|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE** | | | | | **DEPARTMENT OF COMPUTER SCIENCE ENGINEERING** | | | | |
| **Program Name:** B. Tech | | | | **Assignment Type: Lab** | | | **Academic Year:**2025-2026 | | |
| **Course Coordinator Name** | | | | 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) | | | | | | |
| **Course Code** | | | 24CS002PC215 | **Course Title** | | AI Assisted Coding | | | |
| **Year/Sem** | | | II/I | **Regulation** | | R24 | | | |
| **Date and Day**  **of Assignment** | | | Week5 - Monday | **Time(s)** | |  | | | |
| **Duration** | | | 2 Hours | **Applicable to**  **Batches** | |  | | | |
| **AssignmentNumber: 9.1**(Present assignment number)/**24**(Total number of assignments) | | | | | | | | | |
|  | **Q.No.** | **Question** | | | | | | ***Expected Time***  ***to complete*** |  |
|  | 1 | **Lab 9 – Documentation Generation: Automatic Documentation and Code Comments**  **Lab Objectives**   * Inline comments * Docstrings * Auto-documentation tools * AI-assisted summarization   **Task Description #1** (Automatic Code Commenting)  **Scenario:** You have been given a Python function without comments.  def calculate\_discount(price, discount\_rate):  return price - (price \* discount\_rate / 100)   * Use an AI tool (or manually simulate it) to generate line-by-line comments for the function. * Modify the function so that it includes a docstring in Google-style or NumPy-style format. * Compare the auto-generated comments with your manually written version.   def calculate\_discount(price, discount\_rate):      """Calculates the discounted price based on the original price and discount rate.      Args:          price: The original price of the item.          discount\_rate: The discount rate in percentage.      Returns:          The discounted price.      """      # Calculate the discount amount      discount\_amount = price \* discount\_rate / 100      # Subtract the discount amount from the original price to get the discounted price      discounted\_price = price - discount\_amount      # Return the calculated discounted price      return discounted\_price  # Example usage with the modified function  original\_price = 100  discount\_percentage = 15  final\_price = calculate\_discount(original\_price, discount\_percentage)  print(f"Original price: ${original\_price}")  print(f"Discount rate: {discount\_percentage}%")  print(f"Discounted price: ${final\_price}")  OUTPUT:Original price: $100 Discount rate: 15% Discounted price: $85.0  **Task Description #2** (API Documentation Generator)  **Scenario:** A team is building a **Library Management System** with multiple functions.  def add\_book(title, author, year):  # code to add book  pass  def issue\_book(book\_id, user\_id):  # code to issue book  Pass   * Write a Python script that uses docstrings for each function (with input, output, and description). * Use a documentation generator tool (like pdoc, Sphinx, or MkDocs) to automatically create HTML documentation. * Submit both the code and the generated documentation as output.   # library\_manager.py  """  Library Management System Core Functions.  This module provides essential functions for a simple library management system,  including adding, issuing, and removing books.  """  from typing import Dict, Any, List  # A simple in-memory database to simulate book storage.  books\_db: Dict[int, Dict[str, Any]] = {}  next\_book\_id: int = 1  def add\_book(title: str, author: str, year: int) -> Dict[str, Any]:      """      Adds a new book to the library database.      This function creates a new book entry with a unique ID and stores      it in the in-memory database. The book is initially available.      Args:          title (str): The title of the book.          author (str): The name of the book's author.          year (int): The publication year of the book.      Returns:          Dict[str, Any]: A dictionary representing the newly added book,                          including its unique ID.      """      global next\_book\_id      new\_book = {          "id": next\_book\_id,          "title": title,          "author": author,          "year": year,          "is\_issued": False,          "issued\_to": None,      }      books\_db[next\_book\_id] = new\_book      next\_book\_id += 1      return new\_book  def issue\_book(book\_id: int, user\_id: str) -> bool:      """      Issues a book to a user.      This function marks a specific book as 'issued' and assigns it to a user.      It returns `True` if the book was successfully issued, `False` otherwise      (e.g., if the book does not exist or is already issued).      Args:          book\_id (int): The unique identifier of the book to be issued.          user\_id (str): The unique identifier of the user.    Returns:          bool: `True` if the book was successfully issued, `False` otherwise.      """      if book\_id in books\_db and not books\_db[book\_id]["is\_issued"]:          books\_db[book\_id]["is\_issued"] = True          books\_db[book\_id]["issued\_to"] = user\_id          return True      return False  def remove\_book(book\_id: int) -> bool:      """      Removes a book from the library database.      This function deletes a book entry from the database. It can only      be removed if it is not currently issued.      Args:          book\_id (int): The unique identifier of the book to be removed.      Returns:          bool: `True` if the book was successfully removed, `False` otherwise                (e.g., if the book does not exist or is currently issued).      """      if book\_id in books\_db and not books\_db[book\_id]["is\_issued"]:          del books\_db[book\_id]          return True      return False  # Example usage of the functions  if \_\_name\_\_ == "\_\_main\_\_":      print("--- Adding a new book ---")      book1 = add\_book("The Catcher in the Rye", "J.D. Salinger", 1951)      print(f"Added book: {book1}")      print("\n--- Issuing the book ---")      is\_issued = issue\_book(book1["id"], "user123")      if is\_issued:          print(f"Book ID {book1['id']} successfully issued to user123.")      else:          print(f"Failed to issue book ID {book1['id']}.")      print("\n--- Attempting to remove the issued book ---")      is\_removed = remove\_book(book1["id"])      if is\_removed:          print(f"Book ID {book1['id']} was removed.")      else:          print(f"Failed to remove book ID {book1['id']} because it is issued.")    print("\n--- Adding another book and removing it ---")  --- Adding a new book ---  Added book: {'id': 1, 'title': 'The Catcher in the Rye', 'author': 'J.D. Salinger', 'year': 1951, 'is\_issued': False, 'issued\_to': None}  --- Issuing the book ---  Book ID 1 successfully issued to user123.  --- Attempting to remove the issued book ---  Failed to remove book ID 1 because it is issued.  --- Adding another book and removing it ---  Added book: {'id': 2, 'title': 'Dune', 'author': 'Frank Herbert', 'year': 1965, 'is\_issued': False, 'issued\_to': None}  Book ID 2 was removed successfully.  === Code execution complete ===  **Task Description #3** (AI-Assisted Code Summarization)  **Scenario:** You are reviewing a colleague’s codebase containing long functions.  def process\_sensor\_data(data):  cleaned = [x for x in data if x is not None]  avg = sum(cleaned)/len(cleaned)  anomalies = [x for x in cleaned if abs(x - avg) > 10]  return {"average": avg, "anomalies": anomalies}   * Generate a summary comment explaining the purpose of the function in 2–3 lines. * Create a flow-style comment (step-by-step explanation). * Write a short paragraph of documentation describing possible use cases of this function in real-world scenarios.   # The function first creates a new list called 'cleaned' by iterating through the input 'data' # and only keeping values that are not None. This ensures we work with valid numbers. # Next, it computes the 'avg' by summing the 'cleaned' data and dividing by the count of items in it. # Finally, it finds 'anomalies' by creating another list that includes any value from 'cleaned' # where the absolute difference between the value and the 'avg' is greater than 10. # The function then returns a dictionary containing both the calculated average and the list of anomalies. Real-world Use Cases This function is highly useful in scenarios where you need to quickly assess the health or stability of a system based on live data. For example, in a manufacturing plant, sensors on an assembly line could measure machine temperature, and this function could detect sudden spikes or drops that might indicate a malfunction. Similarly, in environmental monitoring, it could process data from air quality or water level sensors to flag unusual readings that may signal a pollution event or a flood risk. In health technology, the same logic could be applied to wearable devices to identify irregular heart rate or activity levels for immediate alerting.  **Task Description #4** (Real-Time Project Documentation)  **Scenario:** You are part of a project team that develops a Chatbot Application. The team needs documentation for maintainability.   * Write a README.md file for the chatbot project (include project description, installation steps, usage, and example). * Add inline comments in the chatbot’s main Python script (focus on explaining logic, not trivial code). * Use an AI-assisted tool (or simulate it) to generate a usage guide in plain English from your code comments. * Reflect: How does automated documentation help in real-time projects compared to manual documentation?  Chatbot ProjectProject Description This is a simple, rule-based chatbot designed to provide automated responses to common user queries. The bot is built to handle basic greetings, answer simple questions, and respond to specific commands. It serves as a foundational component for more complex conversational AI applications. Features Handles general greetings and farewells.  Responds to a predefined set of simple questions.  Recognizes and executes specific commands (e.g., help, quit).  Provides a clear and friendly conversational flow. Installation To get started with the chatbot, follow these steps.  Clone the repository:  git clone [https://your-repository-url.git](https://your-repository-url.git)  cd chatbot-project  Create a virtual environment: It's recommended to work within a virtual environment to manage dependencies.  python -m venv venv  Activate the virtual environment:  On Windows:  .\venv\Scripts\activate  On macOS/Linux:  source venv/bin/activate  Install dependencies: The project has a single dependency.  pip install colorama Usage Run the main script from your terminal.  python chatbot.py  The chatbot will greet you, and you can begin typing your messages. The conversation continues until you type the quit command. Example Here is a sample conversation with the bot:  Hello, I am a simple chatbot. How can I help you today?  > Hello  Hello there!  > What is your purpose?  I'm here to answer simple questions and respond to commands.  > help  Available commands:  - `help`: Shows this help message.  - `quit`: Exits the chatbot.  > quit  Goodbye! | | | | | | Week5 - Monday |  |