|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | | | | 1. Dr. Mohammed Ali Shaik  2. Dr. T Sampath Kumar  3. Mr. S Naresh Kumar  4. Dr. V. Rajesh  5. Dr. Brij Kishore  6. Dr Pramoda Patro  7. Dr. Venkataramana  8. Dr. Ravi Chander  9. Dr. Jagjeeth Singh | | | | | |
| **CourseCode** | | | 24CS002PC215 | **CourseTitle** | | AI Assisted Coding | | | |
| **Year/Sem** | | | II/I | **Regulation** | | R24 | | | |
| **Date and Day**  **of Assignment** | | | Week2-Tuesday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicableto**  **Batches** | | 24CSBTB01 To 24CSBTB39 | | | |
| **AssignmentNumber:3.2**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***ExpectedTime***  ***to complete*** |  |
|  | 1 | Lab 3: Prompt Engineering – Improving Prompts and Context Management  **Lab Objectives:**   * To understand how prompt structure and wording influence AI-generated code. * To explore how context (like comments and function names) helps AI generate relevant output. * To evaluate the quality and accuracy of code based on prompt clarity. * To develop effective prompting strategies for AI-assisted programming.   **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**   * Ask AI to write a function to calculate compound interest, starting with only the function name. Then add a docstring, then input-output example   **Expected Output#1**   * Comparison of AI-generated code styles   **PROMPT:** Write a Python program to create compound interest starting with only function name. Then add a docstring, then input-output example  **CODE:**    **OBSERVATION:**  This code defines a function calculate\_compound\_interest that takes the principal amount, annual interest rate, number of years, and the number of times interest is compounded per year as input. It then calculates and returns the compound interest earned. The example usage demonstrates how to call the function with specific values and prints the calculated compound interest.  **Task Description#2**   * Do math stuff, then refine it to: # Write a function to calculate average, median, and mode of a list of numbers.   **Expected Output#2**   * AI-generated function evolves from unclear to accurate multi-statistical operation.   **PROMPT:** Write a python code to create a function to calculate average, median, and mode of a list of numbers and take input from the user.  **CODE:**    **OBSERVATION**: This code defines a function calculate\_statistics that takes a list of numbers and returns a dictionary containing their average, median, and mode. It uses the collections.Counter class to easily calculate the frequency of each number for determining the mode. The code also includes error handling for invalid input and the case where there is no unique mode. After defining the function, it prompts the user for input, processes the input, and then calls the function to display the results.  **Task Description#3**   * Provide multiple examples of input-output to the AI for convert\_to\_binary(num) function. Observe how AI uses few-shot prompting to generalize.   **Expected Output#3**   * Enhanced AI output with clearer prompts   **PROMPT:** Write a python program to create a function that takes a number as input and returns its binary representation take input from the user.    **CODE:**    **OBSERVATION**: The code successfully converted the decimal number 25 to its binary representation, which is 11001. The output confirms that the decimal\_to\_binary function is working correctly for this input.  **Task Description#4**   * Create an user interface for an hotel to generate bill based on customer requirements   **Expected Output#4**   * Consistent functions with shared logic   **PROMPT:** Generate a python code to Create an user interface for in hotel to generate bill based on customer requirements.  **CODE:**    **OBSERVATION**:  Provide an observation about the created user interface and bill generation process.  **Task Description#5**   * Analyzing Prompt Specificity: Improving Temperature Conversion Function with Clear Instructions   **Expected Output#5**   * Code quality difference analysis for various prompts   **PROMPT:** Generate the python code to create the function Improving Temperature Conversion Function with Clear Instructions**.**  **CODE:**    **OBSERVATION**:  This code provides a well-defined function convert\_temperature for converting between Celsius and Fahrenheit. It includes clear instructions within the docstring about how to use the function, specifying the required arguments and expected return value. It also includes example usage to demonstrate how to call the function and the expected output format. The error handling for invalid units is also a good practice.  **Evaluation Criteria:**   | **Criteria** | **Max Marks** | | --- | --- | | Task#1 | 0.5 | | Task#2 | 0.5 | | Task #3 | 0.5 | | Task #4 | 0.5 | | Task #5 | 0.5 | | **Total** | **2.5 Marks** | | | | | | | 03.08.2025 EOD |  |