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 : write a python code that generate a login system.review the generated code for hardcoded password,plain-text storage,or lack of encryption with(register,login,exit).

sk1.py > main

```python
import hashlib
import getpass

def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()

def register(users):
    username = input("Enter a new username: ").strip()
    if username in users:
        print("Username already exists. Please try another.")
        return
    password = getpass.getpass("Enter a new password: ")
    confirm = getpass.getpass("Confirm password: ")
    if password != confirm:
        print("Passwords do not match.")
        return
    users[username] = hash_password(password)
    print("Registration successful.")
```

```
def login(users):
    username = input("Enter your username: ").strip()
    if username not in users:
        print("Username not found.")
        return
    password = getpass.getpass("Enter your password: ")
    if users[username] == hash_password(password):
        print("Login successful. Welcome,", username)
    else:
        print("Incorrect password.")

def main():
    users = {}
    while True:
        print("\n1. Register\n2. Login\n3. Exit")
        choice = input("Select an option: ").strip()
        if choice == '1':
            register(users)
        elif choice == '2':
            login(users)
        elif choice == '3':
            print("Exiting program.")
            break
        else:
            print("Invalid option. Please try again.")

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

OUTPUT:

```
1. Register
2. Login
3. Exit
Select an option: 1
Enter a new username: Deeksha
Enter a new password:
```

```
Confirm password:
Registration successful.

1. Register
2. Login
3. Exit
Select an option: 2
Enter your username: Deeksha
Enter your password:
Login successful. Welcome, Deeksha

1. Register
2. Login
3. Exit
1. Register
2. Login
3. Exit
Select an option: 3
Exiting program.
```

EXPLANATION :

This code implements a simple command-line user registration and login system in Python.. Passwords are securely hashed using SHA-256 before being stored in a dictionary.The 'register' function allows new users to create an account with a unique username and password. The 'login' function checks user credentials by comparing the hashed password input to the stored hash. The main loop provides a menu for users to register, log in, or exit the program.

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 : write a python function that generate 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.

```python
def loan_approval_john(income, credit_score):
    if income > 30000 and credit_score > 650:
        return "Approved"
    else:
        return "Denied"

def loan_approval_priya(income, credit_score):
    if income > 35000 and credit_score > 700:
        return "Approved"
    else:
        return "Denied"

def loan_approval_ahmed(income, credit_score):
    if income > 32000 and credit_score > 670:
        return "Approved"
    else:
        return "Denied"

def loan_approval_emily(income, credit_score):
    if income > 28000 and credit_score > 630:
```

```python
def loan_approval_emily(income, credit_score):
        return "Approved"
    else:
        return "Denied"
applicants = [
    ("John", loan_approval_john, 32000, 660),
    ("Priya", loan_approval_priya, 36000, 710)
]

for name, func, income, credit_score in applicants:
    result = func(income, credit_score)
    print(f"Applicant: {name}, Income: {income}, Credit Score: {credit_score} => {result}")

print("\nAnalysis:")
print("The approval criteria differ for each applicant based on their name, which may reflect bias in the logic."
```

OUTPUT:

```
ython3.11.exe c:/Users/DEEKSHA/OneDrive/Desktop/AIAC/Lab-5.2/Task2.py
Applicant: John, Income: 32000, Credit Score: 660 => Approved
Applicant: Priya, Income: 36000, Credit Score: 710 => Approved

Analysis:
The approval criteria differ for each applicant based on their name, which may reflect bias in the logic.
```

EXPLANATION :

Defines different loan approval rules for each applicant.
Associates each applicant with their rule and data.
Applies the rule to decide approval or denial.
Prints each applicant's result.
Notes possible bias from using different criteria.

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 the python function to calculate the nth Fibonacci number using recursion with user input.

```
k3.py > ...
def fibonacci(n):
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci(n-1) + fibonacci(n-2)

try:
    num = int(input("Enter the value of n: "))
    print(f"The {num}th Fibonacci number is: {fibonacci(num)}")
except ValueError:
    print("Please enter a valid integer.")
```

OUTPUT:

```
Enter the value of n: 15
The 15th Fibonacci number is: 610
```

EXPLANATION :

This code defines a recursive function to compute the nth Fibonacci number.
It prompts the user to input a value for n.
If the input is valid, it prints the nth Fibonacci number.
If the input is invalid (not an integer), it prints an error message.
The Fibonacci function uses recursion: fibonacci(n) = fibonacci(n-1) + fibonacci(n-2)

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

PROMPT: write a python function code that 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.

