible AI usage**.
- This task shows that AI-generated code should be **robust, safe, and transparent** in failure scenarios.

Task Description #4 (Security in User Authentication)

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

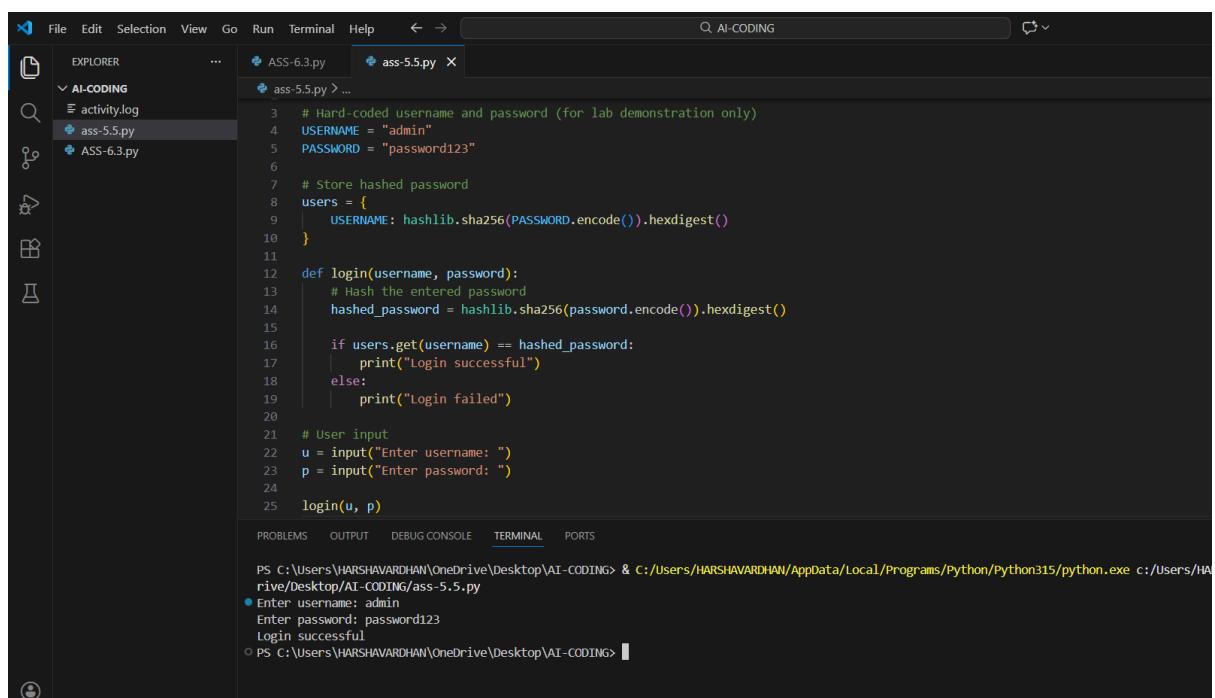
Analyze: Check whether the AI uses secure password handling practices.

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 Used

“Generate a Python-based login system. Review whether it follows secure password handling practices. Identify security flaws and revise the code using password hashing and proper input validation.”



The screenshot shows a dark-themed instance of Visual Studio Code. In the Explorer sidebar, there are two files: 'ass-5.5.py' and 'ASS-6.3.py'. The 'ass-5.5.py' file is currently selected. The code editor displays the following Python script:

```
3 # Hard-coded username and password (for lab demonstration only)
4 USERNAME = "admin"
5 PASSWORD = "password123"
6
7 # Store hashed password
8 users = {
9     USERNAME: hashlib.sha256(PASSWORD.encode()).hexdigest()
10 }
11
12 def login(username, password):
13     # Hash the entered password
14     hashed_password = hashlib.sha256(password.encode()).hexdigest()
15
16     if users.get(username) == hashed_password:
17         print("Login successful")
18     else:
19         print("Login failed")
20
21 # User input
22 u = input("Enter username: ")
23 p = input("Enter password: ")
24
25 login(u, p)
```

The terminal tab at the bottom shows the execution of the script and its output:

```
PS C:\Users\HARSHAVARDHAN\OneDrive\Desktop\AI-CODING> & C:/Users/HARSHAVARDHAN/AppData/Local/Programs/Python/Python315/python.exe c:/Users/HARSHAVARDHAN/OneDrive/Desktop/AI-CODING/ass-5.5.py
● Enter username: admin
Enter password: password123
Login successful
○ PS C:\Users\HARSHAVARDHAN\OneDrive\Desktop\AI-CODING>
```

Justification / Ethical Analysis

- Initial AI-generated code may store passwords in plain text, which is insecure.
- Password hashing (SHA-256) protects credentials even if data is leaked.
- This task demonstrates that developers must **review and improve AI output**.
- Highlights ethical responsibility to protect user data and prevent security breaches

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 Used

“Generate a Python script that logs user activity including username, IP address, and timestamp. Analyze privacy risks and improve the code using anonymized or masked logging.”

The screenshot shows a code editor interface with a dark theme. In the Explorer sidebar, there are files named 'activity.log', 'ass-5.5.py', and 'ASS-6.3.py'. The 'ass-5.5.py' file is open in the main editor area, displaying the following Python code:

```
import logging
from datetime import datetime
logging.basicConfig(level=logging.INFO)
def log_activity(username):
    masked_user = username[0] + "****"
    time = datetime.now()
    logging.info(f"User {masked_user} accessed system at {time}")
user = input("Enter username: ")
log_activity(user)
```

In the bottom right corner, there is a terminal window showing the execution of the script and its output:

```
PS C:\Users\HARSHAVARDHAN\OneDrive\Desktop\AI-CODING> & C:/Users/HARSHAVARDHAN/AppData/Local/Programs/Python/Python315/python.exe c:/Users/HARSHAVARDHAN/OneDrive/Desktop/AI-CODING/ass-5.5.py
● Enter username: admin
Enter password: password123
Login successful!
PS C:\Users\HARSHAVARDHAN\OneDrive\Desktop\AI-CODING> & C:/Users/HARSHAVARDHAN/AppData/Local/Programs/Python/Python315/python.exe c:/Users/HARSHAVARDHAN/OneDrive\Desktop/AI-CODING/ass-5.5.py
● Enter username: admin
INFO:root:User a*** accessed system at 2026-02-04 10:42:51.117698
○ PS C:\Users\HARSHAVARDHAN\OneDrive\Desktop\AI-CODING>
```

Justification / Ethical Analysis

- Logging full usernames and IP addresses can violate user privacy.
- Masking or minimizing logged data reduces risk of misuse.
- Privacy-aware logging follows ethical and legal standards such as data minimization.
- This task reinforces that AI-generated logging must respect **user privacy and confidentiality**