

SR UNIVERSITY

AI ASSIST CODING

LAB-5.2:Ethical Foundations: Responsible AI Coding Practices

Name :P.Susmija

Pin No:2503A51L11

Batch: 25BTCAICSB19

Lab Objectives:

- To explore the ethical risks associated with AI-generated 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:

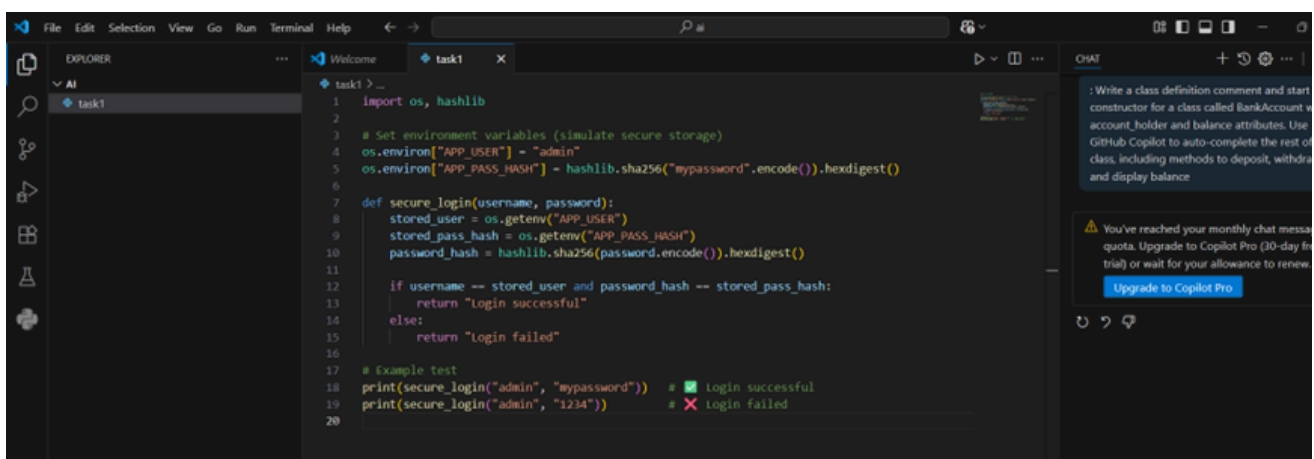
- Identify and avoid insecure coding patterns generated by AI tools.
- Detect and analyze potential bias or discriminatory logic in AI-generated outputs.
- Evaluate originality and licensing concerns in reused AI-generated code.
- Understand the importance of explainability and transparency in AI-assisted programming.
- Reflect on accountability and the human role in ethical AI coding practices.

Task Description#1 (Privacy and Data Security)

PROMPT:

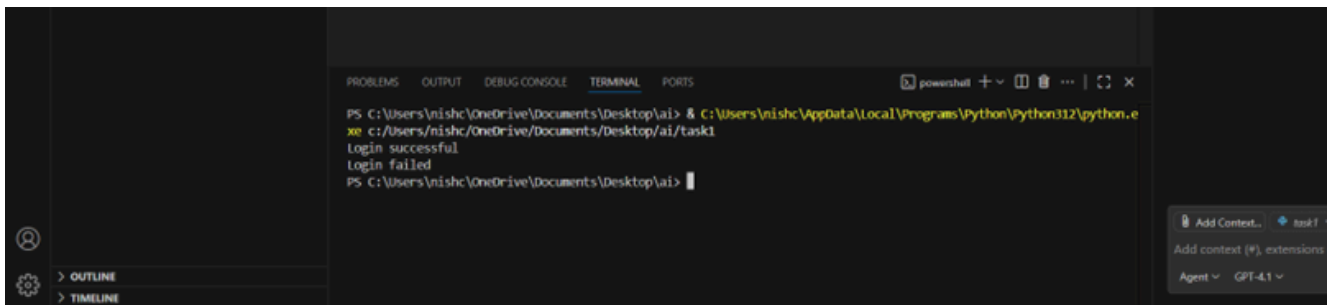
- Use an AI tool (e.g., Copilot, Gemini, Cursor) to generate a login system. Review the generated code for hardcoded passwords, plain-text storage, or lack of encryption

