**Task Management System**

**1. Understanding Linked Lists**

Singly Linked List: A singly linked list is a data structure comprising nodes. Each of those nodes contains a data part and a reference, or a link, pointing toward the next node in sequence. The very first node of the linked list is called a 'head', while the last node points to null; this marks its end.

Doubly Linked List: In a doubly linked list, each node contains the reference to the next node in sequence and the one for the previous node in sequence. This can enable both forward and backward traversals. Therefore, some operations like deleting or inserting before a given node are more efficient.

**4. Time Complexity**

**Time Complexity:**

• Adding a Task: It takes O(n) time for adding a new task at the end of the Singly Linked List, where n is the number of nodes present in the list. This is explained by the fact that we need to traverse the entire list to reach the last node.

• Search a Task: In searching a task by its ID, we need to traverse the list; hence, it takes O(n) time in the worst case.

• Traversal of Tasks: The time complexity for traversal, that is, viewing all tasks, is O(n) since all nodes have to be visited.

• Task Deletion: Deleting a task requires searching for a task, which is O(n), and then once found, the deletion itself is O(1), hence net O(n).

**Advantages of using a linked list over an array in dynamic data:**

• Dynamic Size: This means that links in a linked list can grow or shrink by inserting and deleting nodes. This is in contrast to arrays, which are of a fixed size or alternatively resized at great cost in memory.

• Insertion/Deletion Potency: In general, insertion or deletion of elements in a linked list is easier compared to in an array, mostly under very huge data. This is because operations like insertion and deletion in the former would involve a change in references only, whereas they may require the shifting of elements in arrays.

• Memory Consumption: For a large number of insertions and deletions, linked lists use memory much more effectively than an array, which sometimes allocates more memory than required or may require resizing with some overhead.