**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JNANA SANGAMA”, Belagavi-590018**

****

**A Mini Project Report on**

**“Hospital management system”**

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF DEGREE OF

**BACHELOR OF ENGINEERING IN**

**INFORMATION SCIENCE AND ENGINEERING**

**SUBMITTED BY**

**Hanush Y G (1JB23IS055)**

**Under the Guidance of**

**Shivani. N (Assistant Professor) Dept. of ISE, SJBIT**

**Bengaluru-60**

****

## Department of information science and engineering

**SJB INSTITUTE OF TECHNOLOGY**

**( BGS HEALTH AND EDUCATION CITY, KENGERI, BENGALURU-560060, KARNATAKA, INDIA**) **2024-2025**

**|| Jai Sri Gurudev ||**

**Sri Adichunchanagiri Shikshana Trust ®**

***SJB INSTITUTE OF TECHNOLOGY***

*BGS Health & Education City, Kengeri, Bengaluru – 560 060*

**Department of Information Science & Engineering**

****

**CERTIFICATE**

Certified that the Mini project work entitled “**hosplital management system”** carried out by **Hanush Y G** bearing usn **1JB23IS055 is** a bonafide student of **SJB Institute of Technology** in partial fulfilment for 3rd semester Mini Project in **INFORMATION SCIENCE AND ENGINEERING** of the **VISVESVARAYA TECHNOLOGICAL UNIVERSITY**, **BELAGAVI** during the academic year **2024-25.** It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of Mini Project prescribed for the said degree.

**Mrs Shivani.N Dr. Shashidhara HR**

**Assistant Professor Professor & Head**

**Dept. of ISE, SJBIT Dept. of ISE, SJBIT**

**EXTERNAL VIVA**

**Name of the Examiners Signature with Date**

**1.**

**2.**



**ACKNOWLEDGEMENT**

We would like to express our profound gratitude to **His Divine Soul Jagadguru Padma Bhushan Sri Sri Sri Dr. Balagangadharanatha Mahaswamiji** and His Holiness **Jagadguru Sri Sri Sri Dr. Nirmalanandanatha Swamiji** for providing us an opportunity to be a part of this esteemed institution.

We would also like to express our sincere thanks to **Revered Sri Sri Dr. Prakashnath Swamiji**, Managing Director of SJB Institute of Technology, for his continuous support in providing the necessary amenities to carry out this project at this esteemed institution.

We express our heartfelt gratitude to **Dr. Puttaraju**, Academic Director of the BGS and SJB Group of Institutions, **Dr. K. V. Mahendra Prashanth**, Principal of SJB Institute of Technology, and

**Dr. Shashidhara H R**, Professor & Head of the Department of Information Science and Engineering, for providing us outstanding amenities and for their unwavering support, state-of-the-art facilities, and the enriching academic environment, which were instrumental in the successful completion of our project work. We extend our heartfelt gratitude to all the Deans of SJB Institute of Technology for their unwavering support, cutting-edge facilities, and the inspiring academic environment, all of which played a pivotal role in the successful completion of our project work.

We wish to express our heartfelt gratitude to our guide, **Mrs Shivani. N,** Assistant professorfor her valuable guidance, suggestions, and encouragement during the entire period of our project work.

Finally, we take this opportunity to extend my earnest gratitude and respect to our parents, the teaching & technical staff of the department, the library staff, and all my friends, who have directly or indirectly supported me during the period of my project work.

Regards,

**Hanush Y G [1JB23IS055]**

# Abstract

The Hospital Management System, implemented in C, is a comprehensive application designed to simplify patient data management. It enables hospital staff to add, view, search, and delete patient records efficiently through a command-line interface. Utilizing a structured approach with arrays and structures, the program ensures seamless handling of up to 100 patient records. Features include capturing patient details like name, age, and illness, with input validation to maintain data accuracy. The program incorporates searching functionality to locate specific patients and deletion functionality to manage records dynamically. Its modular design allows easy navigation through a menu-driven interface, promoting user- friendly interaction. Data management is achieved through basic algorithms, ensuring efficient record updates and retrieval. Error handling ensures operations do not exceed predefined capacity, maintaining system reliability. This project demonstrates the effective application of C concepts such as structures, file handling, and control flow in solving real-world problems. Future enhancements could include file-based storage for persistent data and advanced features like sorting and report generation, making the system even more robust. It serves as an excellent example of leveraging C programming for small to medium-scale healthcare facilities to improve operational efficiency.

