

Hospital patient management system (HPMS):

# Alsdd project made by genesis team :

* Saibi Mohamed Abdelilah
* Benachour Mohamed Fouad

GROUPE:B1

Assigned professor: Dr.Lakhdari

# Abstract :

**The Hospital Patient Management System (HPMS)** automates the management of patient data through the use of key data structures such as queues, stacks, and trees. It supports essential features like CRUD operations, appointment scheduling, and an undo discharge function. With both console and GUI interfaces, HPMS is designed for flexibility and scalability. The system showcases how fundamental data structures can be effectively applied to optimize real-world healthcare workflows.

Data structures ………………………………………………………………………………….

Algorithms ...............................................................................................................................

Conclusion.....................................................................................................................................

Screenshots.....................................................................................................................................

38

Key questions ansewrs .................................................................................................................

39

How to run program......................................................................................................................

[© TemplateLab.com](https://templatelab.com/)

34

30

21

15

11

System design............................................................................................................................... .....................................................................................................................................

10

Objectives......................................................................................................................................

7

Problem.........................................................................................................................................

5

Introduction...................................................................................................................................

**CONTENTS**

# Introduction:

The **Hospital Patient Management System (HPMS)** is a software solution developed to streamline and automate patient-related operations within healthcare facilities. In today’s fast-paced medical environment, the ability to efficiently manage patient data, doctor assignments, appointment scheduling, and discharge processes is essential to delivering high-quality care. Traditional paper-based systems are often prone to errors, delays, and inefficiencies—highlighting the need for a digital solution that prioritizes accuracy, speed, and reliability.

This project leverages core data structures—such as arrays, linked lists, queues, stacks, and trees—to replicate and manage real-world hospital workflows. Key features include patient registration, dynamic queue management, doctor-patient assignment, and an undo discharge functionality. These elements demonstrate the practical application of algorithmic concepts in solving challenges common in healthcare administration.

HPMS offers two user interfaces:

* A **console-based menu** for demonstrating algorithmic behavior and testing core functionalities.
* A **GTK-based graphical user interface (GUI)** designed to enhance usability and accessibility for hospital staff for basic operations.

To ensure data persistence, the system incorporates file handling capabilities, allowing patient and appointment data to be saved and retrieved across sessions.

Through this project, we explore the integration of computer science principles with healthcare needs—emphasizing modular programming, efficient data handling, and intuitive interface design.

# Problem:

Traditional hospital management systems often rely on manual processes or outdated digital tools, leading to several operational challenges that harms people’s life:

## **1. Inefficient Patient Tracking**

Paper-based records make it difficult to access or update patient data promptly. This results in delays in treatment and decision-making.

## **2. Poor Queue Management**

Without automated scheduling, patient flow becomes disorganized. Long wait times and inefficient prioritization of cases negatively affect both patients and hospital staff.

## **3. Error-Prone Processes**

Manual data entry increases the likelihood of mistakes in patient records, prescriptions, and discharge summaries—potentially compromising patient safety.

#### **4. Lack of System Integration**

Disjointed systems across departments make coordination difficult. The absence of real-time updates hinders effective communication and delays care delivery.

#### **5. No Undo or Backup Mechanism**

Once data is mistakenly deleted or a patient is discharged incorrectly, recovery is difficult or impossible, leading to permanent data loss and compromised care continuity.

### **How HPMS Solves These Problems**

The **Hospital Patient Management System (HPMS)** offers a structured, algorithm-driven solution that addresses the core issues of traditional systems:

* **Efficient Data Handling:**  
  Uses structured data types (arrays, linked lists, queues, stacks and trees) to allow fast and reliable retrieval and modification of patient records.
* **Automated Queue Management:**  
  Implements a **FIFO queue** system to manage patient appointments and walk-ins, improving flow and reducing wait times.
* **Accurate CRUD Operations:**  
  Ensures consistent and error-minimized creation, retrieval, update, and deletion of records through user-friendly interfaces.
* **Undo Functionality:**  
  Utilizes a **stack** to implement an undo feature for discharged patients, preserving data integrity and supporting error recovery.
* **Dual Interface Support:**  
  Provides both a **console interface** (ideal for testing algorithms and data structures) and a **GTK-based GUI** for enhanced usability by non-technical staff for simple operations.

### **Objectives:**

### **Build a management system for hospital using c language**

1. Implement data structures studied in alsds and alsdd (array until trees)
2. Add, View, Update, and Delete Patient and Doctor Data (CRUD)
3. Search Patient and Doctor by ID/Name
4. Assign Doctor to Patient and Schedule Appointments
5. Manage Discharge Patient with Undo Functionality of Last Discharge
6. View and Manage Waiting Queues
7. Display Hospital Structure
8. Save and Load Patient and Doctors and Departments
9. Console Mode
10. GTK GUI Mode (for basic operations
11. List all patients in the two modes
12. Generate Reports of patients in .txt file with important information.
13. Basic user authentication.
14. Pharmacy program
15. System to track hospital balance (income/expenses).

