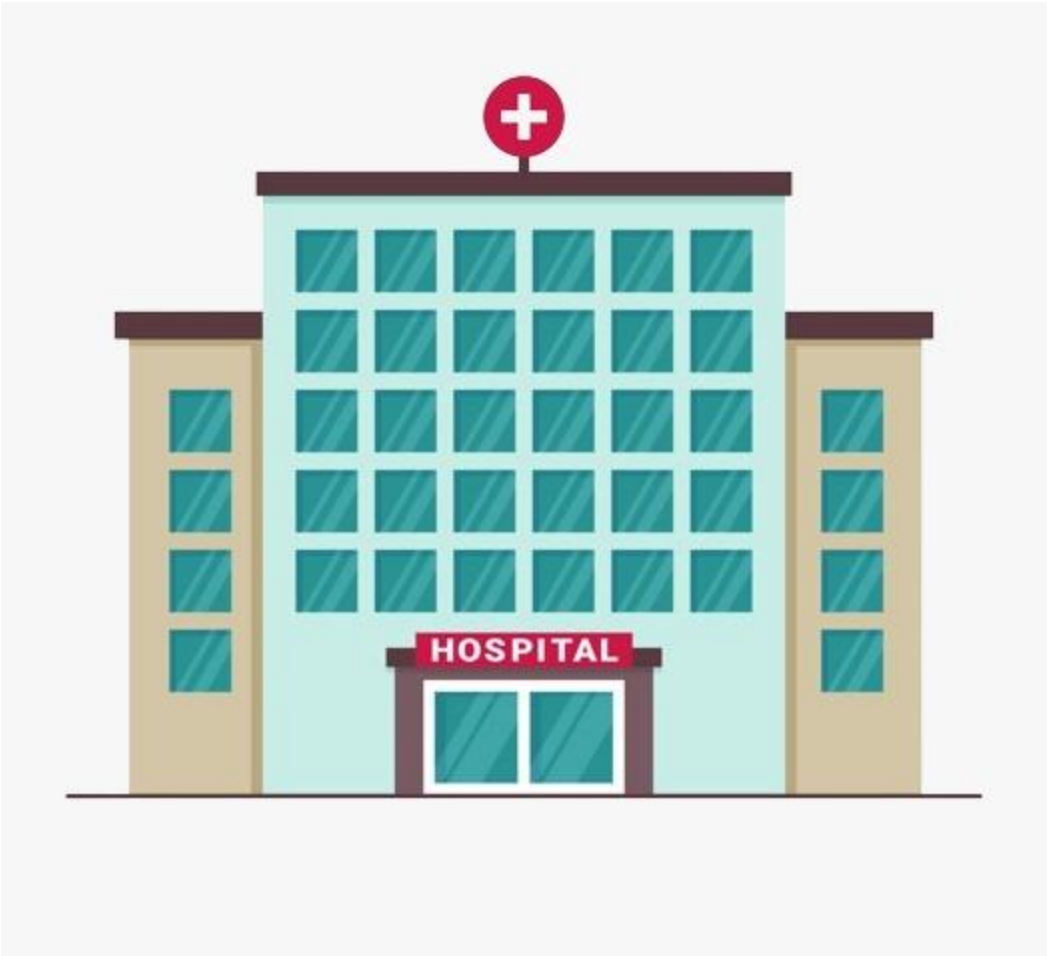


HOSPITAL MANAGEMENT SYSTEM

DATA STRUCTURE AND ALGORITHM

PROJECT REPORT:

GROUP MEMBERS ARE:

ASIFA SIRAJ(CT-22070)
SUMIYA FARHAT(CT-22054)
FILZA TANVEER(CT-22057)

PROJECT BACKGROUND:

OBJECTIVE:

The project aims to create a menu-driven application for managing patient and doctor records in a medical setting. It utilizes different classes for administrators, patients, and doctors to handle various functionalities such as adding, editing, searching, and displaying records.

FEATURES:

- **Administrator Functionality:**
  Includes managing patient and doctor records, ensuring access through a password, and navigating through sub-menus.

- **Patient Functionality:**
  Allows patients to search for their records and return to the main menu.

- **Doctor Functionality:**
  Enables doctors to search for patient and doctor records, display all patient records, and return to the main menu.

- **User Authentication:**
  Users are required to enter passwords for administrative and patient functionalities.

PROS AND CONS:

The pros and cons of the project are:

PROS:

- **Modular Structure:**
  The code is organized into classes and functions, promoting modularity and code reusability.

- **User-Friendly Interface:**
  The menu-driven interface makes it user-friendly and easy to navigate.

