**HOSPITAL MANAGEMENT SYSTEM**

**Project report submitted in partial fulfillment of the Requirements for the Award of the Degree of**

**BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE AND ENGINEERING**

**By**

**Md.Sana Fathima Sulthana -24KB1A05AM**

**M.Muni Saranya-24KB1A05AF**

**M.Nandini-24KB1A05AP**

**N.Jashwitha-24KB1A05BU**

**Under the Guidance of**

**B.Sruthi**

****

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**N.B.K.R.INSTITUTE OF SCIENCE AND TECHNOLOGY**

**N.B.K.R.INSTITUTE OF SCIENCE AND TECHNOLOGY(AUTONOMOUS)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



###### CERTIFICATE

This is to certify that the project report entitled **HOSPITAL MANAGEMENT SYSTEM** being submitted by

MD.SANA FATHIMA SULTHANA 24KB1A05AM

M.MUNI SARANYA 24KB1A05AF

M.NANDINI 24KB1A05AP

N.JASHWITHA 24KB1A05BU

in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering to the N.B.K.R.INSTITUTE OF SCIENCE AND TECHNOLOGY is a record of bonafied work carried out under my guidance and supervision.

|  |  |
| --- | --- |
| **SMT.SRUTHI**  **Assistant Professor**  **Designation** | **Dr A Raja Sekhar Reddy**  **M.Tech, Ph.D**  **Head of the Department** |

**DECLARATION**

I hereby declare that the dissertation entitled **<** **HOSPITAL MANAGEMENT SYSTEM>** submitted for the B.Tech Degree is my original work and the dissertation has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles.

PLACE:NELLORE

DATE:05-05-2025

Md.Sana Fathima Sulthana-24KB1A05AM

M.Muni Saranya-24KB1A05AF

M.Nandini-24KB1A05AP

N.Jaswitha-24KB1A05BU

ACKNOWLEDGEMENT

I sincerely thank my faculty guide, Ms. B. Sruthi, for her guidance and support throughout my project, "Hospital Management System". I also thank Dr. A. Raja Sekhar Reddy, Head of Department, for providing the necessary resources and encouragement.

Gratitude to my friends and teammates for their cooperation and motivation, and to my family for their constant support.

This project enhanced my knowledge of C Programming, especially in Linked Lists, Structures, and Dynamic Memory Management, while giving me valuable insight into real-world application development.

Thank you all once again!

### 

### ABSTRACT OF THE PROJECT

### PROJECT TITLE:-HOSPITAL MANAGEMENT SYSTEM

The Hospital Management System is a C-based console application designed to assist healthcare institutions in efficiently managing patient records and appointment scheduling.The system utilizes fundamental data structures—Arrays to store patient information and Linked Lists to manage appointments-demonstrating the practical use of core programming concepts.

The primary objective of this project is to reduce the complexities associated with Manual hospital record keeping.It allows users to add,search,display patient details as well as schedule,view and delete appointments.The program is structured with a user-friendly menu interface,enabling smooth navigation and operation.

The project serves as an educational tool for understanding ho structured programming and dynamic memory management can be applied in real life world problem solving scenarios.

### TABLE OF CONTENTS

 **Chapter 1: Introduction** --------------------------------------------------------------------Page 1

 **Chapter 2: Literature Survey / Existing System** -------------------------------------- Page 2

 **Chapter 3: Software Requirement Analysis** -------------------------------------------- Page 3

 **Chapter 4: Software Design** -----------------------------------------------------------------Page 4

 **Chapter 5: Proposed System** --------------------------------------------------------------- Page 5

 **Chapter 6: Coding** ---------------------------------------------------------------------------- Page 6

 **Chapter 7: Output Screens / Results** ------------------------------------------------------ Page 7

 **Chapter 8: Conclusion and Further Work** ----------------------------------------------- Page 8

### CHAPTER 1: INTRODUCTION

### The Hospital Management System is a basic software solution aimed at streamlining patient record management and appointment scheduling.Built using C programming language,it is useful for small scale medical facilities.

### 1.1 Objective of the Project

### To design and implement a basic hospital management system using C programming language.To store and manage patient data using arrays.

### Appointments using linked lists.To stimulate real-world hospital operations and demonstrate practical use of data structures.

### 1.2 Scope of the Project

### The program is limited to command-line interface operations but serves as a foundation for future enhancements.It emphasizes the implementation of core programming logic without relying on external libraries or databases.

