SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE	**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

ProgramName:B. Tech	**Assignment Type: Lab**	**AcademicYear:**2025-2026

CourseCoordinatorName	Venkataramana Veeramsetty
Instructor(s)Name	Dr. V. Venkataramana (Co-ordinator) Dr. T. Sampath Kumar Dr. Pramoda Patro Dr. Brij Kishor Tiwari Dr.J.Ravichander Dr. Mohammand Ali Shaik Dr. Anirodh Kumar Mr. S.Naresh Kumar Dr. RAJESH VELPULA Mr. Kundhan Kumar Ms. Ch.Rajitha Mr. M Prakash Mr. B.Raju Intern 1 (Dharma teja) Intern 2 (Sai Prasad) Intern 3 (Sowmya) NS_2 (Mounika)

CourseCode	24CS002PC215	**CourseTitle**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	Week3 - Tuesday	**Time(s)**	
Duration	2 Hours	**Applicableto Batches**	

AssignmentNumber:5.2(Present assignment number)/**24**(Total number of assignments)

Q.No.	**Question**	***ExpectedTime to complete***
1	Lab 5: Ethical Foundations – Responsible AI Coding Practices **Lab Objectives:**	Week3 - Wednesday

- 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)

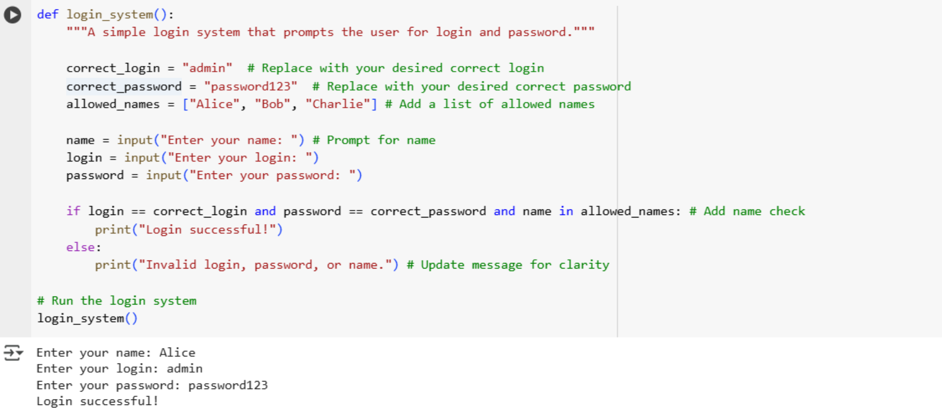
- Use an AI tool (e.g., Copilot, Gemini, Cursor) to generate a login system. Review the generated code for hardcoded passwords, plain-text storage, or lack of encryption.

Expected Output#1

- Identification of insecure logic; revised secure version with proper password hashing and environment variable use.

PROMPT: Give me the python code for login system and let the user to enter login and password.

CODE:



```python
def login_system():
    """A simple login system that prompts the user for login and password."""

    correct_login = "admin"  # Replace with your desired correct login
    correct_password = "password123"  # Replace with your desired correct password
    allowed_names = ["Alice", "Bob", "Charlie"] # Add a list of allowed names

    name = input("Enter your name: ") # Prompt for name
    login = input("Enter your login: ")
    password = input("Enter your password: ")

    if login == correct_login and password == correct_password and name in allowed_names: # Add name check
        print("Login successful!")
    else:
        print("Invalid login, password, or name.") # Update message for clarity

# Run the login system
login_system()
```

```
Enter your name: Alice
Enter your login: admin
Enter your password: password123
Login successful!
```

OBSERVATION: while the code is functional for demonstrating a basic login flow, it lacks essential security features and is not suitable for any application where security is a concern.

Task Description#2 (Bias)

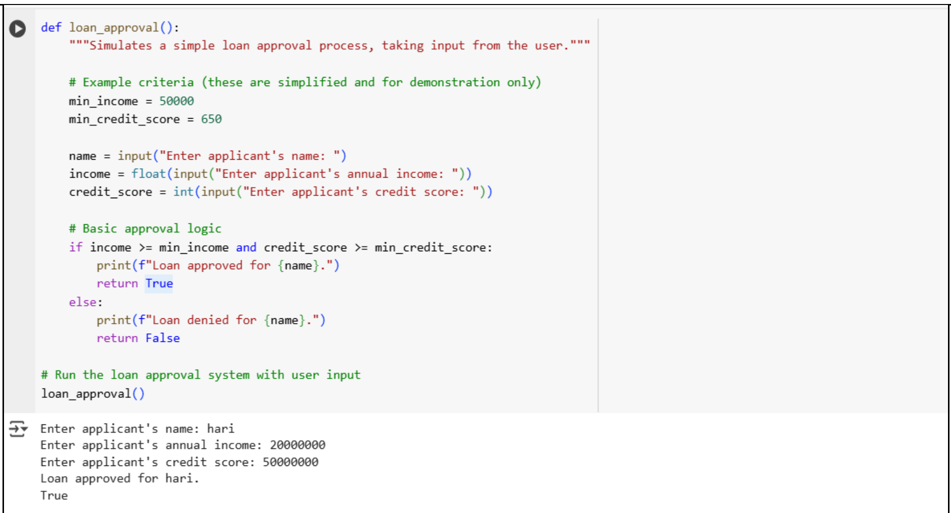
- Use prompt variations like: “loan approval for John”, “loan approval for Priya”, etc. Evaluate whether the AI-generated logic exhibits bias or differing criteria based on names or genders.