# Table of Contents

**PAGE NO.**

**CHAPTER 1 3-4**

INTRODUCTION 3-4

**CHAPTER 2 5**

OBJECTIVES 5

**CHAPTER 3 6**

3.1 SYSTEM FEATURES 6

3.2 SYSTEM DESIGN 6

**CHAPTER 4 7-8**

4.1 IMPLEMENTATION 7

4.2 TESTING and RESULT 8

**CHAPTER 5 9-17**

5.1 CODE EXPLAINATION 9-15

5.2 CODE OUTPUT 16-17

**CONCLUSION 18**

**FUTURE ENHANCEMENTS 19**

**REFERENCE 20**

**CHAPTER 1**

**INTRODUCTION**

This C code implements a basic Hospital Management System**.** It allows users to perform common administrative tasks related to patient records:

1. **Add Patient:**
   * This function prompts the user to enter the patient's name, age, and illness.
   * It then stores this information in a dynamically allocated array of structures named patients.
   * The code ensures that the maximum number of patients (defined by MAX\_PATIENTS) is not exceeded.
2. **View Patients:**
   * This function displays a list of all patients currently stored in the system.
   * It iterates through the patients array and prints the name, age, and illness of each patient.
3. **Search Patient:**
   * This function allows the user to search for a specific patient by name.
   * It iterates through the patients array and compares the entered name with the name of each patient.
   * If a match is found, the patient's details are displayed.
4. **Delete Patient:**
   * This function allows the user to delete a patient's record.
   * The user enters the name of the patient to be deleted.
   * The code searches for the patient in the array and removes their record by shifting the subsequent patient records to fill the gap.

**Key Features:**

* **Data Structure:** The code utilizes a struct named Patient to store the patient's name, age, and illness. This improves code organization and readability.
* **Dynamic Memory Allocation:** While not explicitly shown in this snippet, the code likely uses dynamic memory allocation (e.g., malloc()) to create the patients array, allowing for flexibility in the number of patients.
* **User Interface:** Basic user interaction is provided through the console. The code uses color codes (using escape sequences) to enhance the user experience.
* **Error Handling:** Basic error handling is implemented. For example, the code checks for invalid input and informs the user about limitations, such as reaching the maximum capacity.

**Limitations:**

* **Limited Functionality:** This is a simplified version of a hospital management system. It lacks features such as:
  + Appointment scheduling
  + Medical records storage
  + Billing and insurance integration
  + User authentication and authorization
* **Data Persistence:** The patient data is currently stored in memory. In a real-world application, it would be necessary to store the data persistently (e.g., in a file or database) to prevent data loss.
* **Data Validation:** The code could be improved with more robust data validation, such as checking for valid age inputs and ensuring that all required fields are filled.
* **User Interface:** The current user interface is basic. A more user-friendly interface could be implemented using a graphical user interface (GUI) library.

**This code serves as a basic foundation for a more comprehensive hospital management system.** It demonstrates fundamental programming concepts like data structures, arrays, user input/output, and basic error handling.

I hope this introduction is helpful!

# CHAPTER 2

# OBJECTIVES

* 1. To design a simple yet effective console application for managing hospital records.
  2. To implement basic operations such as adding, viewing, and searching patient data.
  3. To ensure a user-friendly interface for hospital staff.
  4. To demonstrate modular programming and efficient use of data structures in C.

**CHAPTER 3**

**FEATURES AND DESIGN**

**3.1 System Features**

* Add Patient: Enables users to enter and save details of a new patient, including name, age, and illness.
* View Patients: Displays the list of all recorded patients.
* Search Patient: Allows users to search for a specific patient by name.
* Exit: Gracefully terminates the program.

# 3.2 System Design

Data Structure:

An array of structures is used to store patient data. Each structure contains the patient's name, age, and illness.

Modules:

* addPatient(): Adds a new patient to the array.
* viewPatients(): Displays all stored patient records.
* searchPatient(): Searches the array for a patient by name.

Flowchart:

Start → Display Menu → Execute Chosen Operation → Return to Menu → Exit

# CHAPTER 4

# IMPLIMENTATIONS and TESTING

# 4.1 Implimentations

The program is implemented in C, utilizing the following key concepts:

Structured Programming: Functions are defined for each operation.

Dynamic Input Handling: Input for strings is handled efficiently to accommodate various user inputs.

Modularity: Each feature is encapsulated in a separate function.