# System Design & Modeling:

In a hospital setting, managing doctors, patients, departments, and medicines efficiently is essential for providing timely and effective care. The **Hospital Patient Management System (HPMS)** is designed to simulate these real-world interactions using structured data models and algorithms, enabling smooth coordination across all parts of the hospital.

## **Core Components**

### **Doctors:**

A Doctor in hpms has many fields to permit to manage his responsibilities and relations in the program

His structure has the fields :

* name
* age
* speciality
* address
* rank
* numPatients
* maxPatients
* department
* id
* doctorQueue
* wage
* patientsHead

### **Patients:**

A Patient in hpms has many fields to permit to manage his relations in the program

* name
* age
* medicalCase
* address
* condition
* assignedDoc
* department
* id
* isAssured
* sessionCost

### **Hospital**

**The hospital has 4 departments that are (lab, cardiology, physiology, neurology )**

**The balance of the hospital is the sum of all income for the departments (session costs) minus the sum of the expenses (doctors wages)**

#### **Wages :**

They are directly related to the rank of the doctor

#### **Session cost :**

It is connected to the condition (surgery/visit) and the assurance of the patient

## **Departments**

**Each department in hospital has a number of surgery and visit rooms if they are full u can’t add patients and they will be deported to another hospital**

**Also every department has unique balance**

**Departments are formed by :**

* department
* doctors
* patients
* numDoc
* numPat
* surgeryRoomsDepa(busy/free)
* patientRoomsDepa(busy/free)
* balence
* income
* expenses

## **Relations:**

Patients are assigned to doctors under many conditions :

1. the condition of patient is compatible with rank of doctor (emergency condition cannot be treated by an intern )
2. free surgery or patient rooms
3. existence of doctors in the department
4. the existence of doctors that did not reach max number of patients (generated by functions dynamic)

if no doctor satisfies these condition the patient will be deported to another hospital

# **Data Structures in HPMS**

The **Hospital Patient Management System (HPMS)** utilizes a variety of data structures to effectively manage doctors, patients, departments, and medical workflows. Each structure is carefully selected based on its strengths in handling specific operations, ensuring both efficiency and scalability.

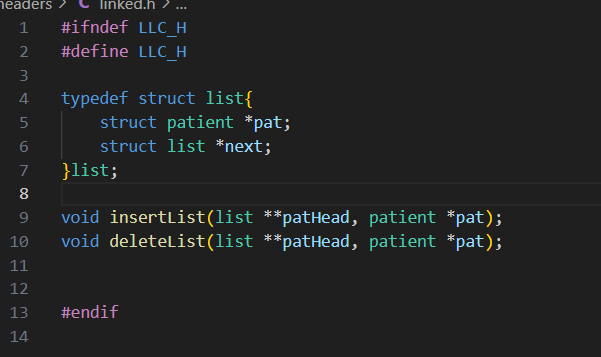
### **1. Linked Lists**

**Purpose:**  
Used to dynamically store and manage the list of patients assigned to each doctor.

**Advantages:**

* No fixed size, allowing flexible patient allocation
* Efficient insertion and deletion operations

**Use Case:**  
Each doctor maintains a personal linked list of their patients, allowing quick updates as patients are added or removed from their care and easily print his patients.



### **2. Queues (FIFO)**

**Purpose:**  
Manage the order of patient sessions and appointments per doctor.

**Advantages:**

* Ensures fair scheduling with first-come-first-served order
* Supports fast addition (enqueue) and removal (dequeue)

**Use Case:**  
Each doctor has a queue for incoming patients, helping manage appointment flow in a fair and organized manner.



### **3. Arrays**

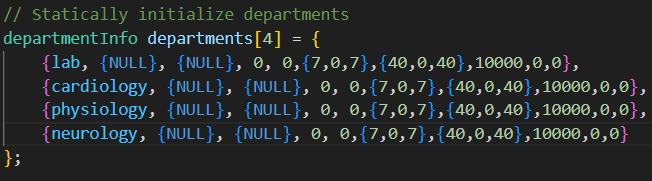
**Purpose:**  
Store fixed-size data such as department lists or predefined values.

**Advantages:**

* Instant access to elements using indices
* Simple and predictable structure for static data

**Use Case:**  
The system stores predefined departments such as Cardiology, Neurology, and Physiology using arrays for easy lookup and iteration.