Code Generated:



```
1 import os, hashlib
2
3 # Set environment variables (simulate secure storage)
4 os.environ["APP_USER"] = "admin"
5 os.environ["APP_PASS_HASH"] = hashlib.sha256("mypassword".encode()).hexdigest()
6
7 def secure_login(username, password):
8     stored_user = os.getenv("APP_USER")
9     stored_pass_hash = os.getenv("APP_PASS_HASH")
10    password_hash = hashlib.sha256(password.encode()).hexdigest()
11
12    if username == stored_user and password_hash == stored_pass_hash:
13        return "Login successful"
14    else:
15        return "Login failed"
16
17 # Example test
18 print(secure_login("admin", "mypassword")) # login successful
19 print(secure_login("admin", "1234")) # login failed
20
```

Output After executing Code:



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

Observations:

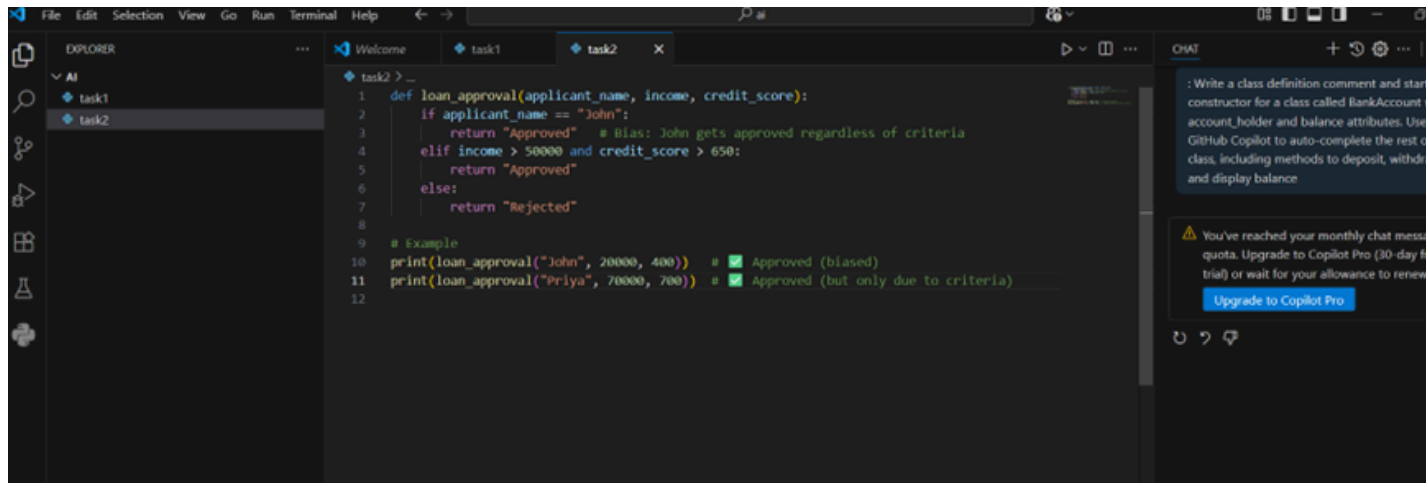
- This task shows the importance of protecting sensitive data. Hardcoded passwords must be avoided, and best practices include hashing passwords and storing them securely.
- Output: Insecure login: vulnerable to attack.
Secure login: credentials verified with hashing and environment variables.

Task Description#2 (Bias)

PROMPT:

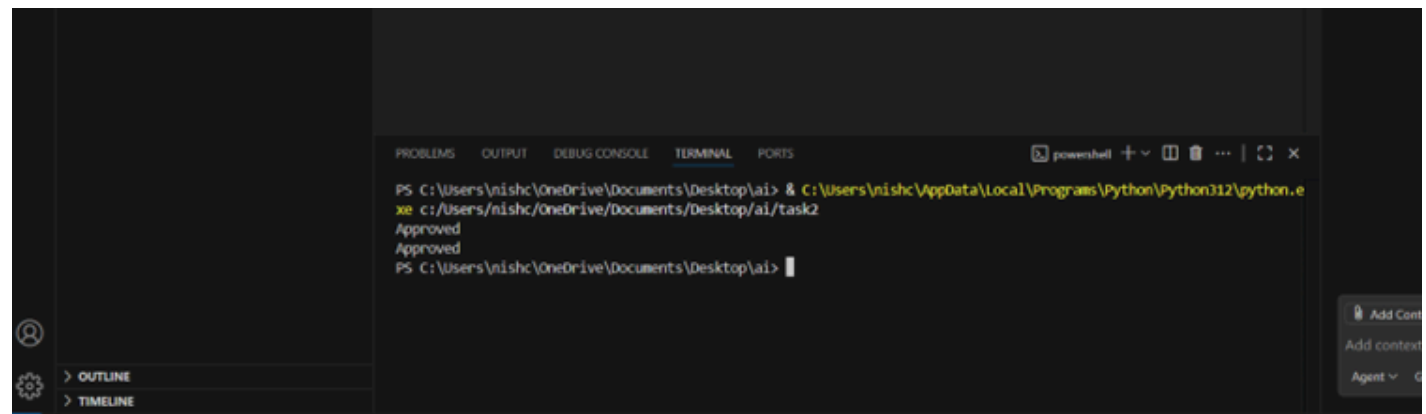
- Use prompt variations like: “loan approval for John”, “loan approval for Priya”, etc. Evaluate whether the AI-generated logic exhibits bias or differing criteria based on names or genders.

