HOSPITAL MANAGEMENT SYSTEM USING RESOURCE MANAGEMENT

MINI-PROJECT REPORT

Submitted by

JAISHREE S 231901011

VEDANTH S 231901060

In partial fulfilment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING(CYBER SECURITY)

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

MAY 2025

BONAFIDE CERTIFICATE

Certified that this project “HOSPITAL MANAGEMENT SYSTEM USING RESOURCE MANAGEMENT ” is the Bonafide work of “**VEDANTH S**, **JAISHREE S**” who carried out the project work under my supervision.

This mini project report is submitted for the viva voce examination to be held on ______

SIGNATURE

V JANANEE

ASSISTANT PROFESSOR

Dept. of Computer Science and Engineering,

Rajalakshmi Engineering College

Chennai

INTERNAL EXAMINER EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our sincere thanks to our Head of the Department

MR. Benedict JN, for encouragement and being ever supporting force during our project work.

We also extend our sincere and hearty thanks to our internal guide

Mrs. V. JANANEE, for her valuable guidance and motivation during the completion of this project.

Our sincere thanks to our family members, friends and other staff members of computer science engineering.

VEDANTH S 231901060

JAISHREE S 231901011

ABSTRACT

The Hospital Management System (HMS) integrates CPU scheduling techniques to improve service efficiency. For hospital bed allocation, it uses First-Come, First-Served (FCFS) to handle general admissions fairly and efficiently. In critical cases, Priority Scheduling ensures patients with urgent needs are given immediate attention. This helps balance fairness with the urgency of medical conditions during bed allocation. The system also includes a resource management module to track and allocate rooms, staff, and medical equipment in real time. This prevents delays and supports better hospital workflow. By combining scheduling algorithms with smart resource management, HMS enhances patient care, reduces wait times, and ensures optimal use of hospital resources. Real-time monitoring allows the system to adapt quickly to changing patient needs and hospital conditions. This dynamic adjustment ensures that both emergency and routine cases are handled smoothly without resource conflicts.Overall, the integration of FCFS and Priority Scheduling in bed allocation, along with efficient resource tracking, makes the HMS a reliable and scalable solution. It improves hospital operations, boosts patient satisfaction, and minimizes operational delays.

TABLE OF CONTENTS

S.NO:	TITLE	PAGE NO.
	Abstract	5
1	Introduction	6
	1.1. Key features	6
2	Scope of Project	8
3	Architecture	11
4	Flowchart	15
5	Code implementation	
	5.1 Program	16
	5.2 Screenshot	17
6	Conclusion	18
7	Future work	19
8	Reference	20

CHAPTER 1

INTRODUCTION

In today's rapidly advancing healthcare sector, the effective management of hospital resources is crucial for providing high-quality care to patients while ensuring optimal utilization of available infrastructure. The integration of technology into healthcare management systems has revolutionized the way hospitals operate, allowing for better coordination, streamlined workflows, and improved patient outcomes. A key aspect of this transformation is the development of sophisticated systems that optimize both administrative and clinical processes.

This project aims to design and implement a **Hospital Management System (HMS)** that leverages **CPU scheduling** and **resource management** techniques to efficiently manage hospital operations. CPU scheduling refers to the method of allocating processing time to various tasks or processes in a computing system, ensuring that critical operations receive the necessary resources in a timely manner. In the context of hospital management, this concept is applied to handle a variety of tasks such as patient admissions, medical record updates, laboratory test scheduling, and more. By effectively scheduling these tasks, the system ensures that high-priority operations are given precedence, reducing delays and optimizing throughput.

Furthermore, **resource management** is a vital aspect of any hospital environment. Resources such as medical staff, equipment, and treatment rooms must be managed efficiently to avoid overbooking, underutilization, or wastage.

1.1. Key Features

● Thread Synchronization: Employs multithreading with Python’s threading module to handle simultaneous patient admissions, discharges, and report updates securely.

● Dual Priority Admission: Implements a two-tier patient management model — First-Come-First-Served (FCFS) for normal cases and Priority Scheduling for emergency cases.

● Deadlock Prevention: Simulates safe state checks during concurrent bed allocation to prevent over-allocation and maintain system stability.