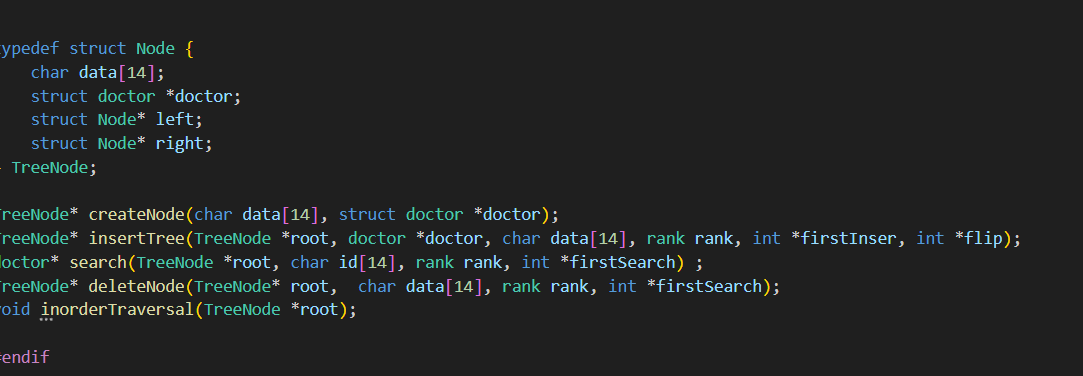
### **4. Trees**

**Purpose:**  
Organize and store patient and doctor records for optimized searching.

**Advantages:**

* Fast lookup by ID when balanced
* Naturally maintains sorted order for generating reports

**Use Case:**  
Patient and doctor records are stored in BSTs, allowing efficient access by ID and organized traversal for display or search operations.



### **5. Structs and Enums**

**Structs:**  
Used to model complex entities like doctors, patients, and departments, grouping related data together in a clean and logical structure.

**Enums:**  
Enhance code clarity and maintainability by defining readable constants for properties such as condition status, department type, or doctor rank.

**Use Case:**  
Each patient or doctor is defined using a struct with fields such as name, age, department, and condition, with enums ensuring readable and consistent values across the system.

### **Why These Structures?**

* **Linked Lists:** Perfect for managing changing patient-doctor assignments without worrying about capacity limits.
* **Queues:** Ensure fair and orderly patient flow for appointments and treatments.
* **Arrays:** Efficient for handling fixed data like department types or static room allocations and linear search.
* **Trees:** Allow fast and organized access to patient and doctor records, especially when dealing with large datasets.
* **Structs & Enums:** Offer a structured and human-readable way to model real-world entities in the healthcare domain.

# **Algorithms Implemented in HPMS:**

The **Hospital Patient Management System (HPMS)** relies on a set of efficient algorithms to manage patient records, doctor assignments, and hospital workflows. These algorithms ensure quick access to data, accurate updates, and seamless user operations.

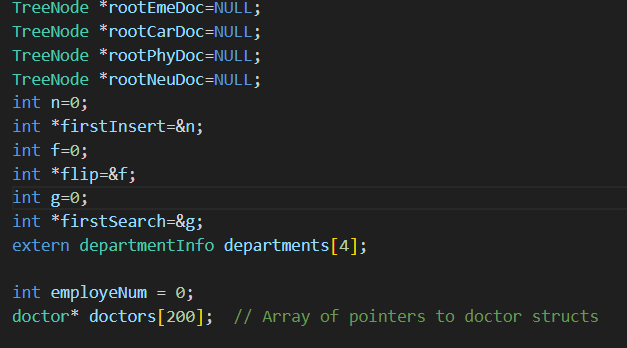
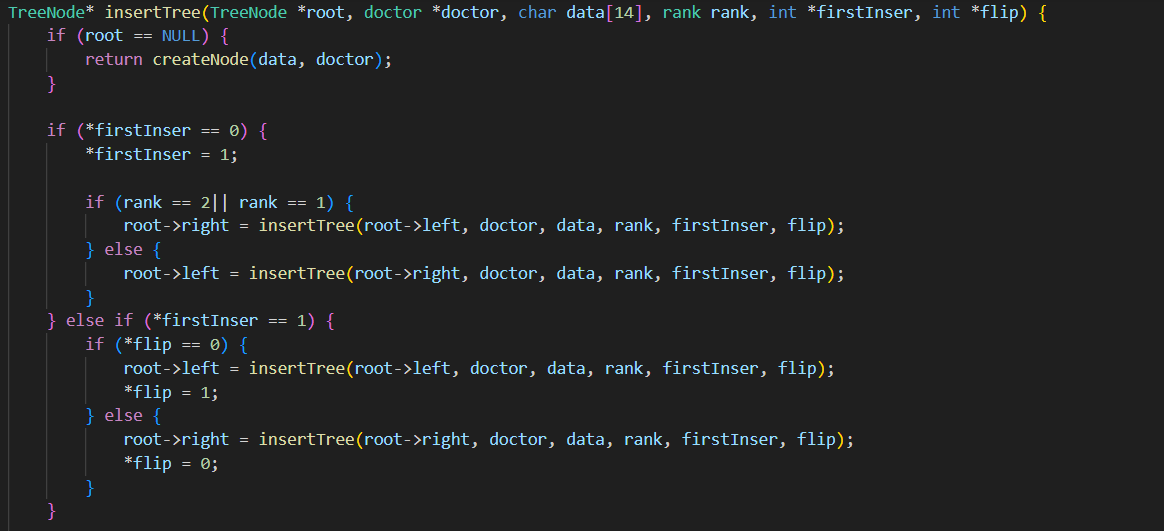
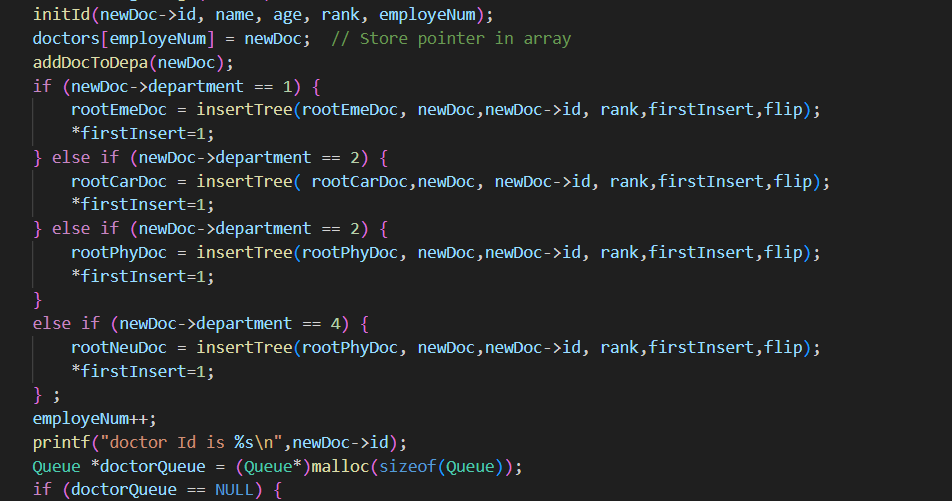
### **1. Balanced Tree Insertion for Doctors and Patients**

