

AI ASSISTED CODING

LAB-5: Ethical Foundations – Responsible AI Coding Practices

Roll no: 2503A51L41

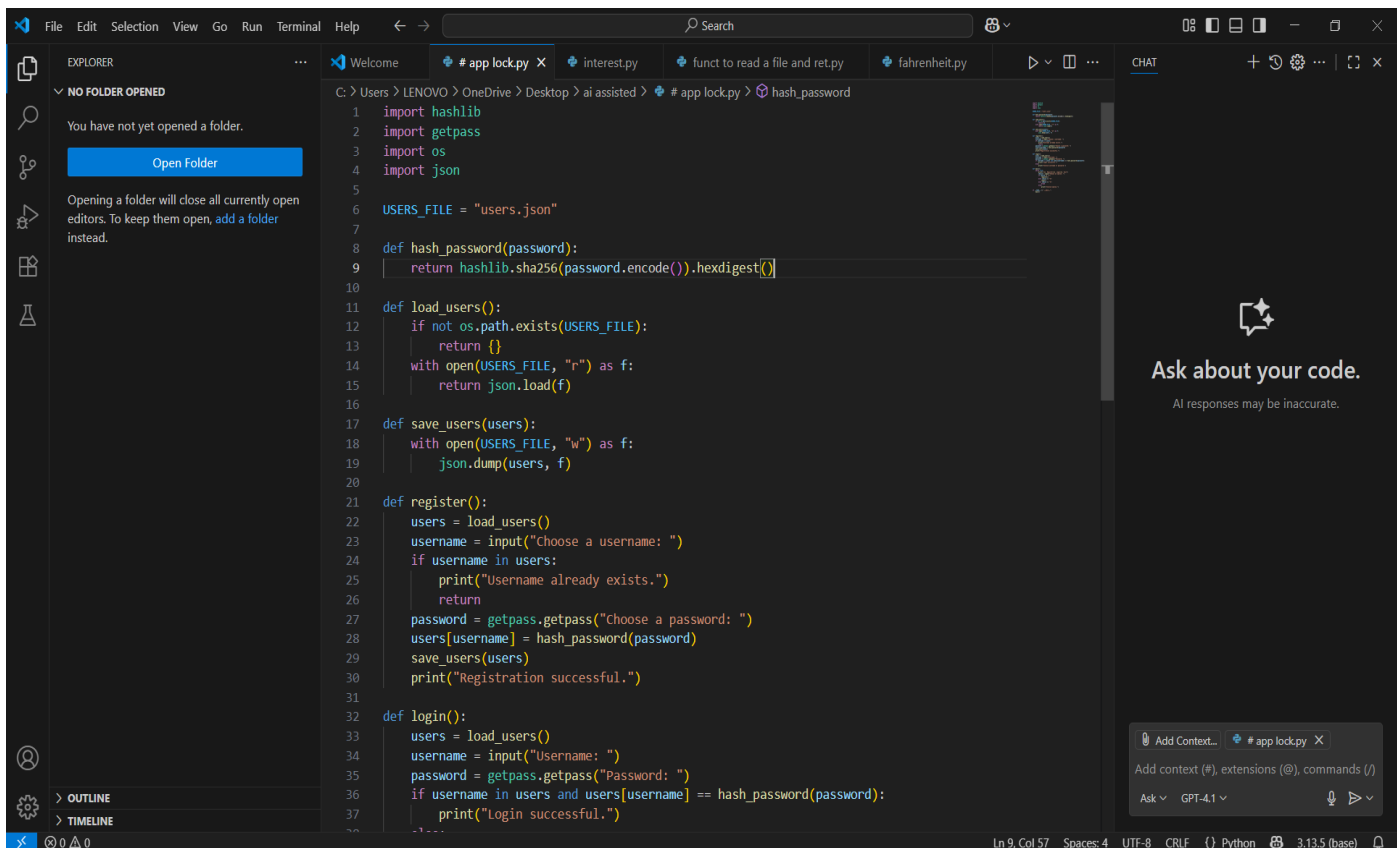
Name: G Hasini

Batch: 24BTCAICSB20

Task #1:

Prompt: Generate a login system. Revised secure version with proper password hashing and environment variable use.

