PROJECT REPORT

Title: Hospital Management System

Course: Programming in C (B.Tech 1 sem)

Submitted To: Mohsin F. Dar

Submitted By: Rishabh Tripathi

SAP ID: 590026253

Date: December 2025

TABLE OF CONTENTS

ABSTRACT

The Hospital Management System (HMS) is a C-based software solution that aims to simplify, automate, and digitalize the workflow of small to medium-scale hospitals. The system manages patient records, personal details, medical conditions, and assigned doctors, ensuring that data is stored securely using binary file handling mechanisms.

This project demonstrates structured programming, dynamic memory allocation, file handling, and modular approach in C. HMS eliminates human errors that occur in manual record keeping and provides a smooth, reliable, and efficient system for hospital data management.

INTRODUCTION

Healthcare institutions handle critical information daily, including patient demographics, medical history, diagnosis, and treatment reports. Traditionally, these records are stored in manual files, which are prone to damage, loss, and misplacement. A computerized hospital management system provides a structured and automated solution.

The purpose of this project is to design and implement a simple yet effective Hospital Management System using the C programming language, which allows hospital staff to manage records digitally. Features include adding new patients, searching for patients by ID, deleting records, and viewing all stored patient data.

PROBLEM DEFINITION

Managing hospital records manually often leads to issues such as redundant data, lost files, slow retrieval, and inaccuracy. Hospitals require a system that ensures reliable data handling, quick retrieval, and minimal errors.

Motivation

The need for a computerized patient record management system is increasing due to growing healthcare demands. A digital system enhances data accessibility, reduces paperwork, improves workflow efficiency, and provides long-term data storage.

SYSTEM DESIGN

The Hospital Management System is built using modular programming. The core components include:
- **Data Layer:** Stores patient information inside a binary file (patients.txt).
- **Logic Layer:** Functions to add, search, delete, and display patient data.
- **User Interface Layer:** Menu-driven command-line interface.

Dynamic memory allocation using malloc() and realloc() ensures flexibility in storing patient data without fixed-size limitations.

MODULE DESCRIPTION

1. Add Patient Module:

Allows the user to enter patient details including ID, name, age, gender, disease, and doctor. Memory is dynamically allocated for each new patient.

2. Search Patient Module:

Enables searching for a patient using their unique ID. If found, details are displayed.

3. Display Patients Module:

Shows all stored patient records, each containing complete patient information.

4. Delete Patient Module:

Deletes a patient record by shifting remaining elements and updating total count.

5. File Handling Module:

Handles permanent storage using binary files. Data is loaded during start and saved on exit.

FLOWCHART

START → Load File → Display Menu → Perform Operation → Update Records → Save File → EXIT

ALGORITHM

Add Patient Algorithm

1. Start

2. Allocate memory for new patient

3. Input patient details

4. Store details in structure array

5. Increase count

6. Save to binary file

7. Stop

Search Patient Algorithm

1. Start

2. Input patient ID

3. Search array

4. If found, display details

5. Else show error

6. Stop

Delete Patient Algorithm

1. Start

2. Input ID

3. Find matching record

4. Shift remaining records

5. Reduce count

6. Save changes

7. Stop

IMPLEMENTATION DETAILS

The system uses a structure to store patient details:

```
struct Patient {
  int id;
  char name[50];
  int age;
  char gender[10];
  char disease[50];
  char doctor[50];
};
```

Major Functions Implemented:

• addPatient() – Adds a new patient.

• displayPatients() – Displays all patients.

• searchPatient() – Searches patient by ID.

• deletePatient() – Deletes a patient.

• saveToFile() – Saves data in binary file.

• loadFromFile() – Loads data from file.

TESTING & RESULTS

✓ Tested adding multiple patients.

✓ Verified search functionality.

✓ Tested delete option for different positions.

✓ Validated file saving and loading.

✓ No memory leaks observed during execution.

LIMITATIONS

• No graphical interface.

• Does not include billing or appointments.

• No multi-user authentication.

CONCLUSION & FUTURE WORK

The Hospital Management System successfully manages patient records and demonstrates the use of dynamic memory allocation and file handling in C. Future updates may include GUI, database connectivity, staff management, appointment system, and auto-generated reports.

REFERENCES

- UPES Course Material – Programming in C

- Online Research Papers on Patient Management Systems