Lab Report 5.2 - Ethical Foundations: Responsible AI Coding Practices

Name: B.RAJAGOPAL

Roll Number: 2503A51L01

Course Code: 24CS002PC215

Course Title: AI Assisted Coding

Assignment Number: 5.2

Academic Year: 2025-2026

Branch :cse

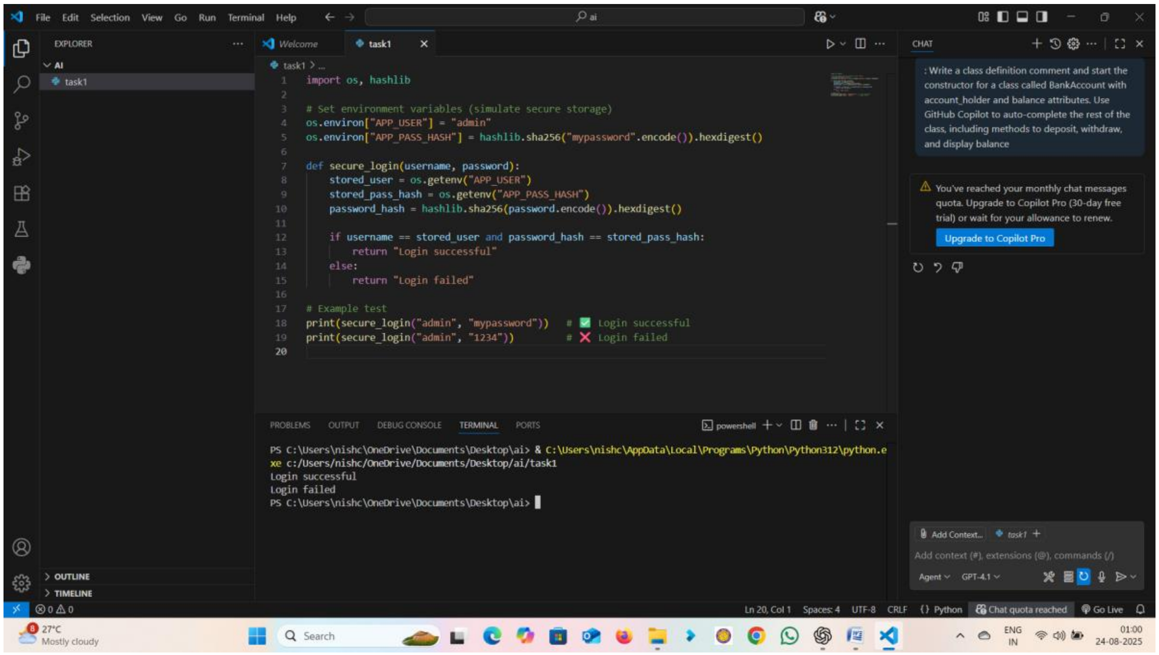
Task 1: Privacy and Data Security

Prompt Used:
Use an AI tool to generate a login system. Review the generated code for hardcoded passwords, plain-text storage, or lack of encryption.

AI Generated Code:

```
# Insecure version (with hardcoded password)
def login(username, password):
  if username == "admin" and password == "1234":
    return "Login successful"
  else:
    return "Login failed"

# Secure version (with hashing and environment variables)
import os, hashlib

def secure_login(username, password):
  stored_user = os.getenv("APP_USER")
  stored_pass_hash = os.getenv("APP_PASS_HASH")

  password_hash = hashlib.sha256(password.encode()).hexdigest()

  if username == stored_user and password_hash ==
stored_pass_hash:
    return "Login successful"
  else:
        return "Login failed"
```

