|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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#1**   1. Write a python code to generate a list of random integers. 2. Sort the list using both methods. 3. Measure and compare the time taken by each method. 4. Print the first 10 elements of the sorted lists. 5. Check if both sorted lists are identical.   **OBSEVATION#1**  Bubble Sort is significantly slower than Python’s built-in sort() function. While both produce the same result, built-in sort() is much more efficient and suitable for real-world use.    **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#2**  1.Write a Python function to count vowels, consonants, and digits in a string.  2. Make a Python program that counts how many vowels, consonants, and digits are in a given string.  3. Create a Python function that checks a string and returns the number of vowels, consonants, and digits.  **OBSERVATION#2**  The function correctly identifies and counts vowels, consonants, and digits by iterating through each character. It ignores spaces and special characters, making it efficient for basic text analysis.    **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#3**  **Write a Python program that:**  1. Creates a text file  2. Writes sample text to it  3. Reads the file and displays its content.  **OBSERVATION#3**  The program successfully demonstrates basic file handling in Python using open() with write and read modes. It creates a text file, writes sample text, then reads and displays the content. The use of with ensures proper file closure and resource management.    **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#4**  **1.Write a Python program** that implements a simple calculator.  **2.** The calculator should use **functions** for the following operations:   * Addition * Subtraction * Multiplication * Division   **3.** After the code, **explain how the program works** step-by-step.  **OBSERVATION#4**  The calculator program is a simple Python script that uses functions to perform basic arithmetic operations: addition, subtraction, multiplication, and division. It prompts the user to select an operation and enter two numbers. Based on the user's choice, the corresponding function is called to calculate and display the result.      **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#5**   1. Check leap year in Python. 2. Python function to check leap year. 3. Leap year checker with input. 4. Python leap year program. 5. One-liner leap year check Python.   **OBSERVATION#5**  The code checks if a year is a leap year using the standard rules: divisible by 4 but not by 100, unless also divisible by 400. It uses a simple condition and provides correct results for most valid inputs.    **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 |  |