**Purpose:**  
Efficiently store and retrieve doctors and patients within departments.

**Method:**  
A flip-flop balancing technique is used during insertions to keep the binary search tree balanced. This ensures fast search performance, especially when looking up patients or doctors by ID.

**Search Modes:**

* **ID-Based Search:** Fast lookup using tree structure.
* **Name-Based Search:** Slower, requires full tree traversal but supports more flexible searches.

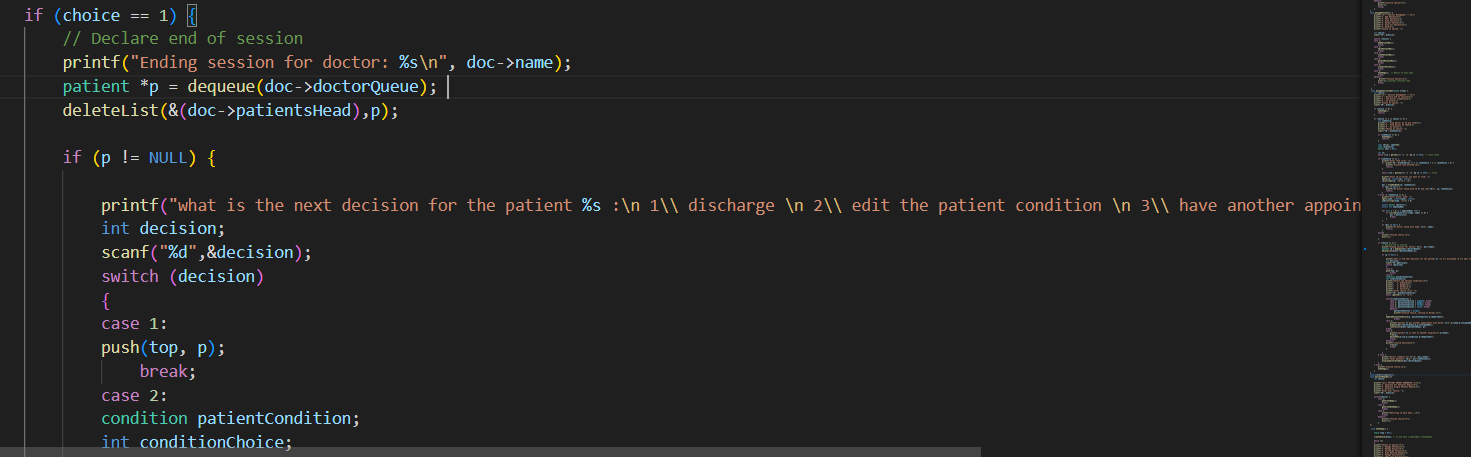
### **2. Session Management (4 Options at Session End)**

**Purpose:**  
Handle the end of a patient’s treatment session with flexible options.

**Available Actions:**

* Transfer to another hospital (if the current one is full)
* Discharge and log the patient
* Reschedule for another session
* Update the patient's condition and reroute

