

## **Team Members and Roles**

Mohammad Mohid – Team Lead & System Integration

Shehroz Khan

Momina Aman

Muskan Fatima

Anas Khan –

Dilshand Ali

Toqueer Khan

# Hospital Emergency Management System

## Contents

<b>Abstract</b>	4
<b>1 Introduction</b>	5
1.1 Problem Statement	5
1.2 Objectives	5
<b>2 System Design</b>	5
2.1 Overall Architecture	5
2.2 System Architecture Diagram	6
2.3 Data Structure Choices and Justification	6
<b>3 Code Architecture</b>	7
3.1 File Structure	7
3.2 Module Interaction	7
<b>4 Implementation Details</b>	8
4.1 Patient Data Structure	8
4.2 Patient Registration Logic	8
4.3 Search Implementation	8
4.4 CSV File Handling	8
<b>5 Data Flow Diagram</b>	8
<b>6 Testing and Validation</b>	9
<b>7 Challenges Faced and Solutions</b>	15
<b>8 AI Tools Used</b>	16
<b>9 Conclusion</b>	17
<b>10 References</b>	18



## Abstract

Hospital emergency departments operate in time-critical environments where efficient patient prioritization and accurate record management are essential. Manual systems are often slow, error-prone, and unable to handle high patient volumes effectively.

This project presents a **Hospital Emergency Management System** developed in **C++** with a graphical user interface using **Dear ImGui**. The system enables hospital staff to register patients, assign medical priority levels, manage emergency queues, search patient records, and store data persistently using CSV files.

A backend module handles patient data storage, priority-based queue management, and searching functionality, while the GUI provides an intuitive interface for interaction. The system demonstrates the practical application of data structures, file handling, and GUI programming to solve real-world healthcare problems.

# 1 Introduction

Emergency departments require rapid decision-making and organized data handling to ensure that critical patients receive immediate care. Traditional paper-based or semi-digital systems often fail to provide efficient prioritization and fast data retrieval.

## 1.1 Problem Statement

Manual emergency room systems do not guarantee optimal patient prioritization, fast searching, or reliable data persistence, leading to increased waiting times and operational inefficiencies.

## 1.2 Objectives

- Develop an automated emergency patient management system
- Prioritize patients based on medical urgency
- Provide fast patient search functionality
- Store patient records persistently using files
- Design a user-friendly graphical interface

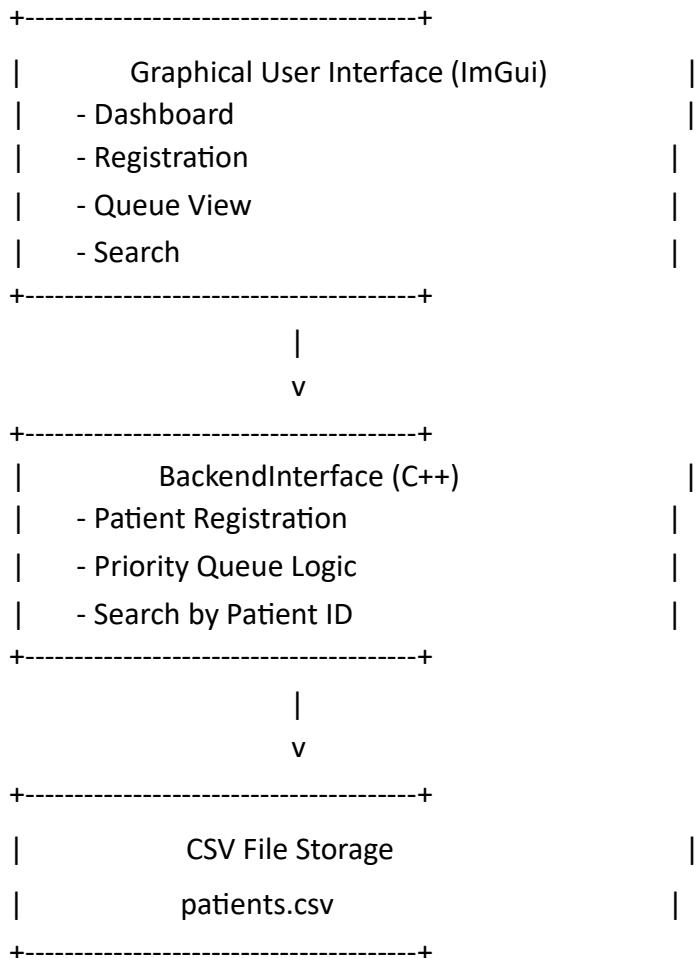
# 2 System Design

## 2.1 Overall Architecture

The system follows a modular layered architecture to ensure separation of concerns and maintainability.

