**REPORT**

**PROJECT NAME: Hospital management system**

**Course code**: CSE135

**Course title:** Data Structure Lab

Submitted to,

Teachers Name: Md Jakaria Zobair & Sheikh Tonmoy

Designation: Lecturer

Daffodil International University

Submitted by,

Tamanna Akter (221-15-5298)

Binota Roy (221-15-5337)

Nahida Akter (221-15-4711)

Section: 61\_M

Department of CSE

**Date: 20 September 2023**

## INDEX:

The Project File contains the following contents-

1. Introduction …………………………………….. 3
2. Objectives ……………………………………….. 3
3. Features …………………………………………. 3
4. Data Structures………………………………….. 4
5. Functions………………………………………… 5
6. Limitations of the Project………………………. 6
7. Future Scope…………………………………….. 7
8. Challenges and Lessons Learned………………… 7
9. Technologies Used……………………………….. 8
10. Project Overview/Summary……………………. 8
11. Key words……………………………………….. 8
12. Conclusion………………………………………. 8
13. References………………………………………... 9
14. Workflow diagram……………………………….. 9

Title: Hospital Management System

**Introduction:**

The Hospital Management System project in C is designed to provide a comprehensive solution for managing patient and doctor records within a hospital setting. This system uses data structures to efficiently organize and manipulate data, offering various functionalities like adding, deleting, searching, and updating patient and doctor information. Below is a detailed report of the project.

**Objectives:**

The primary objectives of the project are as follows:

Develop a system to manage patient and doctor records effectively.

Implement features for adding, deleting, searching, and updating records.

Demonstrate the use of data structures (linked lists) for organizing and accessing data.

Create a user-friendly interface for easy interaction and navigation.

Ensure efficient memory management and error handling.

**Features:**

**Patient Management**:

* Add a new patient record.
* Delete a patient record.
* Update patient information.
* Search for patients by ID.

**Doctor Management:**

* Add a new doctor record.
* Delete a doctor record.
* Update doctor information.
* Search for doctors by ID.

**User-Friendly Interface:**

The project provides a user-friendly menu-driven interface for easy navigation and operation.

**Data Structures:**

**Structures:** The project utilizes two primary data structures: Patient and Doctor structures. These structures are defined to store essential information about patients and doctors. This project employs the following data structures:

**Arrays:** To store and manage patient and doctor records.

**Linked Lists:** To handle appointment scheduling and billing information. Linked lists are used to manage records, and a hash table-like approach is employed for efficient searching.

**Patient Struct (Node):** Stores patient information such as ID, name, stage, and phone number. Contains a pointer to the next patient in the linked list.

Patient Structure:

**Id:** Unique identifier for each patient.

**name:** Name of the patient.

**Stage:** Medical condition or stage of the patient.

**phone:** Contact number of the patient.

**next:** Pointer to the next patient in the linked list.

**Doctor Struct (Node):**

Stores doctor information like ID, name, and specialization.

Contains a pointer to the next doctor in the linked list.

Doctor Structure:

**Id:** Unique identifier for each doctor.

**name:** Name of the doctor.

**specialization:** Medical specialization of the doctor.

**next:** Pointer to the next doctor in the linked list.

**Functions**

**createPatient**(int Id, int phone, const char name, const char Stage):

Creates a new patient struct and initializes its attributes.

Returns a pointer to the newly created patient.

**createDoctor(**int Id, const char name, const char specialization):

Creates a new doctor struct and initializes its attributes.

Returns a pointer to the newly created doctor.

**addPatient**(int Id, int phone, const char name, const char Stage):

Inserts a new patient record into the linked list.

Handles the creation of the first patient node if the list is empty.

**addDoctor**(int Id, const char name, const char specialization):

Inserts a new doctor record into the linked list.

Handles the creation of the first doctor node if the list is empty.

**deleteNode**(int Id, int isPatient):

Deletes a patient or doctor record based on the provided ID.

Handles the removal of the record from the linked list.

**searchRecord**(int Id, int isPatient):

Searches for a patient or doctor record based on the provided ID.

Displays the record if found; otherwise, it indicates that the record was not found.

