

PROJECT REPORT

Computer Programming Lab (CSL-113)



PROJECT TITLE: HOSPITAL MANAGEMENT SYSTEM

BS (AI)-1(A)

Group Members

Name	Enrollment
1. MAIRA MUQADAS	02-136232-036
2. ALISHBA TAHIR	02-136232-045
3. ABDUL MATEEN	02-136232-048

Submitted to:

Miss Zahida Naz

BAHRIA UNIVERSITY KARACHI CAMPUS

Department of Computer Science

TABLE OF CONTENTS

ABSTRACT	3
INTRODUCTION	4
PROBLEM STATEMENT	4
METHODOLOGY	4
PROJECT SCOPE	6
FUTURE DEVELOPMENT	6
CONCLUSION	7
CODE	9
OUTPUT	26

ABSTRACT

The Hospital Management System (HMS) project, developed using C++ programming language, is a comprehensive solution designed to enhance the efficiency of healthcare institutions. The system incorporates modules such as patient management, appointment scheduling, billing, inventory management, and electronic health records. Leveraging C++'s object-oriented features, the project ensures modularity, code reusability, and maintainability. The system prioritizes user-friendly interfaces, data security, and scalability. The implementation aims to modernize hospital management practices, contributing to improved healthcare delivery and patient outcomes.

INTRODUCTION

The Hospital Management System (HMS) presented in this C++ code is a comprehensive software solution designed to streamline and enhance the operational efficiency of healthcare institutions. The code embodies the object-oriented programming paradigm, utilizing structures to represent key entities such as Patients, Doctors, Wards, and Appointments. This approach enables the encapsulation of relevant data and functionalities, fostering code modularity and maintainability.

The system facilitates the addition of doctors and wards, admission of patients to specified wards, scheduling of appointments between patients and doctors, and the display of current appointments and ward occupancies. Leveraging vectors for dynamic data storage, the code showcases the flexibility and scalability of the C++ programming language in managing complex healthcare scenarios.

The example usage within the 'main' function demonstrates the system's capabilities, including doctor and ward addition, patient admission, appointment scheduling, and the subsequent display of appointments and ward occupancies. The Hospital Management System, as exemplified in this code, serves as a foundation for developing robust and user-friendly solutions tailored to the evolving needs of healthcare management practices.

PROBLEM STATEMENT

In the realm of healthcare, the manual handling of administrative tasks, patient records, and staff management poses a significant challenge for hospitals. To address this, there is a need for an advanced Hospital Management System (HMS). The system should offer secure login with role-based access, efficient staff management, streamlined patient registration, appointment scheduling, medical record maintenance, reporting, and analytics. Additionally, it must ensure a user-friendly interface, data security, and adherence to privacy regulations. The goal is to automate processes, enhance operational efficiency, and improve overall healthcare service quality within the hospital.

METHODOLOGY

1. Introduction:

- Briefly introduce the concept of the Hospital Management System (HMS) and its significance in streamlining hospital operations.

2. Objectives:

- Clearly outline the goals of the project, such as efficient user authentication, role-based access control, patient registration, and medical record management.

3. System Architecture:

- Describe the high-level architecture of the system, emphasizing the separation of concerns through role-based structures for admin, doctor, and nurse functionalities.

4. Technologies Used:

- List and briefly explain the programming languages and libraries used (e.g., C++, file handling, and basic input/output operations).

5. User Authentication:

- Discuss the implementation of secure user authentication, including the storage of login credentials in a file, and the verification process during user login.

6. Admin Dashboards:

- Present the structure and functionalities of individual dashboards for the admin, doctor, and nurse, outlining the available options and their corresponding actions.

7. User Registration:

- Detail the steps for adding new users, both administrators and medical staff, including the collection and storage of relevant information.

8. Patient Management:

- Describe patient-related functionalities, covering patient registration, viewing patient details, and updating patient information.

9. Medical Records:

- Discuss the system's ability to manage medical records, including adding records, viewing records, and searching for specific patient records.

10. Doctor's Orders:

- Explain the functionality allowing doctors to input and view patient-specific orders, enhancing communication between medical staff.

11. File Handling:

- Elaborate on the use of file handling techniques to store and manipulate data, such as login credentials, user biodata, patient records, and doctor's orders.

12. Error Handling:

- Briefly mention the implementation of error-handling mechanisms, like validating input data, checking file access and addressing potential user errors.

13. Testing:

- Discuss the testing process, including unit testing for individual functionalities, integration testing for combined functionalities, and user acceptance testing.

14. Results and Impact:

- Reflect on the successful implementation of the system, highlighting its impact on hospital management, staff efficiency, and patient care.

15. Conclusion:

- Summarize the key achievements of the project, emphasizing the successful implementation of a functional Hospital Management System.

16. Future Enhancements:

- Suggest potential improvements or features that could be added to enhance the system's capabilities in the future.

17. References:

- Cite any external resources or references consulted during the development of the project.

This methodology provides a concise overview of the Hospital Management System project, covering its objectives, technical aspects, implementation details, testing procedures, and potential future enhancements.

PROJECT SCOPE

The Hospital Management System (HMS) project is a comprehensive software solution designed to enhance the efficiency of healthcare institutions, encompassing key modules for doctors, nurses and patients. The scope of this project includes the development of functionalities that seamlessly integrate into hospital workflows, facilitating the management of various entities within the healthcare ecosystem.