- **Presentation Layer:** Dear ImGui GUI
- **Application Layer:** BackendInterface
- **Data Storage Layer:** CSV File Storage

## 2.2 System Architecture Diagram



## 2.3 Data Structure Choices and Justification

- **Vector-based Queue**: Used to maintain patient order and priority grouping.
- **Structs**: Used to represent patient records efficiently.
- **Linear Search**: Used for patient lookup due to moderate data size.

## 3 Code Architecture

### 3.1 File Structure

PatientBackend.h / .cpp      -> Backend logic and CSV handling  
GUIManager  
(main.cpp)      -> GUI rendering and interaction  
patients.csv      -> Persistent data storage

### 3.2 Module Interaction

- GUI collects input and sends it to BackendInterface
- BackendInterface stores patient data and saves to CSV
- GUI retrieves queue and search results from backend

## 4 Implementation Details

### 4.1 Patient Data Structure

```
struct Patient { int id; string name;  
    int age; string symptoms; int  
    priority; string priorityLabel;  
    time_t registrationTime;  
};
```

### 4.2 Patient Registration Logic

```
newPatient.id = backend.nextPatientID++; backend.addPatient(newPatient);
```

### 4.3 Search Implementation

```
Patient* BackendInterface::searchPatient(int id) { for (auto& p :  
    patientQueue) { if (p.id == id) return &p;  
    } return nullptr;  
}
```

### 4.4 CSV File Handling

```
outfile << p.id << "," << p.name << "," << p.age << ","  
    << "\"" << p.symptoms << "\"" << ","  
        << p.priority << "," << p.priorityLabel << "\n";
```