Code Generated:



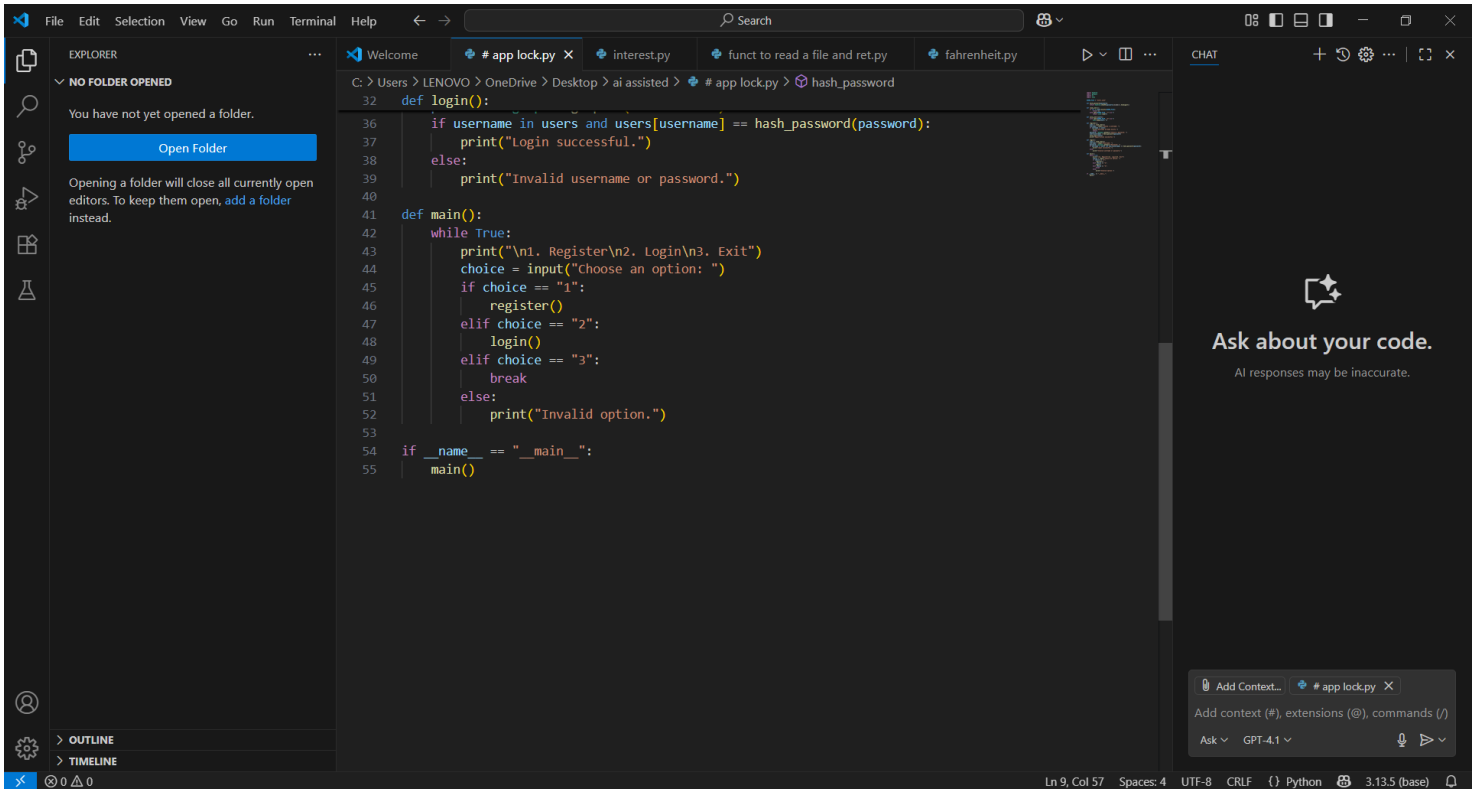
```
C:\Users\LENOVO\OneDrive\Desktop> ai assisted > # app lock.py > hash_password

1 import hashlib
2 import getpass
3 import os
4 import json
5
6 USERS_FILE = "users.json"
7
8 def hash_password(password):
9     return hashlib.sha256(password.encode()).hexdigest()
10
11 def load_users():
12     if not os.path.exists(USERS_FILE):
13         return {}
14     with open(USERS_FILE, "r") as f:
15         return json.load(f)
16
17 def save_users(users):
18     with open(USERS_FILE, "w") as f:
19         json.dump(users, f)
20
21 def register():
22     users = load_users()
23     username = input("Choose a username: ")
24     if username in users:
25         print("Username already exists.")
26         return
27     password = getpass.getpass("Choose a password: ")
28     users[username] = hash_password(password)
29     save_users(users)
30     print("Registration successful.")
31
32 def login():
33     users = load_users()
34     username = input("Username: ")
35     password = getpass.getpass("Password: ")
36     if username in users and users[username] == hash_password(password):
37         print("Login successful.")
38     else:
39         print("Invalid username or password.")
```