FUTURE DEVELOPMENT

Future developments for the Hospital Management System (HMS) project could introduce additional features and improvements to further enhance its functionality and utility in the healthcare environment. Some potential avenues for development include:

1. Billing and Invoicing:

- Implement a billing module to streamline the billing process for medical services.
- Generate invoices for consultations, procedures, and medications.
- Include support for insurance claims and reimbursement tracking.

2. Pharmacy Integration:

- Connect the HMS with the pharmacy to automate medication dispensing and inventory management.
- Facilitate electronic prescriptions and real-time updates on medication availability.

3. Ward appointment:

- Admitting patients to specific wards, and assigning doctors and nurses to respective patients.
- Notifying nurses on duty for timely medication.

4. Appointment Reminders:

- Implement appointment reminder functionalities through SMS or email to reduce no-shows and enhance patient adherence to treatment plans.

5. Expandable Data Model:

- Design the system to accommodate future expansion by incorporating a flexible data model that can adapt to evolving healthcare requirements.

6. Collaboration Features:

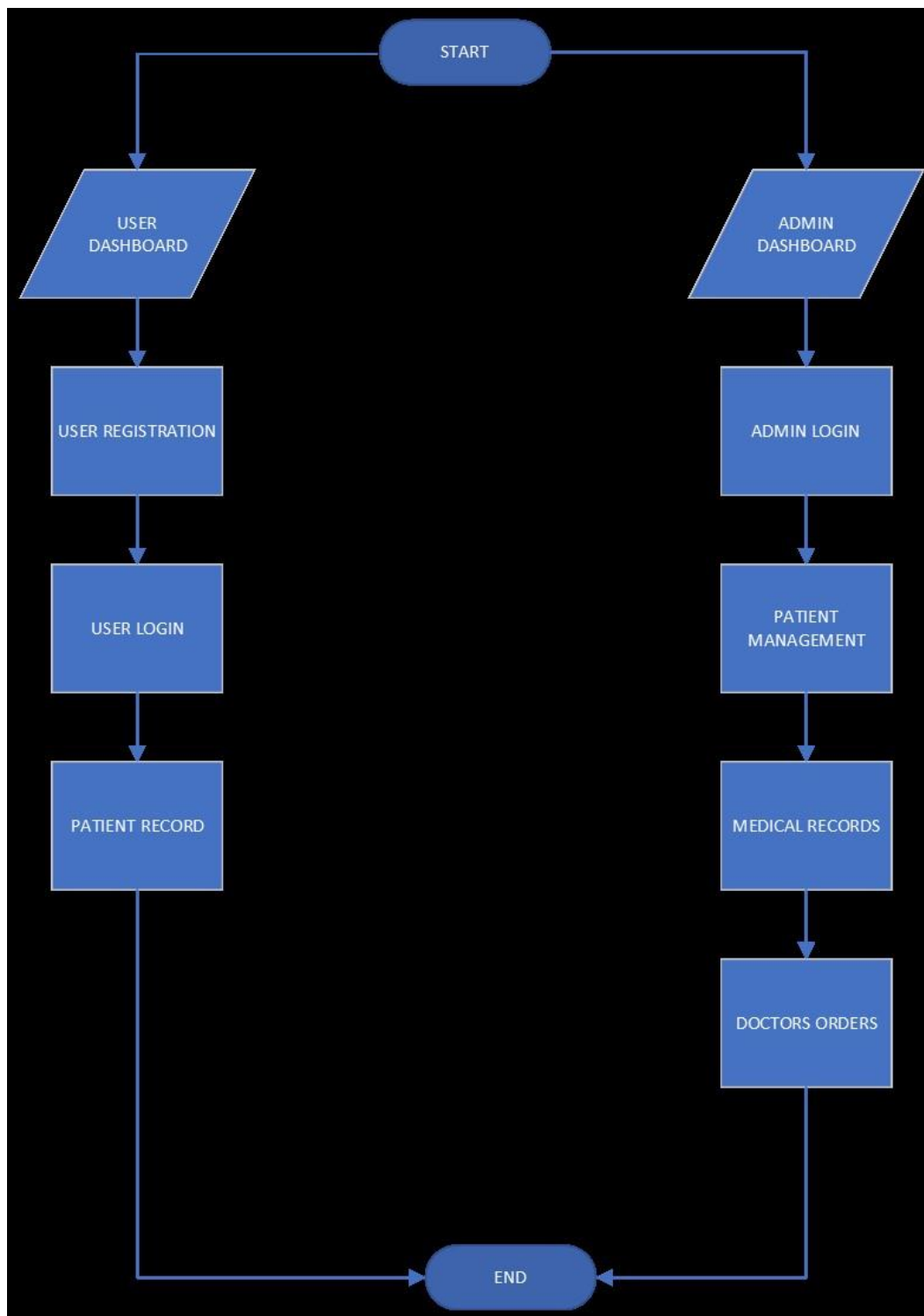
- Introduce collaboration tools to facilitate communication and coordination among healthcare professionals within the system.
- Enable secure sharing of patient information and case discussions.

These future developments aim to transform the Hospital Management System into a comprehensive solution that not only manages day-to-day operations efficiently but also addresses emerging needs in the dynamic healthcare landscape. Regular updates and continuous improvement will be essential to keep the system relevant and valuable for healthcare providers.

