HOSPITAL MANAGEMENT SYSTEM

A Course End Project Report

ADVANCED DATA STRUCTURES LABORATORY (A8513)

In partial fulfillment of the requirements
for the award of the degree of

BATCHELOR OF TECHNOLOGY

IN

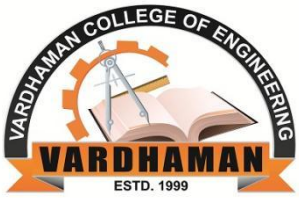
COMPUTER SCIENCE AND ENGINEERING

Submitted By

NIHARIKA DERANGULA
(22881A0512)

Under the guidance of

Mr. Naresh Goud M
Assistant Professor
Department of Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

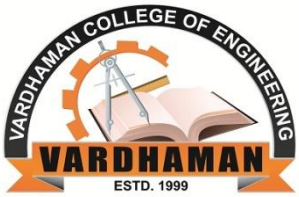
VARDHAMAN COLLEGE OF ENGINEERING
(AUTONOMOUS)
Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC with A++ Grade, ISO 9001:2015 Certified
Kacharam, Shamshabad, Hyderabad – 501218, Telangana, India.

January, 2024

VARDHAMAN COLLEGE OF ENGINEERING

(AUTONOMOUS)
Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC with A++ Grade, ISO 9001:2015 Certified
Kacharam, Shamshabad, Hyderabad – 501218, Telangana, India.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the Course End Project titled **"HOSPITAL MANAGEMENT SYSTEM"** is carried out by **Ms.D. NIHARIKA** , Roll Number **22881A0512** towards **A8513 – Advanced Data Structures Laboratory** course and submitted to **Department of Computer Science and Engineering**, in partial fulfilment of the requirements for the award of degree of **Bachelor of Technology** in **Department of Computer Science and Engineering** during the Academic year 2023-24.

Instructor:

Mr. Naresh Goud M
Assistant Professor,
Dept of Computer Science and Engineering,
Vardhaman College of Engineering,
Hyderabad.

Head of the Department:

Dr. Ramesh Karnati ,
Head of the Department,
Dept. of Computer Science and Engineering,
Vardhaman College of Engineering,
Hyderabad. .

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

We wish to express our deep sense of gratitude to **Mr. Naresh Goud M**, Assistant Professor, Department of Computer Science and Engineering, Vardhaman College of Engineering, for her able guidance and useful suggestions, which helped us in completing the design part of potential project in time.

We particularly thankful to **Dr. Ramesh Karnati**, Associate Professor & Head, Department of Computer Science and Engineering for his guidance, intense support and encouragement, which helped us to mould our project into a successful one.

We show gratitude to our honorable Principal **Dr. J.V.R.Ravindra**, for having provided all the facilities and support.

We avail this opportunity to express our deep sense of gratitude and heartfelt thanks to **Dr. Teegala Vijender Reddy**, Chairman and **Sri Teegala Upender Reddy**, Secretary of VCE, for providing a congenial atmosphere to complete this project successfully.

We also thank all the staff members of Computer Science and Engineering for their valuable support and generous advice. Finally, thanks to all our friends and family members for theircontinuous support and enthusiastic help.

NIHARIKA DERANGULA

(22881A0512)

TABLE OF CONTENTS

INTRODUCTION

A Hospital Management System (HMS) is a software solution designed to streamline and automate various administrative and clinical processes within a healthcare facility. It aims to enhance the efficiency of managing patient data, appointments, billing, and overall hospital operations. The system typically includes modules for patient registration, appointment scheduling, electronic health records (EHR), pharmacy management, and financial transactions. By integrating these functions, HMS contributes to improved patient care, reduced paperwork, and better decision-making for healthcare professionals.

OBJECTIVE OF THE PROJECT

The objectives of a Hospital Management System (HMS) :

1**. Efficient Patient Management**:
- Streamlining the process of managing patient information, appointments, and medical records.

2. **Improved Workflow**;
- Enhancing the overall efficiency of hospital operations, reducing paperwork, and minimizing manual errors.

3. **Enhanced Patient Care**:
- Ensuring better and more informed healthcare delivery through quick access to patient records and histories.