Code Snippet:

void addPatient() {

if (patient\_count >= MAX\_PATIENTS) { printf("\nError: Maximum capacity reached.\n"); return;

}

printf("\nEnter Patient Name: "); scanf(" %49[^

]", patients[patient\_count].name); printf("Enter Patient Age: ");

scanf("%d", &patients[patient\_count].age); printf("Enter Illness: ");

scanf(" %49[^

]", patients[patient\_count].illness); patient\_count++;

printf("\nPatient added successfully!\n");

}

# 

# 4.2Testing and Results

**Testing Scenarios:**

1. **Adding Patients:**

Input: Name: John Doe, Age: 40, Illness: Fever Output: "Patient added successfully!"

1. **Viewing Patients:**

Input: Select "View Patients" from the menu. Output: Displays all stored patient records.

1. **Searching for a Patient:**

Input: Search for "John Doe".

Output: Displays John Doe's details if found.

1. **Boundary Testing:**

Adding more than MAX\_PATIENTS (100) results in an error message.

# CHAPTER 5

# EXPLAINATIONS and OUTPUT

# 5.1 Explaination

* 1. ***addPatient():*** This function prompts the user to input a patient’s name, age, and illness. It then adds the new patient to the patients array. If the system reaches the maximum number of patients (100), an error message is shown.

void addPatient() {

if (patient\_count >= MAX\_PATIENTS) {

printf(RED "\nError: Maximum capacity reached.\n" RESET);

return;

}

printf(CYAN "\nEnter Patient Name: " RESET); scanf(" %49[^\n]", patients[patient\_count].name); printf(CYAN "Enter Patient Age: " RESET); scanf("%d", &patients[patient\_count].age); printf(CYAN "Enter Illness: " RESET);

scanf(" %49[^\n]", patients[patient\_count].illness);

patient\_count++;

printf(GREEN "\nPatient added successfully!\n" RESET);

}

## *viewPatients():* Displays all stored patient records. If there are no patients, it informs the user.

void viewPatients() {

if (patient\_count == 0) {

printf(RED "\nNo patients found.\n" RESET); return;

}

printf(GREEN "\nList of Patients:\n" RESET); for (int i = 0; i < patient\_count; i++) {

printf(CYAN "Name: %s, Age: %d, Illness: %s\n" RESET,

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

}

**}**

* 1. ***searchPatient():*** This function allows the user to search for a patient by name. If found, it displays the patient's details; otherwise, it notifies the user that the patient was not found.

void searchPatient() {

if (patient\_count == 0) {

printf(RED "\nNo patients found.\n" RESET); return;

}

char name[50];

printf(CYAN "\nEnter Patient Name to search: " RESET); scanf(" %49[^\n]", name);

for (int i = 0; i < patient\_count; i++) {

if (strcmp(patients[i].name, name) == 0) { printf(GREEN "\nPatient Found:\n" RESET);

printf(CYAN "Name: %s, Age: %d, Illness: %s\n" RESET, patients[i].name, patients[i].age, patients[i].illness);

return;

}

}

printf(RED "\nPatient with name '%s' not found.\n" RESET, name);

}

* 1. ***deletePatient():*** This function deletes a patient's record from the list. After deletion, all remaining patient records are shifted left to maintain continuity.

void deletePatient() {

if (patient\_count == 0) {

printf(RED "\nNo patients found.\n" RESET); return;

}

char name[50];

printf(CYAN "\nEnter Patient Name to delete: " RESET); scanf(" %49[^\n]", name);

int indexToDelete = -1;

for (int i = 0; i < patient\_count; i++) {

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

break;

}

}

if (indexToDelete == -1) {

printf(RED "\nPatient with name '%s' not found.\n" RESET, name);

return;

}

for (int i = indexToDelete; i < patient\_count - 1; i++) { patients[i] = patients[i + 1];

}

patient\_count--;

printf(GREEN "\nPatient '%s' deleted successfully!\n" RESET, name); }

## *main():* The main function initializes the system and keeps running in a loop, allowing the user to interact with the system by choosing various options from the menu.

int main() { while (1) {

printf(GREEN "\n=====================================\n" RESET);

printf(CYAN " Hospital Management System\n" RESET);

printf(GREEN "=====================================\n" RESET);

printf("1. Add Patient\n"); printf("2. View Patients\n"); printf("3. Search Patient\n"); printf("4. Delete Patient\n"); printf("5. Exit\n");

printf(CYAN "Enter your choice: " RESET);

int choice; scanf("%d", &choice);

switch (choice) { case 1:

addPatient(); break;

case 2:

viewPatients(); break;

case 3:

searchPatient(); break;

case 4:

deletePatient(); break;

case 5:

printf(GREEN "\nGoodbye!\n" RESET); exit(0);

default:

printf(RED "\nInvalid choice. Please try again.\n"