CONCLUSION

In conclusion, the Hospital Management System project, employing C++ structs and functions, stands as a foundational solution for optimizing healthcare operations. Its structured modules for patients, doctors, and nurses, coupled with robust security measures, set the stage for improved patient care and streamlined processes. Future developments, including billing, analytics, and mobile integration, promise to enhance the system's capabilities, ensuring its relevance and adaptability in an ever-evolving healthcare landscape. The commitment to systematic testing and documentation reinforces the reliability and usability of the system, making it an asset for healthcare professionals and administrators alike.

FLOWCHART



CODE

```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
using namespace std;
struct Patient {
    string namep;
    string cnicp;
    int agep;
    int idp;
    long int phnumber;
    int vitals;
    string pres;
    string com;
};
int ID;
struct staffBio {
    string name, cnic;
    int age;
    long int phoneNo;
};
struct staffLogin {
    string username;
    string password;
};

bool userLogin(const string& role, const string& username, const string& password);
bool isUsernameTaken(const string& username);
void adminDashboard(const string& username);
void doctorDashboard(const string& username);
void nurseDashboard(const string& username);
void addNewUser(const string& role);
void removeNurse();
void removeDoctor();
void removeUser(const string& role, const string& username);
void adminRegistration();
void viewStaff(const string& fileName, const string& role);
void add_patientrecord();
void register_patient();
void view_patient();
void update_patient();
void remove_patient();
void register_patient();
void add_doctor_orders();
void view_doctor_orders();
void readMedicalRecord();
bool containsNumeric(const string& str);
```

```

void searchMedicalRecord();

int main() {
    int choice;
    string Role, Username, Password;
    bool cond = false;

    do {

        cout << "Welcome to the Hospital Management System\n";
        cout << "1. Admin Login\n";
        cout << "2. Doctor Login\n";
        cout << "3. Nurse Login\n";
        cout << "4. Exit\n";
        cout << "Enter your choice: ";
        // Check if the input is an integer
        if (!(cin >> choice)) {
            // If not an integer, clear the error state and consume invalid input
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Invalid input. Please enter a valid numeric choice.\n";
            continue;
        }

        switch (choice) {
            case 1:
                Role = "admin";
                break;
            case 2:
                Role = "doctor";
                break;
            case 3:
                Role = "nurse";
                break;
            case 4:
                cout << "Exiting...\n";
                cond = true;
                return 0;
            default:
                cout << "Invalid choice. Please try again.\n";
                continue;
        }
        cout << "Enter id > ";
        cin.ignore();
        getline(cin, Username);
        cout << "Enter password > ";
        getline(cin, Password);
        if (userLogin(Role, Username, Password))
        {
            if (Role == "admin")

```

```

        {
            adminDashboard(Username);
        }
        else if (Role == "doctor")
        {
            doctorDashboard(Username);
        }
        else if (Role == "nurse")
        {
            nurseDashboard(Username);
        }
    }
    else
    {
        cout << "wrong credentials!";
    }
} while (true);

return 0;
}

bool userLogin(const string& role, const string& username, const string& password) {
    ifstream credentialsFile("logincredentials.txt");
    if (!credentialsFile.is_open()) {
        cerr << "Unable to open credentials file." << endl;
        return false;
    }

    string storedRole, storedUsername, storedPassword;
    while (getline(credentialsFile, storedRole, ':') &&
        getline(credentialsFile, storedUsername, ':') &&
        getline(credentialsFile, storedPassword)) {
        if (role == storedRole && username == storedUsername && password == storedPassword)
        {
            credentialsFile.close();
            return true; // Authentication successful
        }
    }

    credentialsFile.close();
    return false; // Authentication failed
}

void adminDashboard(const string& username) {
    cout << "Welcome Admin: " << username << endl;
    int choice;
    bool cond = false;

    do {
        cout << "Admin Menu:" << endl;

```

```

cout << "1. Add Doctor" << endl;
cout << "2. Add Nurse" << endl;
cout << "3. Remove Doctor" << endl;
cout << "4. Remove Nurse" << endl;
cout << "5. View Doctors" << endl;
cout << "6. View Nurses" << endl;
cout << "7. Register Patient" << endl;
cout << "8. Display All Patients" << endl;
cout << "9. Logout" << endl;
cout << "Enter your choice: ";
if (!(cin >> choice)) {
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    cout << "Invalid input. Please enter a valid numeric choice.\n";
    continue;
}

switch (choice) {
case 1:
    addNewUser("doctor");
    break;

case 2:
    addNewUser("nurse");
    break;

case 3:
    removeDoctor();
    break;

case 4:
    removeNurse();
    break;

case 5:
    viewStaff("doctorBiodata.txt", "doctor");
    break;

case 6:
    viewStaff("nurseBiodata.txt", "nurse");
    break;

case 7:
    register_patient();
    break;
case 8:
    view_patient();
    break;
case 9:
    cout << "Exiting Admin Dashboard..." << endl;

```

```

        cond = true;
        break;

    default:
        cout << "Invalid choice. Please enter a valid option." << endl;
        continue;
    }
} while (!cond);
}

void doctorDashboard(const string& username)
{
    int op;
    bool cond = false;
    do {
        cout << "Welcome Doctor: " << username << endl;
        cout << "1. Add Patient's Medical record" << endl;
        cout << "2. View Patient Data" << endl;
        cout << "3. Display Patients Medical record" << endl;
        cout << "4. Search for a Patient's Medical record" << endl;
        cout << "5. Exit.." << endl;
        if (!(cin >> op)) {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Invalid input. Please enter a valid numeric choice.\n";
            continue;
        }

        switch (op) {

            case 1:
                add_patientrecord();
                break;
            case 2:
                view_patient();
                break;
            case 3:
                readMedicalRecord();
                break;
            case 4:
                searchMedicalRecord();
                break;
            case 5:
                cond = true;
                break;
            default:
                cout << "Invalid input" << endl;
                continue;
        }

    } while (!cond);
}