4. **Appointment Scheduling Optimization**:
- Creating a systematic and effective appointment scheduling syssssssssstem to manage patient flow.

5**. Financial Management**:
- Tracking and managing financial transactions related to patient care, billing, and invoicing.

6**. Pharmacy Management**:
- Efficiently managing drug inventory, prescriptions, and dispensing.

7. **Staff Management:**
- Organizing and tracking information related to hospital staff, their roles, and responsibilities.

8. **Emergency Handling:**
- Providing a mechanism for quick response and efficient handling of emergency cases.

9. **Reporting and Analytics**:

- Generating comprehensive reports and analytics for hospital performance evaluation.

10**. Integration and Interoperability**:
- Facilitating seamless communication and data exchange between different modules and systems within the healthcare facility.

11. **Security and Compliance**:
- Ensuring the security of patient data and compliance with healthcare regulations and standards.

12. **User-Friendly Interface**:
- Designing an intuitive and user-friendly interface for easy navigation and use by healthcare professionals.

13. **Scalability**:
- Designing the system to accommodate growth and changes in the hospital's size and requirements.

14. **Patient Privacy**:
- Emphasizing the importance of maintaining patient privacy and confidentiality in all aspects of the system.

15. **Training and Support:**
- Providing adequate training and support to hospital staff for effective utilization of the system.

16. **Cost-effectiveness**:
- Implementing a solution that optimizes resource utilization and contributes to cost-effectiveness in hospital management.

PROBLEM STATEMENT

n the current healthcare environment, many hospitals face challenges in managing patient information, appointments, and overall operational efficiency. The manual processes involved in maintaining records often lead to errors, delays, and difficulties in accessing critical patient data. To address these issues, the development of a Hospital Management System (HMS) using linked lists in C is proposed.

Key Problems:

1. **Inefficient Patient Data Management**:
   - Manual handling of patient records leads to inefficiencies, making it challenging to organize and retrieve information promptly.

2. **Appointment Scheduling Bottlenecks**:
   - The absence of a systematic approach to appointment scheduling results in congestion and delays in patient flow.

3. **Limited Accessibility to Electronic Health Records (EHR):**
   - Lack of a structured EHR system hampers healthcare providers' ability to access and update patient records efficiently.

4. **Financial Transaction Complexity:**
   - The existing financial management system lacks optimization, causing complications in tracking transactions related to patient care.

5. **Pharmacy Management Challenges:**
   - Current pharmacy management systems may not be integrated, leading to difficulties in tracking drug inventory and prescriptions.

6**. Ineffective Staff Management:**
   - The manual organization of staff information may result in mismanagement of roles, responsibilities, and communication.

7. **Emergency Handling Inefficiencies**:
   - The absence of a streamlined system for handling emergencies can lead to delays in response and resource allocation.

SOFTWARE AND HARDWARE REQUIREMENTS

Software Requirements:

1. C Compiler:
   - A C compiler such as GCC (GNU Compiler Collection) for compiling and executing C code.

2. Integrated Development Environment (IDE):
   - An IDE like Code::Blocks, Visual Studio Code, or Dev-C++ for an organized development environment.

3. Operating System:
   - Compatibility with Windows, Linux, or macOS, depending on the hospital's infrastructure.

4.Database Management System (Optional):
   - For enhanced data management, integration with a database system like MySQL or SQLite can be considered.

5. Graphics Library (Optional):
   - If graphical user interface (GUI) elements are desired, a library like GTK or Simple DirectMedia Layer (SDL) can be included.

Hardware Requirements:

1. Processor:
   - A modern processor with sufficient speed to handle data processing and system operations efficiently.

2. Memory (RAM):
   - Adequate RAM to ensure smooth execution of the application, especially when handling large datasets.

3.Storage Space:
   - Sufficient storage space for storing the compiled application and any associated data files.

4. Input/Output Devices:
   - Standard input devices such as a keyboard and mouse for user interaction.
   - Display monitor for visual output.
- If the system requires network capabilities for data sharing or remote access, appropriate networking hardware may be needed.

PROJECT DESCRIPTION

The Hospital Management System (HMS) in C using linked lists is a comprehensive software solution designed to streamline and enhance the efficiency of healthcare operations. This system focuses on organizing patient information, managing appointments, and optimizing various aspects of hospital administration through the implementation of linked list data structures.

