Patient Management System

GitHub: https://github.com/Franklin-Festo-Francis/Banking-Managment-System.git

Introduction

The **Patient Management System** is a console-based application developed in **C++** that helps manage patient records such as name, age, disease, phone number, and address. It provides a simple interface for performing **CRUD (Create, Read, Update, Delete)** operations and includes basic input validation and user authentication for secure access.

Objectives

To design a small-scale patient record management system.

To implement structured programming concepts like functions, arrays, loops, and conditionals.

To ensure data accuracy through input validation and authentication.

System Features

Login Verification: Only authorized users (Name: *Varun*, Password: *123#*) can access the system.

Add Patient Record: Add new patient details with checks for valid input.

Search Record: Find patient data by Name, ID, or Disease.

Update Record: Modify existing patient information.

Delete Record: Remove records based on patient ID.

Display Records: View all stored patient information.

Technical Overview

The program uses a **structure array** to store patient data temporarily. Each patient record includes fields such as **ID, Name, Age, Address, Disease, Phone,** and **Doctor Assigned**.
Validation ensures that names contain only alphabets, while phone and age fields contain only numeric values. The interface uses system("color"), Sleep(), and formatted text output to enhance user experience.

Results and Limitations

All major functions perform correctly, allowing efficient patient data management. However, since the program uses only **runtime storage (arrays)**, all data is lost once the program exits. Additionally, it supports a maximum of **50 records** and only one **admin account**.

Conclusion

The project successfully demonstrates the use of **C++ structures, loops, functions, and conditional logic** to create a functional Patient Management System. With features like authentication, input validation, and CRUD operations, it provides a strong foundation for understanding practical data handling in C++.
Future improvements could include **file handling for permanent storage**, **multiple user logins**, and a **graphical user interface (GUI)** for enhanced usability and better accessibility.