

SR UNIVERSITY

AI ASSIST CODING

Lab-5.2

ROLL NO:2503A51L13

NAME: Begala Hasini

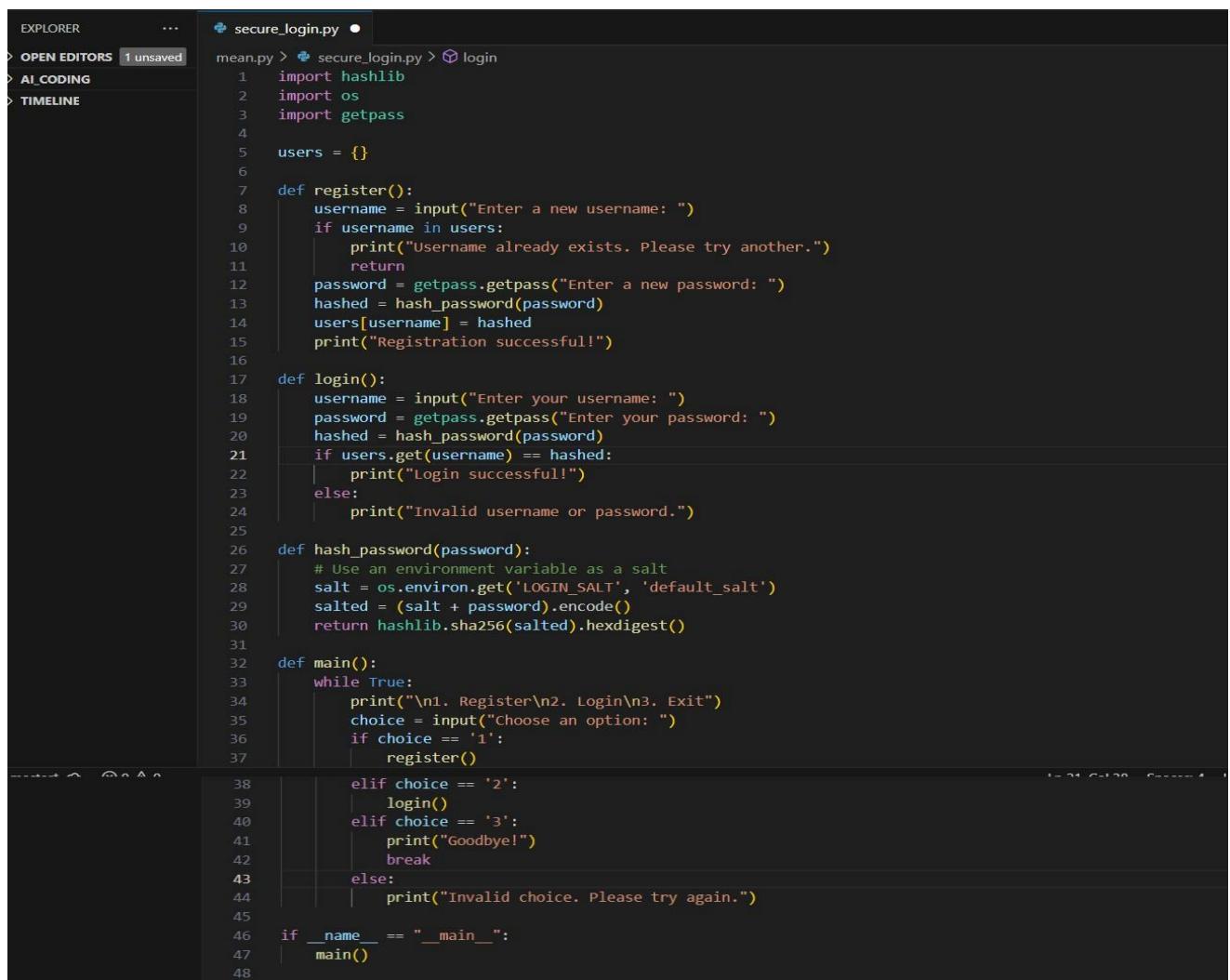
BATCH:19

TASK #1:

Prompt Used:

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:



The screenshot shows a code editor interface with a dark theme. The left sidebar has sections for EXPLORER, OPEN EDITORS (1 unsaved), AI_CODING, and TIMELINE. The main area displays a Python script named 'secure_login.py'. The code implements a simple registration and login system using the hashlib library for password hashing. It includes a salted hashing function and a main menu with three options: Register, Login, and Exit. The code is numbered from 1 to 48.

```
EXPLORER ...  
OPEN EDITORS 1 unsaved  
AI_CODING  
TIMELINE  
  
secure_login.py ●  
mean.py > secure_login.py > login  
1 import hashlib  
2 import os  
3 import getpass  
4  
5 users = {}  
6  
7 def register():  
8     username = input("Enter a new username: ")  
9     if username in users:  
10         print("Username already exists. Please try another.")  
11         return  
12     password = getpass.getpass("Enter a new password: ")  
13     hashed = hash_password(password)  
14     users[username] = hashed  
15     print("Registration successful!")  
16  
17 def login():  
18     username = input("Enter your username: ")  
19     password = getpass.getpass("Enter your password: ")  
20     hashed = hash_password(password)  
21     if users.get(username) == hashed:  
22         print("Login successful!")  
23     else:  
24         print("Invalid username or password.")  
25  
26 def hash_password(password):  
27     # Use an environment variable as a salt  
28     salt = os.environ.get('LOGIN_SALT', 'default_salt')  
29     salted = (salt + password).encode()  
30     return hashlib.sha256(salted).hexdigest()  
31  
32 def main():  
33     while True:  
34         print("\n1. Register\n2. Login\n3. Exit")  
35         choice = input("Choose an option: ")  
36         if choice == '1':  
37             register()  
38         elif choice == '2':  
39             login()  
40         elif choice == '3':  
41             print("Goodbye!")  
42             break  
43         else:  
44             print("Invalid choice. Please try again.")  
45  
46 if __name__ == "__main__":  
47     main()
```

Output After executing Code:

```

D:\IVE\Documents\AI_CODING\mean.py    secure_login.py
1. Register
2. Login
3. Exit
Choose an option: 1
Enter a new username: Simra
Enter a new password:
Registration successful!

1. Register
2. Login
3. Exit
Choose an option: 2
Enter your username: Simra
Enter your password:
Login successful!

1. Register
2. Login
3. Exit
choose an option: 2
3. Exit
Choose an option: 2
choose an option: 2
Enter your username: Simra
Enter your username: Simra
Enter your password:
Enter your password:
Invalid username or password.

1. Register
2. Login
3. Exit
choose an option: 3
Goodbye!

```

Observations:

- The first AI login system had unsafe practices, like storing passwords in plain text.
- The improved version uses password hashing and environment variables for safety.
- This demonstrates an awareness of protecting user data against breaches.

TASK #2:

Prompt Used:

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:

```

EXPLORER      ...
OPEN EDITORS  1 unsaved
AI CODING
TIMELINE

loan.py

mean.py > loan.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)

```

Output After executing Code:

```

DRIVE/DOCUMENTS/AI_CODING/mean.py/loan.py
Enter the number of applicants: 2

--- Applicant 1 ---
Enter applicant's name: John
Enter monthly salary: 50000
Enter monthly salary: 50000
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: 60000
Enter credit score: 620
Enter number of outstanding loans: 0

Loan Approved for John

--- Applicant 2 ---
Enter applicant's name: Priya
Enter monthly salary: 60000
Enter credit score: 620
Enter number of outstanding loans: 0

Evaluating loan application for Priya...
Loan Rejected for Priya

```

Observations:

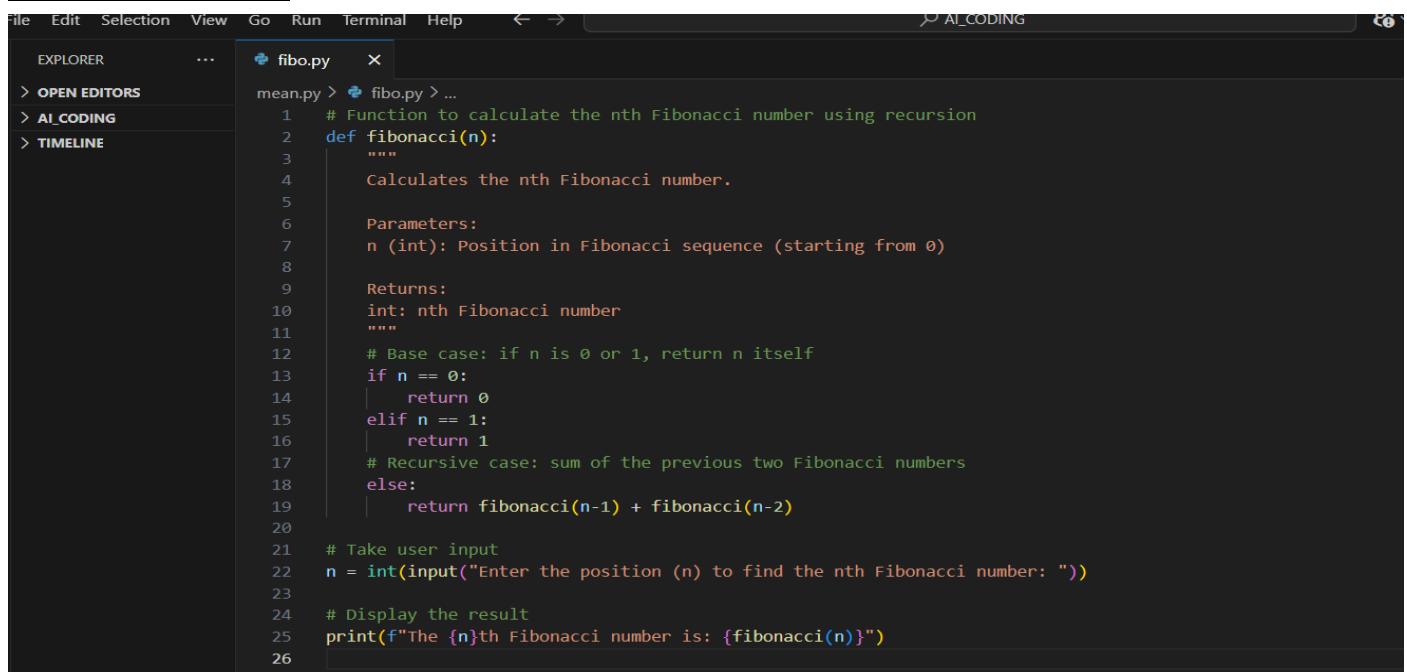
- The AI-generated loan approval system allowed users to input details for evaluation.
- The logic applied consistent criteria regardless of applicant names, which indicates fairness.
- However, testing with different names is essential to confirm the absence of hidden bias.
- This task emphasizes that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

TASK #3:

Prompt Used:

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

Code Generated:



```

File Edit Selection View Go Run Terminal Help ⌘ ⌘ AI_CODING
EXPLORER ... fibo.py ...
mean.py > fibo.py > ...
1 # Function to calculate the nth Fibonacci number using recursion
2 def fibonacci(n):
3     """
4         Calculates the nth Fibonacci number.
5
6         Parameters:
7             n (int): Position in Fibonacci sequence (starting from 0)
8
9         Returns:
10            int: nth Fibonacci number
11        """
12        # Base case: if n is 0 or 1, return n itself
13        if n == 0:
14            return 0
15        elif n == 1:
16            return 1
17        # Recursive case: sum of the previous two Fibonacci numbers
18        else:
19            return fibonacci(n-1) + fibonacci(n-2)
20
21        # Take user input
22        n = int(input("Enter the position (n) to find the nth Fibonacci number: "))
23
24        # Display the result
25        print(f"The {n}th Fibonacci number is: {fibonacci(n)}")
26

```

Output After executing Code:

```
Drive/Documents/AI_CODING/mean.py/fibo.py"
Enter the position (n) to find the nth Fibonacci number: 6
The 6th Fibonacci number is: 8
```

Observations:

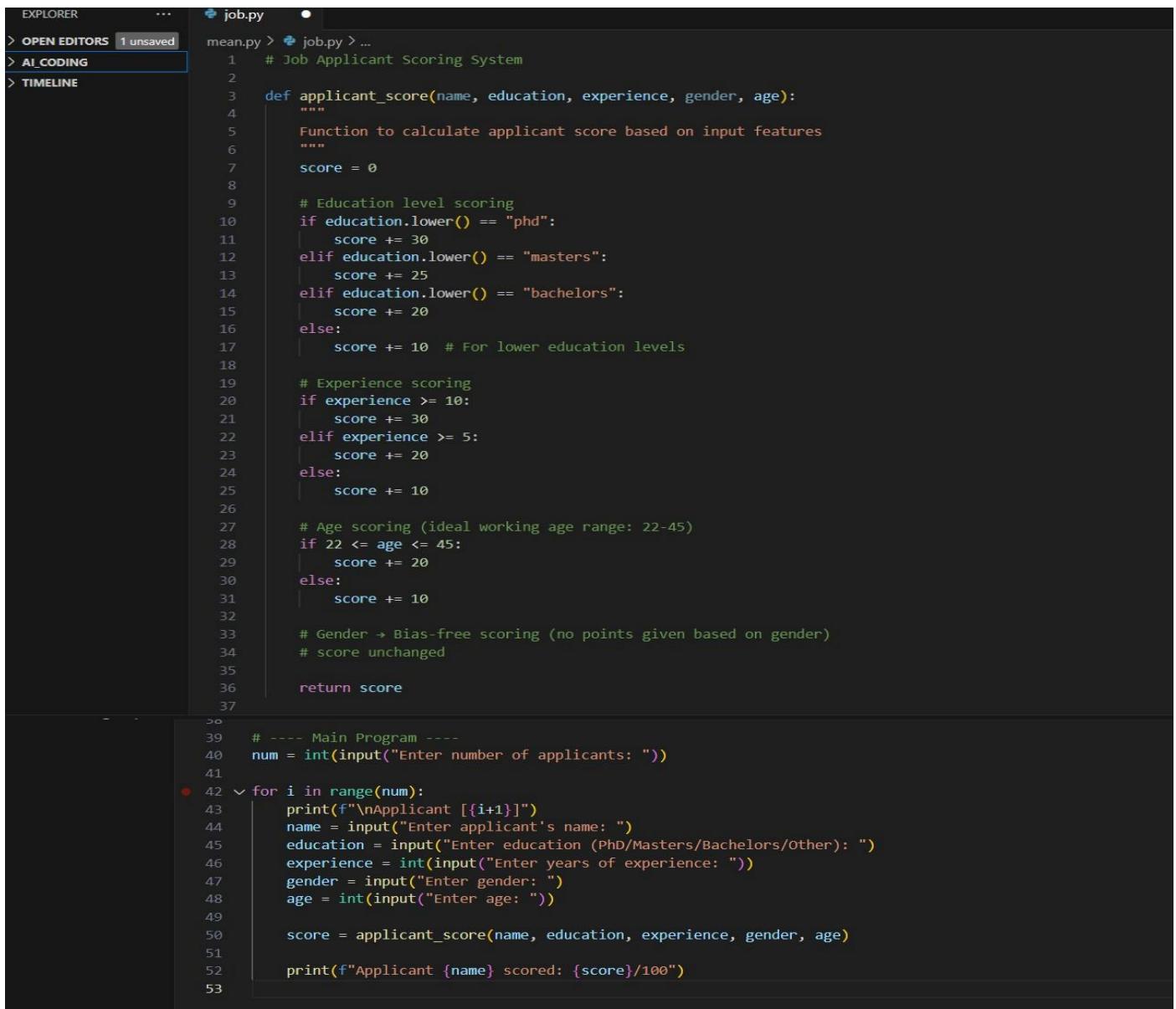
- The program correctly calculates the nth Fibonacci number using recursion.
- It allows dynamic input from the user instead of a fixed value.
- Demonstrates the use of base cases and recursion to solve problems.

TASK #4:

Prompt Used:

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:



```
EXPLORER ... ● job.py
> OPEN EDITORS 1 unsaved
> AI_CODING
> TIMELINE

job.py
mean.py > job.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"\nApplicant [{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")
```

Output After executing Code:

```

Enter number of applicants: 2
Applicant [1]
Enter applicant's name: Simra
Enter education (PhD/Masters/Bachelors/Other): Bachelors
Enter years of experience: 1
Enter gender: Female
Enter age: 21
Applicant Simra scored: 40/100

Applicant [2]
Enter applicant's name: Sam
Enter education (PhD/Masters/Bachelors/Other): PhD
Enter years of experience: 2
Enter gender: Male
Enter age: 24
Applicant Sam scored: 60/100

```

Python
Python

Observations:

- Education, experience, and age are main factors affecting the score.
- Gender is not used in scoring → bias-free approach.
- Scores are capped at 100 points.
- Ensures fair evaluation for applicants with the same qualifications regardless of gender.

TASK #5:

Prompt Used:

```

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

```

Code Generated:

```

2 def greet_user(name, gender):
3     name = (name or "").strip() or "User"
4     gender = (gender or "").strip().lower()
5
6     if gender == "male":
7         greeting = f"Hello Mr. {name}"
8     elif gender == "female":
9         greeting = f"Hello Ms. {name}"
10    elif gender == "neutral":
11        greeting = f"Hello Mx. {name}"
12    else:
13        greeting = f"Hello {name}"
14
15    return greeting
16
17 if __name__ == "__main__":
18     name = input("Enter your name: ").strip()
19     gender = input("Enter your gender (male/female/neutral): ").strip()
20     message = greet_user(name, gender)
21     print(message)
22 # ...existing code...

```

Output After executing Code:

```

Enter your name: Hasini
Enter your gender (male/female/neutral): female
Hello Ms. Hasini
PS C:\Users\HASINI\OneDrive\Desktop\ai_code>

```

Observations:

- The program defines a greet_user function that returns a greeting with an appropriate title.
- Gender input is converted to lowercase to handle different cases (e.g., Male, MALE, male).
- Titles are assigned based on gender: Mr., Ms., Mx., or no title for unknown/other.