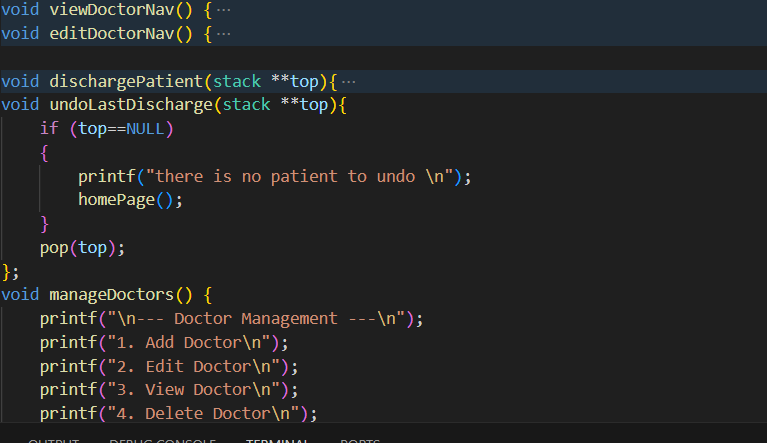
This allows the system to adapt based on patient status and hospital capacity.



### **3. Undo Last Discharge (Using a Stack)**

**Purpose:**  
Reverse an accidental discharge of a patient.

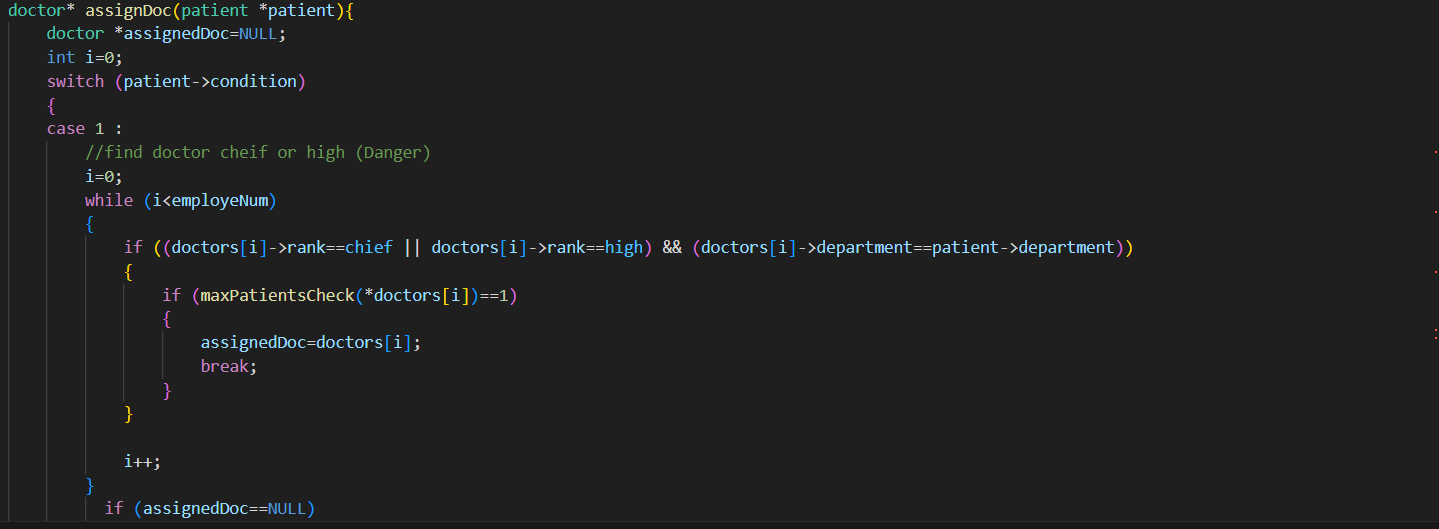
**Method:**  
Discharged patients are stored in a stack. The last discharged patient can be quickly restored by popping from the stack, ensuring data is never permanently lost by mistake.



### **4. Doctor Assignment Based on Patient Condition**

**Purpose:**  
Match patients to the most suitable available doctor.

This helps balance the load and ensures critical patients get timely care.

