**Name: Thokala Srivalli**

**Enrollment No: 2403A510B2**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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:**   * 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:** give me a python code in a manner sorting of a list using both the bubble sort algorithm and Python’s built-in sort() function. Compare the two implementations using dynamic input.    **Observation:** This code compares two sorting approaches: a basic algorithm (bubble sort) and an optimized one (Python’s built-in sort, Timsort).   * **Bubble Sort:** Has a time complexity of **O(n²)** in both worst and average cases. This means the sorting time grows quadratically as the list size increases. While easy to understand and implement, it’s inefficient for large datasets. * **Python’s Built-in Sort (sorted()):** Uses **Timsort**, with a time complexity of **O(n log n)** in both average and worst cases. This makes it significantly faster than bubble sort, especially as the dataset grows, since the time increase is much slower.   **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:**  Write a Python function that returns the number of vowels, consonants, and digits in a given string.  **Observation:**  In this program, the function **count\_chars** processes a given string by iterating over its characters and classifying them into vowels, consonants, digits, or other symbols. It produces a dictionary summarizing the counts of each category, which is then printed in the example usage.  **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:** Write a Python program that creates a .txt file, writes sample text into it, then reads and displays the file content.    **Observation:**  In this code, Python’s file handling is demonstrated by first defining a file name and sample content, writing the content in write mode, and later reading it back in read mode to display. Any I/O errors that occur are caught using try...except.  **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 for a simple calculator using functions for add, subtract, multiply, and divide. Include user input for numbers and operation selection.    **Observation:**  In this program, a basic calculator is implemented with functions for add, subtract, multiply, and divide. A menu is displayed for the user to select an operation, after which two numbers are entered. Invalid inputs are handled, and division by zero is checked separately. The calculator runs repeatedly using a while True loop, but it lacks a specific exit command. is provided in this version.  **Task Description #5**  • Use Cursor AI to create a Python program that checks if a given year is a leap year or not. Trydifferent 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:** Write a Python program to check if a given year is a leap year.    **Observation:**  In this Python program, the **is\_leap\_year** function checks if a year qualifies as a leap year using the common conditions: divisible by 4, but not by 100 unless also divisible by 400. The program requests a year from the user, ensures the input is valid, and then shows whether it is 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** | | | | | | | Week1 - Thursday |  |