● Customizable Bed Configuration: Allows dynamic setup of total general beds and ICU beds, enabling flexible simulation of different hospital sizes.

● Live Admission Graphs: Integrates Matplotlib to display real-time bed occupancy statistics for both ICU and general wards as patients are admitted or discharged.

● Real-Time Visualization: Offers an animated GUI panel built with Tkinter to reflect patient admissions, discharges, and bed assignments in real time.

● Efficiency Reporting: Logs discharge histories and generates cumulative and average utilization reports to assess hospital resource performance.

● Educational Value: Bridges textbook OS scheduling principles and real-world hospital resource management, ideal for academic demonstrations and mini-projects.

CHAPTER 2
SCOPE OF THE PROJECT

This project presents a complete Hospital Bed Allocation and Management System — designed to simulate key OS-level concepts such as resource allocation, deadlock handling, priority scheduling, and real-time performance visualization in a healthcare setting. The system enhances both conceptual understanding and practical experimentation with multithreaded operations.

Concurrent Resource Allocation

● Simulates patient inflow in a multi-threaded environment where ICU and general beds are treated as limited shared resources.

● Emergency cases receive priority access to ICU beds, while normal cases follow a First-Come-First-Served (FCFS) strategy for general beds.

● Patient admissions and discharges happen concurrently, representing real-world dynamic resource contention and scheduling.

Deadlock Prevention and Priority Scheduling

● Before bed allocation, the system verifies the availability of resources to avoid deadlocks during concurrent patient handling.

● Priority scheduling logic ensures that critical patients (emergencies) are always given precedence over normal cases when assigning ICU beds.

● Real-time patient queues reflect OS-level queue manipulation for scheduling simulations.

Interactive GUI and Real-Time Visualization

● Built with Python’s Tkinter library, the system offers a clear visual representation of hospital bed allocation and patient movement.

● Separate zones for ICU and general beds allow users to observe real-time allocation and discharge events in the visual panel.

● Patient names and admission types are displayed in bed slots, reinforcing the link between OS allocation logic and real-world scenarios.

Live Performance Monitoring

● Real-time occupancy rates and patient admission statistics are visualized via Matplotlib bar graphs.

● Discharge logs are continuously recorded to evaluate bed turnover efficiency and historical usage trends.

● Interactive reports and export options allow for external data analysis.

Custom Configuration and Flexibility

● Users can configure:

○ Total ICU beds and general beds for the hospital.

○ Admission priority levels for incoming patients.

○ Patient name entries, admission policies, and simulation parameters.

● Flexible architecture allows easy extension into advanced resource management scenarios.

Academic and Research Relevance

● Designed for students, teachers, and OS enthusiasts studying:

○ Resource allocation algorithms in a shared, constrained environment.

○ Priority scheduling and FCFS queue handling in multi-threaded applications.

○ Deadlock prevention via safe state resource checks.

○ Real-time system monitoring and performance reporting.

● Provides a practical foundation for extending concepts into topics like starvation, fairness, real-time scheduling, and distributed systems.