```python
def score_applicant(education, experience, gender, age):
    score = 0
    education_weights = {
        'highschool': 10,
        'bachelor': 20,
        'master': 30,
        'phd': 40
    }
    score += education_weights.get(education.lower(), 0)
    if experience < 0:
        experience_points = 0
    elif experience < 2:
        experience_points = 5
    elif experience < 5:
        experience_points = 15
    elif experience < 10:
        experience_points = 25
    else:
        experience_points = 35
    score += experience_points
def score_applicant(education, experience, gender, age):
    if gender.lower() in ['male', 'female', 'other']:
        gender_points = 0
    else:
        gender_points = 0
    score += gender_points
    if age < 18:
        age_points = 0
    elif 18 <= age < 25:
        age_points = 5
    elif 25 <= age < 40:
        age_points = 10
    elif 40 <= age < 60:
        age_points = 8
    else:
        age_points = 5
    score += age_points

    return score
```

```
def analyze_scoring_logic():
    print("Analyzing scoring logic for bias or unfair weightings...")
    print("- Education: Higher degrees are weighted more, which may disadvantage those with less access to education.")
    print("- Experience: More experience is rewarded, but may disadvantage younger applicants.")
    print("- Gender: No points are given for gender, which is fair. Gender should not affect scoring.")
    print("- Age: Some points are given based on age brackets, but extreme ages are not rewarded. Be cautious of age discrimination.")
    print("Recommendation: Regularly review and validate the scoring system to ensure fairness and compliance with anti-discrimination laws.")

def main():
    print("Job Applicant Scoring System")
    education = input("Enter education level (highschool, bachelor, master, phd): ").strip()
    try:
        experience = int(input("Enter years of relevant experience: ").strip())
    except ValueError:
        print("Invalid input for experience. Setting to 0.")
        experience = 0
    gender = input("Enter gender (male, female, other): ").strip()
    try:
        age = int(input("Enter age: ").strip())
    except ValueError:
        print("Invalid input for age. Setting to 0.")
        age = 0

    score = score_applicant(education, experience, gender, age)
    print(f"Applicant Score: {score}")

    analyze = input("Would you like to analyze the scoring logic for bias? (yes/no): ").strip().lower()
    if analyze == 'yes':
        analyze_scoring_logic()

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

OUTPUT:

```
Job Applicant Scoring System
Enter education level (highschool, bachelor, master, phd): bachelor
Enter years of relevant experience: 2
Enter gender (male, female, other): female
Enter age: 24
Applicant Score: 40
Would you like to analyze the scoring logic for bias? (yes/no): n0
```

EXPLANATION :

This code implements a job applicant scoring system based on education, experience, gender, and age.

The score_applicant function assigns points for each attribute and sums them for a total score.

The main function collects user input for each attribute and displays the calculated score.

The analyze_scoring_logic function explains potential biases or fairness issues in the scoring logic.

Users can choose to review the scoring system for bias after seeing their score.

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

```python
def greet_user(name, gender):
    gender_lower = gender.lower()
    if gender_lower == "male":
        title = "Mr."
    elif gender_lower == "female":
        title = "Ms."
    else:
        title = ""
    if title:
        return f"Hello, {title} {name}! Welcome."
    else:
        return f"Hello, {name}! Welcome."
if __name__ == "__main__":
    name = input("Enter your name: ")
    gender = input("Enter your gender (male/female/other): ")
    print(greet_user(name, gender))
```

OUTPUT:

```
thon3.11.exe C:/Users/DEEKSHA/OneDrive/Desktop/
Enter your name: Deeksha
Enter your gender (male/female/other): female
Hello, Ms. Deeksha! Welcome.
```

EXPLANATION :

This code defines a function greet_user that greets a user with a title based on their gender.
If the gender is "male", it uses "Mr."; if "female", it uses "Ms."; otherwise, no title is used.
The main block prompts the user to enter their name and gender.
It then prints a personalized greeting using the greet_user function.
The greeting is formatted to include the title if applicable, followed by the user's name.

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**