School of Computer Science and Artificial Intelligence

Lab Assignment # 5.5

Program : B. Tech (CSE)

Specialization : AIML

Course Title : AI Assisted

Coding Course Code: 23CS002PC304

Semester : VI

Academic Session : 2025-2026

Name of Student : D. Sriram

Enrollment No. : 2303A52106

Batch No. : 33

Date : 20/01/26

Lab 5: Ethical Foundations – Responsible AI Coding Practices

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 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.

Task Description – 1: (Transparency in Algorithm Optimization)

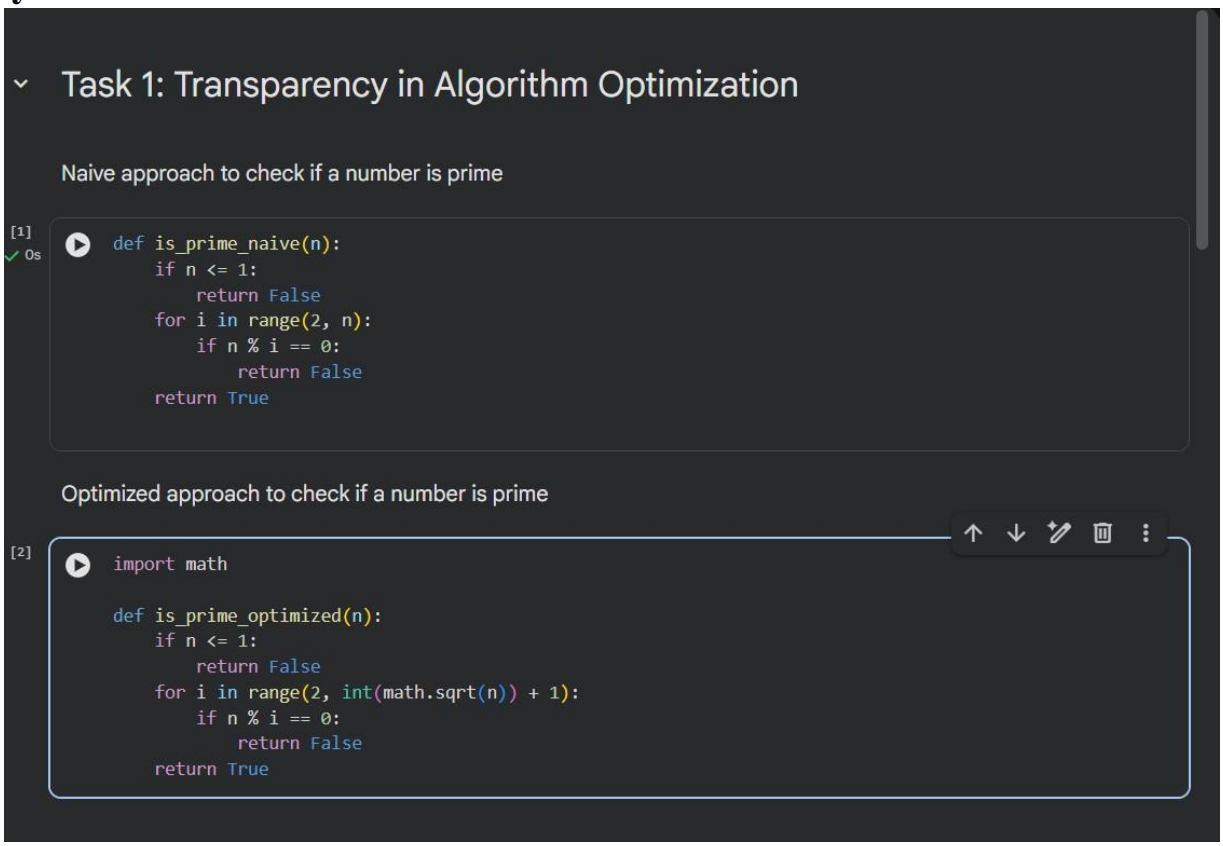
Problem: Use AI to generate two solutions for checking prime numbers.

