Saad Ali Khan

SP23-BSE-048

DS theory

Assignment 01

DESCRIPITION:

Here are the functions we used in this code

·1 createTask(int id, string desc, int priority):

* Creates a new task node with a unique ID, description, and priority.
* Returns the pointer to the newly created task.

·2 addTask(Task\*& head, int id, string desc, int priority):

* Adds a new task to the list, ensuring tasks are sorted by priority (higher priority tasks come first).
* Inserts the task at the correct position in the list.

3· viewTasks(Task\* head):

* Displays all tasks in the list with their ID, description, and priority.
* If the list is empty, it shows a message saying no tasks are available.

4· removeHighestPriorityTask(Task\*& head):

* Removes the task with the highest priority (first task in the list).
* If the list is empty, it shows a message that no tasks can be removed.

5· removeTaskByID(Task\*& head, int id):

* Removes a task by its unique task ID.
* If the task is not found, it shows a message indicating the task wasn't located in the list.

6· main():

* Implements a menu-based system that allows the user to add tasks, view all tasks, remove tasks (by highest priority or ID), or exit the program.

LOGIC BEHIND CODE:

Here is the logic we used in code.

· 1 **Task Node Structure**:

* Each task is represented as a node containing the following:
* taskID: A unique identifier for the task.
* description: Details of the task.
* priority: A numeric value representing the importance of the task (higher numbers indicate higher priority).
* next: A pointer to the next node (task) in the list.

· 2 **Adding Tasks**:

* When a new task is added, the list is traversed to find the correct position based on its priority.
* The task is inserted either at the beginning (if it has the highest priority) or in the appropriate position where tasks with higher priority come before it.

·3 **Viewing Tasks**:

* The function traverses the entire list starting from the head (first node) and displays the task details of each node.
* If the list is empty, a message indicating "No tasks available" is displayed.

· 4 **Removing the Highest Priority Task**:

* Since the list is sorted by priority, the task with the highest priority is always the first node (head of the list).
* The head pointer is moved to the next task, and the previous first task is deleted from memory.

· 5 **Removing a Task by ID**:

* The list is traversed to find the task with the given taskID.
* Once found, the task is removed by adjusting the pointers of the previous node to skip over the node to be deleted.
* If the task is not found, an error message is shown.

·6 **Main Menu Interaction**:

* A loop presents a menu to the user, allowing them to add tasks, view all tasks, remove tasks by ID or highest priority, or exit the system.
* The program continues prompting the user until they choose the exit option.