Key Features:

1. **Patient Information Management**:

- Utilizes linked lists to efficiently store and manage patient details, including name, age, gender, and medical history.

- Dynamic updating and retrieval of patient records.

2. **Appointment Scheduling:**

- Implements linked list-based scheduling for effective management of patient appointments.

- Prioritizes appointments based on urgency and optimizes patient flow.

3. **Electronic Health Records (EHR):**

- Establishes a structured EHR system using linked lists for quick access to patient records.

- Enables healthcare providers to update and retrieve medical histories seamlessly.

4. **Financial Management:**

- Efficiently tracks financial transactions related to patient care, billing, and invoicing.

- Utilizes linked lists for organized and accessible financial records.

5. **Pharmacy Management:**

- Manages drug inventory and prescriptions using linked lists.

- Connects pharmacy data with patient records for comprehensive healthcare management.

6. **Staff Management**:

- Organizes and tracks hospital staff information, roles, and responsibilities.

- Implements a hierarchical linked list structure for staff management.

7. **Emergency Handling:**

- Establishes a linked list-based system for quick response and resource allocation during emergencies.

- Prioritizes emergency cases for efficient handling.

Technical Implementation:

- **C Programming Language**:
  - Utilizes the C programming language for system development.
  - Leverages the simplicity and efficiency of C for optimal performance.
- **User-Friendly Interface**:
  - Designs an intuitive command-line interface for easy user interaction.
  - Prioritizes user experience for healthcare professionals using the system.
- 4 .**Scalability:**
  - Develops the system with scalability in mind to accommodate future growth and additional functionalities.
- 5. **Security Measures**:
  - Implements data security measures to ensure patient confidentiality and compliance with healthcare regulations.
- . **Linked List Data Structures:**
  - Implements linked lists for dynamic and efficient data organization.
  - Enhances data retrieval and management through linked list functionalities.

PROCEDURE

1. Start:
- Begin the program.

2. Initialize Linked List:
- Initialize the linked list to store patient information (name, ID, etc.).

3. Menu:
- Display a menu for operations (Add patient, Remove patient, View patient list, Exit).

4. **User Input**:
- Prompt the user to choose an operation.

5. Switch Case
- Implement a switch-case structure based on the user's choice.

a. **Add Patient**:
- Prompt user for patient details.
- Create a new node in the linked list.
- Add the node to the list.

b. **Remove Patient**:
- Prompt user for patient ID to be removed.
- Search and delete the corresponding node.

c. **View Patient List**:
- Display details of all patients in the linked list.

d. **Exit**
- End the program.

6. Loop:
- Repeat steps 4-5 until the user chooses to exit.

7. **End**
- Terminate the program.

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Patient {
   int id;
   char name[50];
   int age;
   char gender;
   struct Patient *next;
};


struct Patient* addPatient(struct Patient *head, int id, const char *name, int age, char gender)
 {
   struct Patient newPatient = (struct Patient)malloc(sizeof(struct Patient));
   newPatient->id = id;
   strcpy(newPatient->name, name);
   newPatient->age = age;
   newPatient->gender = gender;
   newPatient->next = head;

   return newPatient;
}


struct Patient* deletePatient(struct Patient *head, int id)
{
   struct Patient *current = head;
   struct Patient *prev = NULL;

   while (current != NULL && current->id != id)
 {
       prev = current;
       current = current->next;
   }
   if (current == NULL)
 {
       printf("Patient with ID %d not found.\n", id);
       return head;
   }
   if (prev == NULL)
 {
       head = current->next;
   }
 else {
```

```
        prev->next = current->next;
        }
    printf("Patient with ID %d deleted.\n", id);
    free(current);
    return head;
}


struct Patient* searchPatient(struct Patient *head, int id)
 {
    struct Patient *current = head;
    while (current != NULL && current->id != id)
    {
      current = current->next;
     }
    if (current == NULL)
    {
       printf("Patient with ID %d not found.\n", id);
       return NULL;
    }

    printf("Patient found - ID: %d, Name: %s, Age: %d, Gender: %c\n",
          current->id, current->name, current->age, current->gender);
    return current;
}

