

BISMILLAHIR RAHMANIR RAHEEM

6/26/2024

DATA STRUCTURE AND ALGORITHM

FINAL PROJECT

HOSPITAL MANAGEMENT SYSTEM

SUBMITTED TO:

MUHAMMAD SHAMSHUL ALAM

DEAN, SCIENCE AND ENGINEERING FACULTY

INTERNATIONAL ISLAMIC UNIVERSITY
CHATTOGRAM.

SUBMITTED BY:

MUNYIM MAHMUD MAHI, C231025

SAJEDUL ALAM FAHIM, C231031

MUHAMMAD MOINUL ISLAM, C231008



PUSH CODER

TEAM D-S-A

INTRODUCTION

Hospital Management System

This C++ program is designed to manage patient records in a hospital efficiently. The system leverages various data structures and file handling mechanisms to store, retrieve, and manage patient details. The main features include admitting patients, displaying patient lists, and discharging patients, with added functionalities such as automatic timestamping for admission times and doctor assignment.

Key Components and Features:

1. Patient Structure:

- Contains details such as patient ID, name, address, disease, admission date and time, admission timestamp for priority handling, and doctor name.
- Automatically sets the current date and time upon creation.

2. Node Structure:

- Defines a node for the linked list, holding patient data and a pointer to the next node.

3. PatientList Class:

- Manages the patient records using a linked list for sequential storage and a priority queue for priority-based operations.
- Provides functions to add, display, and remove patients, ensuring the list remains updated.
- Supports file operations to load patient data from a file on startup and save it back on exit, ensuring data persistence.

-
- 4. Main Function:**
 - Acts as the user interface for the hospital management system, presenting a menu for admitting patients, displaying the patient list, and discharging patients.
 - Includes a predefined list of doctors to be assigned to patients upon admission.

Workflow:

- **Admitting Patients:**
 - Prompts for patient details including ID, name, address, and disease.
 - Allows selection of a doctor from a predefined list.
 - Automatically records the current date and time as the admission timestamp.
 - Adds the patient to the linked list and priority queue for efficient management.
- **Displaying Patients:**
 - Displays a list of all admitted patients with their details in a tabular format.
- **Discharging Patients:**
 - Prompts for the patient ID to be discharged.
 - Requires confirmation from the assigned doctor before removing the patient from the list.
- **File Handling:**
 - Loads patient data from a file at startup.
 - Saves patient data to a file upon exiting, ensuring no loss of data.

Example Usage:

1. Admit a New Patient:

- User inputs patient details and selects a doctor from the list.
- System assigns the current date and time as the admission timestamp.

2. Display All Patients:

- User selects the option to display all patients.
- System prints a detailed list of all patients.

3. Discharge a Patient:

- User inputs the patient ID to be discharged.
- System verifies the patient ID and requires doctor confirmation before discharge.

By integrating these functionalities, the system ensures a streamlined and efficient process for managing patient records in a hospital setting. This implementation highlights the use of structures, linked lists, priority queues, and file handling in C++ to create a robust hospital management system.

IMPLEMENTATION

The project is a comprehensive hospital management system designed to handle patient admissions, display patient lists, and manage patient discharges. Here's a detailed explanation of its implementation:

1. Patient Structure

This structure holds the details of a patient including:

- `id`: Unique identifier for the patient.
- `patientName`: Name of the patient.
- `patientAddress`: Address of the patient.
- `disease`: Disease the patient is suffering from.
- `date`: Date of admission.
- `admissionTime`: Time of admission.
- `admissionTimestamp`: Timestamp for admission, used for priority comparison.
- `doctorName`: Name of the doctor assigned to the patient.

The `Patient` constructor automatically initializes `date`, `admissionTime`, and `admissionTimestamp` with the current date and time.

2. Node Structure

Defines a node in the linked list:

- `data`: Holds a `Patient` object.
- `next`: Pointer to the next node in the list.

