Explain the Big O complexity of your priority and sorting mechanisms.

In this project, treatment requests are managed using two queues. One queue is for priority patients and other is for normal patients.This method helps the system treat emergency patients first.
Adding a treatment request to the system's time complexity is O(1). Because the request is added directly to the end of the queue.No searching or moving of data is needed.
Processing a treatment request system's time complexity is O(1).The system first checks the priority queue. If it is not empty, it removes the next patient in line. If it is empty, it removes a patient from the normal queue. Queue operations enqueue and dequeue always take fixed time, so the priority system is very fast and efficient.

For sorting patients, the system uses Bubble Sort on a linked list. Patients are sorted severity values.The bubble sort algorithm compare two adjacent patients and swap their positions. This process is repeated until the list is sorted. The time complexity of bubble sort is O(n^2). This means that when the number of patients increases, the sorting time increases.

How would using a heap-based priority queue improve performance?

Heap based priority queue is other way to manage priority patients. Adding and removing elements of a heap's time complexity is O(log n).This is slower than O(1). This allows the system to quickly manage priority levels. In a group, the patient at the top is always the priority patient. This means the system does not need two queues.The system can operate with a single data structure.A heap is useful when there are many priority levels. For example, our Project has severity values from 1 to 10.

In this project, using two queues is smart because we are prioritizing based right or wrong.

Priority queue operations: O(1)
Sorting with Bubble Sort: O(n^2)
Heap based queue: O(log n)