**updateRecord**(int Id, int phone, int isPatient, const char newName, const char newInfo):

Updates patient or doctor information based on the provided ID.

Allows for modifications of name, phone number, and condition/specialization.

**displayPatients():**

Displays all patient records in the linked list.

**displayDoctors():**

Displays all doctor records in the linked list.

**Exit Option:**

Users can exit the system when they are done with their tasks.

Implementation Details.

**Main Function:**

The main function serves as the entry point to the program.

It displays a menu-driven interface where users can select various options to perform actions within the system.

Users can continue using the system until they choose to exit.

**Limitations of the Project:**

While the Hospital Management System provides essential features for record management, it has certain limitations:

**Lack of security features:** The system does not implement user authentication, making it vulnerable to unauthorized access.

**Lack of persistent data storage:** The project does not incorporate a database, so records are lost upon program termination.

**Limited error handling:** While basic error handling is in place, the system may not handle all possible edge cases.

**Limited functionality**: The project focuses primarily on patient and doctor record management and does not include advanced features like appointment scheduling, billing, or reporting

**Future Scope:**

The project can be extended and improved in several ways in the future:

**Database Integration:** Implement a database system to ensure persistent storage of patient and doctor records.

**Enhanced User Interface**: Develop a graphical user interface (GUI) for improved usability and aesthetics.

**Additional Functionalities:** Integrate features like appointment scheduling, billing, and patient history for a comprehensive hospital management system.

**Security Enhancements**: Implement authentication mechanisms to secure sensitive patient and doctor data.

**Reporting:** Develop advanced reporting capabilities to generate statistics and insights about hospital operations.

**User Interface Improvement:** Enhance the user interface to make it more user-friendly and intuitive.

**Web or Mobile Application:** Convert the system into a web or mobile application for remote access.

**Challenges and Lessons Learned**

The development of the Hospital Management System presented several challenges and provided valuable learning experiences:

**Data Structure Implementation:** Designing and implementing data structures (linked lists and hash table-like structures) required a deep understanding of data organization and manipulation.

**Dynamic Memory Management**: Proper memory allocation and deallocation were essential to prevent memory leaks and ensure efficient resource usage.

**User Interface Design:** Creating an intuitive menu-driven interface and handling user input effectively posed design and implementation challenges.

**Error Handling:** Implementing robust error handling and validation for user inputs was a crucial aspect of the project.

**Testing and Debugging:** Thorough testing and debugging were required to ensure the correctness and reliability of the system.

**Technologies Used**

Programming Language: C

Integrated Development Environment (IDE): Code: Blocks, XCode

**Project Overview/Summary:**

The "Hospital Management System" is a software application developed in the C programming language. It aims to streamline the management of patient and doctor records within a hospital. It allows users to add, delete, search, and update patient and doctor information, as well as display records. The project employs data structures like linked lists and dynamic memory allocation to efficiently manage the data.

**Key words**

Addition, deletion, search, and update of patient and doctor records. Menu-driven interface for user interactions. Efficient organization of records using linked lists. Basic error handling for input validation.

**Conclusion:**

The Hospital Management System project in C successfully demonstrates the use of data structures and efficient algorithms to manage patient and doctor records in a hospital environment. It offers a user-friendly interface and essential functionalities required for managing healthcare information. This project can be further extended and enhanced to include additional features and security mechanisms to meet the specific needs of a real-world hospital management system.

The project effectively employs data structures like linked lists to manage patient and doctor records.

**References**

The project was developed using knowledge and understanding of C programming, data structures, and software engineering principles. Online tutorials, C programming books, and documentation were referenced for guidance and best practices.

**Workflow diagram:**

**Display Welcome Message**

**Display Main Menu Options:**

**1. Insert a Patient**

**2. Insert a doctor**

**3. Delete a Record**

**4. Search for a Record**

**5. Update Record**

**6. Display Patient Records**

**7. Display Doctor Records**

**8. Exit**

**User Chooses an Option**

**Perform Selected Action**

**Exit Program**

**If Insert Patient Chosen**

**Collect Patient Information**

**Create Patient Record**

**Add Patient Record to Patient Linked List**

**Display Confirmation: Patient Added**