Vivekanand Education Society's Institute Of Technology
Department Of Information Technology
DSA mini Project A.Y. 2025-26

Title:
# Hospital Management System Using Queue Data Structure

Sustainability Goal : To reduce paper usage and promote eco-friendly digital record maintenance in hospitals


Domain: Information Technology

Member: Dhruv Haswani
Roll no : 20
Div : D10B(Batch A)


Mentor Name:


1 NO POVERTY
2 ZERO HUNGER
3 GOOD HEALTH AND WELL-BEING
4 QUALITY EDUCATION
5 GENDER EQUALITY
6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
THE GLOBAL GOALS
For Sustainable Development
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE AND JUSTICE STRONG INSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS

# Content

1. Introduction to the Project
2. Problem Statement
3. Objectives of the Project
4. Scope of the Project
5. Requirements of the System (Hardware, Software)
6. ER Diagram of the Proposed System
7. Data Structure & Concepts Used
8. Algorithm Explanation
9. Time and Space Complexity
10. Front End
11. Implementation
12. Gantt Chart
13. Test Cases
14. Challenges and Solutions
15. Future Scope
16. Code
17. Output Screenshots
18. Conclusion
19. References (in IEEE Format)


# Introduction to Project

**A Hospital Management System is designed to handle patients systematically.**

**In real hospitals, patients register themselves and wait for their turn to be served.**

**We can implement this process using Queue Data Structure.**

**Queue follows FIFO (First In, First Out) principle → the patient who registers first is served first.**

# Problem Statement

**To build a simple hospital management system where:**

**1. Patients can register with an ID and name.**

**2. Patients are served one by one in proper order.**

**3. The list of waiting patients can be displayed at any time.**

**The system should also handle cases like queue full and queue empty**

# Objectives of the project

**To simulate a real-world hospital scenario using data structures.**

**To implement queue operations for patient registration and service.**

**To provide a simple, user-friendly system for managing patients.**

**To understand the application of circular queues in practical problems.**

# Scope Of The Project

**Register and manage patient details.**

**Handle appointments between patients and doctors.**

**Generate and maintain billing records.**

**Manage patient queue using First Come First Serve (FCFS).**

**Provide a simple, user-friendly interface.**

**Extendable to include databases, pharmacy, and analytics.**

# Requirements of the system (Hardware, software)

**Hardware Requirements:**

**Processor: Intel i3 or above**

**RAM: Minimum 2 GB (4 GB recommended)**

**Hard Disk: 200 MB free space**

**Monitor: Standard display (any resolution)**

**Keyboard: For input of patient details**

# Requirements of the system (Hardware, software)

**Software Requirements:**

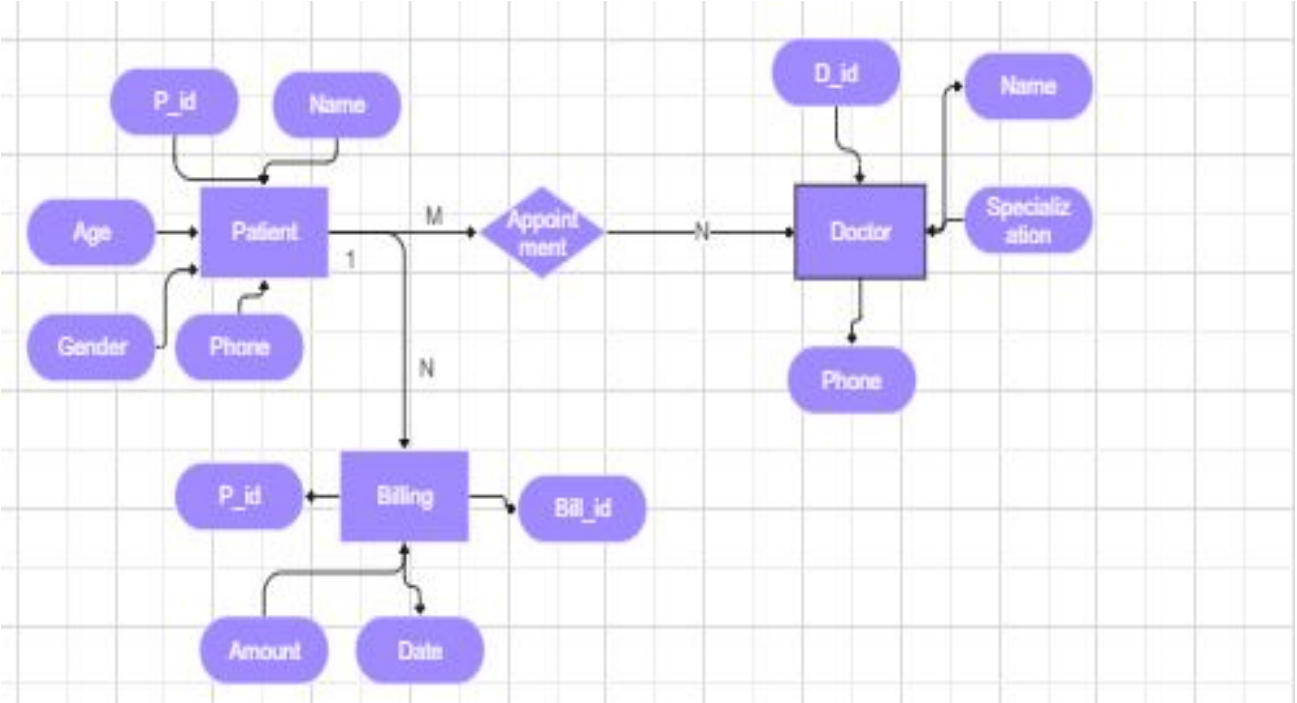
