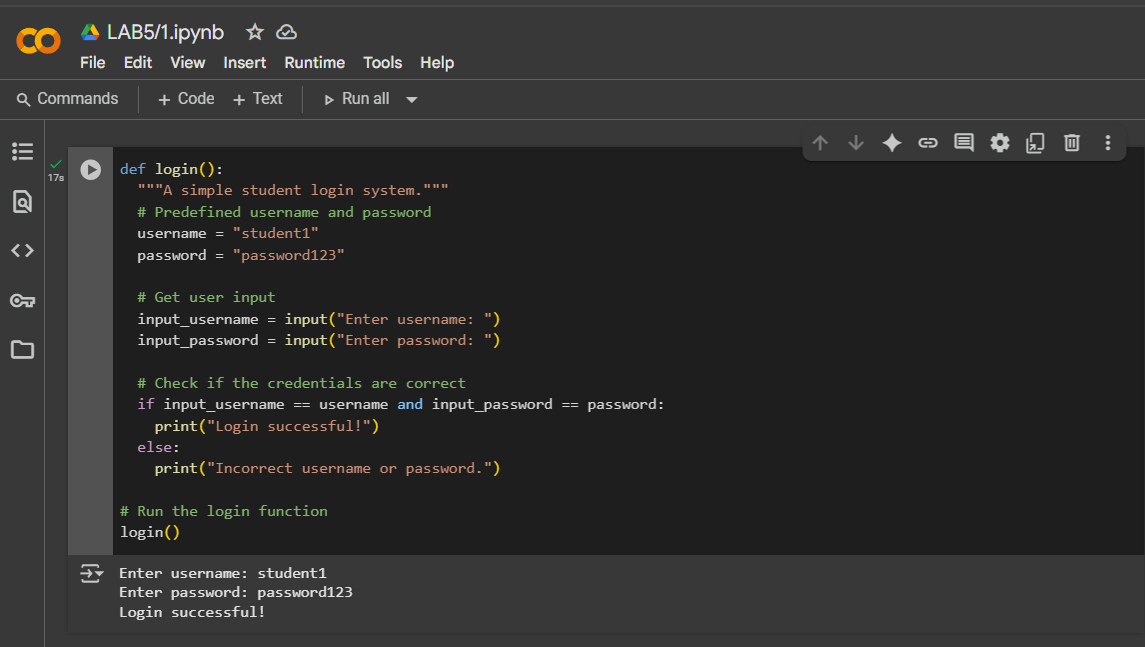
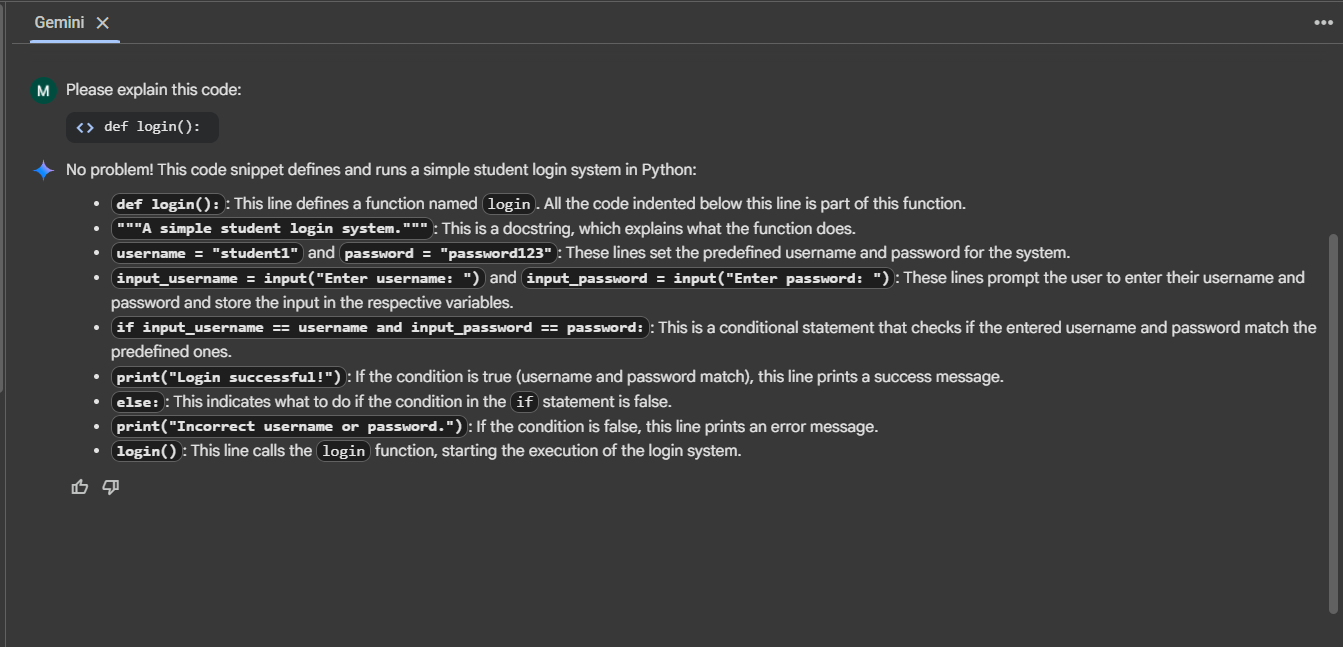
# ASSIGNMENT 5