- **Data Validation:**
  Incorporates input validation for better user experience and data integrity.

- **Clear Flow:**
  The program flow is well-structured with distinct functionalities for different user roles.

CONS:

- **Limited Error Handling:**
  Error handling is somewhat limited, and the program might not gracefully handle unexpected inputs.

- **Code Redundancy:**
  Some code blocks, such as clearing the screen, are repeated, leading to redundancy.

- **No Comments:**

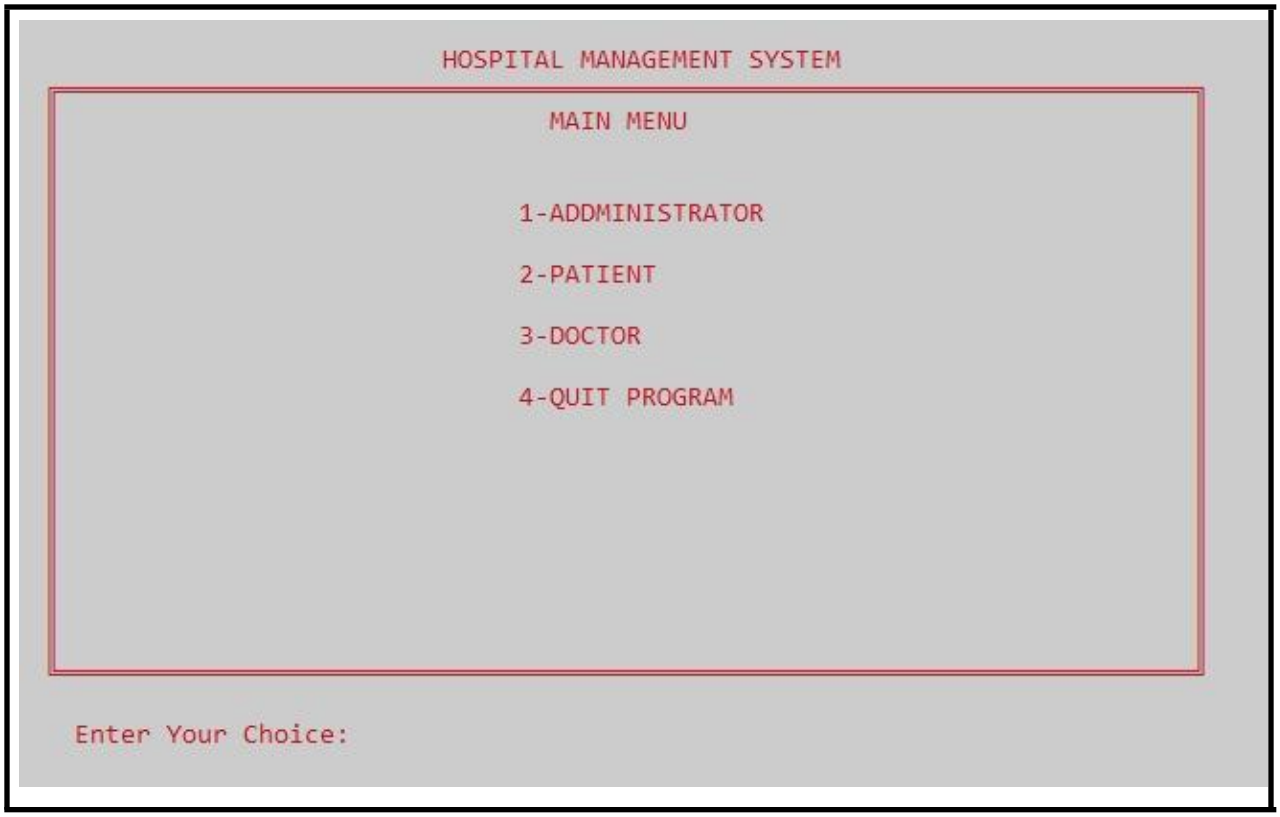
Lack of comments might make it challenging for others to understand the code's logic and purpose.

OUTPUTS:

- **Loading Screen:**
  Displays a loading screen using ASCII characters.

- **Menu Screens:**
  Main menu, sub-menus, and specific screens for adding, editing, searching, and displaying records.

MAIN MENU:



ASKING ADMIN TO ENTER PASSWORD:

```
HOSPITAL MANAGEMENT SYSTEM
ADMIN BLOCK

Kindly Insert The Administrator Password :

admin
```

ADMIN BLOCK:

```
HOSPITAL MANAGEMENT SYSTEM
MAIN MENU

1-Patient Record

2-Doctor Records

3-Main Menu
```

MANAGING RECORD OF PATIENTS
SAME FOR DOCTORS:

```
HOSPITAL MANAGEMENT SYSTEM

MAIN MENU

1-Add Record

2-Edit Record

3-Search Record

4-Delete Record

5-Display Record

6-Main Menu

Enter Your Choice:
```

- **User Prompts:**
  Requests user input for various actions.

