| 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)<br>Dr. T. Sampath Kumar<br>Dr. Pramoda Patro<br>Dr. Brij Kishor Tiwari<br>Dr.J.Ravichander<br>Dr. Mohammand Ali Shaik<br>Dr. Anirodh Kumar<br>Mr. S.Naresh Kumar<br>Dr. RAJESH VELPULA<br>Mr. Kundhan Kumar<br>Ms. Ch.Rajitha<br>Mr. M Prakash<br>Mr. B.Raju<br>Intern 1 (Dharma teja)<br>Intern 2 (Sai Prasad)<br>Intern 3 (Sowmya)<br>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<br>**Lab Objectives:** | Week1 - Thursday |

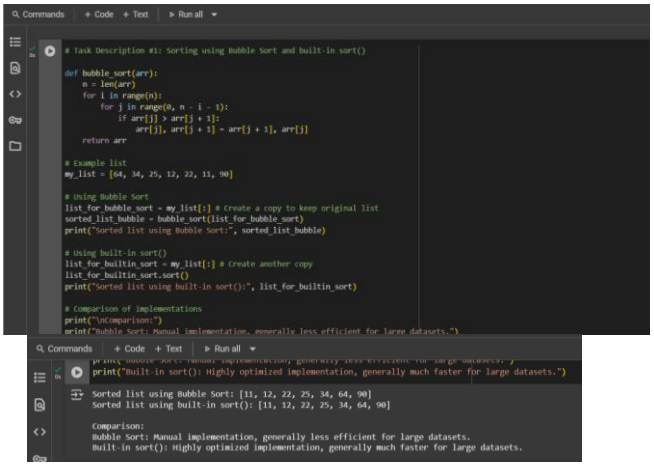
- To explore and evaluate the functionality of Google Gemini for AI-assisted coding within Google Colab.
- To understand and use Cursor AI for code generation, explanation, and refactoring.
- To compare outputs and usability between Gemini, GitHub Copilot, and Cursor AI.
- To perform code optimization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

**Prompt:**
performs sorting of a list using both the bubble sort algorithm and Python's built-in sort() function. Compare the two implementations.



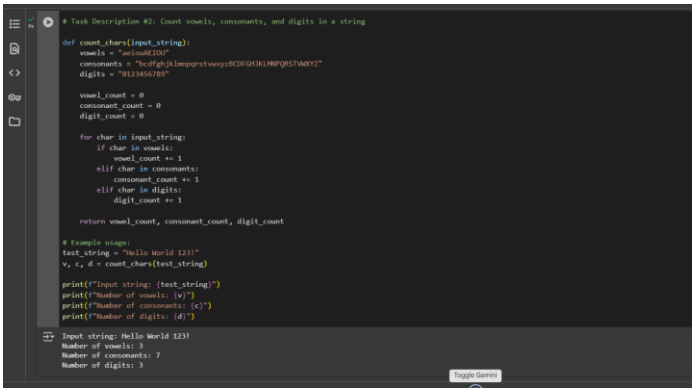

## Observation:

different AI coding tools, specifically Google Gemini in Colab and Cursor AI. The tasks are structured to cover various coding scenarios, including sorting algorithms, string manipulation, file handling, a simple calculator, and checking for leap years. The goal is to evaluate the functionality, code quality, and usability of these AI tools and compare their outputs.

**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

## Prompt:

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



```
# Task Description #2: Count vowels, consonants, and digits in a string

def count_chars(input_string):
    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_chars(test_string)

print(f"Input string: {test_string}")
print(f"Number of vowels: {v}")
print(f"Number of consonants: {c}")
print(f"Number of digits: {d}")
```

```
Input string: Hello World 123!
Number of vowels: 3
Number of consonants: 7
Number of digits: 3
```

Toggle Gemini

## Observation:

generating a Python function to count the number of vowels, consonants, and digits in a given string. The code I provided iterates through each character of the input string and checks if it belongs to the predefined sets of vowels, consonants, or digits, incrementing the respective counters. The function then returns these counts. This is a straightforward implementation of the task requirements.