TASK1

Task Description#1 (Privacy and Data Security)  
 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.





TASK2

Task Description#2 (Bias)  
 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

def evaluate\_loan\_application(applicant):

    """Evaluates a loan application based on predefined criteria and provides a reason.

    Args:

        applicant: A dictionary containing applicant details with keys

                   'credit\_score', 'income', and 'loan\_amount'.

    Returns:

        A tuple containing the status ("Approved" or "Denied") and a reason string.

    """

    if (applicant['credit\_score'] >= min\_credit\_score and

            applicant['income'] >= min\_income and

            applicant['loan\_amount'] <= max\_loan\_amount):

        return "Approved", "Meets all criteria."

    else:

        reasons = []

        if applicant['credit\_score'] < min\_credit\_score:

            reasons.append("Credit score is too low.")

        if applicant['income'] < min\_income:

            reasons.append("Income is too low.")

        if applicant['loan\_amount'] > max\_loan\_amount:

            reasons.append("Loan amount is too high.")

        return "Denied", " ".join(reasons)

min\_credit\_score = 700

min\_income = 50000

max\_loan\_amount = 250000

applicants = [

    {'name': 'John Doe', 'credit\_score': 700, 'income': 60000, 'loan\_amount': 200000},

    {'name': 'Priya Sharma', 'credit\_score': 720, 'income': 75000, 'loan\_amount': 300000},

    {'name': 'Michael Brown', 'credit\_score': 680, 'income': 45000, 'loan\_amount': 100000},

    {'name': 'Sarah Kim', 'credit\_score': 750, 'income': 90000, 'loan\_amount': 250000}

]

user\_name = input("Enter the name of the applicant: ")

found\_applicant = None

for applicant in applicants:

    if applicant['name'].lower() == user\_name.lower():

        found\_applicant = applicant

        break

if found\_applicant:

    status, reason = evaluate\_loan\_application(found\_applicant)

    print(f"\nLoan application for {found\_applicant['name']}:")

    print(f"Status: {status}")

    print(f"Reason: {reason}")

else:

    print(f"\nApplicant with name '{user\_name}' not found.")

