M.KUMARASAMY
COLLEGE OF ENGINEERING
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur – 639 113.

A Project Report

on

HOSPITAL MANAGEMENT SYSTEM

Submitted in partial fulfilment of requirements for the award of the course

of

CGA1121 – DATA STRUCTURES

Under the guidance of

Mrs.S.Ananthi M.E.,

Assistant Professor/CSE

Submitted By

VENKAT RAGAV N (927623BAD123)

DEPARTMENT OF FRESHMAN ENGINEERING

M.KUMARASAMY COLLEGE OF ENGINEERING
(Autonomous)

KARUR – 639 113

MAY 2024

M.KUMARASAMY
COLLEGE OF ENGINEERING
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur – 639 113.

M. KUMARASAMY COLLEGE OF ENGINEERING

(Autonomous Institution affiliated to Anna University, Chennai)

KARUR – 639 113

BONAFIDE CERTIFICATE

Certified that this project report on **"HOSPITAL MANAGEMENT SYSTEM"** is the bonafide work of **VENKAT RAGAV N (927623BAD123)** who carried out the project work during the academic year 2023- 2024 under my supervision.

Signature

Mrs.S.Ananthi M.E.,

SUPERVISOR,

Department of Computer Science and Engineering,

M. Kumarasamy College of Engineering,

Thalavapalayam, Karur -639 113.

Signature

Dr. K.CHITIRAKALA, M.Sc., M.Phil.,Ph.D.,

HEAD OF THE DEPARTMENT,

Department of Freshman Engineering,

M. Kumarasamy College of Engineering,

Thalavapalayam, Karur -639 113.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

VISION OF THE INSTITUTION

To emerge as a leader among the top institutions in the field of technical education

MISSION OF THE INSTITUTION

- Produce smart technocrats with empirical knowledge who can surmount the global challenges
- Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students
- Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations

VISION OF THE DEPARTMENT

To produce competent industry relevant education, skillful research, technical and innovative computer science professionals acquaintance with managerial skills, human and social values.

MISSION OF THE DEPARTMENT

- To impart technical knowledge through innovative teaching, research, and consultancy.
- To develop and to promote student ability thereby to compete globally through excellence in education.
- To facilitate the development of academic-industry Collaboration.
- To produce competent engineers with professional ethics, technical competence and a spirit of innovation and managerial skills.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO 1: To acquire technical knowledge and proficiency required for the employment and higher education in the contemporary areas of computer science or management studies.

PEO 2: To apply their competency in design and development of innovative solutions for real-world problems.

PEO 3: To demonstrate leadership qualities with high ethical standards and collaborated with other industries for the socio-economical growth of the country.

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

1. **PSO1:** Ability to apply the analytical and business skills to provide sustainable solutions as an engineer/researcher for the real-time applications using Machine Learning, Internet of Things and Data analytics.
2. **PSO2:** Ability to practice ethical and human values with soft-skills qualities in computer science and business disciplines to emerge as an entrepreneur for the growth and development of the society.

ABSTRACT

Hospital Management System (HMS) is an integrated software solution designed to streamline and automate healthcare facility operations. It encompasses various modules catering to different departments, including patient management, appointment scheduling, billing and invoicing, inventory management, and staff administration. The patient management module stores comprehensive patient records, tracks visits, and provides easy access to information. Appointment scheduling enables online and offline bookings, manages doctor schedules, and sends reminders. Billing and invoicing generate accurate invoices, handle insurance claims, and offer financial reports. Inventory management tracks medical supplies, automates reordering, and monitors expiry dates. Staff administration maintains staff records, manages schedules, and facilitates payroll. Additional modules like EMR, LIS, RIS, and pharmacy management can be integrated for enhanced functionality. The development process involves requirement gathering, system design, development, testing, deployment, training, and ongoing maintenance and support. Key considerations include data security, scalability, user-friendliness, and integration with other hospital systems.

ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	POs MAPPED	PSOs MAPPED
Hospital Management System (HMS) is an integrated software solution designed to streamline and automate the day-to-day operations of a healthcare facility. It encompasses various modules that cater to different departments within the hospital, including patient management, appointment scheduling, billing and invoicing, inventory management, and staff administration.	**PO1(2)** **PO2(3)** **PO3(2)** **PO4(2)** **PO5(3)** **PO6(1)** **PO7(3)** **PO8(2)** **PO9(3)** **PO10(3)** **PO11(2)** **PO12(2)**	**PSO1(3)** **PSO2(2)**

Note: 1- Low, 2-Medium, 3- High

SUPERVISOR **HEAD OF THE DEPARTMENT**

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Introduction

A hospital management system is a useful tool that allows users to manage and organize contact information efficiently. In the world of programming, implementing such an application can be an exciting and educational task. One effective way to structure the data for a hospital management system is by using a doubly linked list.