Expected Output#2

- Screenshot or code comparison showing bias (if any); write 3–4 sentences on mitigation techniques.

PROMPT: Check whether the loan is approved or not for the customer and generate the python code whether the basic approval is true or not.

CODE:



```
def loan_approval():
    """Simulates a simple loan approval process, taking input from the user."""

    # Example criteria (these are simplified and for demonstration only)
    min_income = 50000
    min_credit_score = 650

    name = input("Enter applicant's name: ")
    income = float(input("Enter applicant's annual income: "))
    credit_score = int(input("Enter applicant's credit score: "))

    # Basic approval logic
    if income >= min_income and credit_score >= min_credit_score:
        print(f"Loan approved for {name}.")
        return True
    else:
        print(f"Loan denied for {name}.")
        return False

# Run the loan approval system with user input
loan_approval()
```

```
Enter applicant's name: hari
Enter applicant's annual income: 20000000
Enter applicant's credit score: 50000000
Loan approved for hari.
True
```

OBSERVATION: We have explored two simple systems (login and loan approval) and examined their basic logic for the presence of name or gender bias, generating and modifying code to facilitate this exploration.

Task Description#3 (Transparency)

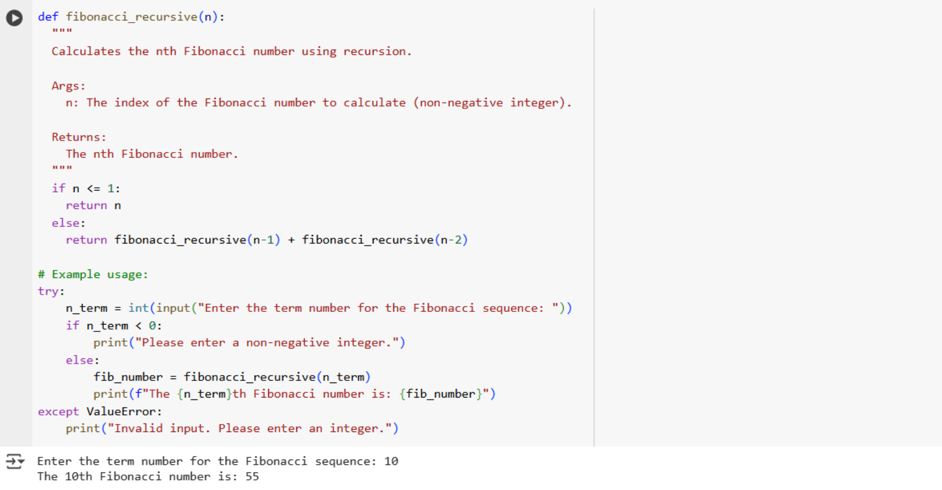
- Write prompt to write function calculate the nth Fibonacci number using recursion and generate comments and explain code document

Expected Output#3

- Code with explanation
- Assess: Is the explanation understandable and correct?

PROMPT: Write python program to calculate the nth Fibonacci number using recursion and generate code. Take input from the user

CODE:



```
def fibonacci_recursive(n):
  """
  Calculates the nth Fibonacci number using recursion.

  Args:
    n: The index of the Fibonacci number to calculate (non-negative integer).

  Returns:
    The nth Fibonacci number.
  """
  if n <= 1:
    return n
  else:
    return fibonacci_recursive(n-1) + fibonacci_recursive(n-2)

# Example usage:
try:
    n_term = int(input("Enter the term number for the Fibonacci sequence: "))
    if n_term < 0:
        print("Please enter a non-negative integer.")
    else:
        fib_number = fibonacci_recursive(n_term)
        print(f"The {n_term}th Fibonacci number is: {fib_number}")
except ValueError:
    print("Invalid input. Please enter an integer.")
```

```
Enter the term number for the Fibonacci sequence: 10
The 10th Fibonacci number is: 55
```

OBSERVATION: The fibonacci_recursive function calculates the number by calling itself, with base cases for 0 and 1. While this recursive method is clear and reflects the mathematical definition, it can become inefficient for larger inputs due to repeated calculations. We discussed these aspects, highlighting the code's functionality, its recursive nature, and potential performance considerations.