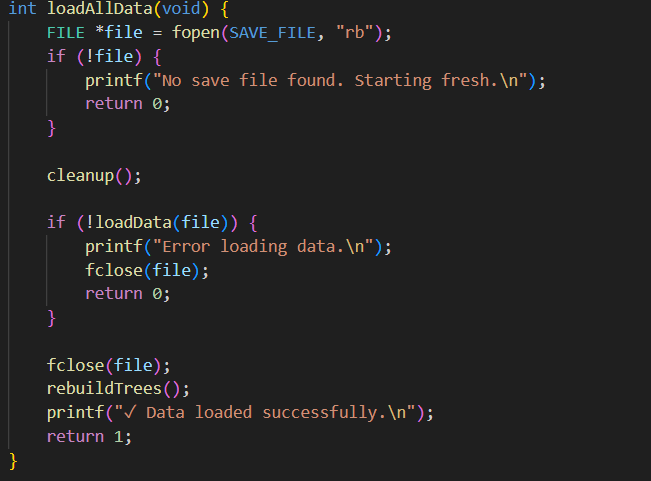
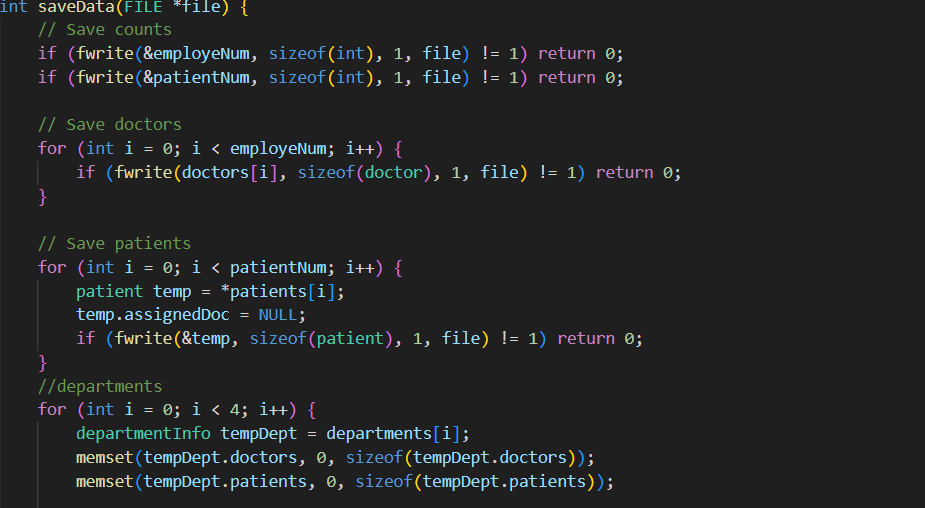


### **5. Data Persistence (Save and Load)**

**Purpose:**  
Preserve patient and doctor data between sessions.

**How It Works:**

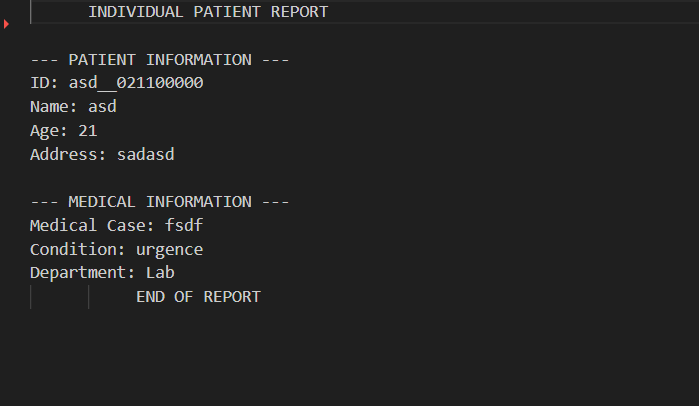
* When saving: Data from trees, queues, and other structures is written to a binary file.
* When loading: The file is read and used to rebuild the system state.  
  This allows the hospital to recover from shutdowns or restart with full data.

### **6. Report Generation**

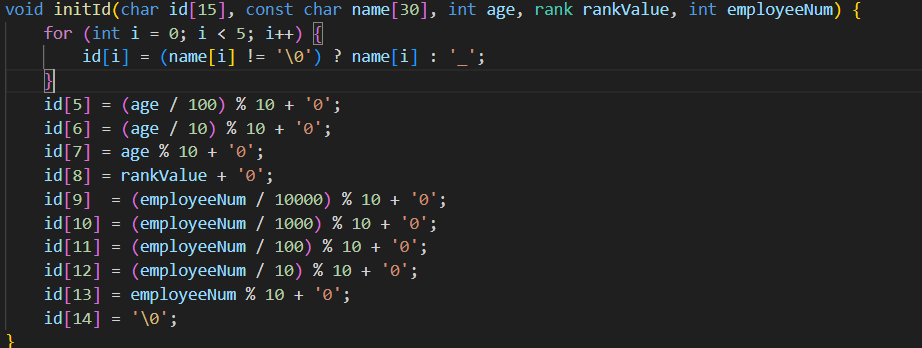
**Purpose:**  
Create readable summaries of system activity.

