SR UNIVERSITY

AI ASSIST CODING

Lab-5.2

ROLL NO:2503A51L13

NAME: Begala Hasini

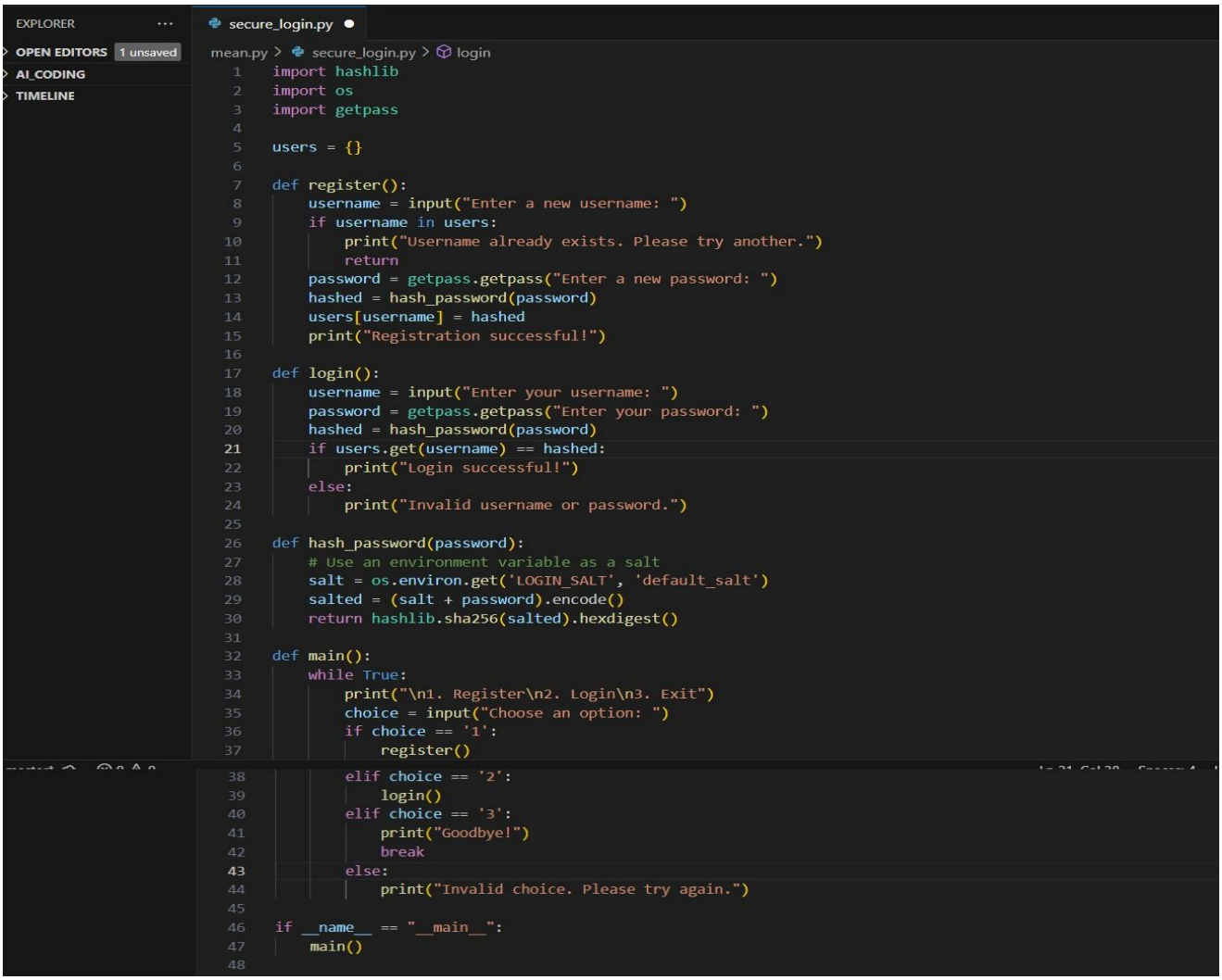
BATCH:19

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:



Output After executing Code:

```
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:
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 > ...

```python
# Loan approval program with user input

def evaluate_loan(name, salary, credit_score, outstanding_loans):
    print(f"\nEvaluating loan application for {name}...")

    # Simple approval criteria
    if salary >= 50000 and credit_score >= 700 and outstanding_loans == 0:
        print(f"Loan Approved for {name} ")
    else:
        print(f"Loan Rejected for {name} ")


# Take number of applicants
n = int(input("Enter the number of applicants: "))

for i in range(n):
    print(f"\n--- Applicant {i+1} ---")
    name = input("Enter applicant's name: ")
    salary = int(input("Enter monthly salary: "))
    credit_score = int(input("Enter credit score: "))
    outstanding_loans = int(input("Enter number of outstanding loans: "))

    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:

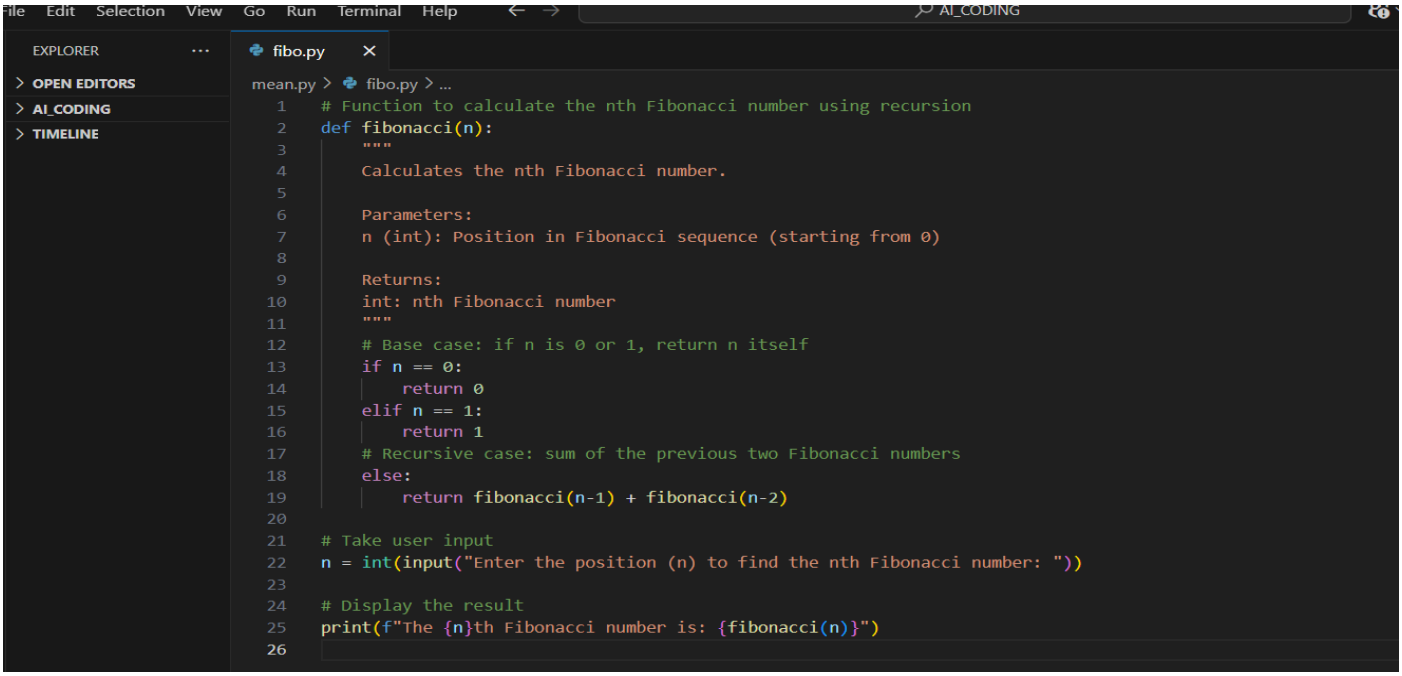
- The AI-generated loan approval system allowed users to input details for evaluation.
- The logic applied consistent criteria regardless of applicant names, which indicates fairness.
- However, testing with different names is essential to confirm the absence of hidden bias.
- This task emphasizes that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



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:

```python
# Job Applicant Scoring System

def applicant_score(name, education, experience, gender, age):
    """
    Function to calculate applicant score based on input features
    """
    score = 0

    # Education level scoring
    if education.lower() == "phd":
        score += 30
    elif education.lower() == "masters":
        score += 25
    elif education.lower() == "bachelors":
        score += 20
    else:
        score += 10  # For lower education levels

    # Experience scoring
    if experience >= 10:
        score += 30
    elif experience >= 5:
        score += 20
    else:
        score += 10

    # Age scoring (ideal working age range: 22-45)
    if 22 <= age <= 45:
        score += 20
    else:
        score += 10

    # Gender → Bias-free scoring (no points given based on gender)
    # score unchanged

    return score


# ---- Main Program ----
num = int(input("Enter number of applicants: "))

for i in range(num):
    print(f"\nApplicant [{i+1}]")
    name = input("Enter applicant's name: ")
    education = input("Enter education (PhD/Masters/Bachelors/Other): ")
    experience = int(input("Enter years of experience: "))
    gender = input("Enter gender: ")
    age = int(input("Enter age: "))

    score = applicant_score(name, education, experience, gender, age)

    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
```

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:

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

Code Generated:

```python
def greet_user(name, gender):
    name = (name or "").strip() or "User"
    gender = (gender or "").strip().lower()

    if gender == "male":
        greeting = f"Hello Mr. {name}"
    elif gender == "female":
        greeting = f"Hello Ms. {name}"
    elif gender == "neutral":
        greeting = f"Hello Mx. {name}"
    else:
        greeting = f"Hello {name}"

    return greeting

if __name__ == "__main__":
    name = input("Enter your name: ").strip()
    gender = input("Enter your gender (male/female/neutral): ").strip()
    message = greet_user(name, gender)
    print(message)
# ...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.