3. PatientList Class

This class manages the patient records using a linked list for sequential storage and a priority queue for priority-based operations. Key functionalities include:

- **Constructor:** Initializes the `head` and `tail` pointers of the linked list to `nullptr`.
- **Destructor:** Cleans up all nodes in the linked list to prevent memory leaks.
- **addPatient:** Adds a new patient to both the linked list and the priority queue.
- **displayPatients:** Displays the details of all patients in the linked list.
- **removePatient:** Removes a patient from the linked list by their ID, with a confirmation prompt from the assigned doctor.
- **loadPatientsFromFile:** Loads patient records from a specified file.
- **savePatientsToFile:** Saves the current patient records to a specified file.

Main Function

The main function serves as the user interface for the hospital management system. It performs the following tasks:

1. Initialization

- Creates an instance of `PatientList`.
- Loads patients from a file named `patients.txt`.

2. Menu Display and User Input

- Continuously displays a menu with options to admit a patient, display the patient list, discharge a patient, or exit.

- Takes user input to select the desired operation.

3. Operation Handling

- **Admit Patient:**
 - Prompts the user for patient details.
 - Displays a list of available doctors for selection.
 - Adds the new patient to the `PatientList`.
- **Display Patient List:**
 - Calls the `displayPatients` function to print all patients' details.
- **Discharge Patient:**
 - Prompts the user for the patient ID to be discharged.
 - Calls the `removePatient` function to remove the patient after doctor confirmation.
- **Exit:**
 - Saves the patient list to `patients.txt` before exiting.

Example Usage

- 1. Admitting a Patient:**
 - User enters patient details (ID, name, address, disease).
 - Selects a doctor from the list.
 - System adds the patient with the current date and time as admission details.
- 2. Displaying Patients:**
 - User selects the option to display all patients.
 - System prints a tabular list of all admitted patients.
- 3. Discharging a Patient:**
 - User enters the patient ID for discharge.
 - System verifies the patient and requires confirmation from the assigned doctor.
 - Upon confirmation, the patient is removed from the list.

4. Saving and Loading:

- Patient data is loaded from `patients.txt` on startup.
- Patient data is saved to `patients.txt` upon exiting.