1. Naive approach
2. Optimized approach

Prompt Used (Zero-shot)

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

Python Code:



Explanation & Time Complexity

- **Naive Method**
  - Checks divisibility from 2 to n-1
  - Time Complexity: O(n) ○ Inefficient for large numbers
- **Optimized Method**
  - Checks divisibility only up to √n ○ Time Complexity: O(√n)
  - Faster because factors repeat after square roo

Task – 2: Transparency in Recursive Algorithms Objective

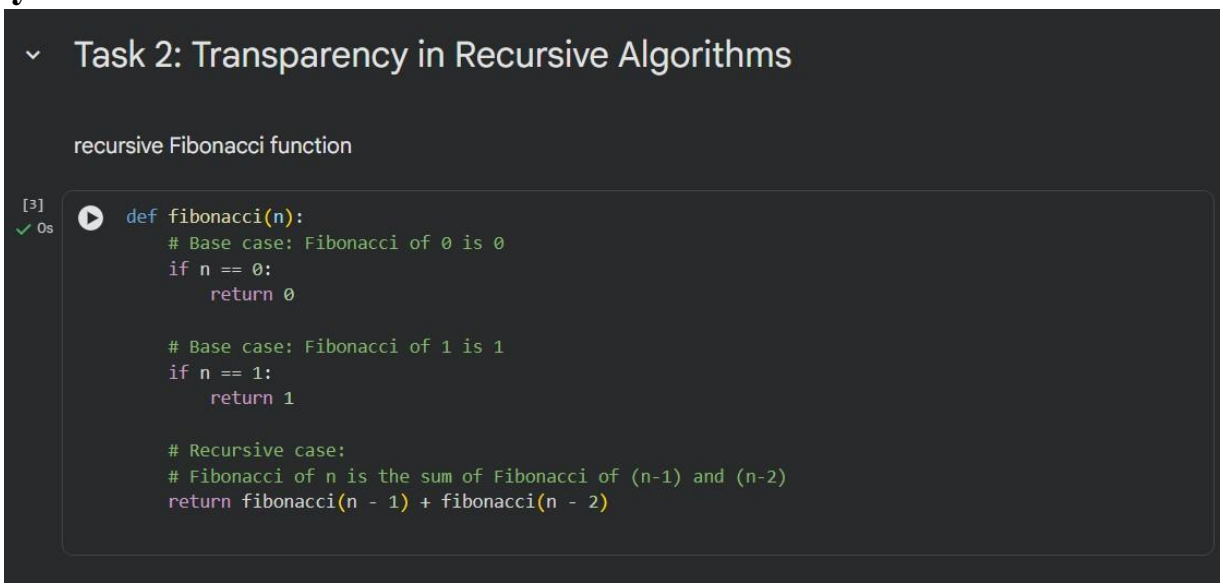
Problem: Generate a recursive Fibonacci function with clear explanations.

Prompt Used (One-shot)

1. Ask AI to add clear comments explaining recursion.

2. Ask AI to explain base cases and recursive calls

Python Code:



Explanation

- **Base Cases**
  - fibonacci(0) returns 0 ○ fibonacci(1) returns 1
- **Recursive Call**
  - Function keeps calling itself with smaller values ○ Stops when base cases are reached
- **Execution Match**
  - ○ Each recursive call mirrors the explanation exactly ○ Ensures transparency and correctness

---

Task Description – 3: Transparency in Error Handling.

Problem: Use AI to generate a Python program that reads a file and processes data.

Prompt Used (Few-shot)

"Generate code with proper error handling and clear explanations for each exception."