```

```

}

void nurseDashboard(const string& username) {
    int select;
    bool cond = false;
    do {
        cout << "Welcome Nurse: " << username << endl;
        cout << "1. View Patient Information" << endl;
        cout << "2.Update Patient Medical Information" << endl;
        cout << "3. Add Doctor's Orders" << endl;
        cout << "4. View Doctor's Orders" << endl;
        cout << "5. Exit" << endl;
        cout << "Enter your choice: ";
        if (!(cin >> select)) {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Invalid input. Please enter a valid numeric choice.\n";
            continue;
        }

        switch (select) {
            case 1:
                view_patient();
                break;
            case 2:
                update_patient();
                break;
            case 3:
                add_doctor_orders();
                break;
            case 4:
                view_doctor_orders();
                break;
            case 5:
                cout << "Exiting Nurse Dashboard..." << endl;
                cond = true;
                break;
            default:
                cout << "Invalid selection, try again" << endl;
                break;
        }
    } while (!cond);
}

void adminRegistration() {
}

void addNewUser(const string& role) {
    staffBio newstaffBio;
    staffLogin newstaffLogin;
}

```

```

do {
    cout << "Enter new " << role << "'s username: ";
    cin >> newstaffLogin.username;

    if (isUsernameTaken(newstaffLogin.username)) {
        cout << "Username already exists. Try a different username." << endl;
    }
    else {
        break;
    }
} while (true);

cout << "Enter new " << role << "'s password: ";
cin >> newstaffLogin.password;

cout << "Enter new " << role << "'s name: ";
cin.ignore();
getline(cin, newstaffBio.name);

while (containsNumeric(newstaffBio.name)) {
    cout << "Name should not contain numbers. Enter " << role << "'s name again: ";
    getline(cin, newstaffBio.name);
}

cout << "Enter new " << role << "'s age: ";
while (!(cin >> newstaffBio.age) || newstaffBio.age < 0) {
    cout << "Invalid input. Please enter a valid age: ";
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
}

cout << "Enter new " << role << "'s phone number: ";
while (!(cin >> newstaffBio.phoneNo) || newstaffBio.phoneNo < 0) {
    cout << "Invalid input. Please enter a valid phone number: ";
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
}

cout << "Enter new " << role << "'s CNIC (11121-2223322-4): ";
cin.ignore();
getline(cin, newstaffBio.cnic);

while (newstaffBio.cnic.length() != 15 || newstaffBio.cnic[5] != '-' || newstaffBio.cnic[13] != '-') {
    cout << "Invalid CNIC format. Please enter a valid CNIC: ";
    getline(cin, newstaffBio.cnic);
}
ofstream userFile("logincredentials.txt", ios::app);
if (userFile.is_open()) {

```

```

        userFile << endl << role << ":" << newstaffLogin.username << ":" <<
newstaffLogin.password;
        userFile.close();
        cout << role << " added successfully!" << endl;
    }
    else {
        cerr << "Error opening credentials file!" << endl;
    }

    ofstream bioFile(role + "Biodata.txt", ios::app);
    if (bioFile.is_open()) {
        bioFile << newstaffLogin.username << ":" << newstaffBio.name << ":" << newstaffBio.age
<< ":" << newstaffBio.phoneNo << ":" << newstaffBio.cnic << endl;
        bioFile.close();
        cout << role << "'s biodata added successfully!" << endl;
    }
    else {
        cerr << "Error opening biodata file!" << endl;
    }
}

void removeUser(const string& role, const string& username) {
    ifstream loginFile("logincredentials.txt");
    ofstream temploginFile("temp.txt");
    ifstream bioFile(role + "Biodata.txt");
    ofstream tempbioFile("Biotemp.txt");

    if (!loginFile.is_open() || !temploginFile.is_open() || !bioFile.is_open() || !tempbioFile.is_open())
    {
        cerr << "Error opening file!" << endl;
        return;
    }

    string line;
    bool userFound = false;

    while (getline(loginFile, line)) {
        stringstream ss(line);
        string storedRole, storedUsername;
        getline(ss, storedRole, ':');
        getline(ss, storedUsername, ':');

        if (storedRole == role && storedUsername == username) {
            userFound = true;
        }
        else {
            temploginFile << line << endl;
        }
    }
}

```



```

loginFile.close();
temploginFile.close();

while (getline(bioFile, line)) {
    stringstream ss(line);
    string storedName;
    getline(ss, storedName, ':');

    if (storedName == username) {
        userFound = true;
    }
    else {
        tempbioFile << line << endl;
    }
}

bioFile.close();
tempbioFile.close();

if (remove("logincredentials.txt") != 0 || remove((role + "Biodata.txt").c_str()) != 0) {
    cerr << "Error deleting file!" << endl;
    return;
}

if (rename("temp.txt", "logincredentials.txt") != 0 || rename("Biotemp.txt", (role +
"Biodata.txt").c_str()) != 0) {
    cerr << "Error renaming file!" << endl;
    return;
}

if (userFound) {
    cout << "User removed successfully!" << endl;
}
else {
    cout << "User not found!" << endl;
}
}

void removeDoctor() {
    string username;
    cout << "Enter doctor's username to remove: ";
    cin >> username;
    removeUser("doctor", username);
}

void removeNurse() {
    string username;
    cout << "Enter nurse's username to remove: ";
    cin >> username;
    removeUser("nurse", username);
}