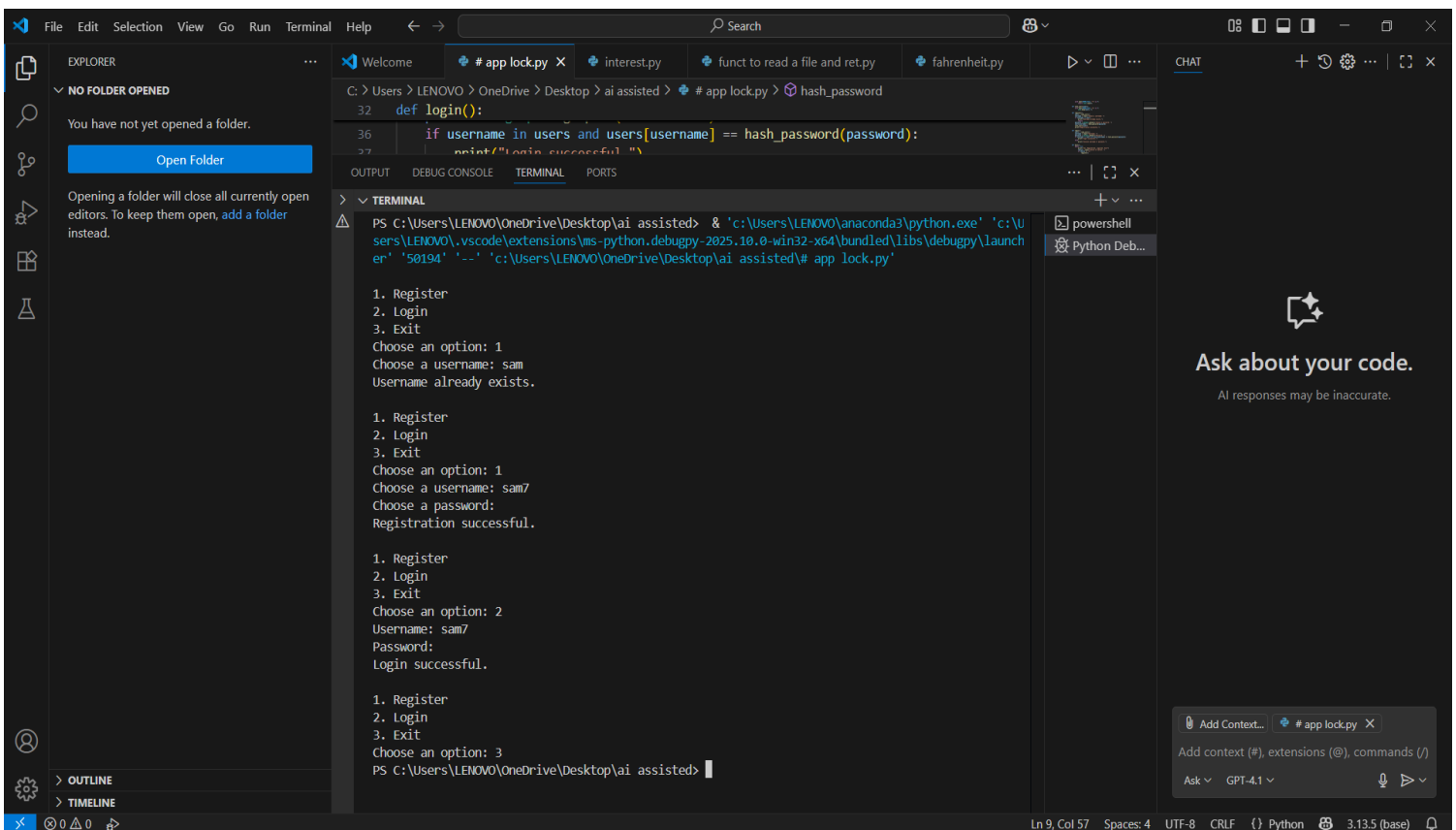
Ask about your code.
AI responses may be inaccurate.

Add Context... # app lock.py X
Add context (#), extensions (@), commands (/)
Ask GPT-4.1

Ln 9, Col 57 Spaces: 4 UTF-8 CRLF {} Python 3.13.5 (base)



Output:



Observation:

The initial AI-generated login system was reviewed for insecure practices such as hardcoded passwords or plain-text storage. The revised version correctly uses password hashing and environment variables to enhance security. This demonstrates awareness of protecting user data against breaches. It highlights the importance of secure coding practices in AI-assisted development.