CHAPTER 3

ARCHITECTURE

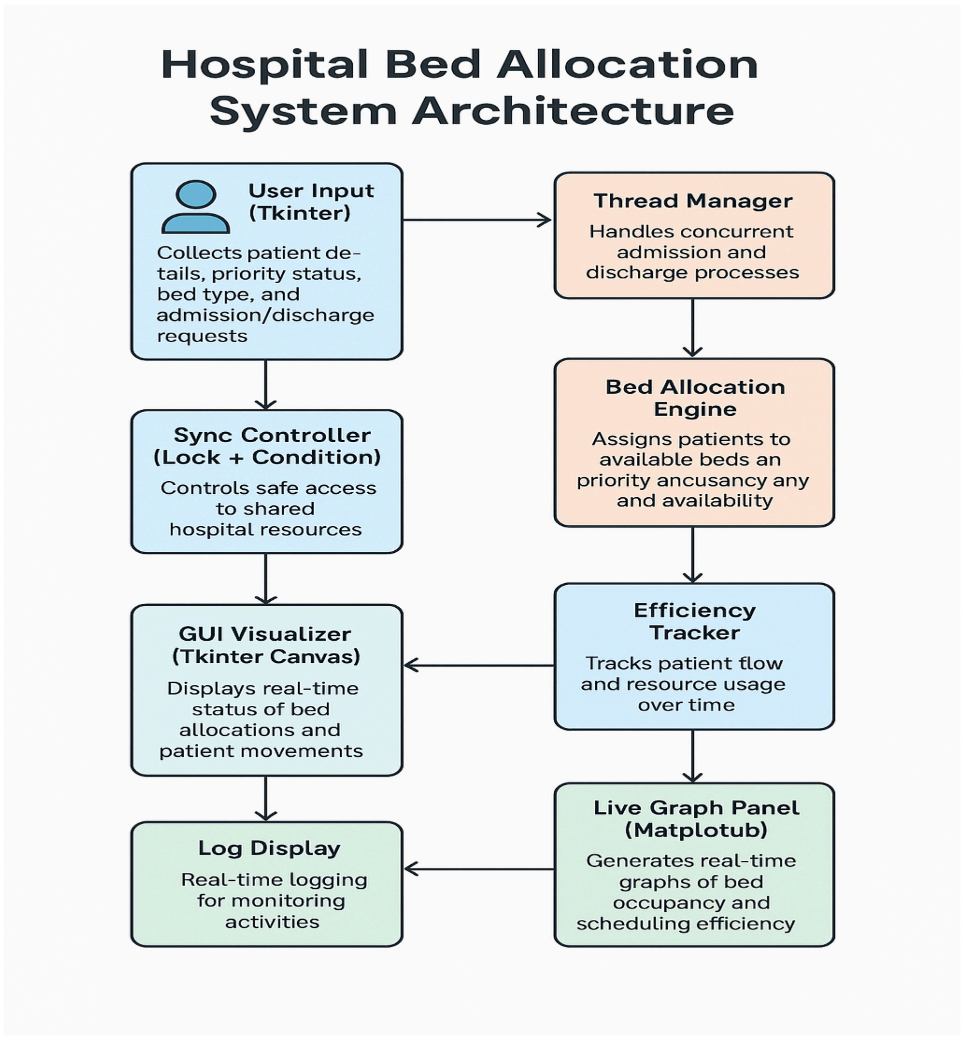

Hospital Bed Allocation System Architecture
User Input (Tkinter)
Collects patient details, priority status, bed type, and admission/discharge requests
Thread Manager
Handles concurrent admission and discharge processes
Sync Controller (Lock + Condition)
Controls safe access to shared hospital resources
Bed Allocation Engine
Assigns patients to available beds an priority ancusancy any and availability
GUI Visualizer (Tkinter Canvas)
Displays real-time status of bed allocations and patient movements
Efficiency Tracker
Tracks patient flow and resource usage over time
Log Display
Real-time logging for monitoring activities
Live Graph Panel (Matplotub)
Generates real-time graphs of bed occupancy and scheduling efficiency

Figure 1. Hospital Bed Allocation System Architecture

● User Input (Tkinter)
Collects patient details, priority status (normal/emergency), bed type (ICU/General), and admission/discharge requests via user-friendly input forms.

● Thread Manager
Handles concurrent patient admission and discharge processes using Python’s threading module for simulating real-world multi-user hospital operations.

● Sync Controller (Lock + Condition)
Controls safe access to shared hospital resources (beds):

○ Emergency patients: assigned priority access to ICU beds.
○ Normal patients: assigned based on First-Come-First-Served (FCFS) for General beds.
○ Ensures proper thread synchronization to avoid race conditions and deadlocks.

● Bed Allocation Engine
Handles dynamic assignment of patients to available beds based on availability, priority, and admission type.

● GUI Visualizer (Tkinter Canvas)
Displays real-time status of bed allocations, patient admissions, and discharges. Animates patient movements and bed occupancy using labeled visual elements.

● Log Display
Real-time event logging for all admission, discharge, and allocation activities in a dedicated Tkinter text area for system monitoring.

● Efficiency Tracker
Continuously tracks hospital resource usage:
Records total patients admitted, discharged, waiting, and average bed occupancy over time.