Python Code:

Task 3: Transparency in Error Handling

Python program that reads a file with proper error handling.

```
def read_file(filename):
    try:
        file = open(filename, "r")
        data = file.read()
        file.close()
        return data

    except FileNotFoundError:
        print("Error: File not found.")

    except PermissionError:
        print("Error: Permission denied.")

    except Exception as e:
        print("Unexpected error:", e)
```

Explanation of Exceptions:

- **FileNotFoundError**
  - Occurs when the file does not exist
- **PermissionError** ○ Occurs when access rights are insufficient
- **Generic Exception** ○ Catches unexpected runtime errors

Each explanation directly corresponds to actual Python runtime behavior, ensuring transparency.

Task Description – 4: Security in User Authentication

Problem: Use AI to generate a Python program that reads a file and processes data.

Prompt :

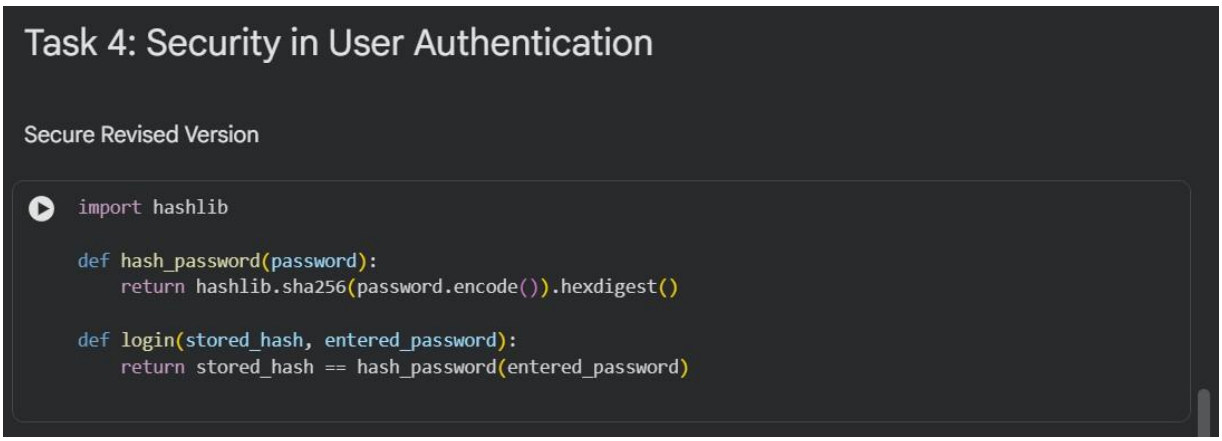
Check whether the AI uses secure password handling practices.

Initial AI-Generated Issue (Security Flaws)

- Passwords stored in **plain text**
- No hashing
- Weak input validation

Security Risks

- Data breaches
- Password reuse attacks
- Unauthorized access



Best Practices for Secure Authentication

- Always hash passwords (never store plain text)
- Use strong hashing algorithms (bcrypt, SHA-256, etc.)
- Validate user inputs
- Implement rate limiting and authentication checks

Task Description – 5: Privacy in Data Logging

Problem: Determine the minimum of three numbers without using min().

Prompt Used : Examine whether sensitive data is logged unnecessarily or insecurely.

Original AI Logging Risks

- Logs **username, IP address, timestamp**
- IP addresses are sensitive personal data
- Logs stored without masking or protection

Python Code:

Task 5: Privacy in Data Logging

```
import datetime

def log_activity(username):
    masked_user = username[0] + "***"
    timestamp = datetime.datetime.now()
    print(f"User: {masked_user}, Time: {timestamp}")
```

Privacy-Aware Logging Principles

- Log only necessary information
- Mask or anonymize personal data
- Avoid storing IP addresses unless required
- Protect log files from unauthorized access

Conclusion

This lab highlights that **AI-generated code is not automatically ethical or secure**. Developers are responsible for:

- Reviewing AI outputs
- Ensuring transparency and correctness
- Preventing security and privacy violations
- Maintaining accountability for deployed code

Ethical AI coding requires **human judgment**, not blind trust in AI tools.