// Function to count the number of patients in the linked list

int countPatients(struct Patient *head)
 {
    struct Patient *current = head;
    int count = 0;
    while (current != NULL)
 {
       count++;
       current = current->next;
    }
    return count;
}

// Function to display the list of patients

void displayPatients(struct Patient *head)
{
    struct Patient *current = head;
    printf("\nPatient List:\n\n");

    while (current != NULL)
 {
        printf("ID:\t%d\nName:\t%s\nAge:\t%d\nGender:\t%c\n\n",
```

```
            current->id, current->name, current->age, current->gender);
        current = current->next;
    }
}


void freePatients(struct Patient *head)
{
    struct Patient *current = head;
    struct Patient *next;
    while (current != NULL)
{
        next = current->next;
        free(current);
        current = next;
    }
}
int main()
 {
    struct Patient *patientList = NULL;
    int choice, id, age;
    char name[50], gender;
    do {
        printf("\nHospital Management System\n");
        printf("1. Add Patient\n");
        printf("2. Delete Patient\n");
        printf("3. Search Patient\n");
        printf("4. Count Patients\n");
        printf("5. Display Patients\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                printf("Enter patient ID: \n");
                scanf("%d", &id);
                printf("Enter patient name: \n");
                scanf("%s", name);
                printf("Enter patient age: \n");
                scanf("%d", &age);
                printf("Enter patient gender (M/F): \n");
                scanf(" %c", &gender);
                patientList = addPatient(patientList, id, name, age, gender);
                break;
            case 2:
                printf("Enter patient ID to delete: \n");
                scanf("%d", &id);
                patientList = deletePatient(patientList, id);
                break;
```

```
        case 3:
            printf("Enter patient ID to search: \n");
            scanf("%d", &id);
            searchPatient(patientList, id);
            break;
        case 4:
            printf("Number of patients: %d\n", countPatients(patientList));
            break;
        case 5:
        displayPatients(patientList);
            break;
        case 6:
            freePatients(patientList);
            printf("Exiting program.\n");
            break;
        default:
            printf("Invalid choice. Please try again.\n");
    }
  } while (choice != 6);
  return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Patient {
    int id;
    char name[50];
    int age;
    char gender;
    struct Patient *next;
};


struct Patient* addPatient(struct Patient *head, int id, const char *name, int age, char gender)
{
    struct Patient newPatient = (struct Patient)malloc(sizeof(struct Patient));
    newPatient->id = id;
    strcpy(newPatient->name, name);
    newPatient->age = age;
    newPatient->gender = gender;
    newPatient->next = head;
    return newPatient;
}


struct Patient* deletePatient(struct Patient *head, int id)
{
    struct Patient *current = head;
    struct Patient *prev = NULL;
   while (current != NULL && current->id != id) {
        prev = current;
        current = current->next;
    }
```

```
    while (current != NULL && current->id != id) {
        prev = current;
        current = current->next;
    }
    if (current == NULL) {
        printf("Patient with ID %d not found.\n", id);
        return head;
    }
    if (prev == NULL) {
        head = current->next;
    } else {
        prev->next = current->next;
    }
    printf("Patient with ID %d deleted.\n", id);
    free(current);
    return head;
}


struct Patient* searchPatient(struct Patient *head, int id) {
    struct Patient *current = head;
    while (current != NULL && current->id != id) {
       current = current->next;
}
    if (current == NULL) {
        printf("Patient with ID %d not found.\n", id);
        return NULL;
    }

    printf("Patient found - ID: %d, Name: %s, Age: %d, Gender: %c\n",
           current->id, current->name, current->age, current->gender);
    return current;
}
```

```c
int countPatients(struct Patient *head) {
    struct Patient *current = head;
    int count = 0;
    while (current != NULL) {
        count++;
        current = current->next;
    }
    return count;
}