A doubly linked list is a data structure that consists of nodes, where each node contains data and two pointers – one pointing to the next node in the sequence and another pointing to the previous node. This bidirectional linkage allows for easy traversal in both directions, making it a suitable choice for applications that require frequent insertions, deletions, and modifications.

In the context of a hospital management system, a doubly linked list can be employed to create a dynamic and flexible structure for managing contact details. Each node in the list can represent an entry in the phone directory, storing information such as name, phone number, address, and any other relevant data.

1.2 Objective

The objective of this response is to discuss the data structure choice for a Hospital Management System. This involves understanding the various aspects of data structure, including the representation of patient data, the use of tables and graphs, and the importance of data integrity and scalability.

1.3 Data Structure Choice

When considering data structures for storing contact information in a phone directory application, the choice is critical for optimizing operations like insertion, deletion, and search. One commonly compared option is the doubly linked list. Doubly linked lists offer advantages in terms of efficient insertion and deletion at any position, allowing for easy reorganization of contacts. The bidirectional traversal capability enables smooth navigation, supporting functionalities like sorting and iterative processing. However, the drawback lies in the additional memory required to store two pointers per node, potentially impacting space efficiency.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Linked List

Introduction

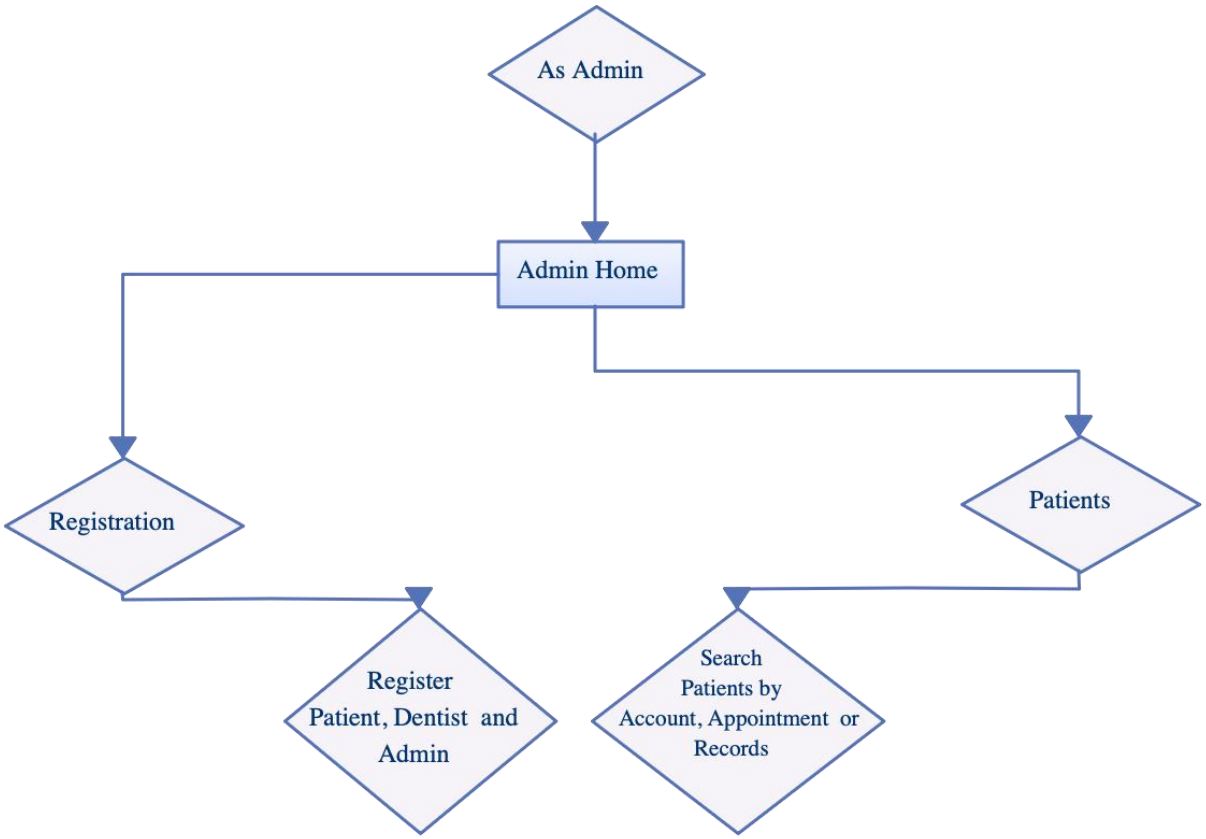
A linked list is a linear data structure where each element, called a node, contains data and a reference (or pointer) to the next node in the sequence. Unlike arrays, linked lists do not require contiguous memory locations and can dynamically grow and shrink, making them flexible and efficient for certain applications.