### 1.2.1 Features

### Add,Search and View Patients

### Add Appointment,View Appointment and delete Appointment

### User-friendly interface

### 1.2.2 Limitations

### The system uses a text based interface,which may be intuitive for all users.

### Records are lost after the program is closed(No database).

### 1.3 Methodology

The development of the Hospital Management System followed a structured step-by-step approach,combining Problem analysis,System design,Implementation ,Testing and debugging,Evaluation and Iteration.

### CHAPTER 2: LITERATURE SURVEY / EXISTING SYSTEM

### 2.1 Overview of Hospital Management System

### The Hospital Management System is a C-based application that streamlines basic administrative tasks in a healthcare setting.It enables he management of patient data and

### appointments through a console-based interface.It aims to simulate real-world system within a simplified,educational framework.

### 2.2 Existing Systems

Current Hospital Management Solutions are Manual Systems and Advanced digital Systems.

🡪*Manual Systems*:Paper based logs and registers for patient records and appointments.This have a

Prone to human error and difficult to update or track over time.

🡪*Advanced Digital Systems*:Full\_Fledged ERP systems like MediSoft or OpenEMR.Feature-rich

Databaseintegration,billing,inventory and reporting.

----------------------------------------------------------------------------------------------------------------------

### 2.3 Limitations of Existing Systems

### 🡪*Manual Systems:*

* Inefficient and error-prone.
* Difficult to retrieve or search data.
* High risk of data loss or duplication.

🡪*Advanced Systems:*

* Require high-end infrastructure.
* Costly licensing and maintenance.
* Often too complex for small clinics or educational settings.
* Dependence on trained staff and IT Support.

---------------------------------------------------------------------------------------------------------------------

### 2.4 Need for a New System

### A light weight,easy to use system is needed for smaller healthcare setups that do not require extensive features.

🡪**The New System should:**

\* Be simple to operate and understand.

\* Require minimal system resources.

\* Demonstrate fundamental data structure concepts in a real-world context.

\* Provide essential functionalities without the overhead of complex software.

This C-based system serves both practical and educational purposes,making it suitable for

Small Clinics,learning environments or early-stage development of a larger system.

**CHAPTER 3: SOFTWARE REQUIREMENT ANALYSIS**

Software requirement analysis is the process of identifying and documenting the functional and non-functional needs of the software system. For the Hospital Management System project developed in C language, this phase is crucial to define how the system should behave and what features it must provide. This project focuses on implementing arrays to store patient records and a linked list to Schedule appointments.

* **3.1 Functional requirements**

Functional requirements define the operations and services the system should perform. The following are the key functional requirements of the proposed system:

1. **Patient Mangement:**

* Add new patient records(name,age,gender,ID)
* Display all patient records.
* Search patients by ID or name.

2**.Appointment Management**:

* Schedule new Appointments (linked to patient ID,with date and doctor name).
* View all Appointments.
* Delete Appointments based on patient ID.

**3.User interface**

A menu-driven interface will be provided to allow the user to navigate the program easily—perform conversions, view history, or exit the program.

* **3.2 Non-functional requirements**

Non-functional requirements describe how the system should perform and define its quality attributes.

1. **Usability**  
   The system should be user-friendly and intuitive. All prompts and messages must be clear and informative.
2. **Reliability**  
   The application must handle invalid inputs gracefully, such as unsupported currencies or negative values, and display appropriate error messages without crashing.
3. **Efficiency**  
   The program should be lightweight and should execute quickly without unnecessary memory usage. Memory used by linked lists should be freed properly before program termination.
4. **Portability**  
   The program should be able to run on multiple operating systems, including Windows and Linux, as long as a C compiler is available.
5. **Maintainability**  
   The code should be modular and well-documented to make future changes, debugging, or enhancements easier.

* **3.3 Software and hardware requirements**

1. **Software requirements:**

* C language compiler (e.g., GCC, Turbo C)
* Text editor or IDE (e.g., Visual Studio Code, CodeBlocks)
* Operating system: Windows or Linux

1. **Hardware requirements:**

* Minimum 1 GB RAM
* Minimum 100 MB of disk space
* Standard input/output devices (keyboard and monitor)
* **3.4 Assumptions**
* Each patient will have a unique ID manually entered during registration.
* Appointments are only created for existing patient IDs.
* **3.5 Constraints**:
* Patients record are stored in a fixed-size array,which limits scalability.
* The system will be a console-based application, without graphical interfaces.