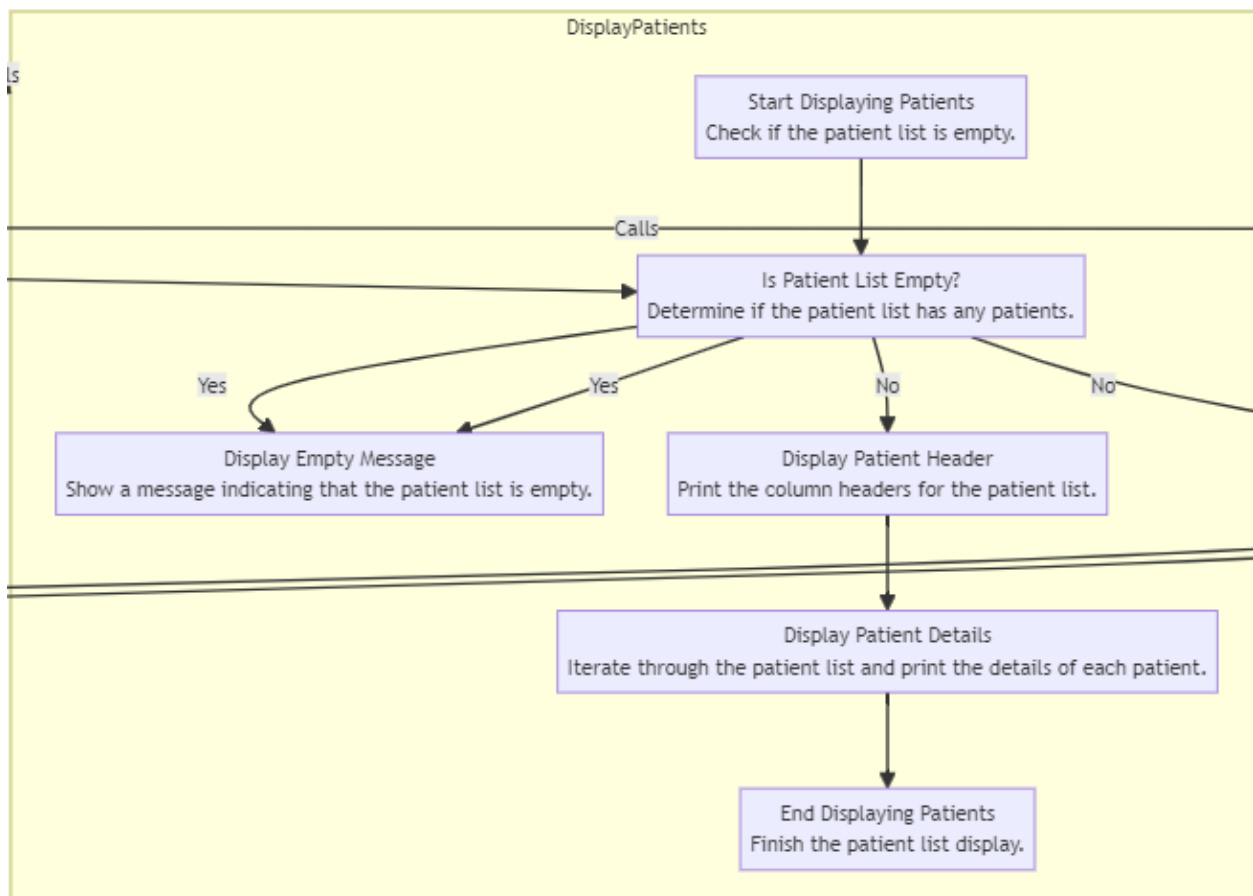
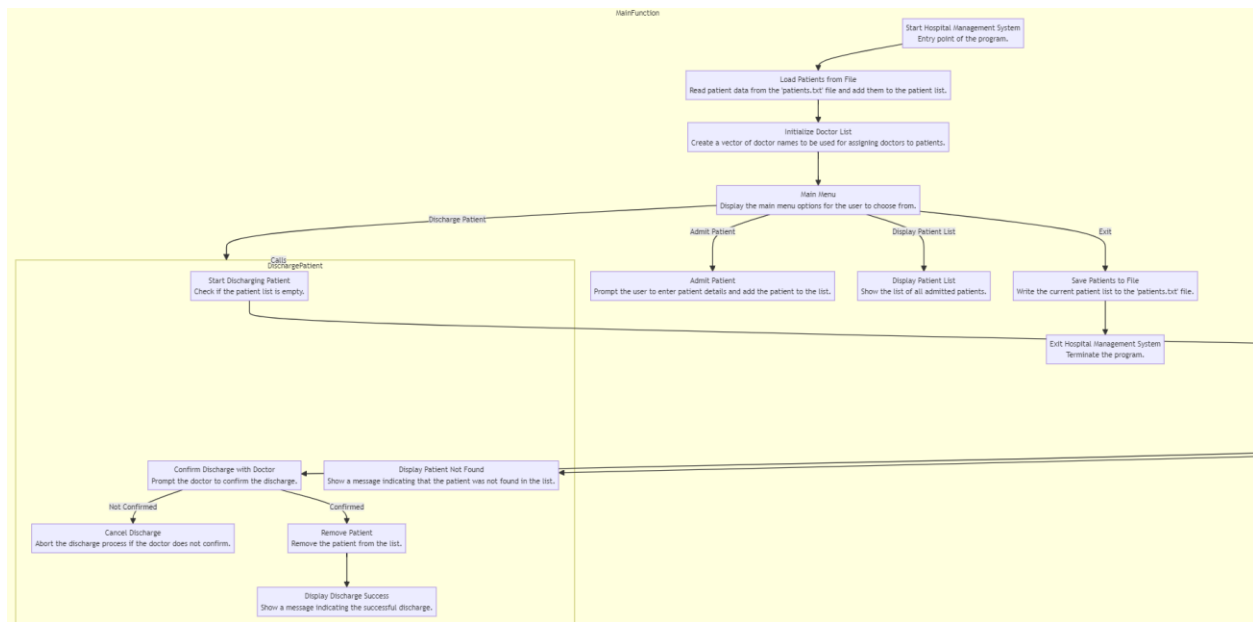
Priority Queue

