

National University of Computer and Emerging Sciences



**Hospital Patient Monitoring and Bed Allocation System**

**Group Members:**

Shayan Nemat ………… 23K-0899

Syed Waleed Hussain………… 23K-0885

Muhammad Huzaifa Altaf ………… 23K-0627

Sanjna ………… 23K-2022

**Supervised by:**

**Miss Anaum Hamid**

**FAST School of Computing**

**National University of Computer and Emerging Sciences**

**Karachi, Pakistan**

**April 2025**

# 1. Introduction

The Hospital Bed Management System Simulation project was developed to simulate real-world hospital bed management using multithreading and semaphores in C. The project began in March 2025 and was scheduled for completion by April 2025. It was developed by a team of four students of Section 4-H from FAST University.

The primary organization involved is the university itself, with the project aiming to model critical aspects of hospital patient management, including ICU and General Bed allocations, emergency prioritization, and bed release after patient discharge.

This report provides a detailed overview of the project's purpose, activities undertaken, features implemented, technologies used, sample code snippets, and a brief analysis of project outcomes.

**2. Features**

* Dynamic Patient Management: Patients can be added with different severity levels and types (Emergency or Regular).
* Priority Queue Implementation: Emergency and high-severity patients are prioritized for bed allocation.
* Multithreading: Each patient allocation runs on a separate thread.
* Bed Type Management: Separate handling for ICU Beds (3) and General Beds (7).
* Emergency Handling: Immediate bed assignment and thread creation for emergency patients.
* Logging System: Two separate logs — one for event tracking (hospital\_log.txt) and another for bed status (bed\_status\_log.txt).
* Resource Synchronization: Proper use of mutex locks and semaphores to prevent race conditions.
* Real-time Monitoring: Console outputs show system status, bed occupancy, and patient processing in real-time.
* Simulation Termination: Graceful exit from the system on demand.

**3. Technology Used**

1. Programming Language: C
2. Libraries Used: pthread.h, semaphore.h, stdio.h, stdlib.h, string.h, time.h, unistd.h
3. Synchronization Primitives: Mutex locks, Semaphores
4. Data Structures: Custom Priority Queue based on patient severity