### CHAPTER 4:SOFTWARE DESIGN

### 4.1 Introduction

Software design is the process of defining the architecture,components,interfaces and data for a system to satisfy specified requirements.It serves as a blue print for both the system and project development.For this Hospital Management System,the design focuses o two main modules-patient management using arrays and Appointment Scheduling using Linked Lists.

### Project Flow

Data Structures:

* Array for storing patient records.
* Linked List for dynamic appointment scheduling.

User-Interface:

Menu driven command-line interface.

Control Flow:

Main Menu🡪User Input🡪Function call🡪Output🡪Return to menu

**4.2.Control Flow diagram**:

+-------------------------+

| Start Program |

+-----------+-------------+

|

v

+-------------------------+

| Display Main Menu |

| (1-Patients, 2-Appointments) |

+-----------+-------------+

|

+--------------------+-------------------+

| |

v v

+-------------------------+ +----------------------------+

| Patient Management | | Appointment Management |

+-------------------------+ +----------------------------+

| 1. Add Patient | | 1. Add Appointment |

| 2. Search Patient | | 2. View Appointments |

| 3. Display Patients | | 3. Delete Appointment |

+-------------------------+ +----------------------------+

| |

+--------------------+-------------------+

|

v

+-------------------------+

| Return to Menu |

+-----------+-------------+

|

v

+-------------------------+

| Exit Program? |

| (Y)es / (N)o |

+-----------+-------------+

|

+-----------------------------+

| |

v v

+---------------------+ +-------------------------+

| End Program | | Display Main Menu |

+---------------------+ +-------------------------+

**CHAPTER 5: PROPOSED SYSTEM**

**5.1 Introduction**

The proposed system is a console-based Hospital Management System application developed in the C programming language. The proposed System allows hospital staff to:

* Register new patients,view and search patients.
* Manage appointments(add,display and delete).
* Avoid redundancy and data entry errors.
* Perform quick lookups using ID or name-based search.

**5.2 Objectives**

* To ensure modular and maintainable code.
* To separate logic between patient data and appointment handling.
* To implement data structures efficiently in C.
* Uses modules for Patient data and Appointment Scheduling.

---------------------------------------------------------------------------------------------------------------------

**5.3 System Architecture**

The system is based on a modular procedural architecture,where each task is handled by a separate function.

**1.Presentation Layer(Use Interface):**

* Acts as the interaction between the user and the system.it is console-based,menu-driven interface.
* Displays options to the user.Takes input from user.Sends input to the corresponding logic layer.

**2. Logic Layer:**

* Handles all the core processing tasks of the system.

**Modules:**

**Patient Module:**

🡪addPatient()-Adds a patient.

🡪searchPatient()-Searches a patient by ID or name

🡪displayPatients()-Display all patient records.

**Appointment Module:**

**🡪**addAppointment()-Adds a node to the linked list.

**🡪**viewAppointments()-Display all Appointments.

**🡪**deleteAppointment()-Removes appointment by patient ID.

**3.Data Layer:**

Stores and manages data in memory.

Data Structures:

Array(Patient records):

Fixed size of struct patient.Sequential access based on index.

Linked List(Appointments):

Dynamically allocated nodes of struct Appointment.Allows flexible insertion,deletion and traversal.

**5.4 Features of the Proposed System**

* Structured Patient Information Management.
* Dynamic Appointment Scheduling.
* Search Functionalities.
* Interactive console interface
* Memory efficient design.
* Custom deletion operations.
* Educational Value and Modularity.
* **5.5 Advantages of the Proposed System**
* Simple and suitable for small clinics or academic use.
* Fast data access and manipulation using basic Data Structures.
* No need for external databases.
* Good practical implementation of Array and Linked List.

### CHAPTER 6: CODING

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_PATIENTS 100

struct Patient {

int id;

char name[100];

int age;

char gender[10];

};

struct Appointment {

int patientId;

char doctorName[100];

char date[20];

struct Appointment\* next;

};

struct Patient patients[MAX\_PATIENTS];

int patientCount = 0;

struct Appointment\* appointmentHead = NULL;

void addPatient();

void displayPatients();

void searchPatient();

void addAppointment();

void displayAppointments();

void deleteAppointment();

void addPatient() {

if (patientCount >= MAX\_PATIENTS) {

printf("Maximum patient limit reached.\n");

return;

