|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | | | Week4 - Wednesday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicableto**  **Batches** | |  | | | |
| **AssignmentNumber:9.3**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | | | | | | | | | |
|  | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***ExpectedTime***  ***to complete*** |  |
|  | 1 | Lab 9: Documentation Generation: Automatic documentation and code comments  **Lab Objectives:**   * To understand the importance of documentation and code comments in software development. * To explore how AI-assisted coding tools can generate meaningful documentation and inline comments. * To practice generating function-level and module-level docstrings automatically. * To evaluate the quality, accuracy, and limitations of AI-generated documentation. * To develop a small automated tool for documentation generation in Python..     **Lab Outcomes (LOs):**  After completing this lab, students will be able to:   * Apply AI-assisted coding tools to generate docstrings and inline comments for Python code. * Critically analyze AI-generated documentation for correctness, completeness, and readability. * Create structured documentation (function-level, module-level) following standard formats. * Design and implement a mini documentation generator tool to automate code commenting and docstring creation.   **Task Description#1 Basic Docstring Generation**   * Write python function to return sum of even and odd numbers in the given list. * Incorporate manual **docstring** in code with Google Style * Use an AI-assisted tool (e.g., Gemini, Copilot, Cursor AI) to generate a docstring describing the function. * Compare the AI-generated docstring with your manually written one.   **Expected Outcome#1:** Students understand how AI can produce function-level documentation. Generate python functionSubtask: Create a Python function that takes a list of numbers as input and returns the sum of even and odd numbers.  def sum\_even\_odd(numbers):    """Calculates the sum of even and odd numbers in a list.    Args:      numbers: A list of numbers.  Returns:      A tuple containing the sum of even numbers and the sum of odd numbers.    """    even\_sum = 0    odd\_sum = 0    for number in numbers:      if number % 2 == 0:        even\_sum += number      else:        odd\_sum += number    return even\_sum, odd\_sum Add manual docstringSubtask: Manually write a docstring for the generated function following the Google Style Guide.  def sum\_even\_odd(numbers):    """Calculates the sum of even and odd numbers in a list.    This function iterates through the input list of numbers and separates    them into even and odd categories, computing the sum for each category.    Args:      numbers: A list of integers or floats.    Returns:      A tuple containing two elements:      - even\_sum (int or float): The sum of all even numbers in the input list.      - odd\_sum (int or float): The sum of all odd numbers in the input list.    """    even\_sum = 0    odd\_sum = 0    for number in numbers:      if number % 2 == 0:        even\_sum += number      else:        odd\_sum += number    return even\_sum, odd\_sum  # Comparison Analysis:  # Similarities:  # - Both docstrings provide a concise summary of the function's purpose.  # - Both docstrings include an "Args:" section to describe the input parameter.  # - Both docstrings include a "Returns:" section to describe the output.  # Differences:  # - The manual docstring is more verbose, providing a more detailed explanation of the function's process ("iterates through the input list...").  # - The manual docstring specifies the expected data types for both the input list elements ("integers or floats") and the elements in the returned tuple ("int or float"). The AI-generated docstring is less specific, simply stating "A list of numbers" and "A tuple containing the sum of even and odd numbers."  # - The manual docstring explicitly names and describes each element in the returned tuple ("even\_sum", "odd\_sum"). The AI-generated docstring describes the tuple more generally.  # - The manual docstring adheres to the Google Style Guide's structure with a concise summary line, a blank line, and then the detailed explanation and parameter/return descriptions. The AI-generated docstring is also generally in a similar format, but the level of detail differs.  **Task Description#2 Automatic Inline Comments**   * Write python program for **sru\_student** class with attributes like name, roll no., hostel\_status and **fee\_update** method and **display\_details** method. * Write comments manually for each line/code block * Ask an AI tool to add inline comments explaining each line/step. * Compare the AI-generated comments with your manually written one.   **Expected Output#2:** Students critically analyze AI-generated code comments.  class sru\_student:      def \_\_init\_\_(self, name, roll\_no, hostel\_status):          self.name = name  # Assign the input name to the instance variable self.name          self.roll\_no = roll\_no  # Assign the input roll\_no to the instance variable self.roll\_no          self.hostel\_status = hostel\_status  # Assign the input hostel\_status to the instance variable self.hostel\_status          self.fee\_balance = 0  # Initialize the fee\_balance for the student to 0      def fee\_update(self, amount):          """Updates the student's fee balance."""          self.fee\_balance += amount  # Add the input amount to the current fee\_balance          print(f"Fee balance updated for {self.name}. New balance: {self.fee\_balance}") # Print a confirmation message with the updated balance      def display\_details(self):          """Displays the student's details."""          print("Student Details:")  # Print a header for the student details          print(f"Name: {self.name}")  # Print the student's name          print(f"Roll No.: {self.roll\_no}")  # Print the student's roll number          print(f"Hostel Status: {self.hostel\_status}")  # Print the student's hostel status          print(f"Fee Balance: {self.fee\_balance}")  # Print the student's fee balance  class sru\_student:      def \_\_init\_\_(self, name, roll\_no, hostel\_status):          self.name = name          self.roll\_no = roll\_no          self.hostel\_status = hostel\_status          self.fee\_balance = 0      def fee\_update(self, amount):          """Updates the student's fee balance."""          self.fee\_balance += amount          print(f"Fee balance updated for {self.name}. New balance: {self.fee\_balance}")      def display\_details(self):          """Displays the student's details."""          print("Student Details:")          print(f"Name: {self.name}")          print(f"Roll No.: {self.roll\_no}")          print(f"Hostel Status: {self.hostel\_status}")          print(f"Fee Balance: {self.fee\_balance}")  **Task Description#3**   * Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide). * Incorporate manual **docstring** in code with NumPy Style * Use AI assistance to generate a module-level docstring + individual function docstrings. * Compare the AI-generated docstring with your manually written one.   **Expected Output#3:** Students learn structured documentation for multi-function scripts  **de**f add(x, y):      """      Adds two numbers.      Parameters      ----------      x : int or float          The first number.      y : int or float          The second number.      Returns      -------      int or float          The sum of x and y.      Examples      --------      >>> add(2, 3)      5      >>> add(2.5, 3.5)      6.0      """      return x + y  def subtract(x, y):      """      Subtracts the second number from the first.      Parameters      ----------      x : int or float          The first number.      y : int or float          The second number.      Returns      -------      int or float          The difference between x and y.      Examples      --------      >>> subtract(5, 3)      2      >>> subtract(5.5, 3.0)      2.5      """      return x - y  def multiply(x, y):      """      Multiplies two numbers.      Parameters      ----------      x : int or float          The first number.      y : int or float          The second number.      Returns      -------      int or float          The product of x and y.      Examples      --------      >>> multiply(2, 3)      6      >>> multiply(2.5, 3.0)      7.5      """      return x \* y  def divide(x, y):      """      Divides the first number by the second.      Parameters      ----------      x : int or float          The numerator.      y : int or float          The denominator.      Returns      -------      float          The result of the division.    Raises      ------      ZeroDivisionError          If the denominator is zero.      Examples      --------      >>> divide(6, 3)      2.0      >>> divide(5, 2)      2.5      """      if y == 0:          raise ZeroDivisionError("cannot divide by zero")      return x / y  **Push documentation whole workspace as .md file in GitHub Repository**  **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** | | | | | | Week4 - Wednesday |  |