Key Features of a Linked List:

Dynamic Size: Linked lists can grow and shrink dynamically with the insertion and deletion of nodes, as opposed to arrays that have a fixed size.

Efficient Insertions/Deletions: Insertions and deletions can be more efficient than in arrays, especially when these operations are performed at the beginning of the list or at a known position.

2.2 Block Diagram



CHAPTER 3

MODULES

Patient Management Module Functions

3.1 Create a New Patient Record:

1. **Prompt for Patient Information:** Ask the user for the patient's full name, date of birth, contact information, medical history, insurance details, etc.

2. Instantiate Patient Object: Create a data structure (object or struct) to store the gathered information.

3. Create New Node: Allocate memory for a new doubly linked list node, populate it with the patient's data, and set `next` and `previous` pointers to null**.**

4. Determine Insertion Point: Decide on the sorting criteria (e.g., by patient ID, last name).

5. Insert Node:

* If the list is empty, set the new node as both head and tail.

* Otherwise, traverse the list to find the correct position, update neighboring nodes' pointers, and insert the new node.

3.2 Display Patient Records:

1. **Traversal:** Start at the head of the list and iterate through it using the `next` pointer.
2. **Display Information:** For each node visited, retrieve and display the patient's full name, ID, date of birth, and any other relevant information.
3. **Continue:** Proceed until the end of the list is reached.

3.3 Search for Patient Record:

1. **Prompt for Search Term:**Ask the user for the patient's name or ID to search for.
2. **Traversal:**Start at the head of the list and iterate using `next`.
3. **Comparison:** Compare the search term with each patient's data.
4. **Match Found:** If a match is found, display the patient's full record.
5. **No Match:** If no match is found after reaching the end of the list, inform the user.

3.4 Delete Patient Record:

1. **Prompt for Deletion:** Ask the user for the name or ID of the patient to delete.
2. **Traversal and Search:** Similar to the search function.

3. Match Found:

* If the node to be deleted is the head or tail, update the respective pointers accordingly.

* If it's a middle node, update the pointers of its neighbors.

* Deallocate the memory of the deleted node.

4. No Match: Inform the user if the patient record was not found.

3.5 Edit Patient Record:

1. **Prompt for Edit:** Ask the user for the name or ID of the patient to edit.

2. Traversal and Search: Similar to the search function.

3. Match Found: Allow the user to modify the patient's information within the node.

4. No Match: Inform the user if the patient record was not found.

Important Considerations:

Data Integrity: Implement validation and error handling to ensure accurate patient data.

Privacy: Protect sensitive patient information with encryption and access controls.

Error Handling: Handle cases like invalid input or attempting to delete a non-existent record.

User Interface: Design a user-friendly interface for interacting with the patient management module.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Results

4.1.1 Add Patient:

```
Hospital Management System
1. Add Patient
2. View All Patients
3. Search Patient by ID
4. Delete Patient
5. Exit
Enter your choice: 1
Enter Patient ID: 01
Enter Patient Name: John
Enter Patient Age: 20
Enter Patient Gender: Male
Patient added successfully.
```

```
Enter your choice: 1
Enter Patient ID: 02
Enter Patient Name: Akash
Enter Patient Age: 23
Enter Patient Gender: Male
Patient added successfully.
```

```
Enter your choice: 1
Enter Patient ID: 03
Enter Patient Name: Ram
Enter Patient Age: 20
Enter Patient Gender: Male
Patient added successfully.
```

4.1.2 View All Patients:

```
Hospital Management System
1. Add Patient
2. View All Patients
3. Search Patient by ID
4. Delete Patient
5. Exit
Enter your choice: 2
ID=3, Name=Ram, Age=20, Gender=Male
ID=2, Name=Akash, Age=23, Gender=Male
ID=1, Name=John, Age=20, Gender=Male
```

4.1.3 Search Patient by ID:

```
Hospital Management System
1. Add Patient
2. View All Patients
3. Search Patient by ID
4. Delete Patient
5. Exit
Enter your choice: 3
Enter Patient ID to search: 02
Patient found: ID=2, Name=Akash, Age=23, Gender=Male
```

4.1.4 Delete Patient:

```
Hospital Management System
1. Add Patient
2. View All Patients
3. Search Patient by ID
4. Delete Patient
5. Exit
Enter your choice: 4
Enter Patient ID to delete: 02
Patient deleted successfully.
```

4.2 Discussion

Hospital management systems have numerous benefits that improve patient care and streamline hospital operations. By streamlining patient registration and management processes, these systems reduce errors and improve operational efficiency. Additionally, they enhance the quality of patient experiences by providing a centralized database of patient information and facilitating better communication between healthcare providers. Furthermore, hospital management systems help reduce costs through better management and resource allocation**.**