**Operating System: Windows / Linux**

**Compiler: GCC (MinGW / Turbo C / Code::Blocks / Dev C++)**

**Programming Language: C**

**Editor/IDE: Any text editor or IDE supporting C (e.g., Code::Blocks, VS Code, Dev-C++)**

# ER diagram of the proposed system



# Data Structures And Concepts Used

**Circular Queue :**

**Implemented using an array of fixed size.**
**Ensures efficient use of memory.**
**Handles patients in FIFO (First In, First Out) order.**

**Structure (struct) :**

**Used to store patient details: ID and Name.**
**Makes data handling organized and modular.**

**Front & Rear Pointers :**

**front → Tracks the first patient to be served.**
**rear → Tracks the latest patient registered.**

# Data Structures And Concepts Used

**Core Operations :**

**1. Enqueue (Register Patient) – Insert new patient.**

**2. Dequeue (Serve Patient) – Remove and serve patient.**

**3. Display – Show patients currently waiting.**

**Concepts Applied :**

**FIFO Principle (first come, first served).**

**Modular arithmetic for circular queue indexing.**

**Menu-driven programming for user interaction**

# Algorithm Explanation

**1. Register Patient (Enqueue) :**

**Step 1: Check if queue is full → if yes, display “Queue Full”.**
**Step 2: If queue is empty, set front = rear = 0.**
**Step 3: Else, move rear = (rear + 1) % SIZE.**
**Step 4: Insert patient ID & name at queue[rear].**
**Step 5: Display success message.**

**Explanation: Adds a new patient to the waiting list.**

**2. Serve Patient (Dequeue)**

**Step 1: Check if queue is empty → if yes, display “No patients to serve”.**
**Step 2: Print details of patient at front.**
**Step 3: If front == rear, reset queue (front = rear = -1).**
**Step 4: Else, move front = (front + 1) % SIZE.**

**Explanation: Removes and serves the patient who came first.**

# Algorithm Explanation

**3. Display Queue**

**Step 1: Check if queue is empty → if yes, display “No patients in queue”.**
**Step 2: Start from front and print each patient till rear.**
**Step 3: Use circular indexing (i + 1) % SIZE.**

**Explanation: Shows all patients currently waiting for their turn.**

**4. Exit**

**Step 1: Terminate the program when user selects Exit option.**

**Explanation: Ends the hospital management system safely.**

# Time And Space Complexity

**Time Complexity :**

**Enqueue (Register Patient): O(1) → Direct insertion at rear.**
**Dequeue (Serve Patient): O(1) → Direct removal from front.**
**Display Queue: O(n) → Traverses all n patients in the queue.**

**Overall: Each operation is efficient, with constant or linear time.**

**Space Complexity :**

**Queue array of fixed size → O(n), where n = SIZE of queue.**
**Extra variables (front, rear, id, name[]) → O(1).**

**Total Space Complexity: O(n).**

**Conclusion:**
**The system is time-efficient (most operations in O(1)).**

# Front End

**The project uses a Menu-Driven Console Interface as the front end.**

**Users interact with the system through simple numbered options.**

**The main menu includes:**

**1. Register Patient → Add new patient (ID & Name).**
**2. Serve Patient → Serve the next patient in queue.**
**3. Display Queue → Show all waiting patients.**
**4. Exit → Close the system.**

**Inputs are taken using keyboard and outputs are displayed on the console screen.**

**This makes the system easy to use and understand for beginners.**

# Implementation

**The system is implemented in C language using a Circular Queue.**

**Steps followed in implementation:**

**1. Define a Patient structure (ID, Name).**

**2. Create a queue array of fixed size.**

**3. Maintain front and rear pointers for queue operations.**

**4. Implement functions:**

**registerPatient() → Adds patient to queue.**
**servePatient() → Removes and displays served patient.**
**displayQueue() → Prints all waiting patients.**

