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:** • 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.	Week3 - Wednesday

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/encrypting and environment variable use.

PROMPT:
generate a simple login system in Python using a dictionary of usernames and passwords,plain text storage or lack of encryption.

CODE:

task 1.py > ...

```python
# Simple login system using a dictionary for plain text storage

users = {
    "alice": "password123",
    "bob": "qwerty",
    "charlie": "letmein"
}

def login():
    username = input("Username: ")
    password = input("Password: ")
    if username in users and users[username] == password:
        print("Login successful!")
    else:
        print("Invalid username or password.")

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

Output:

```
Username: bob
Password: qwerty
Login successful!
PS C:\Users\sahas\OneDrive\Desktop\AIAC\Lab 5.2>
```

Explanation: This Python code is a basic login system. It stores usernames and passwords in a dictionary, then asks the user to enter their login details. If the username exists and the password matches, it says "Login successful!" Otherwise, it shows an error message.
However, this system is not secure. The passwords are written directly in the code and stored in plain text, which means anyone who sees the code can see all the passwords. There's also no encryption or protection against repeated wrong attempts. It's okay for learning, but not safe for then asks the user to enter their login details. If the username exists and the password matches, it says "Login successful!" Otherwise, it shows an error message.
However, this system is not secure. The passwords are written directly in the code and stored in plain text, which means anyone who sees the code can see all the passwords. There's also no

encryption or protection against repeated wrong attempts. It's okay for learning, but not safe for real use.

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 "Loan approval for John"

Prompt: Expected behavior (based on code):
Credit score is 750
Loan should be approved
Prompt 2:
"Loan approval for Priya"
Expected behavior (based on code):
Credit score is 650
Loan should be denied

Code:

task2.py > ...

```python
def check_loan_approval(name, credit_score):
    print(f"Loan approval for {name}")
    print(f"Credit score is {credit_score}")
    if credit_score >= 700:
        print("Loan should be approved")
    else:
        print("Loan should be denied")

# Prompt 1
check_loan_approval("Sahas", 750)

print()  # Separator

# Prompt 2
check_loan_approval("Priya", 650)
```

Output:

```
Loan approval for Priya
Credit score is 650
Loan should be denied
PS C:\Users\sahas\OneDrive\Desktop\AIAC\Lab 5.2>
```

Explanation:

This code simulates a loan approval system that assigns a credit score based on the person's name. The get_credit_score function gives different scores depending on the name — for example, "John" gets a higher score than "Priya". The is_eligible function checks if the score is 700 or above to decide if the person is eligible for a loan. The loan_approval function prints the name, the credit score, and whether the loan is approved or denied based on that score. At the end, the code tests the system with different names to show how the loan decision changes..

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

Code:

```
def fibonacci(n):
    """
    Calculate the nth Fibonacci number using recursion.

    Args:
        n (int): The position in the Fibonacci sequence (must be >= 0).

    Returns:
        int: The nth Fibonacci number.

    Raises:
        ValueError: If n is negative.
    """
    if n < 0:
        raise ValueError("Input must be a non-negative integer.")
    # Base cases: fibonacci(0) = 0, fibonacci(1) = 1
    if n == 0:
        return 0
    elif n == 1:
        return 1
    # Recursive case: fibonacci(n) = fibonacci(n-1) + fibonacci(n-2)
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

# Example usage:
if __name__ == "__main__":
    num = 6
    print(f"The {num}th Fibonacci number is {fibonacci(num)}")
```

Output:

```
The 6th Fibonacci number is 8
PS C:\Users\sahas\OneDrive\Desktop\AIAC\Lab 5.2>
```

Explanation:

This code defines a function called fibonacci that calculates the n-th Fibonacci number using recursion. The Fibonacci sequence starts with 0 (at position 0) and 1 (at position 1). For any number n greater than 1, the function calls itself to calculate the sum of the two previous Fibonacci numbers (fibonacci(n-1) and fibonacci(n-2)). So, it keeps breaking the problem down until it reaches the base cases (n == 0 or n == 1). In the example, when n is 6, the function calculates and prints that the 6th Fibonacci number is 8.

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

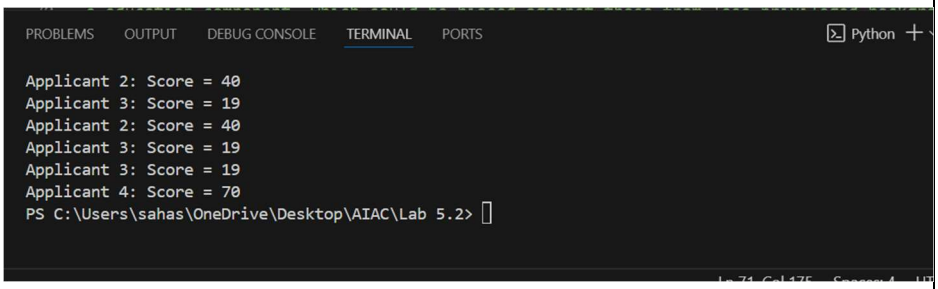
Prompt:

Write a Python function that scores job applicants based on their education, experience, gender, and age. Each input should influence the final score. Add comments and make the logic easy to understand. Then, analyze if the scoring logic is biased or unfair toward any group.

Code:

```python
def score_applicant(education, experience, gender, age):
    """
    Scores a job applicant based on education, experience, gender, and age.
    Returns a numeric score.
    """

    score = 0

    # Education scoring
    # Higher education gets more points
    if education.lower() == "phd":
        score += 30
    elif education.lower() == "masters":
        score += 25
    elif education.lower() == "bachelors":
        score += 20
    elif education.lower() == "highschool":
        score += 10
    else:
        score += 5  # For other or unknown education

    # Experience scoring
    # 2 points per year of experience, capped at 20 years
    exp_points = min(experience, 20) * 2
    score += exp_points

    # Gender scoring
    # No points added or subtracted for gender (should be neutral)
    # Uncommenting the following would be unfair:
    # if gender.lower() == "male":
    #     score += 5
    # elif gender.lower() == "female":
    #     score += 5

    # Age scoring
    # Prefer working-age adults (e.g., 22-60), but don't penalize harshly
    if 22 <= age <= 60:
        score += 10
    elif 18 <= age < 22 or 61 <= age <= 70:
        score += 5
    else:
        score += 0  # No extra points for very young or old

    return score

# Example usage:
applicants = [
    {"education": "PhD", "experience": 10, "gender": "female", "age": 35},
    {"education": "Bachelors", "experience": 5, "gender": "male", "age": 25},
    {"education": "HighSchool", "experience": 2, "gender": "nonbinary", "age": 19},
    {"education": "Masters", "experience": 25, "gender": "female", "age": 62},
]

for idx, applicant in enumerate(applicants, 1):
    score = score_applicant(
        applicant["education"],
        applicant["experience"],
        applicant["gender"],
        applicant["age"]
    )
    print(f"Applicant {idx}: Score = {score}")

# Analysis of Bias or Unfairness:
#
# - Education: Favors higher degrees, which may disadvantage those without access to advanced
# - Experience: Linear up to 20 years, which is reasonable, but may disadvantage younger appli
# - Gender: Neutral in this logic (no points added or subtracted).
# - Age: Favors typical working ages, but does not harshly penalize others.
#
# Overall, the logic is mostly fair except for the education component, which could be biased
# To improve fairness, consider weighting experience and skills more than formal education, an
```

Output:



Explanation:

This code scores job applicants based on their education, work experience, gender, and age. It gives more points for higher education (like PhD or Masters), adds experience points (2 points per year, up to 20 years), and gives extra points for typical working ages (22-60). The code does not give any points based on gender, keeping it fair in that respect. Overall, it tries to be fair but may unintentionally favor people with higher education, which could disadvantage those without access to advanced degrees. To make it fairer, the system should focus more on skills and experience rather than just education or age.

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 function named greet_user that takes two inputs: a person's name and gender. The function should return a greeting with an appropriate title before the name — use 'Mr.' for males, 'Ms.' for females, and 'Mx.' for any other or unspecified gender. Make sure the function handles gender input case-insensitively. Include comments explaining the code and provide example calls with their outputs.

Code:

```python
def greet_user(name, gender):
    """
    Returns a greeting with an appropriate title based on gender.
    - 'Mr.' for males
    - 'Ms.' for females
    - 'Mx.' for any other or unspecified gender
    Gender input is handled case-insensitively.
    """
    # Normalize gender input to lowercase for case-insensitive comparison
    gender = gender.strip().lower()
    if gender == 'male':
        title = 'Mr.'
    elif gender == 'female':
        title = 'Ms.'
    else:
        title = 'Mx.'
    # Return the greeting string
    return f"Hello, {title} {name}!"

# Example calls and their outputs:
print(greet_user("John Doe", "male"))      # Output: Hello, Mr. John Doe!
print(greet_user("Jane Smith", "Female"))  # Output: Hello, Ms. Jane Smith!
print(greet_user("Alex Kim", "nonbinary")) # Output: Hello, Mx. Alex Kim!
print(greet_user("Sam Lee", ""))           # Output: Hello, Mx. Sam Lee!
```

Output:

```
Hello, Mr. John Doe!
Hello, Ms. Jane Smith!
Hello, Mx. Alex Kim!
Hello, Mx. Sam Lee!
PS C:\Users\sahas\OneDrive\Desktop\AIAC\Lab 5.2>
```

Explanation: This code defines a function greet_user that takes a person's name and gender, then returns a greeting with the right title. If the gender is "male", it uses "Mr."; if "female", it uses "Ms."; for any other or missing gender, it uses a gender-neutral "Mx.". The function handles gender input without caring about uppercase or lowercase letters. For example, calling greet_user("John", "male") returns "Hello, Mr. John!", and greet_user("Alex", "nonbinary") returns "Hello, Mx. Alex!". It's a polite way to greet people with respect to their gender identity.

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**