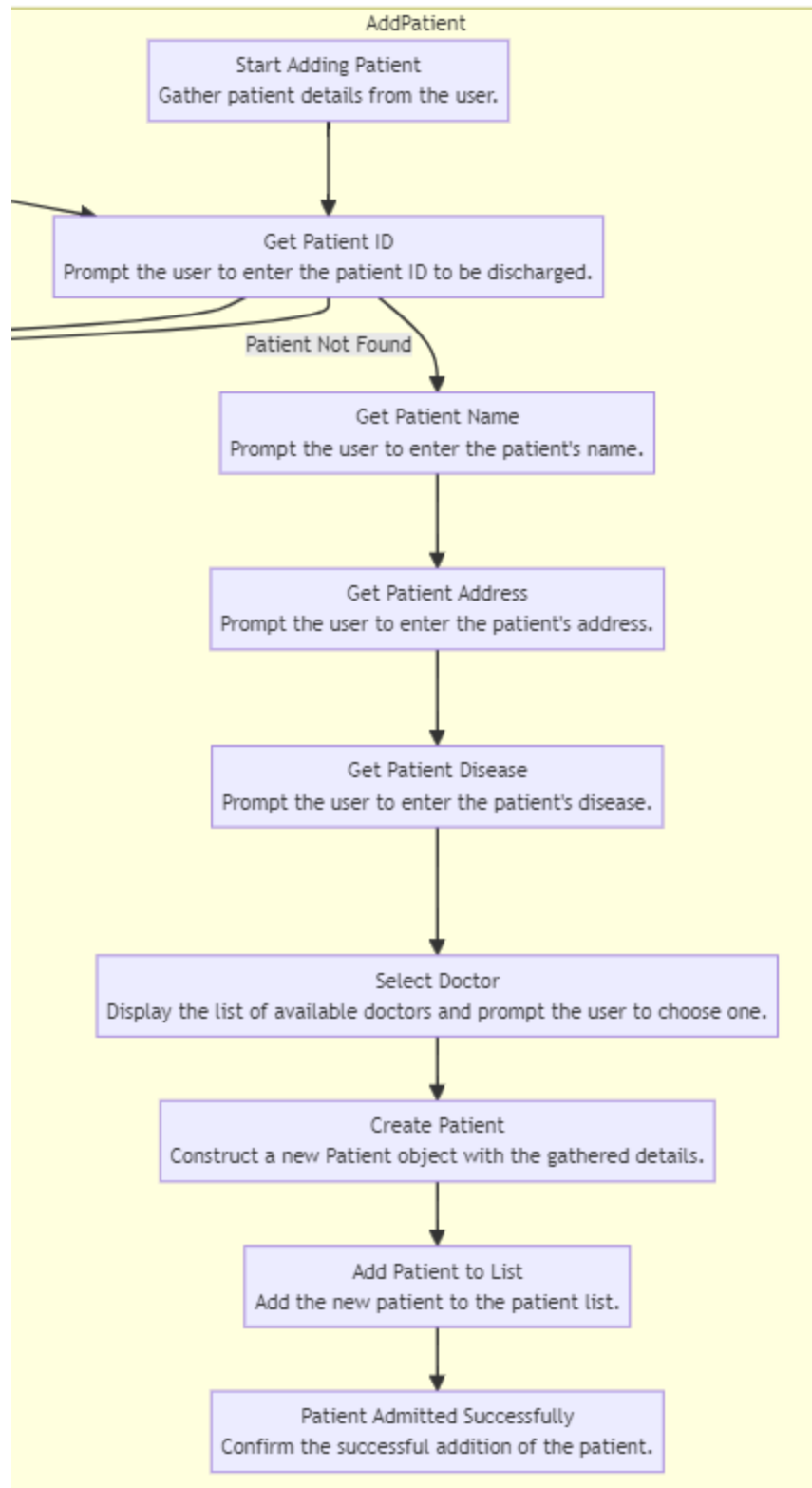
- The priority queue ensures that patients are managed based on their admission times, with earlier admissions having higher priority for discharge and other operations.

File Handling

- Patient data is persisted between sessions using file I/O operations, ensuring that patient records are not lost when the program is closed.