## 5 Data Flow Diagram

User Input (GUI) | v

Patient Registration



+--> BackendInterface



| +--> Save to CSV



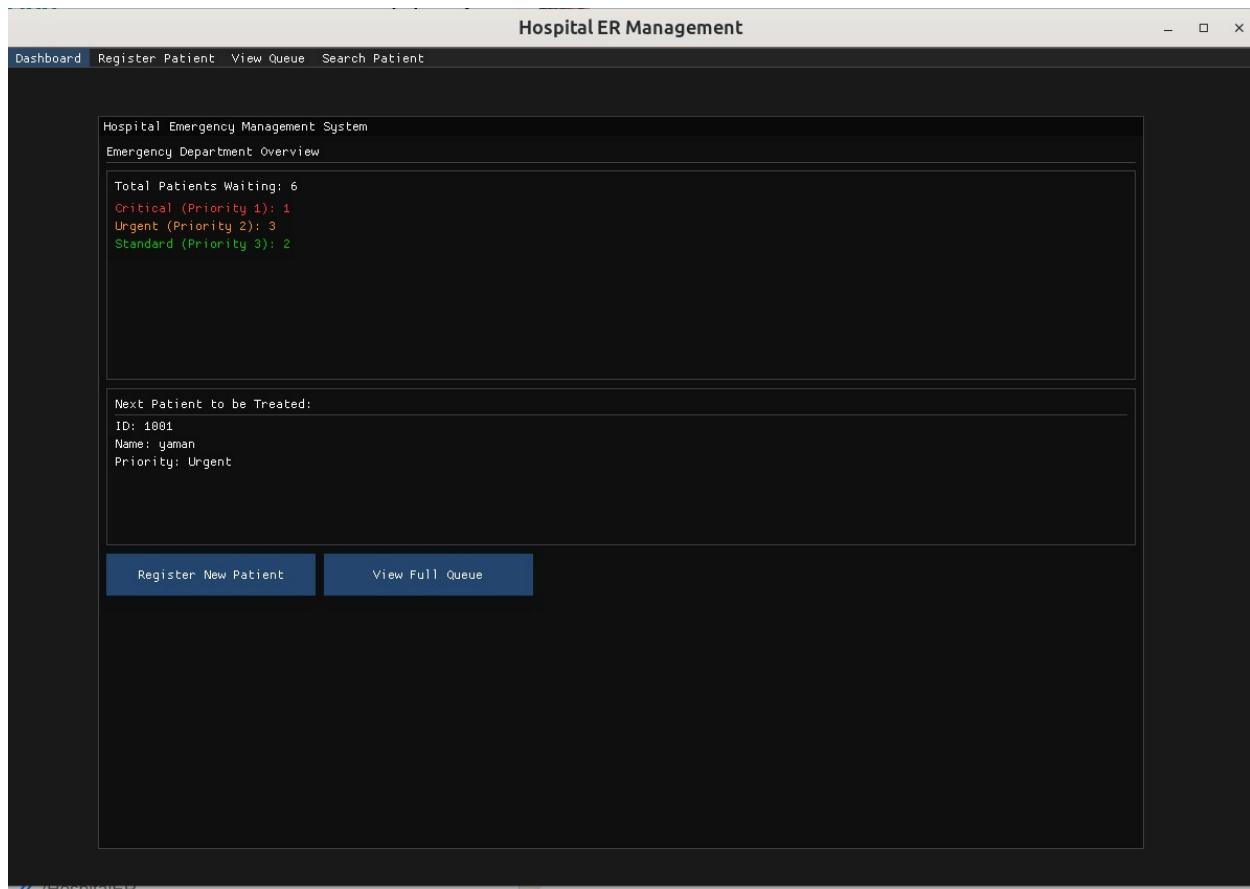
+--> Queue Display / Search

## 6 Testing and Validation

Test No.	Test Description	Result
1	Register patient with valid data	Pass
2	Register patient with empty name	Pass
3	Invalid age input	Pass
4	Priority selection validation	Pass
5	View patient queue	Pass
6	Search existing patient ID	Pass
7	Search non-existing patient	Pass
8	CSV file saving	Pass
9	Reload data from CSV	Pass
10	GUI navigation between screens	Pass

**Screenshots:**

## Dashboard Screen



## Patient insertion:

The screenshot shows a Windows application window titled "Hospital ER Management". The main title bar has a blue header with the text "Hospital Emergency Management System" and "Patient Registration". Below the header, there is a navigation bar with links: "Dashboard", "Register Patient" (which is highlighted in blue), "View Queue", and "Search Patient".

The main content area contains a form for "Patient Registration". The fields are as follows:

- Patient Name: USAMA AFRIDI
- Age: 23
- Symptoms/Reason for Visit: FEVER
- Priority Level:
  - Critical (Heart attack, Severe trauma)
  - Urgent (High fever, Fractures)
  - Standard (Minor injuries, Routine checkup)