● Live Graph Panel (Matplotlib)
Generates real-time bar graphs to visualize bed occupancy status, patient flow, and scheduling efficiency dynamically as new data arrives.

CHAPTER 4
FLOWCHART

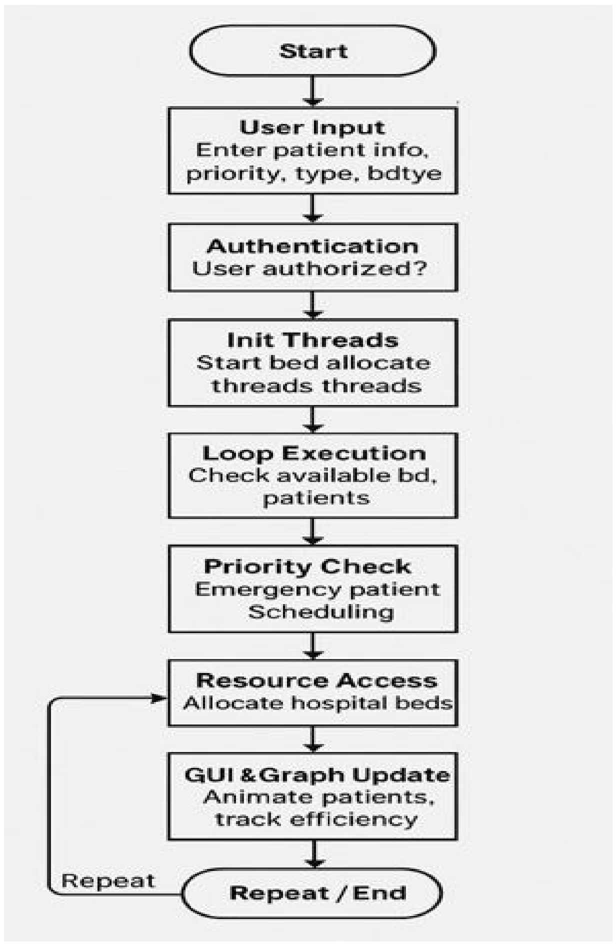

Start
User Input
Enter patient info, priority, type, bdtye
Authentication
User authorized?
Init Threads
Start bed allocate threads threads
Loop Execution
Check available bd, patients
Priority Check
Emergency patient Scheduling
Resource Access
Allocate hospital beds
GUI &Graph Update
Animate patients, track efficiency
Repeat
Repeat / End

Figure 1: Flowchart of Hospital Bed Allocation System

1. **Start** – Launch the app and initialize the GUI.

2. **User Input** – Enter patient information, including priority, type (emergency or general), and the required bed type.
3. **Authentication** – Check if the user (admin, doctor, or nurse) is authorized to access the system.
4. **Init Threads** – Start the threads for managing bed allocation and resources concurrently (using multithreading).
5. **Loop Execution** – The system constantly checks if there are available beds and if patients need to be assigned.
6. **Priority Check** – If there are emergency patients, they are prioritized for bed allocation over general patients.
7. **Bed Allocation** – Allocate beds based on availability:
    - **General Patients** – First-Come-First-Serve (FCFS) Scheduling.
    - **Emergency Patients** – Priority Scheduling.
8. **Resource Access** – Grant access to hospital beds for patients based on scheduling.
9. **GUI & Graph Update** – Show animations of patients moving to beds, update the status and scheduling efficiency graph.
10. **Discharge Process** – Once a patient is discharged, release the bed and update the system.
11. **Repeat / End** – The system continues to operate until closed or the user ends the simulation.