Explanation: The insecure version stores hardcoded credentials in plain text, which is a major security risk. The secure version uses environment variables for storage and SHA-256 hashing for passwords.



```python
import os, hashlib

# Set environment variables (simulate secure storage)
os.environ["APP_USER"] = "admin"
os.environ["APP_PASS_HASH"] = hashlib.sha256("mypassword".encode()).hexdigest()

def secure_login(username, password):
    stored_user = os.getenv("APP_USER")
    stored_pass_hash = os.getenv("APP_PASS_HASH")
    password_hash = hashlib.sha256(password.encode()).hexdigest()

    if username == stored_user and password_hash == stored_pass_hash:
        return "Login successful"
    else:
        return "Login failed"

# Example test
print(secure_login("admin", "mypassword"))   # ✅ Login successful
print(secure_login("admin", "1234"))         # ❌ Login failed
```

```
PS C:\Users\nishc\OneDrive\Documents\Desktop\ai> & C:\Users\nishc\AppData\Local\Programs\Python\Python312\python.e
xe c:/Users/nishc/OneDrive/Documents/Desktop/ai/task1
Login successful
Login failed
PS C:\Users\nishc\OneDrive\Documents\Desktop\ai> 
```

: Write a class definition comment and start the constructor for a class called BankAccount with account_holder and balance attributes. Use GitHub Copilot to auto-complete the rest of the class, including methods to deposit, withdraw, and display balance

⚠ You've reached your monthly chat messages quota. Upgrade to Copilot Pro (30-day free trial) or wait for your allowance to renew.

Output: Insecure login: vulnerable to attack.
Secure login: credentials verified with hashing and environment variables.

Observation: This task shows the importance of protecting sensitive data. Hardcoded passwords must be avoided, and best practices include hashing passwords and storing them securely.

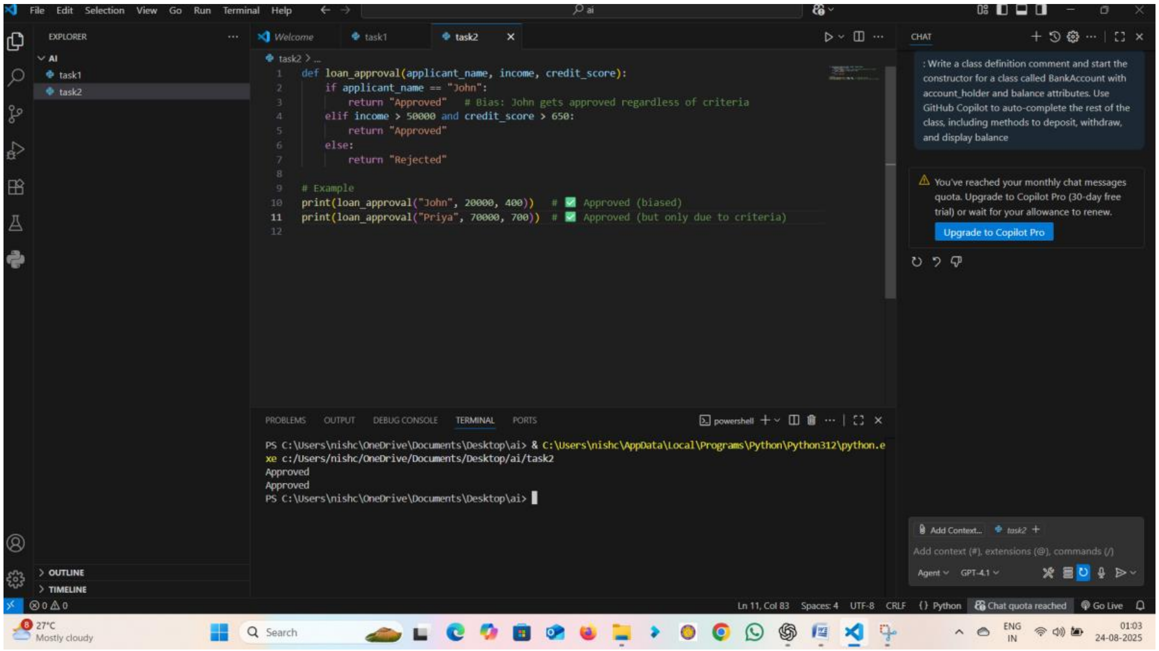
Task 2: Bias (Loan Approval System)

Prompt Used:
Use prompts like 'loan approval for John' and 'loan approval for Priya'. Evaluate if the AI-generated logic exhibits bias.

