**Project:** **St. Joseph’s Hospital Management System**

**Course: CS3400**

**Group Members:**

Khatala 202322723

T’soeu 202322798

Naledi 202002241

Ntokoane 202321215

St. Joseph’s Hospital Management System

This project demonstrates how a small hospital system can be managed with the aid of data structures. Using a straightforward command-line application, it concentrates on patient registration, emergencies, and bed assignment.

# System Overview

The Hospital Management CLI System simulates patient management in a hospital, specifically handling: - Patient registration (using a hash table for quick lookup) - Priority treatment queue (using a priority queue for emergencies) - Bed/room allocation (using a binary tree for efficient room assignment) - Patient logs/history (using linked lists for chronological tracking). All data is stored in text files, ensuring persistence between runs.

# ER Diagram

## C:\Users\User\Downloads\Untitled diagram-2025-10-19-191717.png

## Architecture DiagramData Structures & Their Roles

|  |  |  |
| --- | --- | --- |
| Feature | Data Structure | Description |
| **Patient Registration** | Hash Table | Quick access to patient data by ID or national ID |
| **Emergency Triage Queue** | Priority Queue | Prioritize critical patients by triage level (lower number = higher priority) |
| **Bed Allocation** | Binary Tree | Assign and release beds efficiently based on room hierarchy or availability |
| **Patient Logs** | Linked List | Maintain chronological logs of treatments, admissions, and discharges |

# File Design

|  |  |  |
| --- | --- | --- |
| File Name | Description | Example Fields |
| **patients.txt** | Stores all registered patient details | patient\_id|first\_name|last\_name|dob|gender|phone|address|national\_id |
| **triage.txt** | Maintains all emergency/priority cases in the queue | triage\_id|patient\_id|triage\_level|condition |
| **rooms.txt** | Stores details of each hospital room and bed capacity | room\_id|number\_patients|total\_beds |
| **beds.txt** | Lists all hospital beds and their occupancy | bed\_id|room\_id|occupied |
| **bed\_assignments.txt** | Records which bed is assigned to which patient | assign\_id|patient\_id|bed\_id |
| **patient\_logs.txt** | Logs actions or events linked to a patient | log\_id|patient\_id|timestamp|activity |

# Pseudocode

## Register Patient (Hash Table)

Function RegisterPatient():  
 Input: name, gender, dob, phone, address, national\_id  
 id = generatePatientID()  
 record = {id, name, gender, dob, phone, address, national\_id}  
  
 AppendToFile("patients.txt", record)  
  
 key = hash(id)  
 PatientHashTable[key] = record  
  
 Print "Patient successfully registered."

## Add Emergency Case (Priority Queue)

Function AddEmergencyCase():  
 Input: patient\_id, triage\_level, condition  
 arrival\_time = getCurrentTime()  
  
 entry = (triage\_level, arrival\_time, patient\_id, condition)  
 PriorityQueue.insert(entry)  
  
 AppendToFile("triage.txt", entry)  
 Print "Emergency case added to priority queue."

## Assign Bed (Binary Tree)

Function AssignBed(patient\_id):  
 node = BedTree.root  
  
 while node is not null:  
 if node.bed\_status == false:  
 node.bed\_status = true  
 node.patient\_id = patient\_id  
 UpdateFile("beds.txt", node)  
 AppendToLog(patient\_id, "Assigned to bed " + node.bed\_id)  
 Print "Bed assigned successfully."  
 return  
 else:  
 if node.left != null:  
 node = node.left  
 else:  
 node = node.right  
  
 Print "No available beds found."

## Maintain Patient Logs (Linked List)

Structure LogNode:  
 int log\_id  
 int patient\_id  
 string message  
 datetime timestamp  
 LogNode\* next  
  
Function AppendToLog(patient\_id, message):  
 newNode = new LogNode(generateLogID(), patient\_id, message, getCurrentTime())  
 if PatientLogHead == null:  
 PatientLogHead = newNode  
 else:  
 current = PatientLogHead  
 while current.next != null:  
 current = current.next  
 current.next = newNode  
  
 AppendToFile("patient\_logs.txt", newNode)

# Menu Example (CLI)

====== ST. JOSEPH’S HOSPITAL CLI ======

1. Register Patient

2. Add Emergency Case

3. Assign Bed

4. View Patient Logs

5. Exit

# Conclusion

This system demonstrates the interoperability of various data structures.  
The linked list records each patient's progress, the tree assists with room assignment, the queue assists with emergencies, and the hash table facilitates quick patient discovery.