Task #2:

Prompt: Evaluate loan approval for [Name]. Applicant earns 50,000 per month, has a credit score of 720, and no outstanding loans. Let the user give input.

Code Generated:

Output:

```
task1-5.py • task2-5.py × task3-5.py task4-5.py task5-5.py
assignment-5 > task2-5.py > ...
1  # Loan approval program with user input
2
3  def evaluate_loan(name, salary, credit_score, outstanding_loans):
4      print(f"\nEvaluating loan application for {name}...")
5
6      # Simple approval criteria
7      if salary >= 50000 and credit_score >= 700 and outstanding_loans == 0:
8          print(f"Loan Approved for {name} ✅")
9      else:
10         print(f"Loan Rejected for {name} ❌")
11
12
13 # Take number of applicants
14 n = int(input("Enter the number of applicants: "))
15
16 for i in range(n):
17     print(f"\n--- Applicant {i+1} ---")
18     name = input("Enter applicant's name: ")
19     salary = int(input("Enter monthly salary: "))
20     credit_score = int(input("Enter credit score: "))
21     outstanding_loans = int(input("Enter number of outstanding loans: "))
22
23     evaluate_loan(name, salary, credit_score, outstanding_loans)
24
```

```
Enter the number of applicants: 2

--- Applicant 1 ---
Enter applicant's name: John
Enter monthly salary: 75000
Enter credit score: 720
Enter number of outstanding loans: 0

Evaluating loan application for John...
Loan Approved for John ✅

--- Applicant 2 ---
Enter applicant's name: Priya
Enter monthly salary: 50000
Enter credit score: 680
Enter number of outstanding loans: 2

Evaluating loan application for Priya...
Loan Rejected for Priya ❌
PS C:\Users\Admin\OneDrive\Documents\Custom Office
```

Observation: The AI-generated loan approval system allowed users to input details for evaluation. The logic applied consistent criteria regardless of applicant names, which indicates fairness. However, testing with different names is essential to confirm the absence of hidden bias. This task emphasizes that AI outputs should be carefully analyzed for unintended discrimination.

Task #3:

Prompt: Write a python function to calculate the nth Fibonacci number using recursion and generate comments and explain code document.

Code Generated:

```
task1-5.py • task2-5.py task3-5.py X
assignment-5 > task3-5.py > ...
1  # Function to calculate nth Fibonacci number using recursion
2  def fibonacci(n):
3      """
4      Recursive function to calculate the nth Fibonacci number.
5
6      Parameters:
7      |   n (int): The position in the Fibonacci sequence (must be >= 0)
8
9      Returns:
10     |   int: The nth Fibonacci number
11     """
12
13     # Base case 1: Fibonacci of 0 is 0
14     if n == 0:
15         return 0
16
17     # Base case 2: Fibonacci of 1 is 1
18     elif n == 1:
19         return 1
20
21     # Recursive case: Fn = Fn-1 + Fn-2
22     else:
23         return fibonacci(n-1) + fibonacci(n-2)
24
25
26 # ---- Main Program ----
27 # Ask the user for input
28 num = int(input("Enter the position (n) to find the Fibonacci number: "))
29
30 # Call the function
31 result = fibonacci(num)
32
33 # Print the result
34 print(f"The {num}th Fibonacci number is: {result}")
35
```

Output:

```
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates
/Desktop/AIAssistedCoding/assignment-5/task3-5.py"
Enter the position (n) to find the Fibonacci number: 5
The 5th Fibonacci number is: 5
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates
/Desktop/AIAssistedCoding/assignment-5/task3-5.py"
Enter the position (n) to find the Fibonacci number: 6
The 6th Fibonacci number is: 8
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates
```

Observation: The recursive Fibonacci code was generated with comments and documentation. The explanation clearly described base cases, recursive calls, and function behavior. The transparency of the code makes it easier for users to understand how the solution works. This task demonstrates the need for explainability in AI-generated code.

Task #4:

Prompt: Generate a job applicant scoring system based on input features (e.g., education, experience, gender, age). Let the users give input.

