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	Week1 - Thursday	**Time(s)**	
Duration	2 Hours	**Applicableto Batches**	24CSBTB01 To 24CSBTB39
AssignmentNumber: **2.4**(Present assignment number)/**24**(Total number of assignments)			

Q.No.	Question	*ExpectedTime to complete*
1	Lab 2: Exploring Additional AI Coding Tools – Gemini (Colab) and Cursor AI **Lab Objectives:**	Week1 - Thursday

- To explore and evaluate the functionality of Google Gemini for AI-assisted coding within Google Colab.
- To understand and use Cursor AI for code generation, explanation, and refactoring.
- To compare outputs and usability between Gemini, GitHub Copilot, and Cursor AI.
- To perform code optimization and documentation using AI tools.

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

- Generate Python code using Google Gemini in Google Colab.
- Analyze the effectiveness of code explanations and suggestions by Gemini.
- Set up and use Cursor AI for AI-powered coding assistance.
- Evaluate and refactor code using Cursor AI features.
- Compare AI tool behavior and code quality across different platforms.

Task Description #1
• Open Google Colab and use Google Gemini to generate Python code that performs sorting of a list using both the bubble sort algorithm and Python's built-in sort() function. Compare the two implementations.
Expected Output #1
• Two sorting implementations: Bubble sort (manual logic) and Built-in sort()

```
import time
import random

def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr

# Generate a random list of numbers
data = [random.randint(0, 10000) for _ in range(1000)]

# Bubble Sort
start_time = time.time()
bubble_sorted_data = bubble_sort(data.copy()) # Use a copy to not modify the original list
end_time = time.time()
bubble_sort_time = end_time - start_time

print(f"Bubble Sort Time: {bubble_sort_time:.6f} seconds")

# Built-in Sort
start_time = time.time()
builtin_sorted_data = sorted(data.copy()) # Use a copy
end_time = time.time()
builtin_sort_time = end_time - start_time

print(f"Built-in Sort Time: {builtin_sort_time:.6f} seconds")

# Verify if both sorting methods produced the same result
print(f"Are the results the same? {bubble_sorted_data == builtin_sorted_data}")
```

```
Bubble Sort Time: 0.048998 seconds
Built-in Sort Time: 0.000230 seconds
Are the results the same? True
```

Task Description #2
• In Colab, use Google Gemini to generate a Python function that takes a string and returns: The number of vowels, The number of consonants, The number of digits in the string
Expected Output #2-
• Complete function that Iterates through characters of a string and Counts vowels, consonants, and digits

```python
def count_characters(input_string):
    """
    Counts the number of vowels, consonants, and digits in a string.

    Args:
        input_string: The string to analyze.

    Returns:
        A tuple containing the counts of vowels, consonants, and digits.
    """
    vowels = "aeiouAEIOU"
    consonants = "bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ"
    digits = "0123456789"

    vowel_count = 0
    consonant_count = 0
    digit_count = 0

    for char in input_string:
        if char in vowels:
            vowel_count += 1
        elif char in consonants:
            consonant_count += 1
        elif char in digits:
            digit_count += 1

    return vowel_count, consonant_count, digit_count

# Example usage:
test_string = "Hello World 123!"
v, c, d = count_characters(test_string)
print(f"String: '{test_string}'")
print(f"Vowel count: {v}")
print(f"Consonant count: {c}")
print(f"Digit count: {d}")
```

```
String: 'Hello World 123!'
Vowel count: 3
Consonant count: 7
Digit count: 3
```

Task Description #3

• Install and set up Cursor AI. Use it to generate a Python program that performs file handling:
Create a text file
Write sample text
Read and display the content

Expected Output #3

• Functional code that creates a .txt file, writes content to it, and reads it back.
• Screenshot of Cursor AI interface showing: Prompt used,
Generated code, Output of file operations

```python
# Define the file name
file_name = "why stories matter.txt"

# Write sample text to the file
try:
    with open(file_name, "w") as file:
        file.write("This is a sample line 1.\n")
        file.write("This is a sample line 2.\n")
        file.write("This is a sample line 3.\n")
    print(f"Successfully wrote to {file_name}")
except IOError as e:
    print(f"Error writing to file: {e}")

# Read and display the content of the file
try:
    with open(file_name, "r") as file:
        content = file.read()
        print(f"\nContent of {file_name}:\n")
        print(content)
except IOError as e:
    print(f"Error reading file: {e}")
```

```
Successfully wrote to why stories matter.txt

Content of why stories matter.txt:

This is a sample line 1.
This is a sample line 2.
This is a sample line 3.
```

Task Description #4

• Ask Google Gemini to generate a Python program that implements a simple calculator using functions (add, subtract, multiply, divide). Then, ask Gemini to explain how the code works.

Expected Output #4

• Complete calculator code with user input and operation selection.
• Line-by-line explanation or markdown-style explanation provided by Gemini.
• Screenshot of both the code and explanation in Colab.

```python
def add(x, y):
  """Adds two numbers."""
  return x + y

def subtract(x, y):
  """Subtracts the second number from the first."""
  return x - y

def multiply(x, y):
  """Multiplies two numbers."""
  return x * y

def divide(x, y):
  """Divides the first number by the second, handles division by zero."""
  if y == 0:
    return "Error! Division by zero."
  else:
    return x / y

print("Select operation:")
print("1. Add")
print("2. Subtract")
print("3. Multiply")
print("4. Divide")

while True:
  choice = input("Enter choice(1/2/3/4): ")

  if choice in ('1', '2', '3', '4'):
    try:
      num1 = float(input("Enter first number: "))
      num2 = float(input("Enter second number: "))
    except ValueError:
      print("Invalid input. Please enter numbers.")
      continue

    if choice == '1':
      print(num1, "+", num2, "=", add(num1, num2))
    elif choice == '2':
      print(num1, "-", num2, "=", subtract(num1, num2))
    elif choice == '3':
      print(num1, "*", num2, "=", multiply(num1, num2))
    elif choice == '4':
      print(num1, "/", num2, "=", divide(num1, num2))
    break
  else:
    print("Invalid input. Please enter a valid operation choice.")
```

```
Select operation:
1. Add
2. Subtract
3. Multiply
4. Divide
Enter choice(1/2/3/4): 2
Enter first number: 23
Enter second number: 3
23.0 - 3.0 = 20.0
```

Task Description #5

• Use Cursor AI to create a Python program that checks if a given year is a leap year or not. Try different prompt styles and see how Cursor modifies its code suggestions.

Expected Output #5

• A functional program to check leap year with sample input/output

• At least two versions of the code (from different prompts)

• A short comparison of which version is better and why-

```python
def is_leap(year):
  """Checks if a given year is a leap year."""
  if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
    return True
  else:
    return False

# Example usage:
year = 2024
if is_leap(year):
  print(f"{year} is a leap year.")
else:
  print(f"{year} is not a leap year.")

year = 1900
if is_leap(year):
  print(f"{year} is a leap year.")
else:
  print(f"{year} is not a leap year.")
```

```
2024 is a leap year.
1900 is not a leap year.
```

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
Two sorting implementations: Bubble sort (manual logic) and Built-in sort() (Task#1)	0.5
Counts vowels, consonants, and digits(Task#2)	0.5
Functional code that creates a .txt file, writes content to it, and reads it back- Use cursor (Task#3)	0.5
Complete calculator code with user input and operation selection. (Task#4)	0.5
A functional program to check leap year with sample input/output-use Cursor (Task#5)	0.5
Total	**2.5 Marks**