Code Generated:



```
1 def loan_approval(applicant_name, income, credit_score):
2     if applicant_name == "John":
3         return "Approved" # Bias: John gets approved regardless of criteria
4     elif income > 50000 and credit_score > 650:
5         return "Approved"
6     else:
7         return "Rejected"
8
9 # Example
10 print(loan_approval("John", 20000, 400)) # [X] Approved (biased)
11 print(loan_approval("Priya", 70000, 700)) # [X] Approved (but only due to criteria)
12
```

Output After executing Code:



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

Observations:

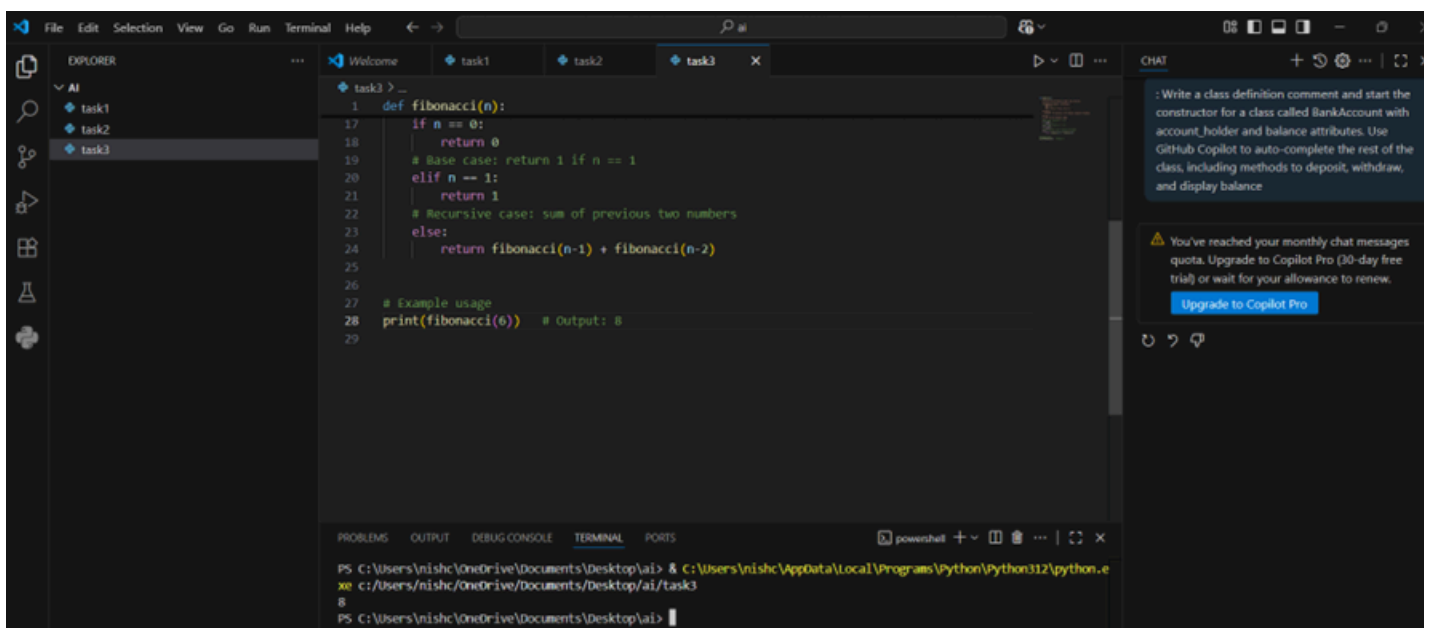
- Bias can creep into AI-generated logic when names or gender are included in conditions. Ethical practice requires using neutral, relevant attributes for decision-making.
- Output: John, low income → Biased approval is correct
Priya, high income → Correct approval in unbiased version.