```

```

}
bool isUsernameTaken(const string& username) {
    ifstream loginFile("logincredentials.txt");
    if (!loginFile.is_open()) {
        cerr << "Error opening login credentials file!" << endl;
        return true;
    }

    string line;
    while (getline(loginFile, line)) {
        size_t pos = line.find(':');
        if (pos != string::npos) {
            string storedUsername = line.substr(pos + 1, line.find(':', pos + 1) - pos - 1);
            if (storedUsername == username) {
                loginFile.close();
                return true;
            }
        }
    }

    loginFile.close();
    return false;
}

void viewStaff(const string& fileName, const string& role) {
    ifstream staffFile(fileName);
    if (!staffFile.is_open()) {
        cerr << "Error opening " << role << " biodata file!" << endl;
        return;
    }

    cout << role << " Details:" << endl;
    cout << "-----" << endl;

    staffBio staffInfo;
    string line;

    while (getline(staffFile, line)) {
        stringstream ss(line);
        getline(ss, line, ':');
        getline(ss, staffInfo.name, ':');
        ss >> staffInfo.age;
        ss.ignore();
        ss >> staffInfo.phoneNo;
        ss.ignore();
        getline(ss, staffInfo.cnic);

        cout << "Name: " << staffInfo.name << endl;
        cout << "Age: " << staffInfo.age << endl;
        cout << "Phone Number: " << staffInfo.phoneNo << endl;
        cout << "CNIC: " << staffInfo.cnic << endl;
    }
}

```

```

        cout << "-----" << endl;
    }

    staffFile.close();
}

void register_patient() {
    system("CLS");
    Patient p;

    ifstream counter("id_count.txt");
    counter >> ID;
    counter.close();

    cout << "Enter Patient Full Name (without numbers): ";
    cin.ignore();
    getline(cin, p.namep);
    while (containsNumeric(p.namep)) {
        cout << "Name should not contain numbers. Enter Patient Full Name again: ";
        getline(cin, p.namep);
    }

    cout << "Enter Patient Age: ";
    while (!(cin >> p.agep) || p.agep <= 0) {
        cout << "Invalid age. Please enter a valid age: ";
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
    }

    cout << "Enter Patient CNIC (in the format 11121-2223322-4): ";
    getline(cin, p.cnicp);

    while (p.cnicp.length() != 15 || p.cnicp[5] != '-' || p.cnicp[13] != '-') {
        cout << "Invalid CNIC format. Please enter a valid CNIC: ";
        getline(cin, p.cnicp);
    }

    cout << "Enter Patient Phone Number: ";
    while (!(cin >> p.phnumber) || p.phnumber <= 0) {
        cout << "Invalid phone number. Please enter a valid phone number: ";
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
    }
    ID++;
    ofstream file("patient_records.txt", ios::app);
    file << ID;
    file << "\n" << p.namep;
    file << "\n" << p.agep;
    file << "\n" << p.cnicp;
    file << "\n" << p.phnumber << "\n";
}

```

```

file.close();
cout << "Patient added successfully!" << endl;
ofstream counts("id_count.txt", ios::out);
counts << ID;
counts.close();

}

void add_patientrecord() {
    view_patient();
    Patient p;
    int ids;

    cout << "Enter patient ID" << endl;
    cin >> ids;

    ifstream file("patient_records.txt");

    bool patientFound = false;

    while (file >> p.idp &&
           getline(file >> ws, p.namep) &&
           file >> p.agep &&
           getline(file >> ws, p.cnicp) &&
           file >> p.phnumber) {
        if (p.idp == ids) {
            patientFound = true;
            break;
        }
    }

    file.close();
    if (!patientFound) {
        cout << "Patient Not Registered" << endl;
        register_patient();
    }

    cout << "Enter Patient Vitals" << endl;
    cin >> p.vitals;

    cout << "Enter Patient Prescription " << endl;
    cin.ignore();
    getline(cin, p.pres);

    cout << "Enter Patient Comments" << endl;
    getline(cin, p.com);

    ofstream report("medical_record.txt", ios::app);
    report << p.idp << "\n" << p.vitals << "\n" << p.pres << "\n" << p.com << "\n";
}

```

```

    report.close();
}

void view_patient() {

    Patient p;
    ifstream read("patient_records.txt");

    if (!read.is_open()) {
        cout << "Error opening patient file!" << endl;
        return;
    }

    while (read >> p.idp &&
           getline(read >> ws, p.namep) &&
           read >> p.agep &&
           getline(read >> ws, p.cnicp) &&
           read >> p.phnumber) {
        cout << "Patient ID: " << p.idp << endl;
        cout << "Patient Name: " << p.namep << endl;
        cout << "Patient Age : " << p.agep << endl;
        cout << "Patient CNIC: " << p.cnicp << endl;
        cout << "Patient Phone Number: " << p.phnumber << endl;
        cout << "-----" << endl << endl;
    }

    read.close();
}

void update_patient() {
    Patient p;
    int id;
    bool found = false;

    cout << "Enter Patient ID to update record" << endl;
    cin >> id;

    ifstream read("medical_record.txt");
    ofstream tempFile("temp.txt", ios::app);

    while (read >> p.idp >> p.vitals) {
        read.ignore();
        getline(read, p.pres);
        getline(read, p.com);

        if (p.idp == id) {
            found = true;