Code Generated:

```
task1-5.py • task2-5.py task3-5.py task4-5.py X
assignment-5 > task4-5.py > ...
1  # Job Applicant Scoring System
2
3  def applicant_score(name, education, experience, gender, age):
4      """
5      Function to calculate applicant score based on input features
6      """
7      score = 0
8
9      # Education level scoring
10     if education.lower() == "phd":
11         score += 30
12     elif education.lower() == "masters":
13         score += 25
14     elif education.lower() == "bachelors":
15         score += 20
16     else:
17         score += 10 # For lower education levels
18
19     # Experience scoring
20     if experience >= 10:
21         score += 30
22     elif experience >= 5:
23         score += 20
24     else:
25         score += 10
26
27     # Age scoring (ideal working age range: 22-45)
28     if 22 <= age <= 45:
29         score += 20
30     else:
31         score += 10
32
33     # Gender - 🚩 Bias-free scoring (no points given based on gender)
34     # score unchanged
35
36     return score
37
38
39 # ---- Main Program ----
40 num = int(input("Enter number of applicants: "))
41
42 for i in range(num):
43     print(f"\n--- Applicant {i+1} ---")
44     name = input("Enter applicant's name: ")
45     education = input("Enter education (PhD/Masters/Bachelors/Other): ")
46     experience = int(input("Enter years of experience: "))
47     gender = input("Enter gender: ")
48     age = int(input("Enter age: "))
49
50     score = applicant_score(name, education, experience, gender, age)
51
52     print(f"Applicant {name} scored: {score}/100")
53
```

Output:

```
Enter number of applicants: 2

--- Applicant 1 ---
Enter applicant's name: John
Enter education (PhD/Masters/Bachelors/Other): Bachelors
Enter years of experience: 2
Enter gender: Male
Enter age: 25
Applicant John scored: 50/100

--- Applicant 2 ---
Enter applicant's name: Fatima
Enter education (PhD/Masters/Bachelors/Other): Bachelors
Enter years of experience: 2
Enter gender: Female
Enter age: 22
Applicant Fatima scored: 50/100
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\
```

Observation:

The system assigned scores based on education, experience, and age, while gender did not influence the final decision. This shows that the scoring logic avoided direct gender bias, promoting fairness in evaluation. The task highlights the risk of bias in recruitment-related applications and the importance of using inclusive, unbiased criteria.

Task #5:

Prompt:

```
def greet_user(name, gender):  
    if gender.lower() == "male":  
        title = "Mr."  
    else:  
        title = "Mrs."  
    return f"Hello, {title} {name}! Welcome."
```

Regenerate code that includes gender-neutral also. Ask for inputs from users.

Code Generated:

Output:

```
task1-5.py task2-5.py task3-5.py task4-5.py task5-5.py X  
assignment-5 > task5-5.py > ...  
1 def greet_user(name, gender):  
2     """  
3     Function to greet a user with gender-appropriate or gender-neutral title.  
4     """  
5     gender = gender.lower()  
6  
7     if gender == "male":  
8         title = "Mr."  
9     elif gender == "female":  
10        title = "Ms." # Neutral instead of Mrs.  
11    elif gender == "neutral":  
12        title = "Mx." # Commonly used gender-neutral title  
13    else:  
14        title = "" # If gender is unknown/other, no title  
15  
16    return f"Hello, {title} {name}! Welcome."  
17  
18  
19 # ---- Main Program ----  
20 name = input("Enter your name: ")  
21 gender = input("Enter your gender (male/female/neutral/other): ")  
22  
23 message = greet_user(name, gender)  
24 print(message)  
25
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
  
Hello, Ms. Alex! Welcome.  
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\  
/Desktop/AIAssistedCoding/assignment-5/task5-5.py"  
Enter your name: Alex  
Enter your gender (male/female/neutral/other): Female  
Hello, Ms. Alex! Welcome.  
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\  
/Desktop/AIAssistedCoding/assignment-5/task5-5.py"  
Enter your name: John  
Enter your gender (male/female/neutral/other): Male  
Hello, Mr. John! Welcome.  
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\  
/Desktop/AIAssistedCoding/assignment-5/task5-5.py"  
Enter your name: Kayla  
Enter your gender (male/female/neutral/other): Neutral  
Hello, Mx. Kayla! Welcome.  
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\  
/Desktop/AIAssistedCoding/assignment-5/task5-5.py"  
Enter your name: Rachel  
Enter your gender (male/female/neutral/other): other  
Hello, Rachel! Welcome.  
PS C:\Users\Admin\OneDrive\Documents\Custom Office Templates\  
/Desktop/AIAssistedCoding/assignment-5/task5-5.py"
```

Observation: The original code only handled male and female greetings, which excluded gender-neutral identities. The regenerated version introduced “Mx.” as a neutral option and allowed flexible input handling. This promotes inclusiveness and avoids discriminatory assumptions. The task highlights the role of ethical responsibility in designing user-centered applications.