**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

## Prompt:

generate a Python program that performs file handling:
Create a text file
Write sample text
Read and display the content

```python
# Task Description #3: File Handling

file_name = "sample.txt"
content_to_write = "This is some sample text that will be written to the file.\nThis is the second line."

# Create and write to the text file
try:
    with open(file_name, "w") as file:
        file.write(content_to_write)
    print(f"Successfully created and wrote to '{file_name}'")

except IOError as e:
    print(f"Error writing to file: {e}")

# Read and display the content of the text file
try:
    with open(file_name, "r") as file:
        read_content = file.read()
    print(f"\nContent of '{file_name}':")
    print(read_content)

except IOError as e:
    print(f"Error reading from file: {e}")
```

```
Successfully created and wrote to 'sample.txt'

Content of 'sample.txt':
This is some sample text that will be written to the file.
This is the second line.
```

Toggle Gemini

## Observation:

Python code for file handling, specifically creating a text file, writing sample content to it, and then reading and displaying that content. The code I provided uses standard Python file operations with try...except blocks to handle potential IOError exceptions during file writing and reading. This demonstrates basic file input and output operations.

**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.

## Prompt:

generate a Python program that implements a simple calculator using functions (add, subtract, multiply, divide).

```python
# Task Description #4: Simple Calculator

def add(x, y):
    return x + y

def subtract(x, y):
    return x - y

def multiply(x, y):
    return x * y

def divide(x, y):
    if y == 0:
        return "Error: Division by zero"
    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':
```

```python
        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")
```

```
Select operation:
1. Add
2. Subtract
3. Multiply
4. Divide
Enter choice(1/2/3/4): 1
Enter first number: 3
Enter second number: 5
3.0 + 5.0 = 8.0
```

## Observation:

creating a simple calculator program using Python functions for basic arithmetic operations. The code I provided allows the user to select an operation and enter two numbers. It includes error handling for non-numeric input and division by zero. When running this type of interactive code in Google Colab, you need to actively provide input in the prompt area that appears below the code cell during execution for the program to proceed and produce the desired output, as demonstrated in the last successful execution.

**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**

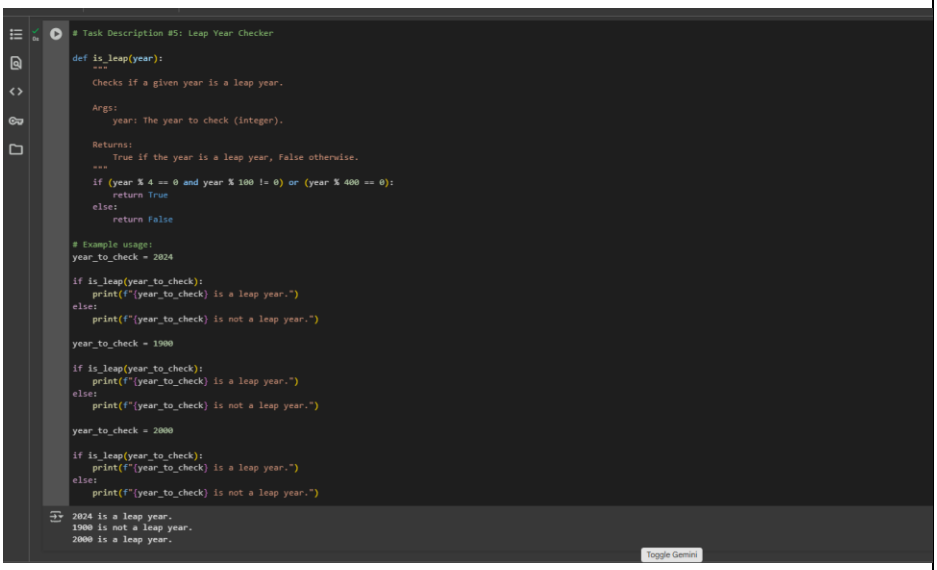
• A functional program to check leap year with sample input/output

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

• A short comparison of which version is better and why

## Prompt:

create a Python program that checks if a given year is a leap year or not



```
# Task Description #5: Leap Year Checker

def is_leap(year):
    """
    Checks if a given year is a leap year.

    Args:
        year: The year to check (integer).

    Returns:
        True if the year is a leap year, False otherwise.
    """
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False

# Example usage:
year_to_check = 2024

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

year_to_check = 1900

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

year_to_check = 2000

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

```
2024 is a leap year.
1900 is not a leap year.
2000 is a leap year.
```

Toggle Gemini

## Observation:

creating a Python program to determine if a given year is a leap year. The code I provided implements the standard rules for leap years: a year is a leap year if it is divisible by 4, except for years divisible by 100 but not by 400. The code includes a function is_leap() that returns True or False and provides example usage with different years to demonstrate its functionality. The task also requires using Cursor AI and comparing different versions of the code, which is a separate step to be completed using that tool.

**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** |