ENTERING DATA OF PATIENT:

```
DATA ENTRY : 1

Enter Patient Name: Sarah

Enter Patient Father Name: Johnson

Enter Patient ID: 1

Enter Patient Birth Year: 2000

Enter Ward Assigned: 1a

Enter Days of Stay: 15

Enter Doctor ID: 1
```

ENTERING DATA OF DOCTOR:

```
DATA ENTRY : 1

Enter Doctor Name: Dr.Jennifer

Enter Doctor Credential: MBBS

Enter Doctor ID: 1

Enter Doctor Birth Year: 1994
```

- **Record Management:**
  Outputs messages for adding, editing, searching, and deleting records.

DISPLAYING DATA OF PATIENT:

```
                              PATIENT RECORD

NAME     FATHER NAME     ID  YEAR    WARD    STAY   DOCTOR ID

Sarah       Johnson     1      2000    1a      14   1


                               Finish
Press any key to continue . . .
```

ENTERING DATA OF DOCTOR:

```
                              DOCTOR RECORD

NAME        CREDENTIALS       ID          YEAR

Dr.Jennifer MBBS            1          1994


                               Finish
Press any key to continue . . .
```

DELETING DATA:

```
Enter The Patient ID In Order To Delete :
1

                              PATIENT RECORD

 NAME     FATHER NAME     ID  YEAR    WARD    STAY   DOCTOR ID


                                Finish

Press any key to continue . . .
```

PATIENT CLASS BLOCK:

```
                    HOSPITAL MANAGEMENT SYSTEM

                           MAIN MENU


                             Michael

                       1-See  Your Record

                       4-Main Menu
```

DOCTOR CLASS BLOCK:

```
HOSPITAL MANAGEMENT SYSTEM
MAIN MENU

Dr.Jeniffer

1-See  Your Record

2-Search Patient

3-All Patient

4-Main Menu
```

- **Error Messages:**
  Displays messages for invalid inputs and other errors.

EXPERIENCE AND DIFFICULTIES:

- **Learning:**
  Overall, the experience was amazing .we learnt a lot of new things such as the better use of linked list and how to manage various functions all together.

- **Representing outputs:**
  We faced difficulty in representing the outputs.

- **Input validations:**
  Ensuring proper input validation can be challenging, especially when dealing with user inputs.

- **Link between patient and doctor:**
  The most difficult situation was how to access the Allpatient function and how to make the link between patient and the doctor.

- **Redundancy:**
  Addressing code redundancy by refactoring and creating more efficient code.

- **User Authentication:**
  Implementing robust user authentication mechanisms might be complex and crucial for security.

- **Error Handling:**

Enhancing error handling and validations to provide more informative and user-friendly error messages.

DATA STRUCTURE:

The primary data structures used in the project include **Linked List**. A linked list is a fundamental data structure in computer science that consists of a sequence of elements, where each element points to the next one in the sequence. Unlike arrays, linked lists do not require contiguous memory locations, providing flexibility in terms of memory allocation. The basic building block of a linked list is a node, and each node contains two components: the data or payload, and a reference or link to the next node in the sequence.
How it is used in project: In our code, linked list data structures are employed to manage doctor records within the admin2 class, which is an extension of the admin class. The doc_node structure is defined to represent a node in the linked list, containing attributes such as docname, doc_credential, docid, and docdob. Additionally, it holds a pointer next2 that points to the next node in the linked list. The admin2 class incorporates pointers head2 and tail2 to respectively indicate the head and tail of the linked list, and an integer count2 keeps track of the number of nodes in the list.

Within the admin2 class, the isEmpty2() function checks whether the linked list is empty by examining if the head is NULL. The docdata function is responsible for adding a new doctor record to the linked list. A new doc_node is dynamically created, populated with data for the new doctor, and appended to the linked list. The docedit function facilitates the editing of an existing doctor record within the linked list. It involves creating a temporary doc_node, obtaining input for the modified doctor details, and updating the relevant attributes of the target node.

Overall, the linked list in this context provides a dynamic and efficient structure for managing doctor records, enabling functionalities such as addition, editing, and traversal of records within the admin2 class.

CONCLUSION:

The project showcases a basic yet functional implementation for managing medical records. Enhancements in error handling, user authentication, and code optimization could improve its robustness and maintainability.