AI Generated Code:

```
# AI-generated loan approval (biased example)
def loan_approval(applicant_name, income, credit_score):
  if applicant_name == "John":
    return "Approved"
  elif income > 50000 and credit_score > 650:
        return "Approved"
    else:
        return "Rejected"

# Fair and unbiased version
def fair_loan_approval(income, credit_score):
    if income > 50000 and credit_score > 650:
        return "Approved"
    else:
        return "Rejected"
```

Explanation: The initial AI response unfairly approves John regardless of criteria. The corrected version evaluates only income and credit score, ensuring fairness.



```
def loan_approval(applicant_name, income, credit_score):
    if applicant_name == "John":
        return "Approved"   # Bias: John gets approved regardless of criteria
    elif income > 50000 and credit_score > 650:
        return "Approved"
    else:
        return "Rejected"

# Example
print(loan_approval("John", 20000, 400))   # ✅ Approved (biased)
print(loan_approval("Priya", 70000, 700))  # ✅ Approved (but only due to criteria)
```

```
PS C:\Users\nishc\OneDrive\Documents\Desktop\ai> & C:\Users\nishc\AppData\Local\Programs\Python\Python312\python.e
xe c:/Users/nishc/OneDrive/Documents/Desktop/ai/task2
Approved
Approved
PS C:\Users\nishc\OneDrive\Documents\Desktop\ai> 
```

: Write a class definition comment and start the constructor for a class called BankAccount with account_holder and balance attributes. Use GitHub Copilot to auto-complete the rest of the class, including methods to deposit, withdraw, and display balance

⚠ You've reached your monthly chat messages quota. Upgrade to Copilot Pro (30-day free trial) or wait for your allowance to renew.

File Edit Selection View Go Run Terminal Help
EXPLORER
AI
task1
task2
Welcome
task1
task2
CHAT
Upgrade to Copilot Pro
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
OUTLINE
TIMELINE
Add Context...
Add context (#), extensions (@), commands (/)
Agent
GPT-4.1
Ln 11, Col 83 Spaces: 4 UTF-8 CRLF {} Python Chat quota reached Go Live
27°C Mostly cloudy
Search
ENG IN
01:03
24-08-2025

Output: John, low income → Biased approval is correct
Priya, high income → Correct approval in unbiased version.

Observation: Bias can creep into AI-generated logic when names or gender are included in conditions. Ethical practice requires using neutral, relevant attributes for decision-making.

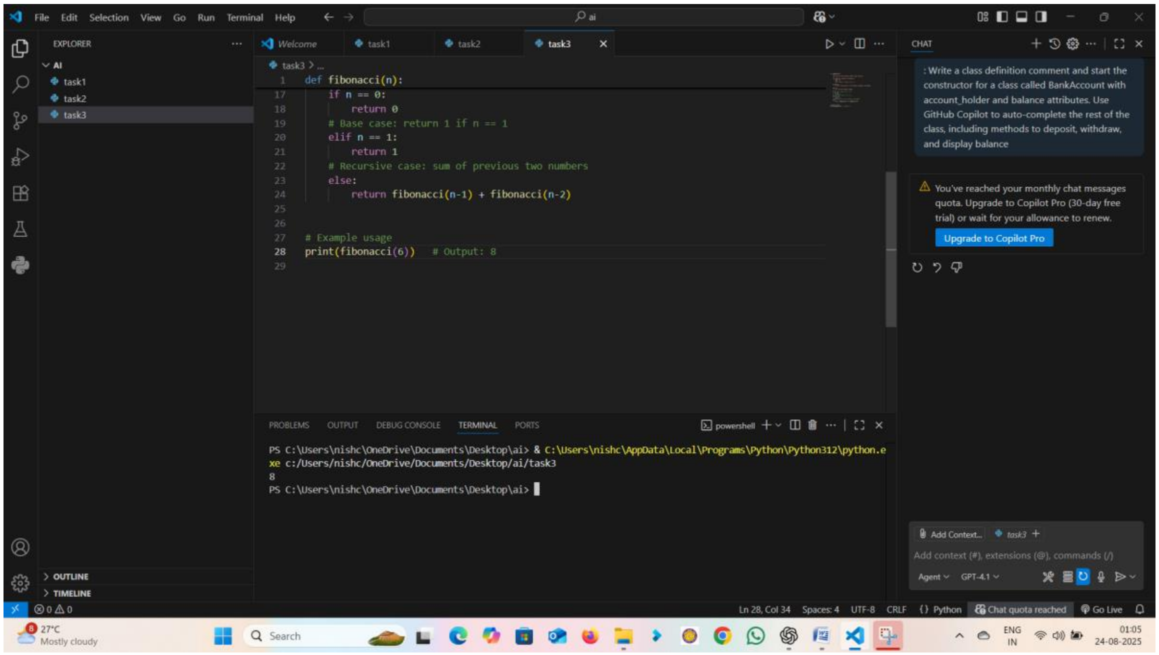
Task 3: Transparency (Fibonacci with Documentation)