This implementation provides a robust and user-friendly system for managing hospital patient records, utilizing efficient data structures and file handling techniques.





FLOWCHART

Here's the flowchart representation in a table format:

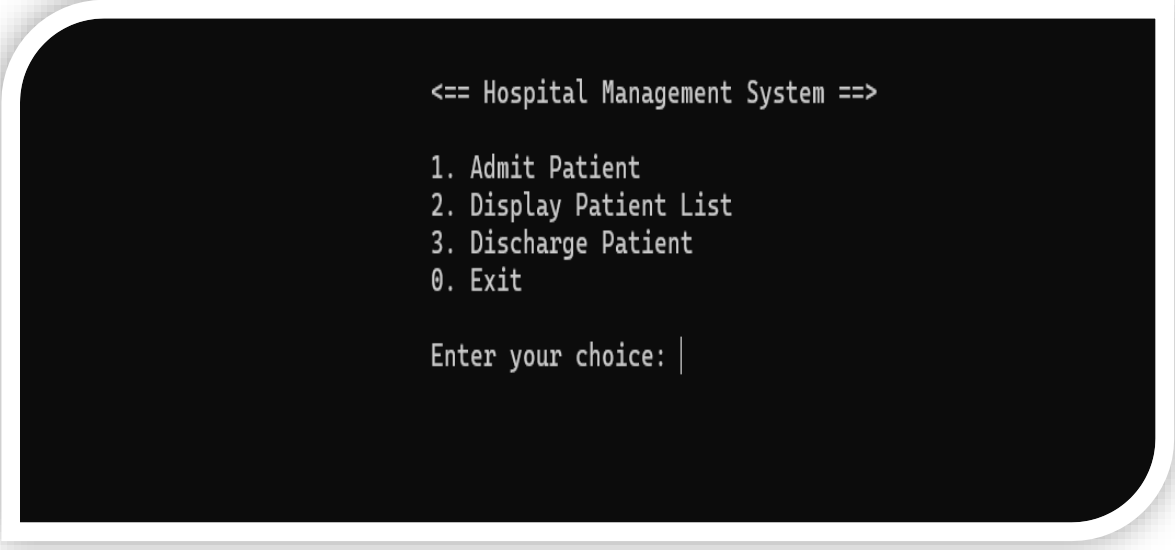
Step	Description	Details
1	Start Program	Initialize <code>PatientList</code> object, load patients from file, and initialize <code>doctorNames</code> vector.
2	Display Menu	Loop to display menu options: Admit Patient, Display Patient List, Discharge Patient, Exit.
3	Handle Choices	Depending on the user's choice, execute the corresponding actions.
3.1	Exit	Save patients to file and end program.
3.2	Admit Patient	Enter patient details, select doctor, add patient to the list, and display success message.
3.3	Display Patient List	Check if the list is empty, then print all patients' details.
3.4	Discharge Patient	Enter patient ID, find patient, confirm discharge with the doctor, and remove patient from the list.
4	PatientList Class	Class to manage patient list with priority queue and linked list.
4.1	Attributes	<code>priority_queue<Patient></code> <code>patientQueue</code> , <code>Node* head</code> , <code>Node* tail</code> .
4.2	Constructor	Initialize <code>head</code> and <code>tail</code> to <code>nullptr</code> .
4.3	Destructor	Delete all nodes in the linked list.

Step	Description	Details
4.4 Methods		addPatient, displayPatients, removePatient, loadPatientsFromFile, savePatientsToFile.
	4.4.1 addPatient	Create a new Node, add to linked list, push patient to priority queue.
	4.4.2 displayPatients	Check if list is empty, print all patients.
	4.4.3 removePatient	Check if list is empty, find patient by ID, confirm discharge with doctor, remove from list.
	4.4.4 loadPatientsFromFile	Open file, read patient details, compute admissionTimestamp, add patient to list.
	4.4.5 savePatientsToFile	Open file, write patient details to file.
Structure to store patient details.		
5 Patient Struct		
5.1 Attributes		int id, string patientName, string patientAddress, string disease, string date, string admissionTime, time_t admissionTimestamp, string doctorName.