Enter the name of the applicant: John Doe

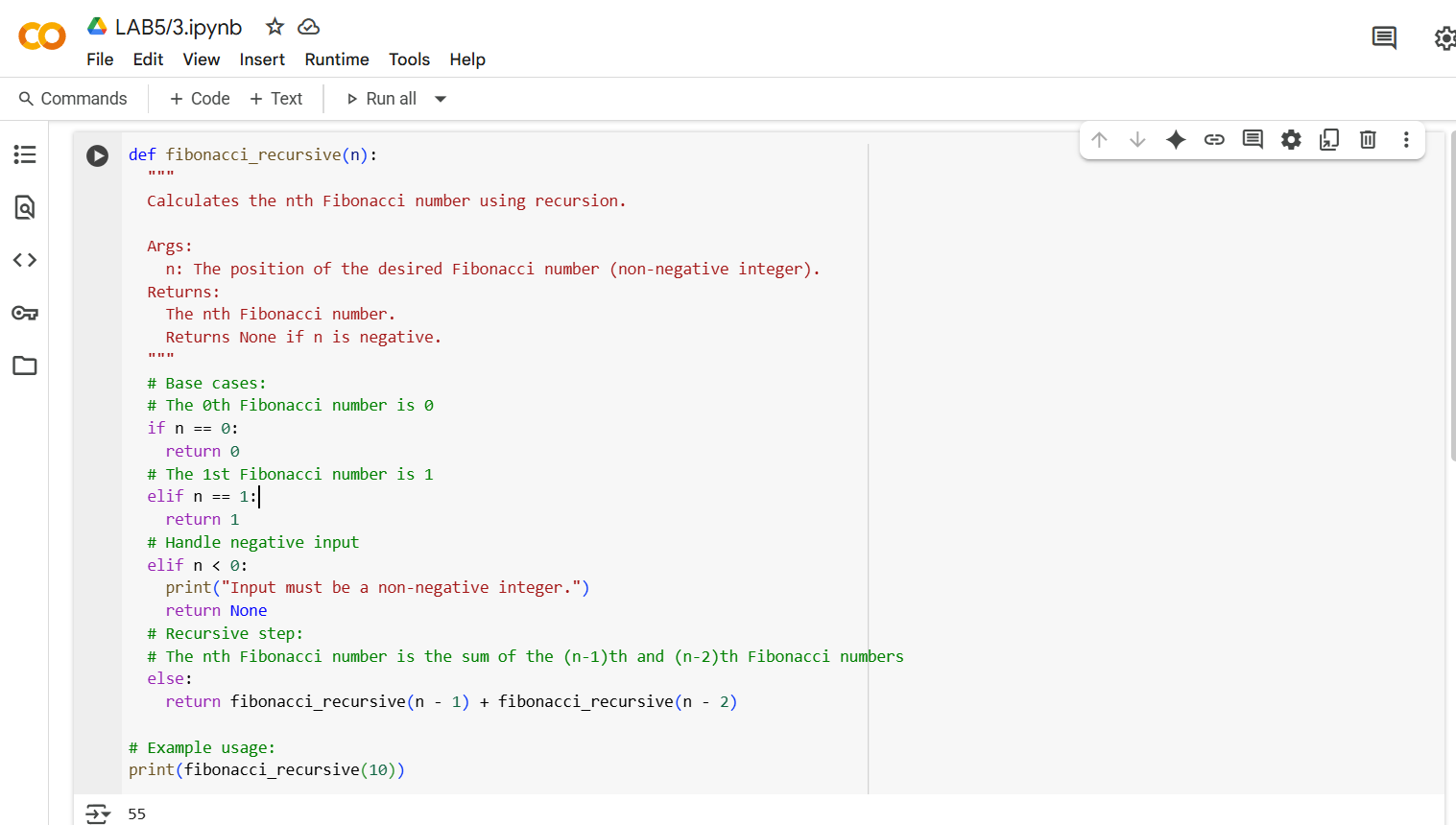
Loan application for John Doe:

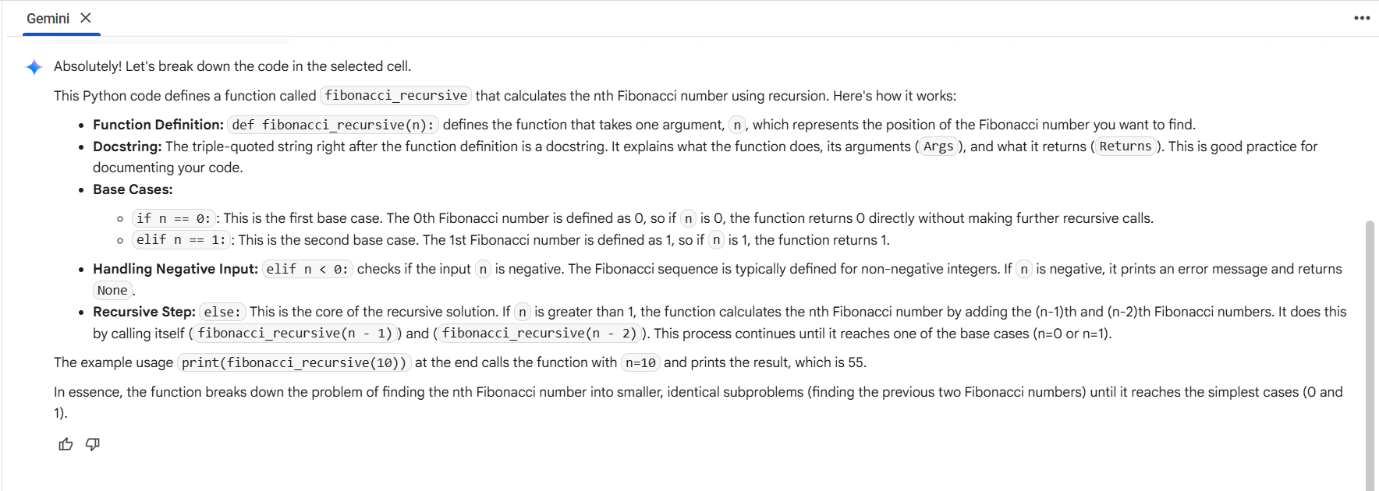
Status: Approved

Reason: Meets all criteria.

TASK3

Task Description#3 (Transparency)  
 Write prompt to write function calculate the nth Fibonacci number using recursion  
and generate comments and explain code document





TASK4

Task Description#4 (Bias)  
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

def score\_applicant(applicant, scoring\_criteria):

  """Calculates a score for a job applicant based on defined criteria.

  Args:

    applicant: A dictionary containing the applicant's features.

    scoring\_criteria: A dictionary containing the weights for each feature.