CHAPTER 5
CODE IMPLEMENTATION

5.1. PROGRAM

```
def load_data():
    # Loads hospital data from JSON file or initializes default data if file not found.

def save_data(data):
    # Saves the current hospital data to JSON file.

def load_discharge_history():
    # Loads discharged patients' history from JSON file or returns an empty list if file not found.

def save_discharge_history(history):
    # Saves the discharge history list to JSON file.

def register_user():
    # Registers a new user with username, password, and role, and saves it to data.

def authenticate():
    # Authenticates user credentials and shows the main menu if valid.

def admit_patient():
    # Admits a patient using priority scheduling and updates bed status, visualization, and report.
```

```
def discharge_patient():
    # Opens a window to select and discharge a patient, updates bed status and discharge history.


def update_visualization():
    # Updates the Tkinter canvas to visually show bed allocations for ICU and General wards.


def update_report_table():
    # Updates the report table with currently admitted patients sorted by priority and timestamp.


def show_main_menu():
    # Switches the interface from login to the main menu and updates the display.


def logout():
    # Logs out the current user and returns to the login screen.


def show_scheduling_efficiency():
    # Displays a bar graph showing the number of patients admitted by FCFS and Priority
scheduling.
```

5.2 SCREENSHOTS

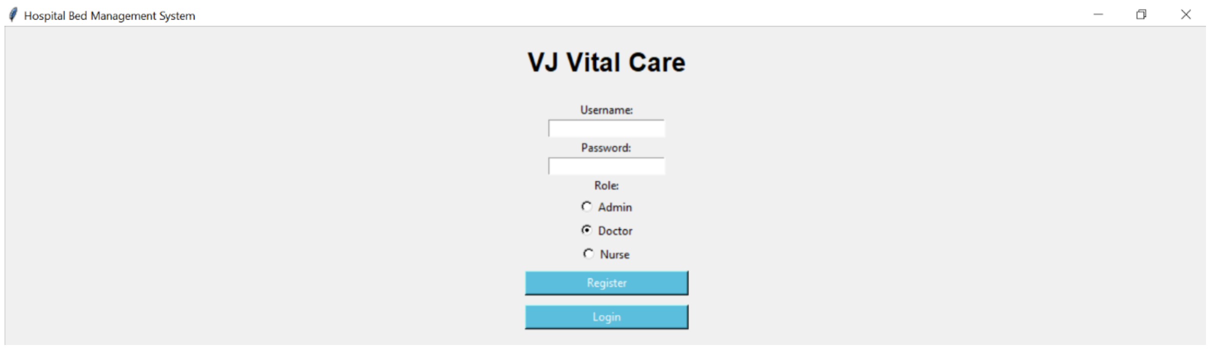
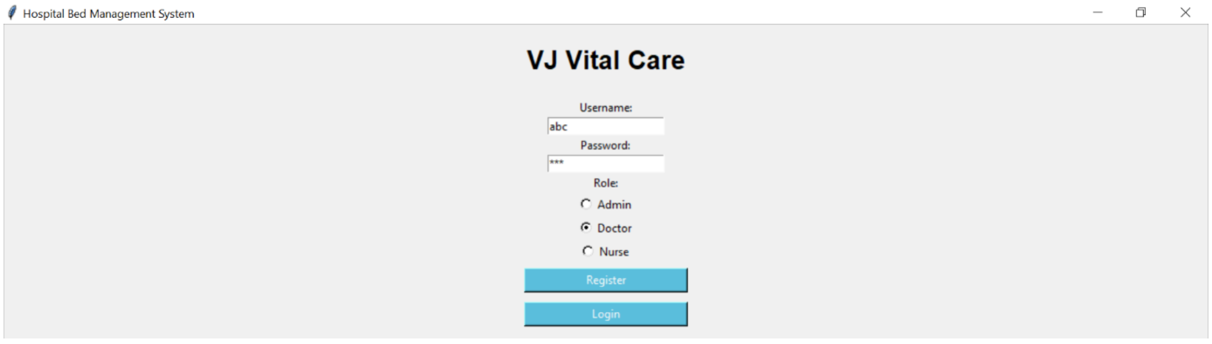