Step	Description	Details
5.2	Constructor	Initialize date, admissionTime, admissionTimestamp.
5.3	Operator <	Custom comparator for priority queue.
6	Node Struct	Structure for linked list node.
6.1	Attributes	Patient data, Node* next.
6.2	Constructor	Initialize data and next.

SCREENSHOTS

HOMEPAGE:

A screenshot of a terminal window with a black background and white text. The text displays the title '<== Hospital Management System ==>', a numbered menu with four options: '1. Admit Patient', '2. Display Patient List', '3. Discharge Patient', and '0. Exit'. Below the menu is a prompt 'Enter your choice: |' with a vertical cursor. The terminal window has a white border and rounded corners, set against a light gray background.

```
<== Hospital Management System ==>
```

- 1. Admit Patient
- 2. Display Patient List
- 3. Discharge Patient
- 0. Exit

```
Enter your choice: |
```

ADMIT PATIENT:

```
<== Hospital Management System ==>
```

1. Admit Patient
2. Display Patient List
3. Discharge Patient
0. Exit

Enter your choice: 1

Enter Patient ID: 01

Enter Patient Name: Munyim

Enter Patient Address: AK KHAN

Enter Patient Disease: Heart Attack

Select a Doctor from the list below:

1. Dr. Debi Sheti(Cardiologist)
2. Dr. Hamid Hasan(Neurologist)
3. Dr. Anisul Awal(Medicine Specialist)
4. Dr. Abdul Karim(Internal Medicine)
5. Dr. Rasel Viper(Poison Specialist)

Enter your choice: 1

Patient Admitted Successfully.

```
<== Hospital Management System ==>
```

1. Admit Patient
2. Display Patient List
3. Discharge Patient
0. Exit

Enter your choice: |

FILE-HANDLING:

```
<== Hospital Management System ==>

1. Admit Patient
2. Display Patient List
3. Discharge Patient
0. Exit

Enter your choice: 2

<== Patient List ==>

ID      Name      Address      Disease      Date      Admission Time      Doctor Name
1      Munyim      AK KHAN      Heart Attack  25/06/2024      04:39 AM      Dr. Debi Sheti(Cardiologist)
2      Fahim      Noyabazar      Stroke      25/06/2024      04:41 AM      Dr. Hamid Hasan(Neurologist)
3      Moinul      Chawkbazar      Body pain      25/06/2024      04:42 AM      Dr. Anisul Awal(Medicine Specialist)

<== Hospital Management System ==>

1. Admit Patient
2. Display Patient List
3. Discharge Patient
0. Exit

Enter your choice: |
```

```
project.txt | project.txt | patients.txt | patients.
File Edit View

1
Munyim
AK KHAN
Heart Attack
25/06/2024
04:39 AM
Dr. Debi Sheti(Cardiologist)
2
Fahim
Noyabazar
Stroke
25/06/2024
04:41 AM
Dr. Hamid Hasan(Neurologist)
3
Moinul
Chawkbazar
Body pain
25/06/2024
04:42 AM
Dr. Anisul Awal(Medicine Specialist)
```

DISCHARGE PATIENT:

```
<== Patient List ==>
ID      Name      Address      Disease      Date      Admission Time  Doctor Name
1      Munyim     AK KHAN     Heart Attack  25/06/2024  04:39 AM       Dr. Debi Sheti(Cardiologist)
2      Fahim      Noyabazar   Stroke       25/06/2024  04:41 AM       Dr. Hamid Hasan(Neurologist)
3      Moinul      Chawkbazar   Body pain    25/06/2024  04:42 AM       Dr. Anisul Awal(Medicine Specialist)

<== Hospital Management System ==>

1. Admit Patient
2. Display Patient List
3. Discharge Patient
0. Exit

Enter your choice: 3

Enter Patient ID to discharge: 2

Doctor Dr. Hamid Hasan(Neurologist) needs to confirm discharge.
Enter 'yes' to confirm: yes

Patient discharged successfully by Dr. Dr. Hamid Hasan(Neurologist).

<== Hospital Management System ==>

1. Admit Patient
2. Display Patient List
3. Discharge Patient
0. Exit

Enter your choice: 2

<== Patient List ==>
ID      Name      Address      Disease      Date      Admission Time  Doctor Name
1      Munyim     AK KHAN     Heart Attack  25/06/2024  04:39 AM       Dr. Debi Sheti(Cardiologist)
3      Moinul      Chawkbazar   Body pain    25/06/2024  04:42 AM       Dr. Anisul Awal(Medicine Specialist)
```