**4. Code Snippets**

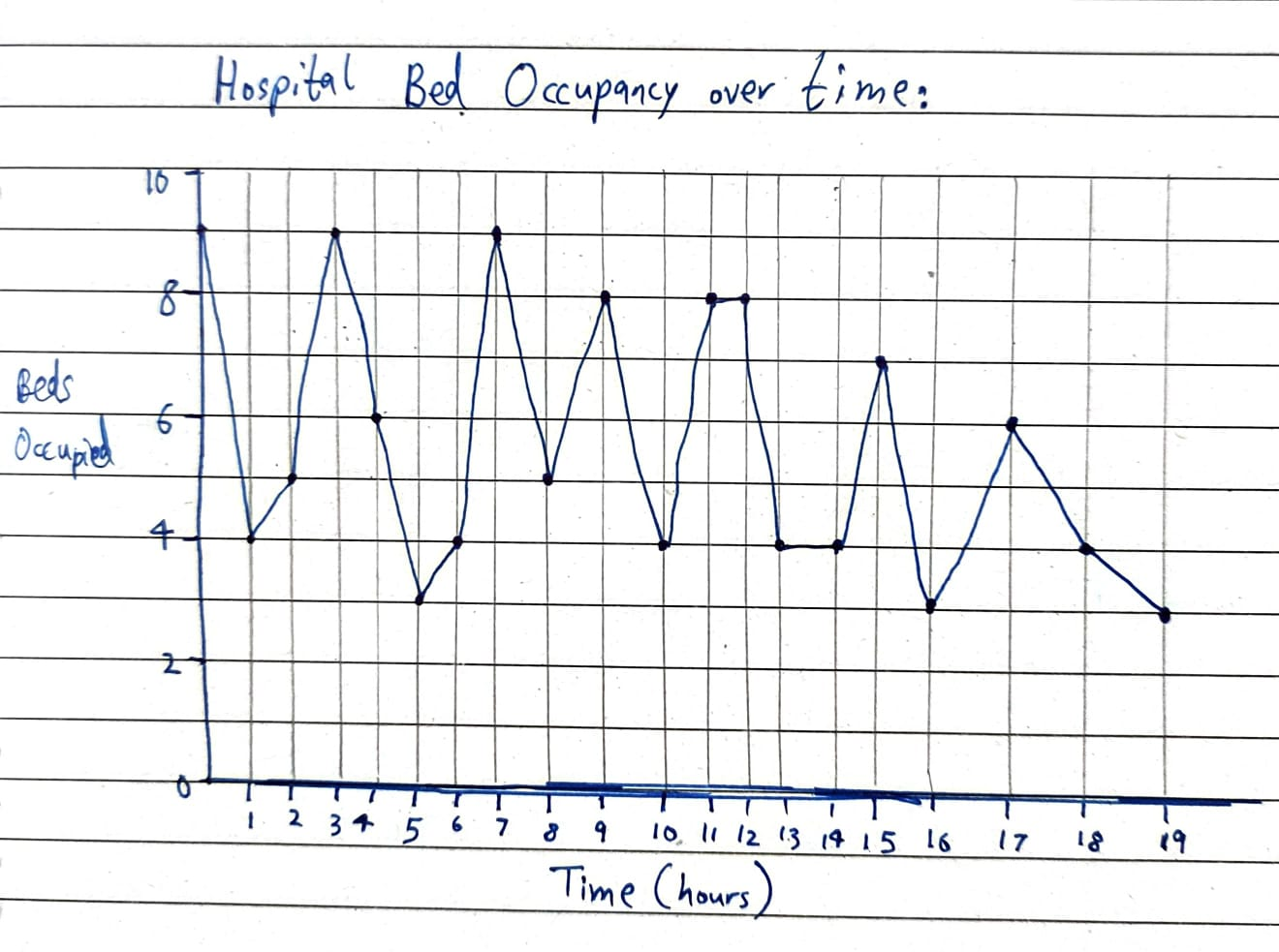
1. ***Priority Queue Insertion:***

void pq\_push(PriorityQueue\* pq, Patient patient) {  
 PQNode\* new\_node = (PQNode\*)malloc(sizeof(PQNode));  
 new\_node->patient = patient;  
 new\_node->next = NULL;  
  
 if (pq->head == NULL || patient.severity > pq->head->patient.severity) {  
 new\_node->next = pq->head;  
 pq->head = new\_node;  
 } else {  
 PQNode\* current = pq->head;  
 while (current->next != NULL && current->next->patient.severity >= patient.severity) {  
 current = current->next;  
 }  
 new\_node->next = current->next;  
 current->next = new\_node;  
 }  
 pq->size++;  
}

1. ***Bed Allocation Thread:***

void\* allocate\_bed(void\* arg) {  
 Patient\* p = (Patient\*)arg;  
 printf("[REQUEST] Patient %d requesting a bed...\n", p->id);  
  
 if (p->type == EMERGENCY) {  
 sem\_wait(&icuBeds);  
 } else {  
 sem\_wait(&generalBeds);  
 }  
  
 pthread\_mutex\_lock(&bed\_lock);  
 occupied\_beds++;  
 pthread\_mutex\_unlock(&bed\_lock);  
  
 logger\_log\_event("Admitted", p);  
 logger\_log\_bed\_status();  
  
 printf("[ALLOCATED] Patient %s (ID: %d) admitted. Beds Occupied: %d\n", p->name, p->id, occupied\_beds);  
  
 sleep(5 + (rand() % 6));  
  
 pthread\_mutex\_lock(&bed\_lock);  
 occupied\_beds--;  
 pthread\_mutex\_unlock(&bed\_lock);  
  
 logger\_log\_event("Discharged", p);  
 logger\_log\_bed\_status();  
  
 if (p->type == EMERGENCY) {  
 sem\_post(&icuBeds);  
 } else {  
 sem\_post(&generalBeds);  
 }  
  
 printf("[RELEASED] Patient %s (ID: %d) discharged. Beds Occupied: %d\n", p->name, p->id, occupied\_beds);  
 free(p);  
 return NULL;  
}

**5. Graph:**



**6. Conclusion**

The Hospital Bed Management System Simulation successfully models a simplified version of hospital resource management under normal and emergency situations. The project demonstrates a practical application of multithreading, semaphores, and priority queues to simulate real-world critical systems.  
  
All planned functionalities were successfully implemented, and no significant deviations from the project scope occurred. Minor improvements such as adding patient discharge notifications are planned for future upgrades.