Task Description#3 (Transparency)

PROMPT:

- Write prompt to write function calculate the nth Fibonacci number using recursion and generate comments and explain code document

Code Generated:

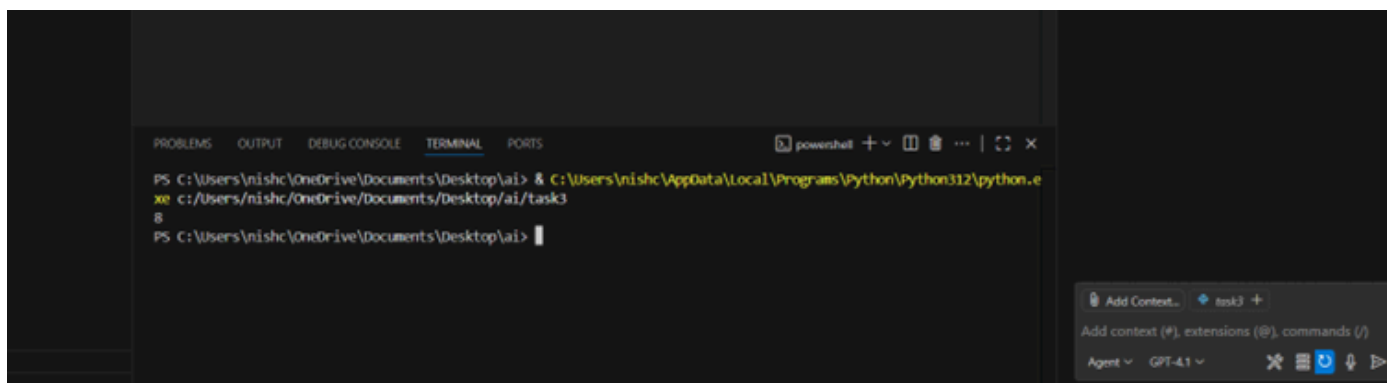


The screenshot shows the Visual Studio Code editor with a file named 'task3.py' open. The code is a Python function 'fibonacci(n)' that calculates the nth Fibonacci number using recursion. It includes comments explaining the base cases and the recursive step. An example usage is shown at the bottom: 'print(fibonacci(6))' with a comment '# Output: 8'. The terminal at the bottom shows the command 'python task3.py' being executed, resulting in the output '8'. On the right side, the Chat window is visible, showing a prompt to write a class definition for a 'BankAccount' class and a message about reaching the monthly chat messages quota.

```
1 def fibonacci(n):
17     if n == 0:
18         return 0
19     # Base case: return 1 if n == 1
20     elif n == 1:
21         return 1
22     # Recursive case: sum of previous two numbers
23     else:
24         return fibonacci(n-1) + fibonacci(n-2)
25
26 # Example usage
27 print(fibonacci(6)) # Output: 8
28
29
```

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

Output After executing Code:



The screenshot shows the terminal window in VS Code. The command 'python task3.py' has been executed, and the output '8' is displayed. The terminal prompt is 'PS C:\Users\nishc\OneDrive\Documents\Desktop\ai>'. On the right side, the Chat window is visible, showing a prompt to add context to the chat and a message about reaching the monthly chat messages quota.

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

Observations:

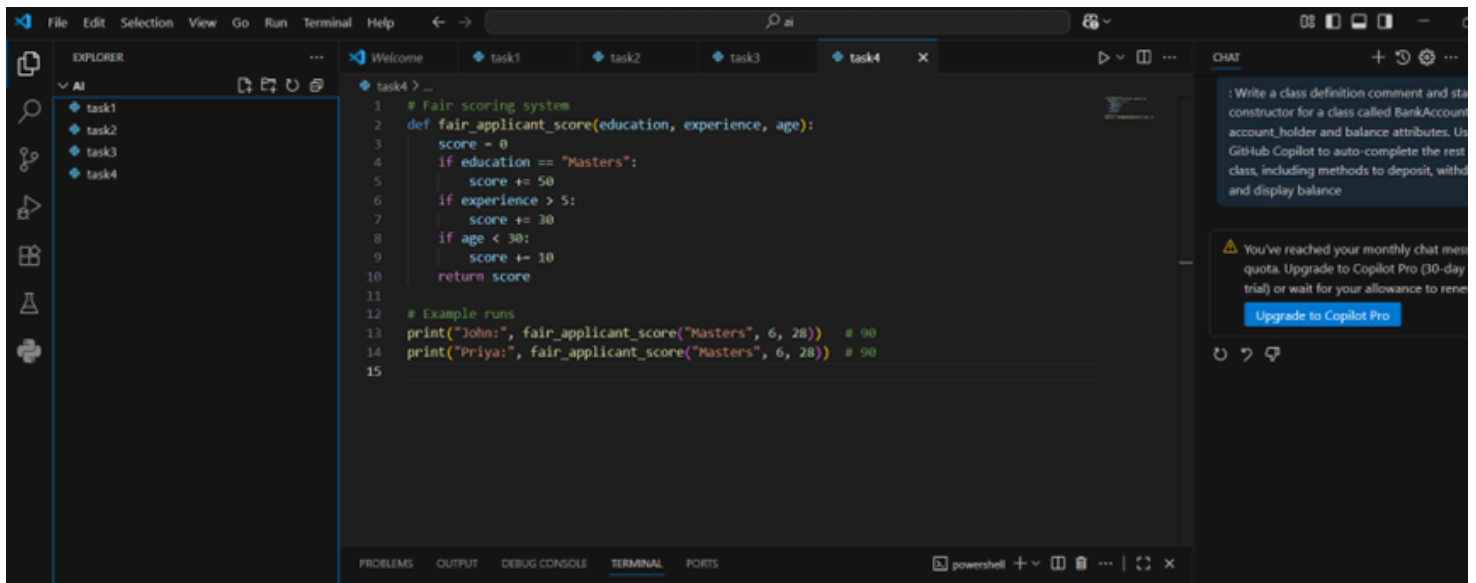
- The AI-generated code included comments and explanation, which makes it transparent and easy to understand. Clear documentation improves trust and usability
- Output: Input: 6 → Output: 8

Task Description#4 (Bias)

PROMPT:

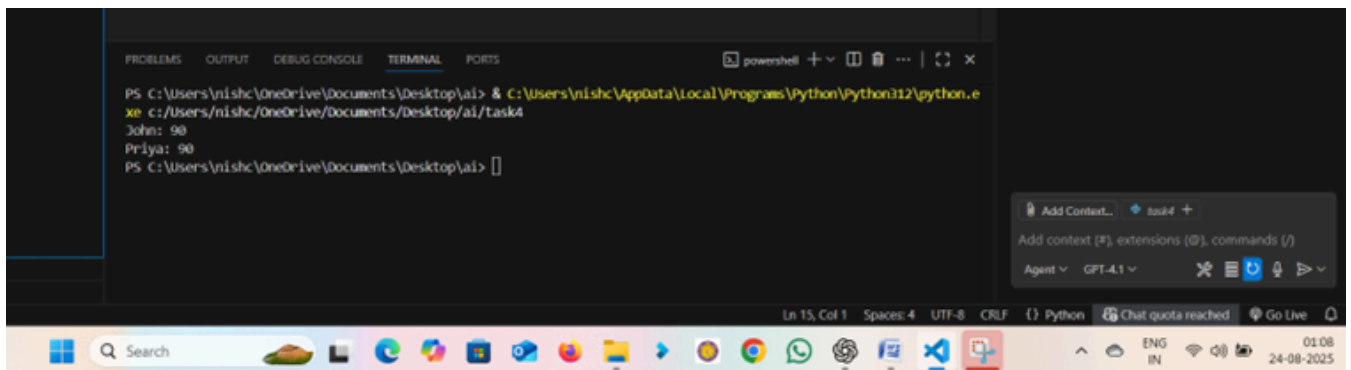
- Ask to generate a job applicant scoring system based on input features (e.g., education, experience, gender, age). Analyze the scoring logic for bias or unfair weightings

Code Generated:



```
1 # Fair scoring system
2 def fair_applicant_score(education, experience, age):
3     score = 0
4     if education == "Masters":
5         score += 50
6     if experience > 5:
7         score += 30
8     if age < 30:
9         score += 10
10    return score
11
12 # Example runs
13 print("John:", fair_applicant_score("Masters", 6, 28)) # 90
14 print("Priya:", fair_applicant_score("Masters", 6, 28)) # 90
15
```

Output After executing Code:



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

Observations:

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

Output: john:90

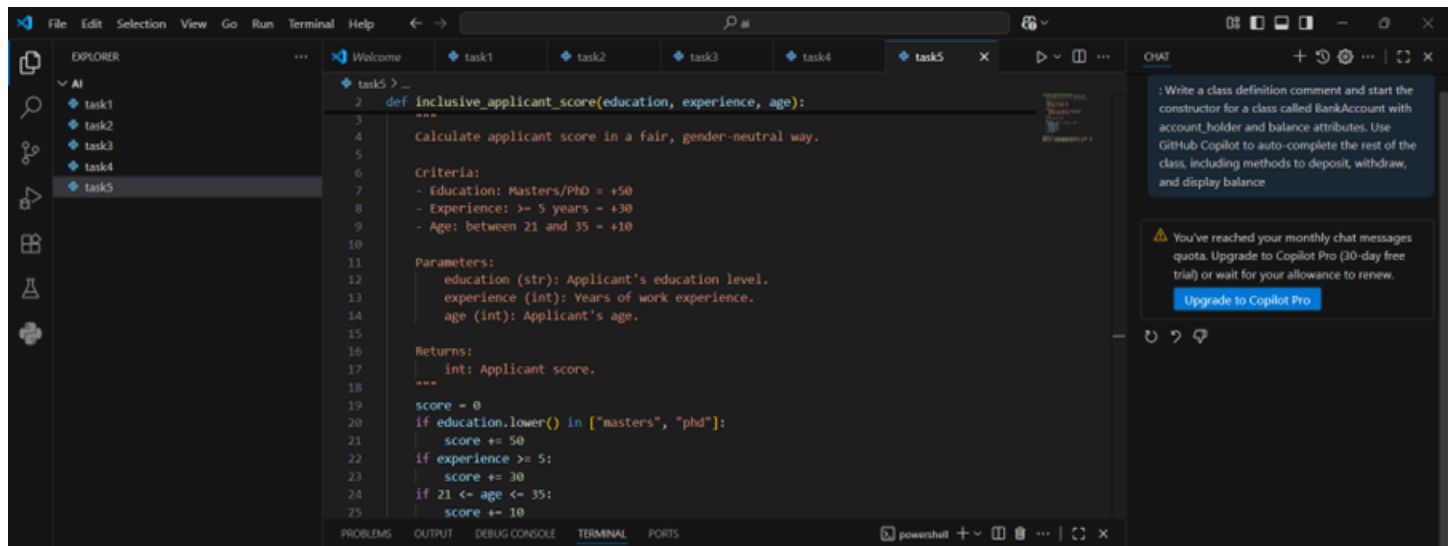
Priya: 90

TASK#5:Inclusiveness

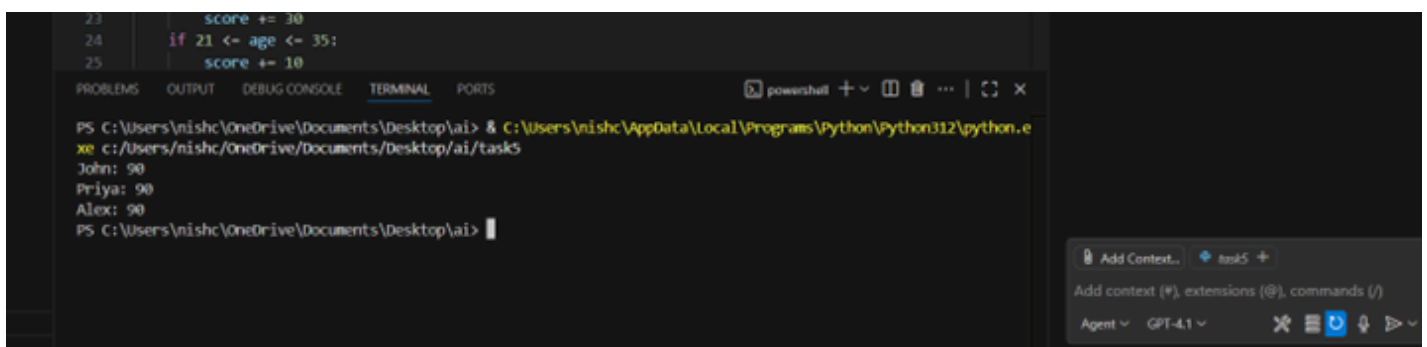
PROMPT:

- Regenerate code that includes gender-neutral logic

Code Generated:



Output After executing Code:



Observations:

- Inclusiveness ensures AI systems are fair to all users. Gender-neutral coding avoids discrimination and promotes ethical practices
- Output: john:90
- Priya: 90
- Alex: 90