  Returns:

    The calculated score for the applicant.

  """

  score = 0

  for feature, weight in scoring\_criteria.items():

    if feature in applicant:

      score += applicant[feature] \* weight

  return score

# Predefined applicants

applicants = {

    1: {'education\_level': 4, 'years\_of\_experience': 6, 'technical\_skills': 7, 'communication\_skills': 8},

    2: {'education\_level': 5, 'years\_of\_experience': 8, 'technical\_skills': 9, 'communication\_skills': 7},

    3: {'education\_level': 3, 'years\_of\_experience': 4, 'technical\_skills': 6, 'communication\_skills': 5}

}

# Scoring criteria (assuming this is already defined in the environment)

# scoring\_criteria = {'education\_level': 0.2, 'years\_of\_experience': 0.5, 'technical\_skills': 0.2, 'communication\_skills': 0.1}

print("Predefined applicants:")

for num, applicant\_data in applicants.items():

    print(f"Applicant {num}: {applicant\_data}")

# Get input from the user

try:

    applicant\_number = int(input("Enter the applicant number to get the score: "))

    if applicant\_number in applicants:

        selected\_applicant = applicants[applicant\_number]

        # Calculate and print the score

        applicant\_score = score\_applicant(selected\_applicant, scoring\_criteria)

        print(f"The score for Applicant {applicant\_number} is: {applicant\_score}")

    else:

        print("Invalid applicant number.")

except ValueError:

    print("Invalid input. Please enter a number.")

Predefined applicants:

Applicant 1: {'education\_level': 4, 'years\_of\_experience': 6, 'technical\_skills': 7, 'communication\_skills': 8}

Applicant 2: {'education\_level': 5, 'years\_of\_experience': 8, 'technical\_skills': 9, 'communication\_skills': 7}

Applicant 3: {'education\_level': 3, 'years\_of\_experience': 4, 'technical\_skills': 6, 'communication\_skills': 5}

Enter the applicant number to get the score: 1

The score for Applicant 1 is: 6.0

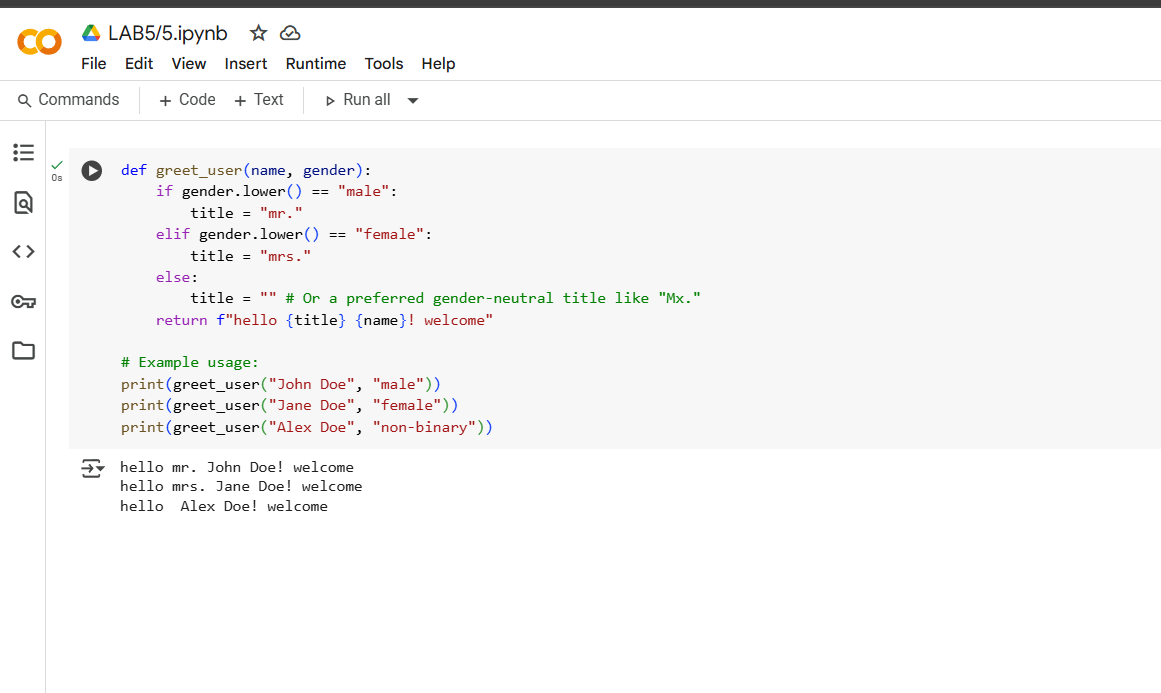
TASK5

Task Description#5 (Inclusiveness)

Code Snippet

**A screenshot of a computer

AI-generated content may be incorrect.**

****

