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

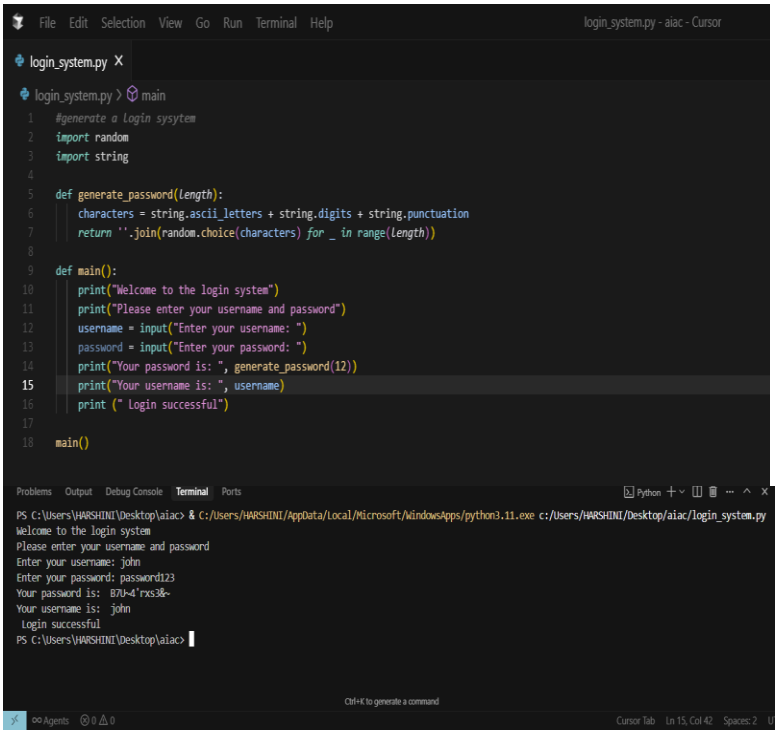
Lab Outcomes (LOs):
After completing this lab, students will be able to:

- 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 and environment variable 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 mitigation techniques.

```python
def approve_loan(name):
  """Simulates loan approval logic based on name."""
  if name.lower() == "john":
    return "Approved"
  elif name.lower() == "priya":
    return "Approved with conditions"
  else:
    return "Denied"

# Example usage:
print(f"Loan status for John: {approve_loan('John')}")
print(f"Loan status for Priya: {approve_loan('Priya')}")
print(f"Loan status for Sarah: {approve_loan('Sarah')}")
```

```
Loan status for John: Approved
Loan status for Priya: Approved with conditions
Loan status for Sarah: Denied
```