# Implementation

**5. Provide a menu-driven interface in main() for user interaction.**

**Error Handling:**

**Displays message if queue is full while registering.**

**Displays message if queue is empty while serving.**

**The system keeps running until the user chooses Exit**

# Gantt Chart

| Activity | Week 1 | Week 2 | Week 3 | Week 4 |
|---|---|---|---|---|
| Problem Identification | 🟩🟩🟩 | | | |
| Requirement Analysis | 🟦🟦🟦 | | | |
| System Design (Flowchart) | | 🟩🟩🟩 | | |
| Coding & Implementation | | | 🟦🟦🟦 | 🟦🟦🟦 |
| Testing & Debugging | | | 🟩🟩🟩 | |
| Documentation & Report | | | | 🟦🟦🟦 |
| Final Presentation | | | | 🟩🟩🟩 |

🟩 = Task 1
🟦 = Task 2

# Test Cases

**Case 1: Register Patient**

**Input → Choice = 1, ID = 101, Name = Raj**

**Output → “Patient Raj (ID: 101) registered successfully.”**

**Case 2: Register Multiple Patients**

**Input → Choice = 1, ID = 102, Name = Meera; ID = 103, Name = Aman**

**Output → All patients added in correct order.**

**Case 3: Serve Patient**

**Input → Choice = 2**

**Output → Displays and removes the first patient in queue.**

# Test Cases

**Case 4: Display Queue**

**Input → Choice = 3**

**Output → Shows all waiting patients in order.**

**Case 5: Queue Empty Condition**

**Input → Choice = 2 (when queue is empty)**

**Output → “No patients to serve.”**

**Case 6: Queue Full Condition**

**Input → Register more than 5 patients (since SIZE = 5)**

**Output → “Queue Full! No more patients can be registered.”**

# Test Cases

**Case 7: Exit Program**

**Input → Choice = 4**

**Output → “Exiting… Goodbye, Take Care!” and program ends.**

# Challenges And Solutions

1. Queue size limited → Use dynamic arrays or linked lists.

2. Name input restricted → Use fgets() for full names.

3. No search/update → Add search & modify functions.

4. No data saving → Use file handling for persistence.

5. Weak error handling → Validate inputs and handle errors gracefully.

6.Too simple for real use → Extend with doctors, billing, appointments, etc.

# Challenges And Solutions

**7.Connect to databases for permanent patient records.**

**8.Add doctor & staff management (availability, duty shifts).**

**9.Introduce billing & pharmacy inventory.**

**10.Enable appointment scheduling and follow-up alerts.**

**11.Improve user interface with GUI for ease of use.**

# Future Scope

**Online access: Patient portal, telemedicine, notifications.**

**Reports & analytics: Daily patient count, disease statistics.**

**Security: Login, data encryption.**

**Web/Mobile integration for remote access.**

**AI & ML for treatment prediction & hospital resource optimization.**

# Output ScreenShots

```
PS C:\c.Dhruv.20> cd 'c:\c.Dhruv.20\.vscode\output'
PS C:\c.Dhruv.20\.vscode\output> & .\'HMS.exe'

--- Hospital Management System ---
1. Register Patient
2. Serve Patient
3. Display Queue
4. Exit
Enter choice: 1
Enter Patient ID: 10
Enter Patient Name: Ramesh
Patient Ramesh (ID: 10) registered.

--- Hospital Management System ---
1. Register Patient
2. Serve Patient
3. Display Queue
4. Exit
Enter choice: 1
Enter Patient ID: Priya
Enter Patient Name: Patient Priya (ID: 10) registered.

--- Hospital Management System ---
1. Register Patient
2. Serve Patient
3. Display Queue
4. Exit
Enter choice: 3
ID: 10 | Name: Ramesh
ID: 10 | Name: Priya

--- Hospital Management System ---
```

```
--- Hospital Management System ---
1. Register Patient
2. Serve Patient
3. Display Queue
4. Exit
Enter choice: 3
ID: 10 | Name: Ramesh
ID: 10 | Name: Priya

--- Hospital Management System ---
1. Register Patient
2. Serve Patient
3. Display Queue
4. Exit
Enter choice: 2
Serving Patient: Ramesh (ID: 10)

--- Hospital Management System ---
1. Register Patient
2. Serve Patient
3. Display Queue
4. Exit
Enter choice: 4
PS C:\c.Dhruv.20\.vscode\output>
```

# Conclusion

**The project successfully demonstrates a Hospital Management System using the Queue data structure.**

**Patients are registered and served in a FIFO (First In, First Out) order.**

**Implementation with Circular Queue ensures efficient memory usage.**

**The system provides a simple, user-friendly, and structured way of managing patients.**

**This project highlights how Data Structures & Algorithms can solve real-life problems effectively.**


# References

**GeeksforGeeks – https://www.geeksforgeeks.org/queue-data-structure**

**Tutorialspoint – https://www.tutorialspoint.com/data_structures_algorithms/dsa_queue.htm**

**Lecture notes & practicals from college DSA course**