// Function to display the list of patients
void displayPatients(struct Patient *head) {
    struct Patient *current = head;
    printf("\nPatient List:\n\n");
    while (current != NULL) {
         printf("ID:\t%d\nName:\t%s\nAge:\t%d\nGender:\t%c\n\n",
                current->id, current->name, current->age, current->gender);
        current = current->next;
    }
}


void freePatients(struct Patient *head) {
    struct Patient *current = head;
    struct Patient *next;
    while (current != NULL) {
        next = current->next;
        free(current);
        current = next;
    }
}

int main() {
    struct Patient *patientList = NULL;
    int choice, id, age;
```

```c
    int choice, id, age;
    char name[50], gender;
    do {
        printf("\nHospital Management System\n");
        printf("1. Add Patient\n");
        printf("2. Delete Patient\n");
        printf("3. Search Patient\n");
        printf("4. Count Patients\n");
        printf("5. Display Patients\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                printf("Enter patient ID: \n");
                scanf("%d", &id);
                printf("Enter patient name: \n");
                scanf("%s", name);
                printf("Enter patient age: \n");
                scanf("%d", &age);
                printf("Enter patient gender (M/F): \n");
                scanf(" %c", &gender);
                patientList = addPatient(patientList, id, name, age, gender);
                break;
            case 2:
                printf("Enter patient ID to delete: \n");
                scanf("%d", &id);
                patientList = deletePatient(patientList, id);
                break;
            case 3:
                printf("Enter patient ID to search: \n");
                scanf("%d", &id);
                searchPatient(patientList, id);
                break;
```

```c
            case 3:
                printf("Enter patient ID to search: \n");
                scanf("%d", &id);
                searchPatient(patientList, id);
                break;
            case 4:
                printf("Number of patients: %d\n", countPatients(patientList));
                break;
            case 5:
                displayPatients(patientList);
                break;
            case 6:
                freePatients(patientList);
                printf("Exiting program.\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 6);
    return 0;
}
```

RESULT:

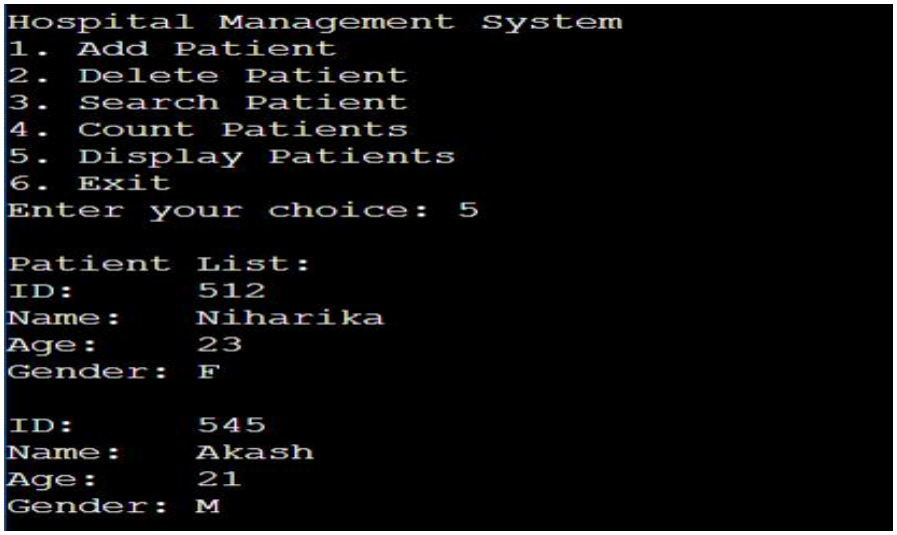
```
Hospital Management System
1. Add Patient
2. Delete Patient
3. Search Patient
4. Count Patients
5. Display Patients
6. Exit
Enter your choice: 1
Enter patient ID: 501
Enter patient name: Arjun
Enter patient age: 20
Enter patient gender (M/F): M

Hospital Management System
1. Add Patient
2. Delete Patient
3. Search Patient
4. Count Patients
5. Display Patients
6. Exit
Enter your choice: 1
Enter patient ID: 502
Enter patient name: yashna
Enter patient age: 21
Enter patient gender (M/F): F

Hospital Management System
1. Add Patient
2. Delete Patient
3. Search Patient
4. Count Patients
5. Display Patients
6. Exit
Enter your choice: 1
Enter patient ID: 545
Enter patient name: Akash
```