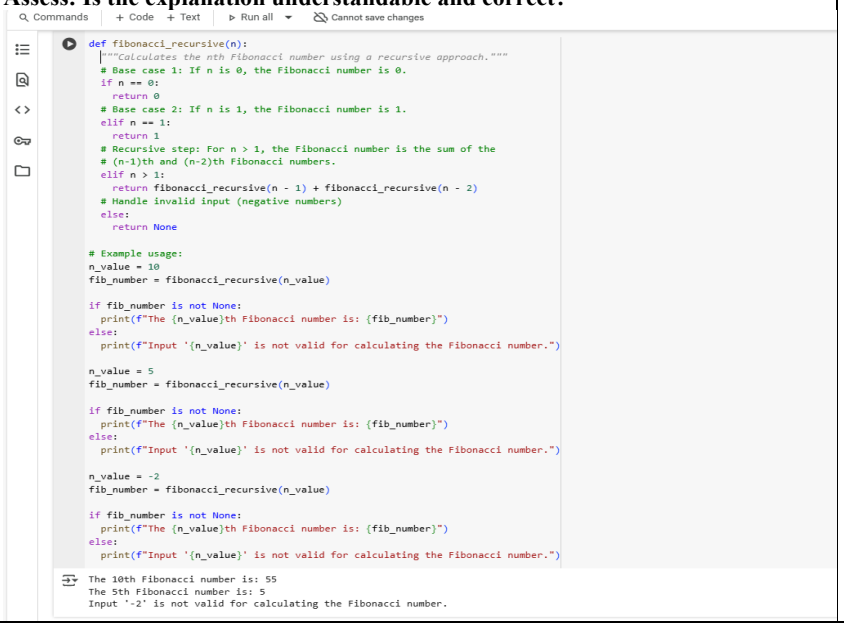
-
-

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

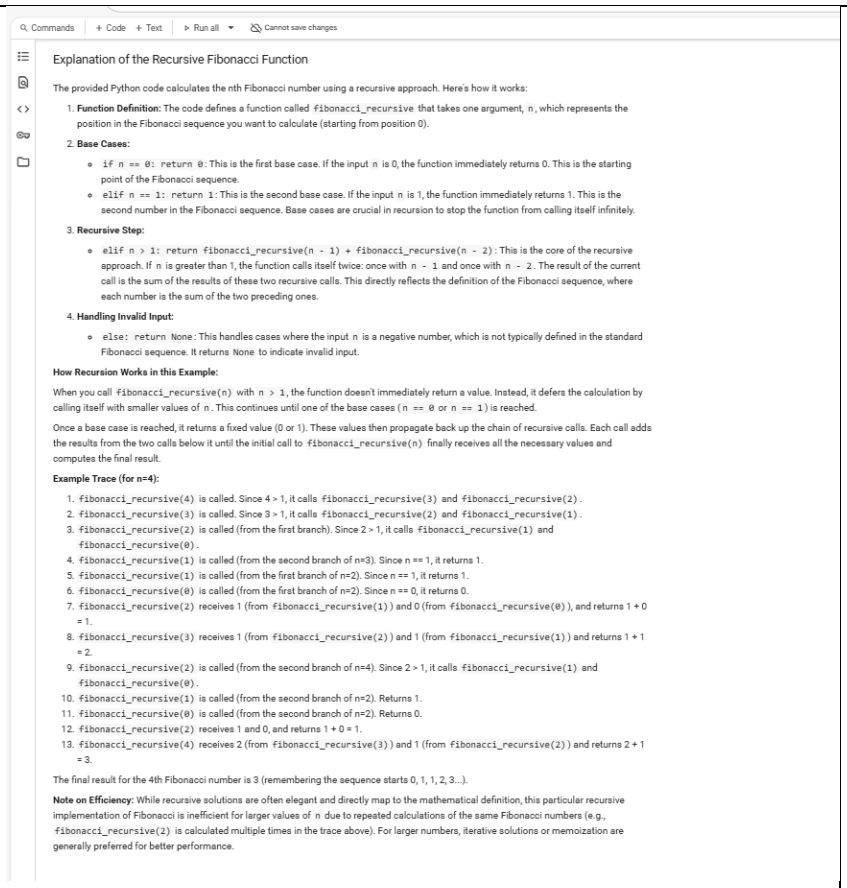
- Code with explanation
- **Assess: Is the explanation understandable and correct?**




```python
def fibonacci_recursive(n):
  """Calculates the nth Fibonacci number using a recursive approach."""
  # Base case 1: If n is 0, the Fibonacci number is 0.
  if n == 0:
    return 0
  # Base case 2: If n is 1, the Fibonacci number is 1.
  elif n == 1:
    return 1
  # Recursive step: For n > 1, the Fibonacci number is the sum of the
  # (n-1)th and (n-2)th Fibonacci numbers.
  elif n > 1:
    return fibonacci_recursive(n - 1) + fibonacci_recursive(n - 2)
  # Handle invalid input (negative numbers)
  else:
    return None

# Example usage:
n_value = 10
fib_number = fibonacci_recursive(n_value)

if fib_number is not None:
  print(f"The {n_value}th Fibonacci number is: {fib_number}")
else:
  print(f"Input '{n_value}' is not valid for calculating the Fibonacci number.")

n_value = 5
fib_number = fibonacci_recursive(n_value)

if fib_number is not None:
  print(f"The {n_value}th Fibonacci number is: {fib_number}")
else:
  print(f"Input '{n_value}' is not valid for calculating the Fibonacci number.")

n_value = -2
fib_number = fibonacci_recursive(n_value)

if fib_number is not None:
  print(f"The {n_value}th Fibonacci number is: {fib_number}")
else:
  print(f"Input '{n_value}' is not valid for calculating the Fibonacci number.")
```

```
The 10th Fibonacci number is: 55
The 5th Fibonacci number is: 5
Input '-2' is not valid for calculating the Fibonacci number.
```

-



-
-

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

```python
# 1. Define a function score_applicant that takes education, experience, gender, and age as input parameters.
def score_applicant(education, experience, gender, age):
    score = 0
    # 2. Implement a scoring logic within the function that assigns points or weights.
    # Scoring based on education
    if education == "PhD":
        score += 50
    elif education == "Master's":
        score += 40
    elif education == "Bachelor's":
        score += 30
    elif education == "High School":
        score += 20
    else:
        score += 10 # Other education levels

    # Scoring based on experience (linear scale, for simplicity)
    score += experience * 5

    # Scoring based on gender (introducing potential bias for demonstration)
    if gender == "Female":
        score += 5 # Arbitrary bonus, can be modified to show bias
    elif gender == "Male":
        score += 0
    else:
        score += 2 # Neutral score

    # Scoring based on age (introducing potential bias for demonstration - e.g., favoring a certain age range)
    if 25 <= age <= 40:
        score += 15
    elif 41 <= age <= 55:
        score += 10
    else:
        score += 5 # Lower score for other age ranges

    # 3. Return the calculated total score from the function.
    return score

# 4. Include example usage of the score_applicant function with different applicant profiles.
print("--- Applicant Scoring Examples ---")

# Example Applicant 1: High education, good experience, Female, middle age
applicant1_score = score_applicant(education="PhD", experience=8, gender="Female", age=35)
print(f"Applicant 1 Score: {applicant1_score}")

# Example Applicant 2: Bachelor's degree, less experience, Male, younger age
applicant2_score = score_applicant(education="Bachelor's", experience=3, gender="Male", age=22)
print(f"Applicant 2 Score: {applicant2_score}")

# Example Applicant 3: Master's degree, average experience, Other gender, older age
applicant3_score = score_applicant(education="Master's", experience=5, gender="Other", age=58)
print(f"Applicant 3 Score: {applicant3_score}")

# Example Applicant 4: High School, no experience, Female, young age
applicant4_score = score_applicant(education="High School", experience=0, gender="Female", age=19)
print(f"Applicant 4 Score: {applicant4_score}")
```

```
--- Applicant Scoring Examples ---
Applicant 1 Score: 110
Applicant 2 Score: 50
Applicant 3 Score: 72
Applicant 4 Score: 30
```

-
-

Task Description#5 (Inclusiveness)

- Code Snippet

```python
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):
    """
    Generates a gender-neutral greeting, ignoring the gender parameter for the greeting itself.

    Args:
      name: The name of the user (string).
      gender: The gender of the user (string). This parameter is accepted but not used in the greeting.

    Returns:
      A gender-neutral greeting string.
    """
    return f"Hello, {name}!"

# Example usage:
print(greet_user("Alex", "Non-binary"))
print(greet_user("Jordan", "Male"))
print(greet_user("Sarah", "Female"))
```

```
Hello, Alex!
Hello, Jordan!
Hello, Sarah!
```

-
-

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**