Fig 1. Login page

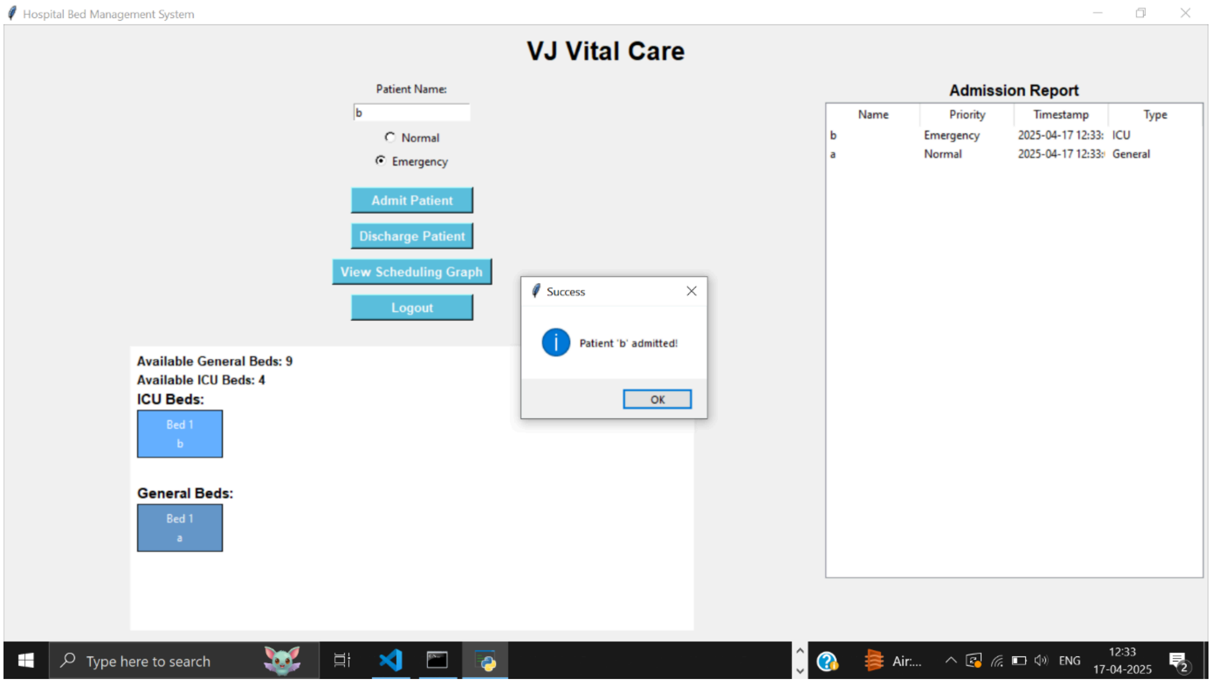


Username: abc

Password: ***

Role: Admin [✓ Doctor] Nurse

Fig 2. Login



Patient Name: b

Normal [✓ Emergency]

Patient 'b' admitted!