**Generated Reports Include:**

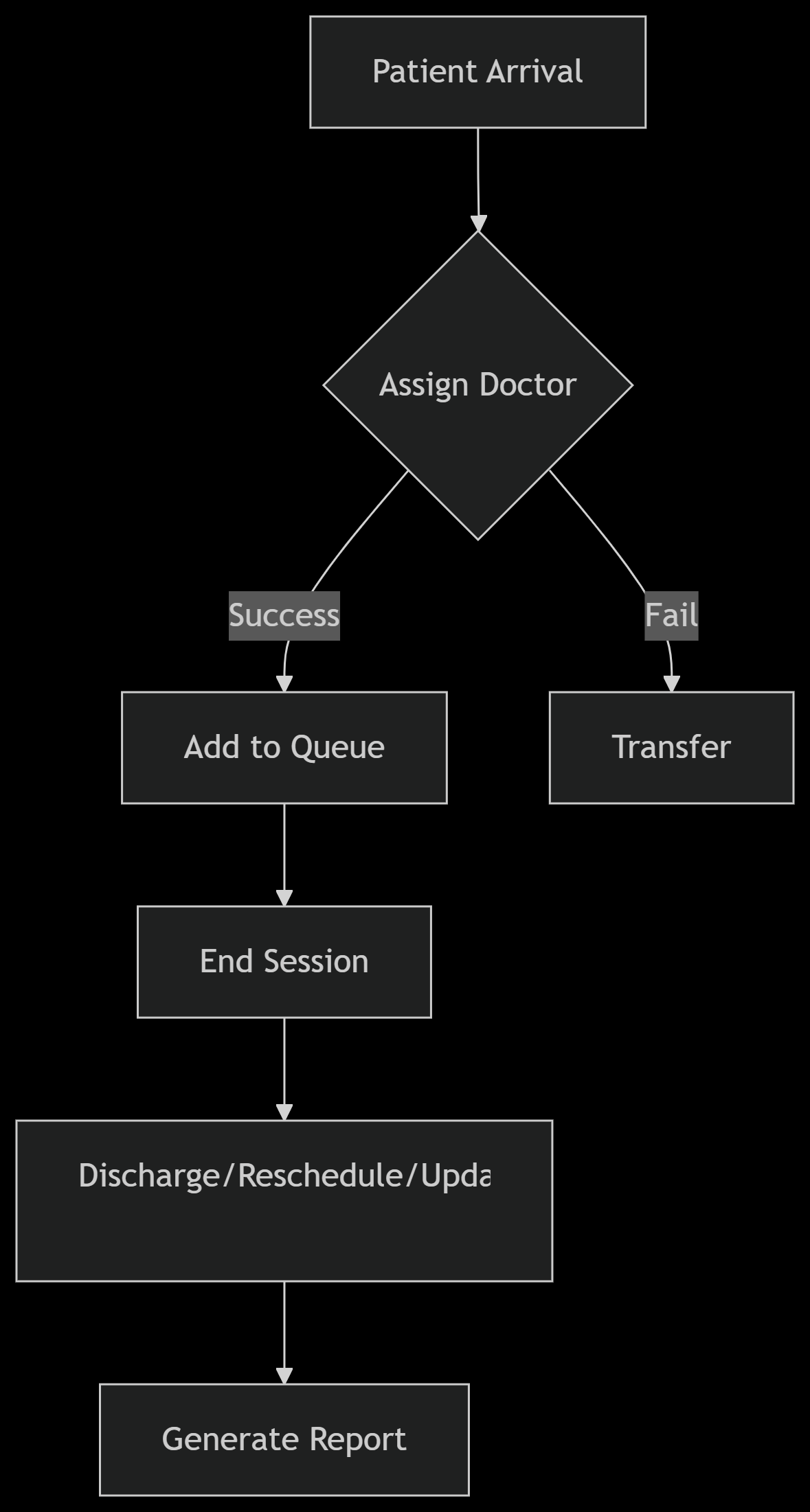
* Discharged patient logs
* Hospital financial summaries (income, expenses, room usage)  
  Reports are saved as .txt files for easy viewing and printing.****

## 7. initializing id :

The initId function generates a unique 14-character identification string for a doctor by combining key attributes: their name, age, rank, and employee number. The process starts by copying the first five characters of the doctor's name into the ID. If the name is shorter than five characters, underscores (\_) are used to fill the remaining positions. Next, the age is converted into a three-digit string and placed in positions 5 to 7 of the ID. The rank, represented as an enum value, is converted to a single digit character and stored at position 8. Finally, a five-digit employee number is split into individual digits and placed in positions 9 to 13. The ID string is null-terminated to ensure it is treated properly as a C string. This structured approach ensures that each ID encodes meaningful, unique information while maintaining a fixed format for easy parsing and comparison.



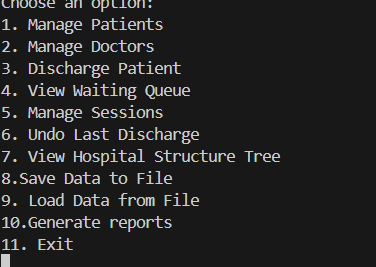
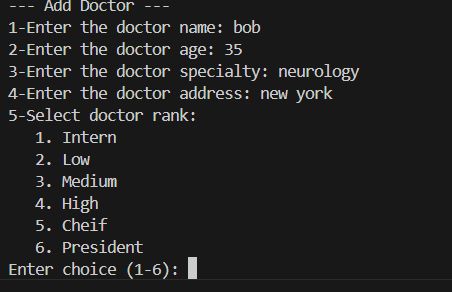
# Patient management walkthrough :