```

```

        cout << "\nEnter new Patient Vital: ";
        cin >> p.vitals;
        cout << "\nEnter new Patient Prescription: ";
        cin.ignore();
        getline(cin, p.pres);
        cout << "\nEnter new Patient Comments: ";
        getline(cin, p.com);

        cout << "\nRecord updated successfully!";
    }
    tempFile << p.idp << "\n" << p.vitals << "\n" << p.pres << "\n" << p.com << "\n";
}
read.close();
tempFile.close();

if (found) {
    remove("medical_record.txt");
    rename("temp.txt", "medical_record.txt");
}
else {
    cout << "\n\tRecord not found!";
}
}

void remove_patient() {
    Patient p;
    int id;

    cout << "Enter Id of the Patient you want to remove" << endl;
    cin >> id;

    ifstream readPatient("patient_records.txt");
    bool foundPatient = false;

    while (readPatient >> p.idp >> p.namep >> p.agep >> p.cnicp >> p.phnumber) {
        if (p.idp == id) {
            foundPatient = true;
            break;
        }
    }

    readPatient.close();

    if (!foundPatient) {
        cout << "\n\tPatient not found in records";
        return;
    }

    ifstream readMedical("medical_record.txt");
    ofstream tempMedical("tempMedical.txt", ios::app);

```

```

bool foundMedical = false;

while (readMedical >> p.idp >> p.vitals) {
    readMedical.ignore();
    getline(readMedical, p.pres);
    getline(readMedical, p.com);

    if (p.idp == id) {
        foundMedical = true;
        cout << "\n\tMedical Record deleted successfully";
    }
    else {
        tempMedical << p.idp << "\n" << p.vitals << "\n" << p.pres << "\n" << p.com << "\n";
    }
}

readMedical.close();
tempMedical.close();

if (foundMedical) {
    remove("medical_record.txt");
    rename("tempMedical.txt", "medical_record.txt");
}
else {
    cout << "\n\tMedical Record not found";
}
}

void add_doctor_orders() {
    int id;
    cout << "\nEnter Patient ID: ";
    if (!(cin >> id)) {
        cout << "Invalid ID, try again!" << endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        return;
    }
    cout << "Enter Doctor's Orders for Patient ID " << id << " (Enter an empty line to finish):" << endl;

    ofstream docOrderFile("doctor_orders.txt", ios::app);
    if (!docOrderFile.is_open()) {
        cerr << "Error opening file!" << endl;
        return;
    }
    docOrderFile << id << '\n';

    string orders;

```

```

cin.ignore();
while (true) {
    getline(cin, orders);
    if (orders.empty()) {
        break;
    }
    docOrderFile << orders << "\n";
}
docOrderFile.close();
cout << "Doctor's Orders added successfully for Patient ID " << id << "." << endl;

}

void view_doctor_orders() {
    int id;
    cout << "\nEnter Patient ID: ";

    if (!(cin >> id)) {
        cout << "Invalid ID, try again!" << endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        return;
    }
    ifstream docOrderFile("doctor_orders.txt");
    if (!docOrderFile.is_open()) {
        cerr << "Error opening file!" << endl;
        return;
    }
    string line;
    bool patientFound = false;
    while (getline(docOrderFile, line)) {
        stringstream ss(line);
        int storedPatientID;
        ss >> storedPatientID;

        if (storedPatientID == id) {
            patientFound = true;
            cout << "Doctor's Orders for Patient ID " << id << ":" << endl;
            while (getline(docOrderFile, line) && !line.empty()) {
                cout << line << endl;
            }
            break;
        }
    }
    docOrderFile.close();
    if (!patientFound) {
        cout << "Doctor's orders not found for Patient ID " << id << "!" << endl;
    }
}

bool containsNumeric(const string& str) {
    for (char c : str) {