```
Enter your choice: 1
Enter patient ID: 545
Enter patient name: Akash
Enter patient age: 21
Enter patient gender (M/F): M

Hospital Management System
1. Add Patient
2. Delete Patient
3. Search Patient
4. Count Patients
5. Display Patients
6. Exit
Enter your choice: 1
Enter patient ID: 512
Enter patient name: Niharika
Enter patient age: 23
Enter patient gender (M/F): F
```


Hospital Management System
1. Add Patient
2. Delete Patient
3. Search Patient
4. Count Patients
5. Display Patients
6. Exit
Enter your choice: 5
Patient List:
ID: 512
Name: Niharika
Age: 23
Gender: F
ID: 545
Name: Akash
Age: 21
Gender: M

```
Name:   Akash
Age:    21
Gender: M

ID:     502
Name:   yashna
Age:    21
Gender: F

ID:     501
Name:   Arjun
Age:    20
Gender: M


Hospital Management System
1. Add Patient
2. Delete Patient
3. Search Patient
4. Count Patients
5. Display Patients
6. Exit
Enter your choice: 2
Enter patient ID to delete: 502
Patient with ID 502 deleted.

Hospital Management System
1. Add Patient
2. Delete Patient
3. Search Patient
4. Count Patients
5. Display Patients
6. Exit
Enter your choice: 3
Enter patient ID to search: 545
Patient found - ID: 545, Name: Akash, Age: 21, Gender: M
```

```
Hospital Management System
1. Add Patient
2. Delete Patient
3. Search Patient
4. Count Patients
5. Display Patients
6. Exit
Enter your choice: 4
Number of patients: 3

Hospital Management System
1. Add Patient
2. Delete Patient
3. Search Patient
4. Count Patients
5. Display Patients
6. Exit
Enter your choice: 5

Patient List:
ID:     512
Name:   Niharika
Age:    23
Gender: F

ID:     545
Name:   Akash
Age:    21
Gender: M

ID:     501
Name:   Arjun
Age:    20
Gender: M
```

```
Hospital Management System
1. Add Patient
2. Delete Patient
3. Search Patient
4. Count Patients
5. Display Patients
6. Exit
Enter your choice: 6
Exiting program.
```

CONCLUSION AND FUTURE WORK

Conclusion:

The hospital management system implemented in C using linked lists provides a foundation for efficiently managing patient records. Linked lists offer dynamic memory allocation, allowing for flexible storage and manipulation of data. The system includes essential functionalities such as adding patients, deleting patients, searching for patient records, counting patients, and displaying the patient list.

The use of linked lists enables easy insertion and removal of patient records, maintaining a dynamic and scalable system. The implementation provides a simple and functional solution for managing patient information.

Future Work:

1. **Enhanced Functionality**
   - Extend the system to include more comprehensive patient details such as medical history, contact information, and treatment records.

2. User Authentication:
   - Implement user authentication and authorization to secure access to sensitive patient information.

3. Database Integration:
   - Integrate a database system to store patient data persistently, allowing for data retrieval even after program termination.

4. GUI Implementation:
   - Develop a graphical user interface (GUI) for a more user-friendly and visually appealing interaction with the system.

5. Appointment Scheduling:
   - Extend the system to include features for scheduling appointments, managing doctors' schedules, and coordinating patient visits.

6. Billing and Insurance Integration:
   - Incorporate billing and insurance-related functionalities to streamline financial transactions and insurance claims processing.

7. Reporting and Analytics:
   - Implement reporting and analytics features to analyze patient data trends, track hospital performance, and support decision-making processes.

8. Multi-User Support:
- Allow multiple users to interact with the system simultaneously, considering concurrency and data integrity.

9. Error Handling and Validation:
- Strengthen the system by incorporating robust error handling and data validation mechanisms to ensure data integrity and reliability.

10. Mobile Application:
- Develop a mobile application for healthcare professionals and staff to access and manage patient records on the go.

REFERENCES

- "Data Structures Using C" by Aaron M. Tenenbaum, Yedidyah Langsam, and Moshe J. Augenstein.

- "C Programming Absolute Beginner's Guide" by Perry and Miller covers basics of C programming.

- GeeksforGeeks (geeksforgeeks.org)