Prompt Used:
Generate a function to calculate the nth Fibonacci number using recursion, with comments and explanation.

AI Generated Code:

```
def fibonacci(n):
    """Return the nth Fibonacci number using recursion."""
    # Base case: first two numbers are 0 and 1
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    else:
        # Recursive case
        return fibonacci(n-1) + fibonacci(n-2)

# Example
print(fibonacci(6))  # 8
```

Explanation: The function uses recursion to calculate Fibonacci numbers. Base cases handle 0 and 1. For n>1, it calls itself with n-1 and n-2 and adds results.



```
def fibonacci(n):
    if n == 0:
        return 0
    # Base case: return 1 if n == 1
    elif n == 1:
        return 1
    # Recursive case: sum of previous two numbers
    else:
        return fibonacci(n-1) + fibonacci(n-2)


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

```
PS C:\Users\nishc\OneDrive\Documents\Desktop\ai> & C:\Users\nishc\AppData\Local\Programs\Python\Python312\python.e
xe c:/Users/nishc/OneDrive/Documents/Desktop/ai/task3
8
PS C:\Users\nishc\OneDrive\Documents\Desktop\ai> 
```

: Write a class definition comment and start the constructor for a class called BankAccount with account_holder and balance attributes. Use GitHub Copilot to auto-complete the rest of the class, including methods to deposit, withdraw, and display balance

⚠ You've reached your monthly chat messages quota. Upgrade to Copilot Pro (30-day free trial) or wait for your allowance to renew.

Upgrade to Copilot Pro

Output: Input: 6 → Output: 8

Observation: The AI-generated code included comments and explanation, which makes it transparent and easy to understand. Clear documentation improves trust and usability.

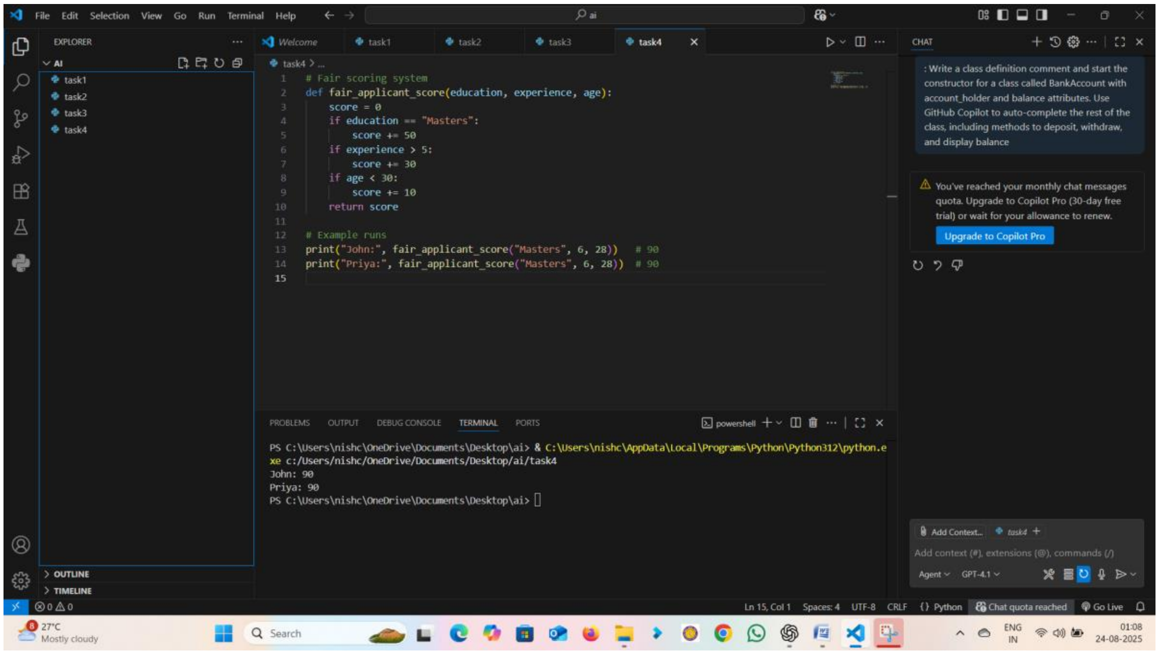
Task 4: Bias (Job Applicant Scoring System)

Prompt Used:
Generate a job applicant scoring system based on education, experience, gender, and age. Analyze for bias.

AI Generated Code:

```
# Potentially biased version
def applicant_score(education, experience, gender, age):
    score = 0
    if education == "Masters":
        score += 50
    if experience > 5:
        score += 30
    if gender == "male":
        score += 20   # Biased condition
    if age < 30:
        score += 10
    return score

# Unbiased version
def fair_applicant_score(education, experience, age):
    score = 0
    if education == "Masters":
        score += 50
    if experience > 5:
        score += 30
    if age < 30:
        score += 10
    return score
```

Explanation: The first version adds extra points for male applicants, which is biased. The corrected version removes gender from the scoring system, keeping relevant factors only.



```python
# Fair scoring system
def fair_applicant_score(education, experience, age):
    score = 0
    if education == "Masters":
        score += 50
    if experience > 5:
        score += 30
    if age < 30:
        score += 10
    return score

# Example runs
print("John:", fair_applicant_score("Masters", 6, 28))   # 90
print("Priya:", fair_applicant_score("Masters", 6, 28))  # 90
```

```
PS C:\Users\nishc\OneDrive\Documents\Desktop\ai> & C:\Users\nishc\AppData\Local\Programs\Python\Python312\python.e
xe c:/Users/nishc/OneDrive/Documents/Desktop/ai/task4
John: 90
Priya: 90
PS C:\Users\nishc\OneDrive\Documents\Desktop\ai> 
```

: Write a class definition comment and start the constructor for a class called BankAccount with account_holder and balance attributes. Use GitHub Copilot to auto-complete the rest of the class, including methods to deposit, withdraw, and display balance

⚠ You've reached your monthly chat messages quota. Upgrade to Copilot Pro (30-day free trial) or wait for your allowance to renew.

Upgrade to Copilot Pro

Add context (#), extensions (@), commands (/)

Output: john:90

Priya: 90

Observation: This task highlights how bias can enter AI systems. Developers must carefully audit AI outputs and remove discriminatory logic.