```



```

        if (isdigit(c)) {
            return true;
        }
    }
    return false;
}

void readMedicalRecord() {
    ifstream file("medical_record.txt");

    if (!file.is_open()) {
        cout << "Error opening medical record file!" << endl;
        return;
    }

    file.seekg(0, ios::end);
    if (file.tellg() == 0) {
        cout << "Error: Medical record file is empty!" << endl;
        file.close();
        return;
    }

    file.seekg(0, ios::beg);

    Patient p;
    while (file >> p.idp >> p.vitals >> p.pres >> p.com) {
        cout << "Patient ID: " << p.idp << endl;
        cout << "Patient Vitals: " << p.vitals << endl;
        cout << "Patient Prescription: " << p.pres << endl;
        cout << "Patient Comments: " << p.com << endl;
        cout << "-----" << endl;
    }

    file.close();
}

void searchMedicalRecord() {
    int patientId;
    cout << "Enter the Patient ID to search: ";
    cin >> patientId;

    ifstream file("medical_record.txt");

    if (!file.is_open()) {
        cout << "Error opening medical record file!" << endl;
        return;
    }

    Patient p;
    bool patientFound = false;

    while (file >> p.idp >> p.vitals >> p.pres >> p.com) {

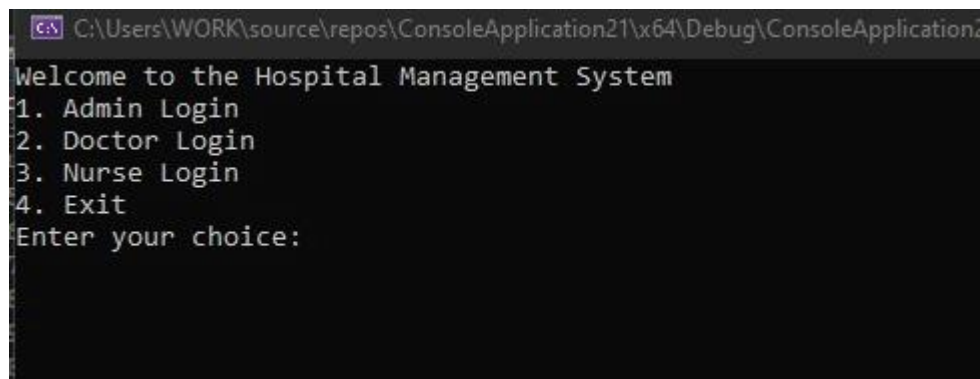
```

```
        if (p.idp == patientId) {
            cout << "Patient ID: " << p.idp << endl;
            cout << "Patient Vitals: " << p.vitals << endl;
            cout << "Patient Prescription: " << p.pres << endl;
            cout << "Patient Comments: " << p.com << endl;
            patientFound = true;
        }
    }

    file.close();

    if (!patientFound) {
        cout << "Patient not found in the medical records." << endl;
    }
}
```

OUTPUT



```
C:\Users\WORK\source\repos\ConsoleApplication21\x64\Debug\ConsoleApplication.exe
Welcome to the Hospital Management System
1. Admin Login
2. Doctor Login
3. Nurse Login
4. Exit
Enter your choice:
```

```
C:\Users\WORK\source\repos\ConsoleApplication21\x64\Debug\ConsoleApplication21.exe
Welcome to the Hospital Management System
1. Admin Login
2. Doctor Login
3. Nurse Login
4. Exit
Enter your choice: 1
Enter id > admin1
Enter password > password1
Welcome Admin: admin1
Admin Menu:
1. Add Doctor
2. Add Nurse
3. Remove Doctor
4. Remove Nurse
5. View Doctors
6. View Nurses
7. Register Patient
8. Display All Patients
9. Logout
Enter your choice:
```

```
Select C:\Users\WORK\source\repos\ConsoleApplication21\x64\Debug\ConsoleApplication21.exe

Welcome to the Hospital Management System
1. Admin Login
2. Doctor Login
3. Nurse Login
4. Exit
Enter your choice: 1
Enter id > admin1
Enter password > password1
Welcome Admin: admin1
Admin Menu:
1. Add Doctor
2. Add Nurse
3. Remove Doctor
4. Remove Nurse
5. View Doctors
6. View Nurses
7. Register Patient
8. Display All Patients
9. Logout
Enter your choice: 1
Enter new doctor's username: doctor23
Enter new doctor's password: doctor1234
Enter new doctor's name: taha
Enter new doctor's age: 41
Enter new doctor's phone number: 0333123123
Enter new doctor's CNIC (11121-2223322-4): 42401-12341232-1
Invalid CNIC format. Please enter a valid CNIC: 42401-2314234-1
doctor added successfully!
doctor's biodata added successfully!
Admin Menu:
```

```
C:\Users\WORK\source\repos\ConsoleApplication21\x64
Admin Menu:
1. Add Doctor
2. Add Nurse
3. Remove Doctor
4. Remove Nurse
5. View Doctors
6. View Nurses
7. Register Patient
8. Display All Patients
9. Logout
Enter your choice: 5
doctor Details:
-----
Name: taha
Age: 24
Phone Number: 333123123
CNIC: 424-1023123-1
-----
Name: Maira
Age: 25
Phone Number: 34234343
CNIC: 11111-2343434-1
-----
Name: taha
Age: 41
Phone Number: 333123123
CNIC: 42401-2314234-1
-----
Admin Menu:
1. Add Doctor
```

```
C:\Users\WORK\source\repos\ConsoleApplication21\x64\Debug\Co
CNIC: 11111-2343434-1
-----
Name: taha
Age: 41
Phone Number: 333123123
CNIC: 42401-2314234-1
-----
Admin Menu:
1. Add Doctor
2. Add Nurse
3. Remove Doctor
4. Remove Nurse
5. View Doctors
6. View Nurses
7. Register Patient
8. Display All Patients
9. Logout
Enter your choice: 4
Enter nurse's username to remove: nurse1
User removed successfully!
Admin Menu:
1. Add Doctor
2. Add Nurse
3. Remove Doctor
4. Remove Nurse
5. View Doctors
6. View Nurses
7. Register Patient
8. Display All Patients
9. Logout
```

```
C:\Users\WORK\source\repos\ConsoleApplication21\x64\Debug\Co
CNIC: 11111-2343434-1
-----
Name: taha
Age: 41
Phone Number: 333123123
CNIC: 42401-2314234-1
-----
Admin Menu:
1. Add Doctor
2. Add Nurse
3. Remove Doctor
4. Remove Nurse
5. View Doctors
6. View Nurses
7. Register Patient
8. Display All Patients
9. Logout
Enter your choice: 4
Enter nurse's username to remove: nurse1
User removed successfully!
Admin Menu:
1. Add Doctor
2. Add Nurse
3. Remove Doctor
4. Remove Nurse
5. View Doctors
6. View Nurses
7. Register Patient
8. Display All Patients
9. Logout
```

```
Select C:\Users\WORK\source\repos\ConsoleApplication21\x64\Debug\ConsoleApplication21.exe

Admin Menu:
1. Add Doctor
2. Add Nurse
3. Remove Doctor
4. Remove Nurse
5. View Doctors
6. View Nurses
7. Register Patient
8. Display All Patients
9. Logout
Enter your choice: 5
doctor Details:
-----
Name: taha
Age: 24
Phone Number: 333123123
CNIC: 424-1023123-1
-----
Name: Maira
Age: 25
Phone Number: 34234343
CNIC: 11111-2343434-1
-----
Name: taha
Age: 41
Phone Number: 333123123
CNIC: 42401-2314234-1
-----
Admin Menu:
1. Add Doctor
```



```
Patient Phone Number: 1231231231
-----

Admin Menu:
1. Add Doctor
2. Add Nurse
3. Remove Doctor
4. Remove Nurse
5. View Doctors
6. View Nurses
7. Register Patient
8. Display All Patients
9. Logout
Enter your choice: 9
Exiting Admin Dashboard...
Welcome to the Hospital Management System
1. Admin Login
2. Doctor Login
3. Nurse Login
4. Exit
Enter your choice: 2
Enter id > doctor1
Enter password > doctor123
Welcome Doctor: doctor1
1. Add Patient's Medical record
2. View Patient Data
3. Display Patients Medical record
4. Search for a Patient's Medical record
5. Exit..
```

```
C:\Users\WORK\source\repos\ConsoleApplication21\x64\Debug\
9. Logout
Enter your choice: 9
Exiting Admin Dashboard...
Welcome to the Hospital Management System
1. Admin Login
2. Doctor Login
3. Nurse Login
4. Exit
Enter your choice: 2
Enter id > doctor1
Enter password > doctor123
Welcome Doctor: doctor1
1. Add Patient's Medical record
2. View Patient Data
3. Display Patients Medical record
4. Search for a Patient's Medical record
5. Exit..
3
Patient ID: 132
Patient Vitals: 9123012
Patient Prescription: podsnda
Patient Comments: oasndansd
-----
Welcome Doctor: doctor1
1. Add Patient's Medical record
2. View Patient Data
3. Display Patients Medical record
4. Search for a Patient's Medical record
5. Exit..
```

```
132
9123012
podsnda
oasndansd
```

```
admin:admin1:password1
doctor:doctor1:doctor123
doctor:maira:maira1
nurse:dekiuki:dekiuki1
doctor:doctor23:doctor1234
```

```
nimra:nimra:123:412304123:412401-12314121  
nasd:asd:123:333123123:4123-123123-1  
dekisuki:Dekisuki:26:123456667:12222-3434433-2
```

```

jasdad
Enter Patient Comments
asdaa
Welcome Doctor: doctor1
1. Add Patient's Medical record
2. View Patient Data
3. Display Patients Medical record
4. Search for a Patient's Medical record
5. Exit..
4
Enter the Patient ID to search: 135
Patient ID: 135
Patient Vitals: 1234
Patient Prescription: jasdad
Patient Comments: asdaa
Welcome Doctor: doctor1
1. Add Patient's Medical record
2. View Patient Data
3. Display Patients Medical record
4. Search for a Patient's Medical record
5. Exit..