Available General Beds: 9
Available ICU Beds: 4

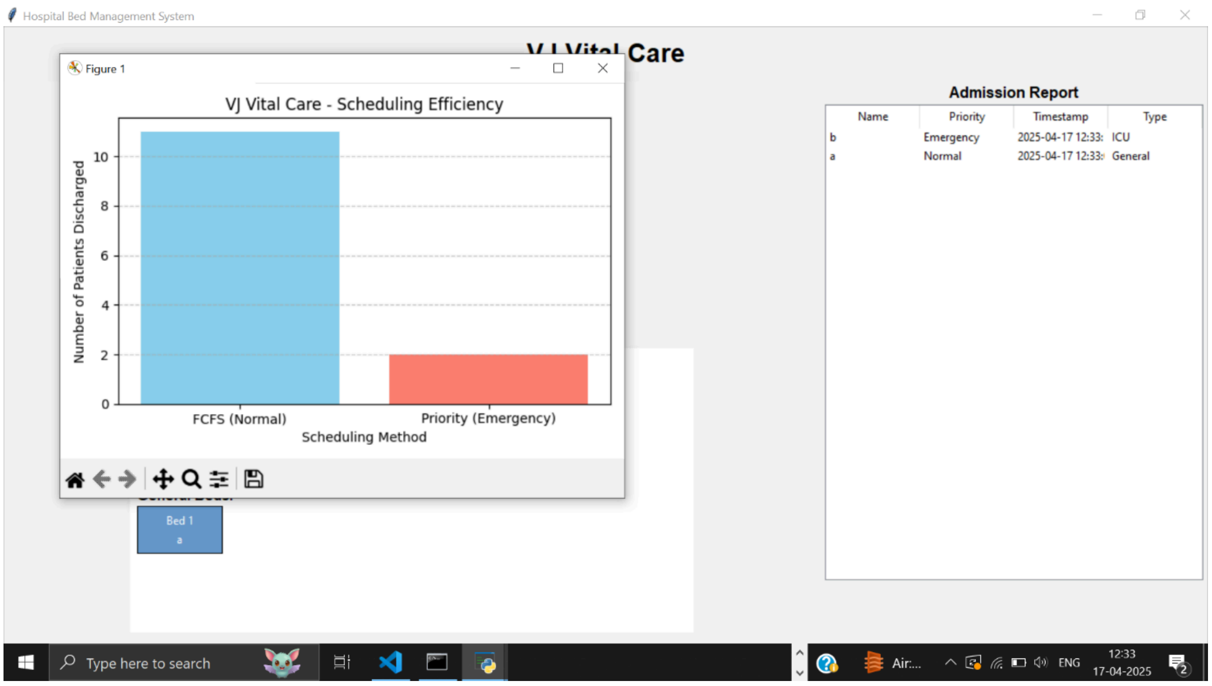
ICU Beds:
Bed 1
b

General Beds:
Bed 1
a

Admission Report

Name	Priority	Timestamp	Type
b	Emergency	2025-04-17 12:33:	ICU
a	Normal	2025-04-17 12:33:	General

Fig 3. Patient admission



Admission Report

Name	Priority	Timestamp	Type
b	Emergency	2025-04-17 12:33:	ICU
a	Normal	2025-04-17 12:33:	General

Fig 4. Admission Report

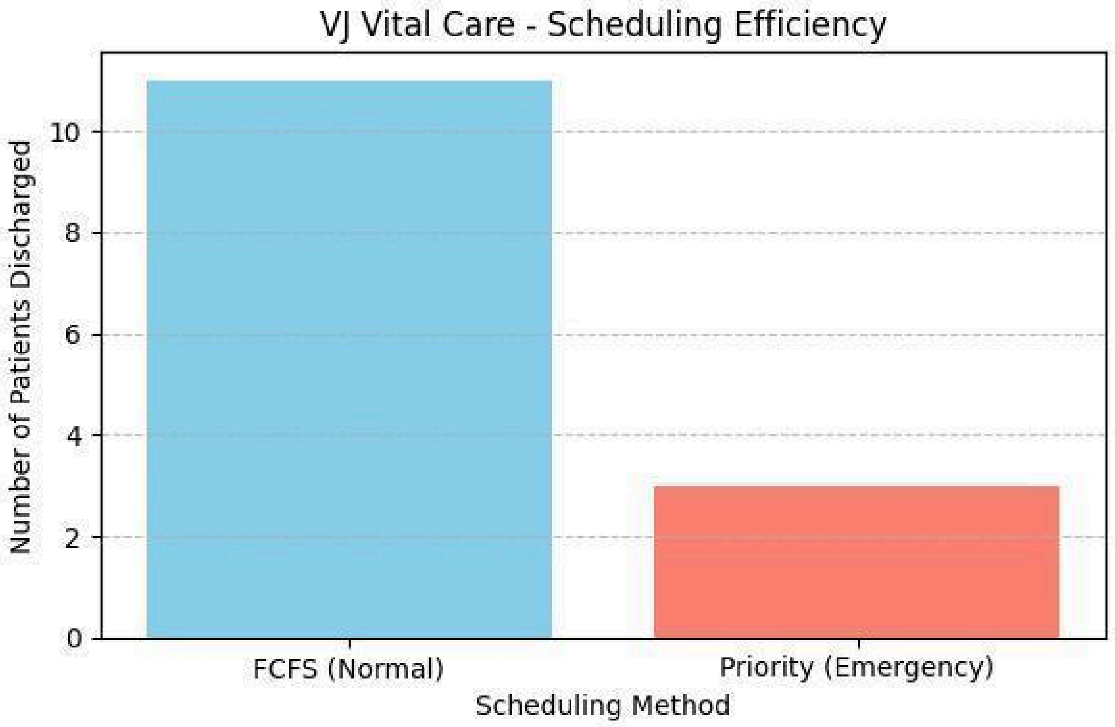


Fig 5. Scheduling Efficiency graph

CHAPTER 6
CONCLUSION

The **Hospital Bed Allocation & Resource Management System** successfully demonstrates the real-world application of core **Operating System concepts** such as **process scheduling, resource allocation, synchronization, and multithreading** using Python. This project simulates patient admission and bed assignment based on **First-Come-First-Serve (FCFS)** and **Priority Scheduling** policies, making it a practical and educational toolkit for understanding concurrent programming principles.