# **Screenshots from the program:**

## Console mode:

### 

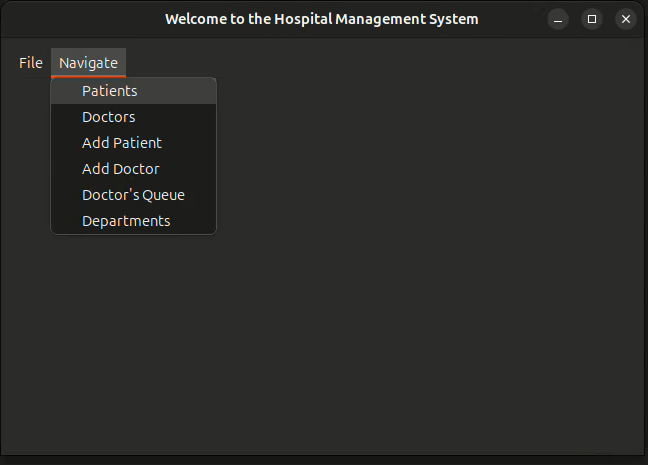
### 

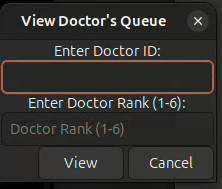
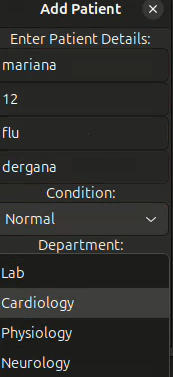
### 

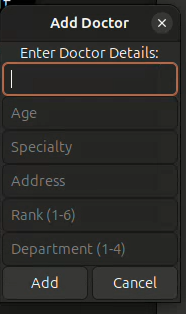
### 

### 

## **GUI MODE:**



**** 



# **Conclusion**

The **Hospital Patient Management System (HPMS)** effectively shows how well designed data structures and algorithms can solve real-world problems in healthcare management. Through its use of trees, queues, stacks, and structured programming, HPMS improves how hospitals handles patients, doctors, sessions, and basic pharmacy operations.

### **Key Features Implemented**

* **Balanced Trees** for fast and organized searches of doctors and patients.
* **Queues** to ensure fair and efficient patient appointment scheduling.
* **Doctor Assignment** based on specialization, availability, and patient condition.
* **Session Management** with multiple possible outcomes (discharge, reschedule, transfer, etc.).
* **File I/O** to store and retrieve hospital data safely across sessions.
* **Basic Pharmacy Integration** that allows:
  + **Buying medicines** .
  + **Stock management** (checking medicine availability).

### **Key Achievements:**

#### **Efficiency**

* Fast patient/doctor search using balanced trees.
* Quick session handling through queue operations.
* Organized handling of medicine purchase and stock status.

#### **Flexibility**

* Supports multiple session outcomes.
* Adapts to different department requirements (e.g., surgery, emergency, general care).
* Basic pharmacy system connects patients to available medicines.

#### **Reliability**

* Binary file saving and loading ensure no patient or hospital data is lost.
* Report generation helps track hospital performance (e.g., discharges, income).
* Pharmacy stock control usage of medicines.

### **Pharmacy System (Basic Integration)**

While simple, the pharmacy module supports essential functions:

* **Buy**: Patients can receive prescribed medicines.
* **Stock**: Hospital can track and check available medicine inventory.

### **Future Enhancements**

To improve and expand the system:

* Add **AI-based triage** for prioritizing patients dynamically.
* Create a **web/mobile interface** for remote access by staff.
* Include **analytics** to predict medicine shortages and patient load.
* Use **barcode scanning** for safer medicine dispensing.

### **Final Note**

This project highlights how core computer science concepts—when applied thoughtfully—can solve real problems in hospital environments. By combining algorithmic logic with practical workflows, HPMS offers a solid foundation for more advanced and scalable healthcare systems in the future

### **Key Questions HPMS Answers**

* **How can hpms reduce patient wait times?**  
   → Through automated and dynamic queue management.
* **How can staff avoid losing critical patient data?**  
  → By save/load option.
* **How can hpms effectively track and manage the distribution of prescribed medications ?**

→ By providing a pharmacy system that has client side that permits the dynamic transaction and usage of medicines with delivery services.

# How to run the program:

For console mode run this command in the hpms directory

gcc src/authentication.c src/customer.c src/departments.c src/doctors.c src/hospital.c main.c src/medic.c src/nav.c src/patients.c src/pharm.c src/pharmaMain.c src/queue.c src/saveLoad.c src/stack.c src/treeDoc.c src/treePat.c src/resources.c src/linked.c -I. -o main –g

then run this command : ./main.exe

For GUI mode run this command in the hpms/gtk directory (you need to have gtk dependencies installed)

Gcc gtk.c ../src/patienrs.c ../src/doctors.c ../src/queue.c ../src/treePat.c ../src/treeDoc.c ../src/resources.c ../src/saveLoad.c ../src/departments.c ../src/nav.c ../src/linked.c ../src/stack.c ../src/hospital.c –o gtk ‘pkg-config –cflags –libs gtk+-3.0’

then run this command : ./gtk