```

```

C:\Users\WORK\source\repos\ConsoleApplication21\x64\Debug\ConsoleApplic
5. Exit..
4
Enter the Patient ID to search: 123
Patient not found in the medical records.
Welcome Doctor: doctor1
1. Add Patient's Medical record
2. View Patient Data
3. Display Patients Medical record
4. Search for a Patient's Medical record
5. Exit..
1
Patient ID: 132
Patient Name: aecs asd
Patient Age : 21
Patient CNIC: 4242-123-12
Patient Phone Number: 4123123
-----
Patient ID: 133
Patient Name: abcd
Patient Age : 23
Patient CNIC: 42401-2412345-1
Patient Phone Number: 1231231234
-----
Patient ID: 134
Patient Name: wieq asd
Patient Age : 21
Patient CNIC: 42401-2341234-1
Patient Phone Number: 333333333

```

```
C:\Users\WORK\source\repos\ConsoleApplication21\x64\Debug\ConsoleApplication21.exe
1. View Patient Information
2.Update Patient Information
3. Add Doctor's Orders
4. View Doctor's Orders
5. Exit
Enter your choice: 3

Enter Patient ID: 135
Enter Doctor's Orders for Patient ID 135 (Enter an empty line to finish):
paisd asdasd

Doctor's Orders added successfully for Patient ID 135.
Welcome Nurse: dekiuki
1. View Patient Information
2.Update Patient Information
3. Add Doctor's Orders
4. View Doctor's Orders
5. Exit
Enter your choice: 4

Enter Patient ID: 135
Doctor's Orders for Patient ID 135:
paisd asdasd
Welcome Nurse: dekiuki
1. View Patient Information
2.Update Patient Information
3. Add Doctor's Orders
4. View Doctor's Orders
5. Exit
Enter your choice:
```

```
C 132
9123012
podsnda
oasndansd
```