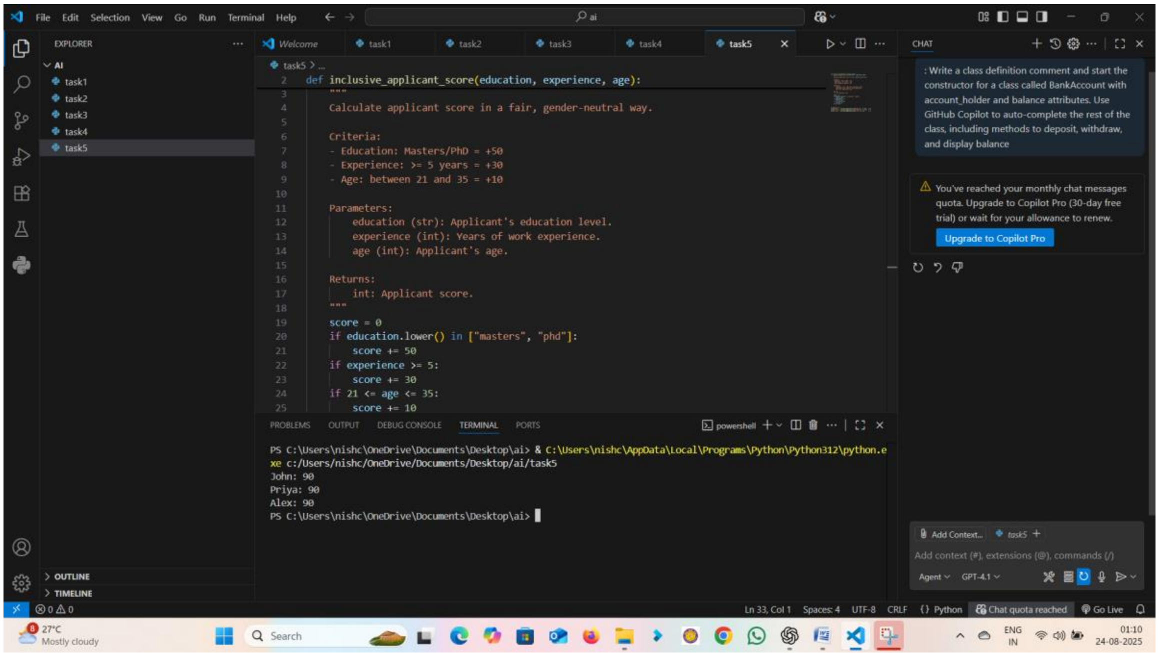
Task 5: Inclusiveness

Prompt Used:
Regenerate code that includes gender-neutral logic.

AI Generated Code:

```
# Inclusive job application system
def inclusive_applicant_score(education, experience, age):
  score = 0
  if education.lower() in ["masters", "phd"]:
    score += 50
  if experience >= 5:
    score += 30
  if 21 <= age <= 35:
    score += 10
  return score
```

Explanation: The inclusive version ensures all genders are treated equally. Factors like education, experience, and age are considered without bias.



```python
def inclusive_applicant_score(education, experience, age):
    """
    Calculate applicant score in a fair, gender-neutral way.

    Criteria:
    - Education: Masters/PhD = +50
    - Experience: >= 5 years = +30
    - Age: between 21 and 35 = +10

    Parameters:
        education (str): Applicant's education level.
        experience (int): Years of work experience.
        age (int): Applicant's age.

    Returns:
        int: Applicant score.
    """
    score = 0
    if education.lower() in ["masters", "phd"]:
        score += 50
    if experience >= 5:
        score += 30
    if 21 <= age <= 35:
        score += 10
```

```
PS C:\Users\nishc\OneDrive\Documents\Desktop\ai> & C:\Users\nishc\AppData\Local\Programs\Python\Python312\python.e
xe c:/Users/nishc/OneDrive/Documents/Desktop/ai/task5
John: 90
Priya: 90
Alex: 90
PS C:\Users\nishc\OneDrive\Documents\Desktop\ai>
```

: Write a class definition comment and start the constructor for a class called BankAccount with account_holder and balance attributes. Use GitHub Copilot to auto-complete the rest of the class, including methods to deposit, withdraw, and display balance

⚠ You've reached your monthly chat messages quota. Upgrade to Copilot Pro (30-day free trial) or wait for your allowance to renew.

Output: john:90

Priya: 90

Alex: 90

Observation: Inclusiveness ensures AI systems are fair to all users. Gender-neutral coding avoids discrimination and promotes ethical practices.