Through the use of **multithreading**, the system handles multiple patient admission requests simultaneously, preventing race conditions and ensuring fair resource (bed) distribution. The integration of a **Tkinter-based GUI** along with **real-time Matplotlib graphs** provides both visual clarity and system performance insights.

This simulation is designed not only as a college mini-project but as a foundation for further learning in **Operating System Design, Synchronization Techniques**, and **Real-Time Systems**. It also reinforces OS theories such as **critical section handling, deadlock prevention, and efficient scheduling algorithms**, offering both theoretical and hands-on learning for students.

Advantages

✔️ **Educational Value**
→ Simplifies complex OS topics like synchronization, deadlocks, and scheduling through interactive hospital-based scenarios.

✔️ **Real-Time Visualization**
→ Live GUI shows patient status transitions (Waiting, Allocated, Discharged) for better conceptual clarity.

✔️ **Dynamic Scheduling Algorithms**
→ Implements both **FCFS** and **Priority-based Scheduling**, demonstrating real-world hospital triage and resource management strategies.

✔️ **Multithreaded Simulation**
→ Uses Python's **threading** module to model concurrent patient arrivals and bed allocation.

✔️ **Performance Reporting**
→ Generates efficiency reports (wait time, occupancy rate) using **Matplotlib graphs** for real-time monitoring.

✔️ **User Configurability**
→ Supports dynamic setup of patient counts, priority levels, bed capacity, and admission speeds.

✔️ **Deadlock Awareness**
→ Built-in scheduling logic avoids deadlock-like scenarios (resource starvation) during simultaneous bed requests.

CHAPTER 7

FUTURE WORK

- **Advanced Scheduling Algorithms**
  Integrate AI-driven dynamic schedulers to optimize patient admissions during peak hours.

- **Real-Time Hospital Data Sync**
  Integrate with real-world hospital management systems or simulated medical IoT devices for real-time data.
- **Deadlock Detection and Recovery**
  Implement techniques like **Wait-Die / Wound-Wait** and enhanced deadlock recovery beyond safe-state validation.
- **IoT Integration**
  Connect with IoT-based medical beds and monitoring systems for real-time automatic updates.
- **Scalability Improvements**
  Expand the simulation to handle large hospitals with multi-ward and cross-department bed allocation logic.
- **Mobile Application Development**
  Extend the GUI into a mobile-friendly interface for staff to monitor and manage bed allocation remotely.
- **Blockchain-Backed Logging**
  Use blockchain for immutable logging of patient admissions, discharges, and resource allocations.
- **Role-Based Access Control (RBAC)**
  Implement user roles like **Admin, Doctor, Nurse, and Staff** for permission-based access to system features.
- **Real-Time Alerts & Notifications**
  Push instant alerts for **bed shortages, emergency admissions, or critical occupancy thresholds**.
- **Cloud Deployment**
  Deploy on cloud infrastructure (AWS, Azure, Google Cloud) for multi-user access and real-time collaborative learning.

CHAPTER 8

REFERENCES

[I] **William Stallings**, "Operating Systems – Internals and Design Principles", 9th Edition, Pearson, 2018.

[II] **Andrew S. Tanenbaum & Herbert Bos**, "Modern Operating Systems", 4th Edition, Pearson, 2016.

[III] **Silberschatz, Galvin, Gagne**, "Operating System Concepts", 10th Edition, Wiley, 2018.

[IV] **Python Official Documentation** — https://docs.python.org/3/

[V] **Tkinter GUI Programming** — https://tkdocs.com/

[VI] **Matplotlib Documentation** — https://matplotlib.org/stable/contents.html