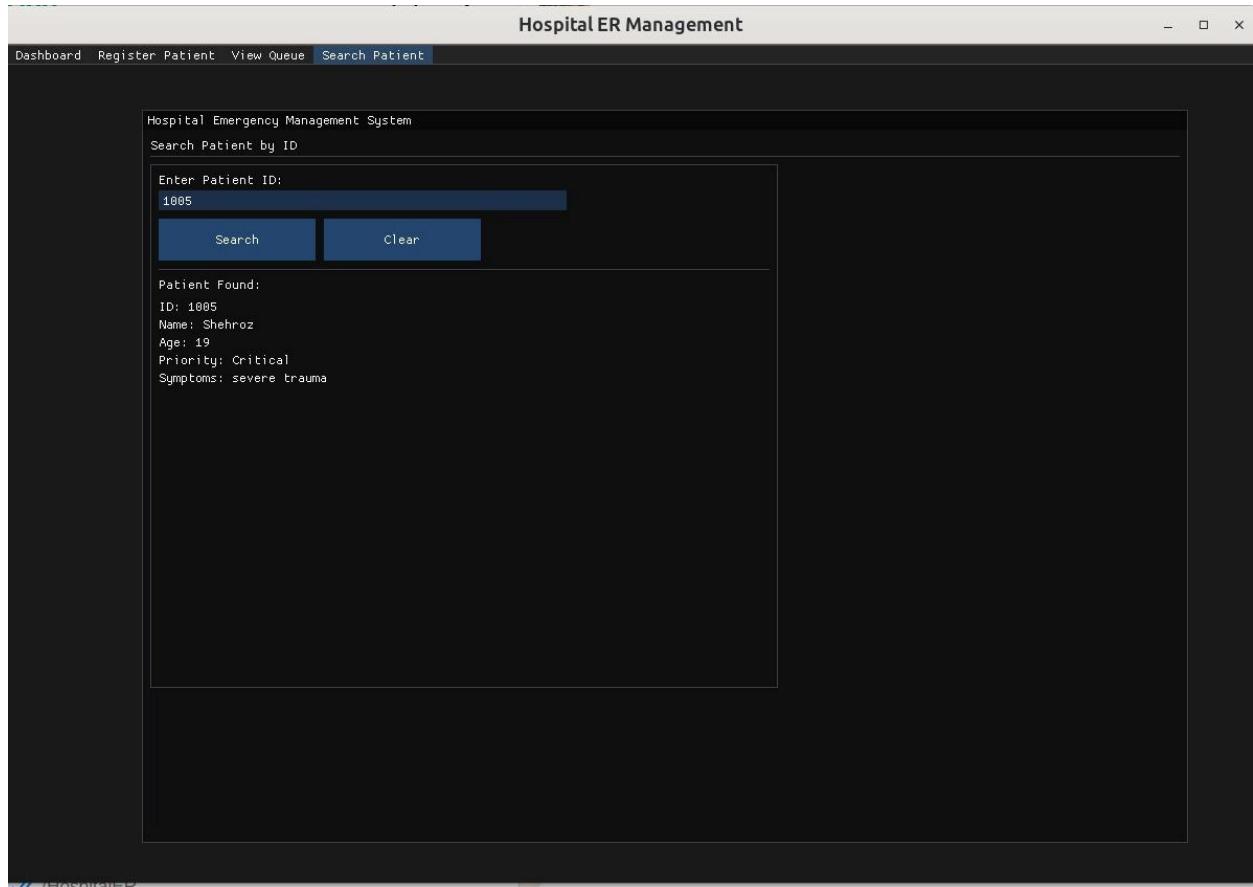
At the bottom of the form are two buttons: "Submit Registration" and "Clear Form". A green message at the bottom of the form area says "Patient registered successfully! ID: 1007".