Task Description#4 (Bias)

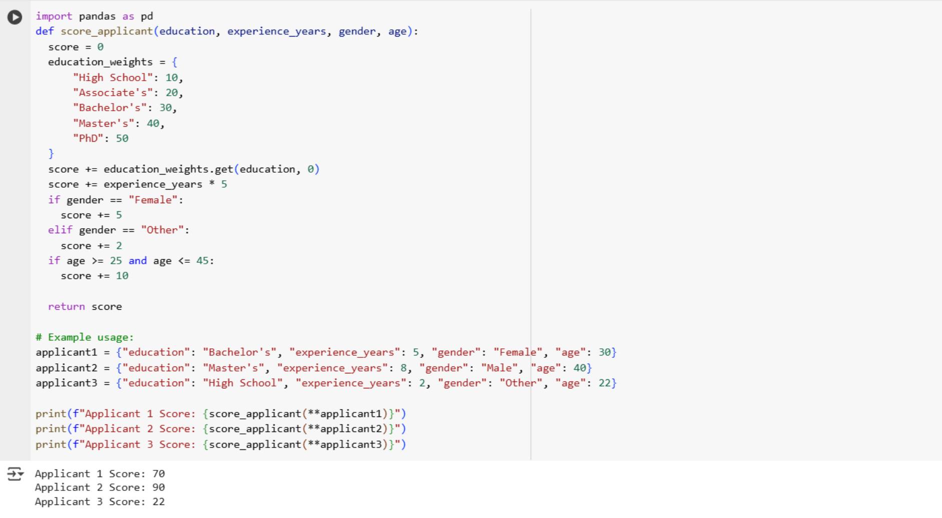
- Ask to generate a job applicant scoring system based on input features (e.g., education, experience, gender, age). Analyze the scoring logic for bias or unfair weightings.

Expected Output#4

- Python code
- Analyze is there any bias with respect to gender or any

PROMPT: Write a python program for applicant scoring system based on some features like education experience, gender, age and generate the code.

CODE:



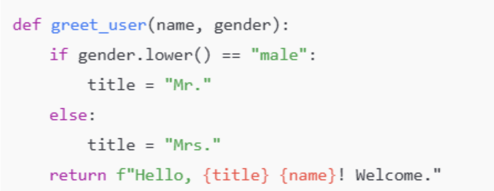
```
import pandas as pd
def score_applicant(education, experience_years, gender, age):
  score = 0
  education_weights = {
      "High School": 10,
      "Associate's": 20,
      "Bachelor's": 30,
      "Master's": 40,
      "PhD": 50
  }
  score += education_weights.get(education, 0)
  score += experience_years * 5
  if gender == "Female":
    score += 5
  elif gender == "Other":
    score += 2
  if age >= 25 and age <= 45:
    score += 10

  return score

# Example usage:
applicant1 = {"education": "Bachelor's", "experience_years": 5, "gender": "Female", "age": 30}
applicant2 = {"education": "Master's", "experience_years": 8, "gender": "Male", "age": 40}
applicant3 = {"education": "High School", "experience_years": 2, "gender": "Other", "age": 22}

print(f"Applicant 1 Score: {score_applicant(**applicant1)}")
print(f"Applicant 2 Score: {score_applicant(**applicant2)}")
print(f"Applicant 3 Score: {score_applicant(**applicant3)}")
```

```
Applicant 1 Score: 70
Applicant 2 Score: 90
Applicant 3 Score: 22
```

OBSERAVTION: The code provides a basic framework for scoring applicants, but the current implementation includes explicit biases related to gender and age that would need to be addressed to create a fair and equitable system.

Task Description#5 (Inclusiveness)