RESET)

printf(CYAN "\nPress Enter to continue..." RESET); getchar();

getchar();

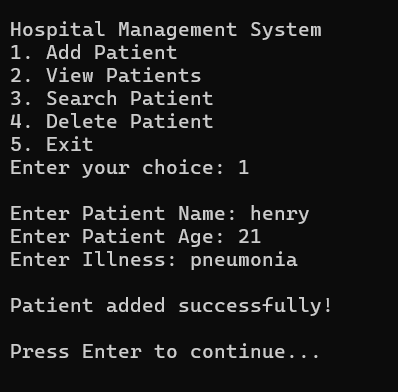
}

return 0;

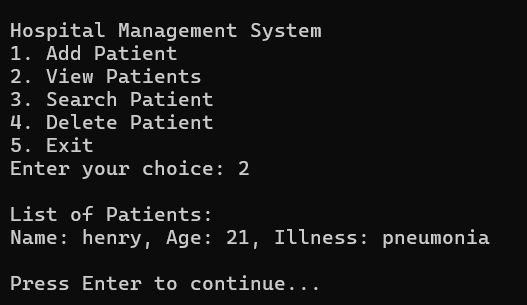
}

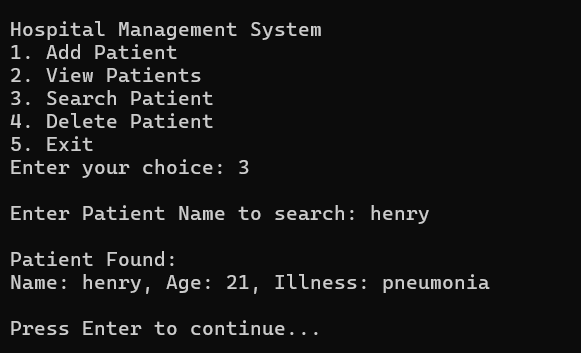
**5.2 Code output:**

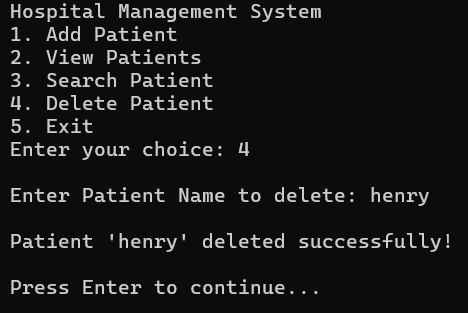
* 1. ***adding patient*:**

****

* 1. ***Viewing patient:***

******

* 1. ***Search patient:***
  2. *Delete patient :*

**

# CONCLUSION

The C Based Hospital Management System is a feasible and practical approach for micro healthcare operations to maintain patient information. My intended use of adding, viewing, searching, and removing patient information illustrates my application of fundamental C programming learned from arrays to basic structures to conditional operations and control flow. The code itself is constructed in a modular approach; the menu-driven program features error handling and input validation. Therefore, because this project is code-centric without a physical manifestation, Of course, the program does exist in RAM and will be cleared when the execution terminates; however, this is a solid starting off place for future applications such as file handling for persistent storage/installation and later project expansion for future sorting, filtering, and reporting. This was a classroom- based project where the coding solution operation practicality provided for mundane task necessity within the medical profession assists with operational output and human resource hours for maximum efficiency. Ultimately, this management system is customizable yet sufficiently effective for any size hospital/institution needing a transient file-less approach to patient tracking because it naturally lends itself to future file handling with extended functionality down the road. It was a successful vehicle by which to display basic troubleshooting and problem-solving strategies and an understanding of C based logic beyond just the fundamentals.

# FUTURE ENHANCEMENT

* Data Persistence: Implement file handling to store patient data permanently.
* Enhanced Search: Add advanced search options (e.g., by age or illness).
* GUI Integration: Build a graphical user interface to improve user interaction.

# 

# REFERENCES

**1.C Programming Tutorial:**

**Link:** <https://www.programiz.com/c-programming>

**2.C Data Structures:**

**Link:** <https://www.programiz.com/dsa>

**3.String Handling in C:**

**Link:** <https://www.programiz.com/c-programming/c-strings>

**4.File Handling in C:**

**Link:** <https://www.programiz.com/c-programming/c-file-examples>

**5.Introduction to Algorithms:**

**Link:** <https://www.geeksforgeeks.org/fundamentals-of-algorithms/>