## Patient Queue:

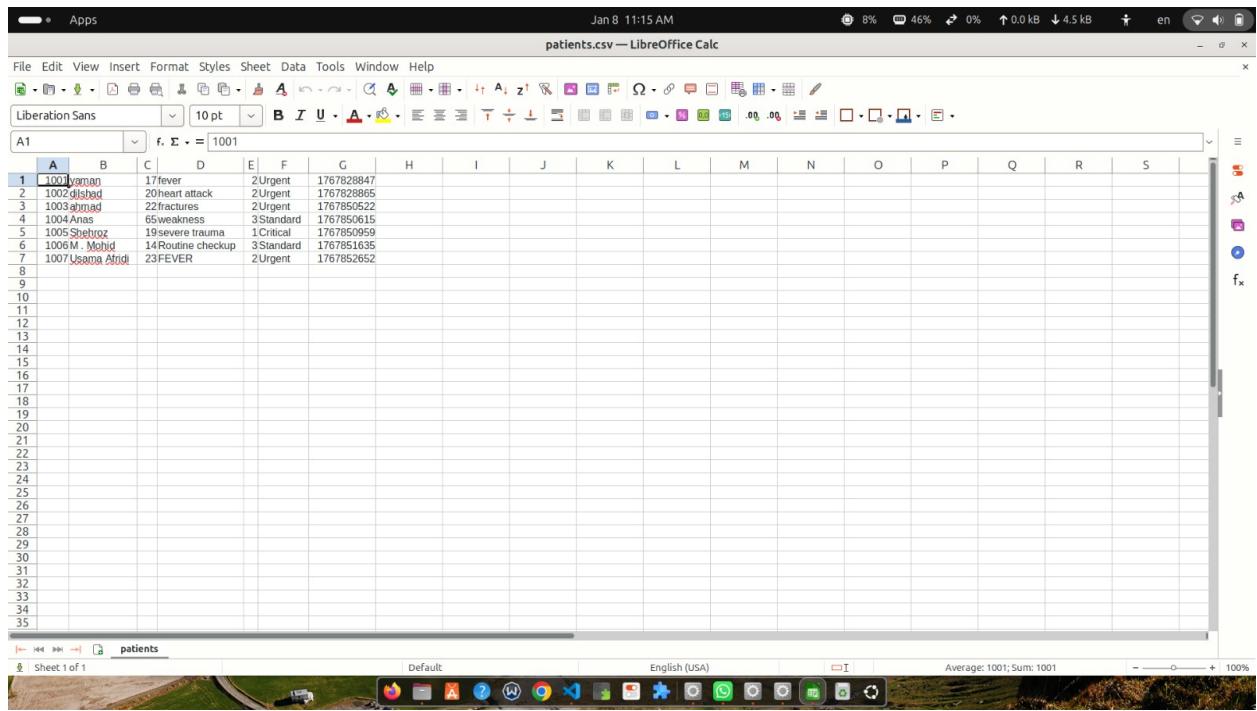
The screenshot shows a window titled "Hospital ER Management". The main content is a table titled "Patient Queue (Ordered by Priority)". The table has columns: ID, Name, Age, Priority, and Symptoms. The data is as follows:

ID	Name	Age	Priority	Symptoms
1001	yaman	17	Urgent	fever
1002	dilishad	20	Urgent	heart attack
1003	ahmad	22	Urgent	fractures
1004	Anas	65	Standard	weakness
1005	Shehroz	19	Critical	severe trauma
1006	M . Mohid	14	Standard	Routine checkup
1007	Usama Afridi	23	Urgent	FEVER

## Patient search:



### Csv file:



The screenshot shows a LibreOffice Calc spreadsheet window titled "patients.csv — LibreOffice Calc". The window includes a toolbar, menu bar, and various icons for file operations. The main area displays a table with 7 rows of data. Row 1 contains column headers: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S. Rows 2 through 7 contain patient information, each starting with a unique ID (e.g., 1001, 1002, 1003, 1004, 1005, 1006, 1007) followed by the patient's name, symptoms, and a numerical value.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	1001	Yaman	17fever	2Urgent	1767828847													
2	1002	dilshad	20heart attack	2Urgent	1767828865													
3	1003	ahmad	22fractures	2Urgent	1767850522													
4	1004	Anas	65weakness	3Standard	1767850615													
5	1005	Shehzad	19severe trauma	1Critical	1767850959													
6	1006	M. Mohid	14Routine checkup	3Standard	1767851635													
7	1007	Usama Afridi	23FEVER	2Urgent	1767852652													

## **7 Challenges Faced and Solutions**

- GUI state management solved using screen enumeration
- CSV parsing issues fixed using string streams
- Input validation implemented to prevent crashes
- Data persistence ensured using file overwrite mode

## **8 AI Tools Used**

- ChatGPT – Code debugging, architecture design, documentation
- Dear ImGui Documentation – GUI components
- C++ Reference – File handling and STL usage

## **9 Conclusion**

The Hospital Emergency Management System successfully automates emergency room operations using a graphical interface and backend logic. The system improves patient prioritization, data organization, and operational efficiency. It demonstrates the practical application of programming concepts and serves as a foundation for future healthcare systems.

## 10 References

- <https://en.cppreference.com>
- <https://github.com/ocornut/imgui>
- <https://www.geeksforgeeks.org>
- ChatGPT – OpenAI