File Edit View

```
1
Munyim
AK KHAN
Heart Attack
25/06/2024
04:39 AM
Dr. Debi Sheti(Cardiologist)
3
Moinul
Chawkbazar
Body pain
25/06/2024
04:42 AM
Dr. Anisul Awal(Medicine Specialist)
```

EXIT:

```
<== Hospital Management System ==>
```

- 1. Admit Patient
- 2. Display Patient List
- 3. Discharge Patient
- 0. Exit

```
Enter your choice: 0
```

```
Exiting... Thank you!
```

FUTURE PLAN/POSSIBLE EXTENTION

The Hospital Management System project can be extended in numerous ways to enhance its functionality, user experience, and robustness. Here are some possible future extensions and improvements:

1. Database Integration:

- **SQL/NoSQL Databases:** Integrate a database system (like MySQL, PostgreSQL, or MongoDB) to handle patient records instead of using text files. This will improve data integrity, security, and scalability.
- **Data Migration:** Provide tools or scripts to migrate data from text files to the database.

2. User Authentication and Authorization:

- **User Roles:** Implement a user management system with different roles (e.g., Admin, Doctor, Nurse, Receptionist) with specific access permissions.
- **Login System:** Add a login system to authenticate users before accessing the system.

3. Appointment Scheduling:

- **Doctor's Schedule:** Allow patients to book appointments with doctors, checking their availability.
- **Reminders and Notifications:** Send reminders to patients and doctors for upcoming appointments via email or SMS.

4. Enhanced User Interface:

- **GUI Development:** Develop a graphical user interface (GUI) using frameworks like Qt (for C++) or web-based interfaces using HTML/CSS/JavaScript.
- **Mobile Application:** Create a mobile app for patients and doctors to access the system on the go.

5. Advanced Patient Management:

- **Electronic Health Records (EHR):** Maintain comprehensive health records including medical history, test results, and prescriptions.
- **Health Monitoring:** Integrate with health monitoring devices to track and log patient vitals automatically.

6. Analytics and Reporting:

- **Reports Generation:** Generate detailed reports on various metrics like patient inflow, doctor performance, and treatment outcomes.
- **Data Analytics:** Use analytics to identify trends and improve decision-making.

7. Security Enhancements:

- **Encryption:** Encrypt sensitive patient data to ensure privacy and comply with regulations like HIPAA.
- **Audit Logs:** Maintain audit logs to track changes and access to patient records for security and accountability.

8. Integration with External Systems:

- **Insurance Systems:** Integrate with insurance companies to handle claims and billing.

- **Laboratory Systems:** Connect with laboratories to directly receive test results.

9. Automated Processes:

- **Billing System:** Develop an automated billing system to generate and manage bills for treatments and services.
- **Inventory Management:** Track and manage hospital inventory, including medications and medical supplies.

10. Patient Portal:

- **Self-service Portal:** Allow patients to view their records, book appointments, and communicate with healthcare providers through a dedicated portal.
- **Telemedicine:** Facilitate online consultations with doctors via video calls or chat.

11. Feedback and Improvement:

- **Patient Feedback:** Collect feedback from patients to improve services and patient care.
- **Continuous Improvement:** Regularly update the system based on user feedback and emerging healthcare trends.

Implementing these extensions can make the Hospital Management System more robust, user-friendly, and comprehensive, ultimately enhancing the quality of healthcare delivery.

CONCLUSION

This project demonstrates the effective use of data structures, file handling, and user interaction to solve real-world problems in hospital management. By following the outlined extensions, this system can evolve into a robust, secure, and user-friendly platform, significantly improving healthcare administration and patient care. The project not only meets the immediate requirements of managing patient information but also lays the groundwork for future innovations and enhancements in the healthcare domain.

THANK YOU!