}

printf("Enter patient name: ");

scanf(" %[^\n]", patients[patientCount].name);

printf("Enter age: ");

scanf("%d", &patients[patientCount].age);

printf("Enter gender: ");

scanf(" %s", patients[patientCount].gender);

patients[patientCount].id = patientCount + 1;

printf("Patient registered with ID: %d\n", patients[patientCount].id);

patientCount++;

}

void displayPatients() {

printf("\n--- Patient List ---\n");

for (int i = 0; i < patientCount; i++) {

printf("ID: %d, Name: %s, Age: %d, Gender: %s\n",

patients[i].id, patients[i].name,

patients[i].age, patients[i].gender);

}

}

void searchPatient() {

int choice, id;

char name[100];

printf("Search by:\n1. ID\n2. Name\nEnter choice: ");

scanf("%d", &choice);

if (choice == 1) {

printf("Enter patient ID: ");

scanf("%d", &id);

for (int i = 0; i < patientCount; i++) {

if (patients[i].id == id) {

printf("Found: %s, Age: %d, Gender: %s\n",

patients[i].name, patients[i].age, patients[i].gender);

return;

}

}

printf("Patient not found.\n");

} else if (choice == 2) {

printf("Enter patient name: ");

scanf(" %[^\n]", name);

for (int i = 0; i < patientCount; i++) {

if (strcmp(patients[i].name, name) == 0) {

printf("Found: ID: %d, Age: %d, Gender: %s\n",

patients[i].id, patients[i].age, patients[i].gender);

return;

}

}

printf("Patient not found.\n");

} else {

printf("Invalid choice.\n");

}

}

void addAppointment() {

int patientId;

printf("Enter patient ID for appointment: ");

scanf("%d", &patientId);

if (patientId < 1 || patientId > patientCount) {

printf("Invalid patient ID.\n");

return;

}

struct Appointment\* newAppt = (struct Appointment\*)malloc(sizeof(struct Appointment));

newAppt->patientId = patientId;

printf("Enter doctor's name: ");

scanf(" %[^\n]", newAppt->doctorName);

printf("Enter appointment date (dd-mm-yyyy): ");

scanf(" %s", newAppt->date);

newAppt->next = appointmentHead;

appointmentHead = newAppt;

printf("Appointment scheduled.\n");

}

void displayAppointments() {

struct Appointment\* temp = appointmentHead;

printf("\n--- Appointments List ---\n");

if(temp !=NULL){

while (temp != NULL) {

printf("Patient ID: %d, Doctor: %s, Date: %s\n",

temp->patientId, temp->doctorName, temp->date);

temp = temp->next;

}

}

else{

printf("No Appointments Scheduled.");

}

}

void deleteAppointment() {

struct Appointment\* temp = appointmentHead;

struct Appointment\* prev = NULL;

while (temp != NULL) {

int patientId;

printf("Enter patient ID to delete appointment: ");

scanf("%d", &patientId);

if (temp->patientId == patientId) {

if (prev == NULL) {

appointmentHead = temp->next;

} else {

prev->next = temp->next;

}

free(temp);

printf("Appointment deleted.\n");

return;

}

prev = temp;

temp = temp->next;

}

printf("No appointments found.\n");

}

int main() {

int choice;

while (1) {

printf("\n--- Hospital Management System ---\n");

printf("1. Add Patient\n");

printf("2. Display Patients\n");

printf("3. Search Patient\n");

printf("4. Add Appointment\n");

printf("5. Display Appointments\n");

printf("6. Delete Appointment\n");

printf("7. Exit\n");

printf("Enter choice: ");

scanf("%d", &choice);

switch (choice) {

case 1: addPatient(); break;

case 2: displayPatients(); break;

case 3: searchPatient(); break;

case 4: addAppointment(); break;

case 5: displayAppointments(); break;

case 6: deleteAppointment(); break;

case 7: printf("Exiting Program...");

exit(0);

default: printf("Invalid choice.\n");

}

}

return 0;

}

* **CHAPTER 7:OUTPUT RESULT/SCREENS**



### CHAPTER 8: CONCLUSION

### The Hospital Management System developed using C provides a foundational solution for organizing patient data and scheduling appointments in a structured,efficient manner.

* The project shows practical application of core Data Structures.
* Although basic,it effectively addresses the challenges of manual-record keeping and can be further enhanced with features like file storage.
* This project not only fulfills it’s objectives but also reinforces the importance of structured programming and logical problem solving in real-world problems.

**Future Enhancements**

**🡪**File-Handling for data Storage ,database integration,Mobile or web-based Access and Data Validation.