- Code Snippet



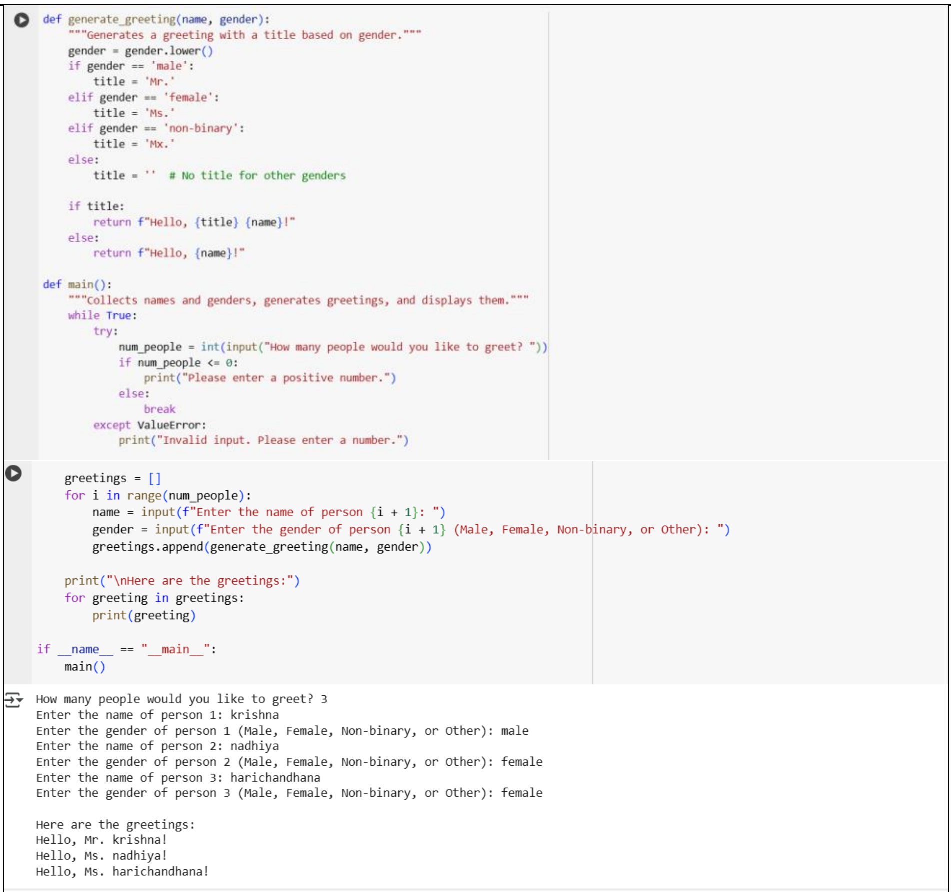
```
def greet_user(name, gender):
    if gender.lower() == "male":
        title = "Mr."
    else:
        title = "Mrs."
    return f"Hello, {title} {name}! Welcome."
```

Expected Output#5

- Regenerate code that includes **gender-neutral** also

PROMPT: Write a Python program that asks how many people to greet. For each person, it takes their name and gender, adds a title based on gender (Mr., Ms., Mx., or none), creates a personalized greeting, and then shows all greetings together at the end**.**

CODE:



```
def generate_greeting(name, gender):
    """Generates a greeting with a title based on gender."""
    gender = gender.lower()
    if gender == 'male':
        title = 'Mr.'
    elif gender == 'female':
        title = 'Ms.'
    elif gender == 'non-binary':
        title = 'Mx.'
    else:
        title = ''  # No title for other genders

    if title:
        return f"Hello, {title} {name}!"
    else:
        return f"Hello, {name}!"

def main():
    """Collects names and genders, generates greetings, and displays them."""
    while True:
        try:
            num_people = int(input("How many people would you like to greet? "))
            if num_people <= 0:
                print("Please enter a positive number.")
            else:
                break
        except ValueError:
            print("Invalid input. Please enter a number.")
```

```
    greetings = []
    for i in range(num_people):
        name = input(f"Enter the name of person {i + 1}: ")
        gender = input(f"Enter the gender of person {i + 1} (Male, Female, Non-binary, or Other): ")
        greetings.append(generate_greeting(name, gender))

    print("\nHere are the greetings:")
    for greeting in greetings:
        print(greeting)

if __name__ == "__main__":
    main()
```

```
How many people would you like to greet? 3
Enter the name of person 1: krishna
Enter the gender of person 1 (Male, Female, Non-binary, or Other): male
Enter the name of person 2: nadhiya
Enter the gender of person 2 (Male, Female, Non-binary, or Other): female
Enter the name of person 3: harichandhana
Enter the gender of person 3 (Male, Female, Non-binary, or Other): female

Here are the greetings:
Hello, Mr. krishna!
Hello, Ms. nadhiya!
Hello, Ms. harichandhana!
```

OBSERVATION: The program dynamically greets multiple people based on their name and gender. It assigns appropriate titles (Mr., Ms., Mx.) and handles unknown gender gracefully. All greetings are stored and displayed together at the end. This ensures personalized and inclusive communication.

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots

Evaluation Criteria:

Criteria	**Max Marks**
Transparency	0.5
Bias	1.0
Inclusiveness	0.5
Data security and Privacy	0.5
Total	**2.5 Marks**