A comprehensive hospital management system typically includes several key modules that work together to provide a seamless patient experience. The patient registration and management module captures demographic and medical data, assigns unique patient IDs, and manages patient onboarding. The appointment scheduling module schedules appointments based on doctor and equipment availability, sends reminders and notifications to patients and staff, and ensures that patients receive timely care. The electronic health records (EHR) module stores and manages patient medical history, insurance, prescriptions, lab results, and other relevant information

CHAPTER 5
CONCLUSION

In conclusion, hospital management systems are crucial for improving patient care, streamlining hospital operations, and reducing costs. By implementing a comprehensive hospital management system, hospitals can streamline patient registration and management processes, improve operational efficiency, and enhance the quality of patient experiences. The system should include key modules such as patient registration and management, appointment scheduling, electronic health records, facility management, billing and pharmacy management, and reporting and analytics. Before implementation, hospitals should assess their specific requirements, choose a scalable and user-friendly system, ensure integration with existing systems, and provide training and support to staff.

REFERENCES

1. - https://mocdoc.com/blog/a-detailed-view-of-hospital-management-system-hms
2. - https://www.linkedin.com/pulse/10-important-modules-hospital-management-system-hms-drlogy-a7gff
3. - https://www.karexpert.com/blogs/what-is-hospital-management-system/
4. - https://www.slideshare.net/neelampriya31/hospital-management-system-business-case
5. - https://ncert.nic.in/vocational/pdf/keda101.pdf

APPENDIX

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

// Define a structure for patient
typedef struct Patient {
    int id;
    char name[100];
    int age;
    char gender[10];
    struct Patient* next;
} Patient;

// Function prototypes
Patient* createPatient(int id, char* name, int age, char* gender);
void addPatient(Patient** head, int id, char* name, int age, char* gender);
void displayPatients(Patient* head);
Patient* searchPatient(Patient* head, int id);
void deletePatient(Patient** head, int id);

int main() {
    Patient* head = NULL;
    int choice, id, age;
    char name[100], gender[10];
```

```
while (1) {
    printf("\nHospital Management System\n");
    printf("1. Add Patient\n");
    printf("2. View All Patients\n");
    printf("3. Search Patient by ID\n");
    printf("4. Delete Patient\n");
    printf("5. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            printf("Enter Patient ID: ");
            scanf("%d", &id);
            printf("Enter Patient Name: ");
            scanf("%s", name);
            printf("Enter Patient Age: ");
            scanf("%d", &age);
            printf("Enter Patient Gender: ");
            scanf("%s", gender);
            addPatient(&head, id, name, age, gender);
            break;
        case 2:
            displayPatients(head);
            break;
        case 3:
            printf("Enter Patient ID to search: ");
            scanf("%d", &id);
            Patient* patient = searchPatient(head, id);
```

```
            if (patient) {
                printf("Patient found: ID=%d, Name=%s, Age=%d, Gender=%s\n",
patient->id, patient->name, patient->age, patient->gender);
            } else {
                printf("Patient with ID=%d not found.\n", id);
            }
            break;
        case 4:
            printf("Enter Patient ID to delete: ");
            scanf("%d", &id);
            deletePatient(&head, id);
            break;
        case 5:
            exit(0);
        default:
            printf("Invalid choice!\n");
    }
  }
  return 0;
}

Patient* createPatient(int id, char* name, int age, char* gender) {
  Patient* newPatient = (Patient*)malloc(sizeof(Patient));
  newPatient->id = id;
  strcpy(newPatient->name, name);
  newPatient->age = age;
  strcpy(newPatient->gender, gender);
  newPatient->next = NULL;
  return newPatient;
```

```
}

void addPatient(Patient** head, int id, char* name, int age, char* gender) {
    Patient* newPatient = createPatient(id, name, age, gender);
    newPatient->next = *head;
    *head = newPatient;
    printf("Patient added successfully.\n");
}

void displayPatients(Patient* head) {
    Patient* current = head;
    if (current == NULL) {
        printf("No patients found.\n");
        return;
    }
    while (current != NULL) {
        printf("ID=%d, Name=%s, Age=%d, Gender=%s\n", current->id, current-
>name, current->age, current->gender);
        current = current->next;
    }
}

Patient* searchPatient(Patient* head, int id) {
    Patient* current = head;
    while (current != NULL) {
        if (current->id == id) {
            return current;
        }
        current = current->next;
```

```
    }
    return NULL;
}

void deletePatient(Patient** head, int id) {
    Patient* current = *head;
    Patient* previous = NULL;

    if (current != NULL && current->id == id) {
        *head = current->next;
        free(current);
        printf("Patient deleted successfully.\n");
        return;
    }

    while (current != NULL && current->id != id) {
        previous = current;
        current = current->next;
    }
```