```cpp
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <exception>
using namespace std;

// Struct definitions
struct DoctorData {
    string name;
    string specialty;
    int age;
    string phone;
};

struct Patient {
    int id;
    string name;
    int age;
    string gender;
    string diagnosis;
    string contact;

    void print() const {
        cout << "ID: " << id << ", Name: " << name << ", Age: " << age
             << ", Gender: " << gender << ", Diagnosis: " << diagnosis
             << ", Contact: " << contact << endl;
    }
};

struct Staff {
    int id;
    string name;
    string position;
    string contact;

    void print() const {
        cout << "ID: " << id << ", Name: " << name << ", Position: " <<
         position
             << ", Contact: " << contact << endl;
    }
};

// Our fucitons

//Zahra Alabdulatif - 2230006809
void writereports();
void viewreports();
void rateHospital();

//Zahra Khalfan - 2230004089
void createPrescription();
void displayPrescription();

//Dana althani 2230005314
```

```cpp
void inputDoctorData(DoctorData& doctor);
void printDoctorData(const DoctorData& doctor);
void inputDoctorAppointments(string* appointments, int numAppointments);
void printDoctorAppointments(const string* appointments, int
 numAppointments);

//Jori Alghamdi -2230002748
void addPatient();
void deletePatient();
void updatePatient();
void searchPatient();

//Asayel Alghamdi -2220000203
void billing();
void payment();
void addStaff();
void searchStaff();

//Leena Alkhudair , 2230006267
void deleteStaff();
void updateStaff();
void displayStaff();

const int MAX_PATIENTS = 100;

Patient patients[MAX_PATIENTS];
int patientCount = 0;

Staff* staff = nullptr;
int staffCount = 0;

void writereports() {
    ofstream file("reports.txt", ios::app);
    try {
        if (file.is_open()) {
            string title, contact;
            cout << "\nEnter the title: ";
            cin.ignore();
            getline(cin, title);
            cout << "Enter the contact information: ";
            getline(cin, contact);

            file << "Title: " << title << endl;
            file << "Contact: " << contact << endl;
            file << "\n--------------------\n";
            file.close();
            cout << "Report saved successfully!" << endl;
        } else {
            throw runtime_error("Error opening file!");
        }
    } catch (const exception& e) {
        cout << e.what() << endl;
    }
}
```

```
void viewreports() {
    ifstream file("reports.txt");
    try {
        if (file.is_open()) {
            cout << "\n=== All Reports ===" << endl;
            string r;
            while (getline(file, r)) {
                cout << r;
                cout << endl;
                if (r == "--------------------")
                    cout << endl;
            }
            file.close();
        } else {
            throw runtime_error("No reports found!!");
        }
    } catch (const exception& e) {
        cout << e.what() << endl;
    }
}

void rateHospital() {
    const int SIZE = 7;
    int ratings[SIZE];
    string categories[SIZE] = {"Cleanliness", "Medical Care Quality",
     "Staff Friendliness", "Wait Times", "Communication", "Complaints
     Handling", "Overall Experience"};

    cout << "=====rating=====" << endl;
    cout << "from (1) to (5)" << endl;
    cout << "(1) represents ( the experience was bad ) and (5) represents (
     the experience was excellent ) " << endl;
    cout << "--------------------------" << endl;

    for (int i = 0; i < SIZE; ++i) {
        bool input = false;
        while (!input) {
            try {
                cout << categories[i] << " : ";
                if (!(cin >> ratings[i])) {
                    throw runtime_error("Please enter a numeric rating.");
                } else if (ratings[i] < 1 || ratings[i] > 5) {
                    throw runtime_error("Please enter a rating between 1
                     and 5.");
                } else {
                    input = true;
                }
            } catch (const exception& e) {
                cout << e.what() << endl;
                cin.clear();
                cin.ignore(1000, '\n');
            }
        }
    }
    char note;
    cout << "Would you like to add a note (y/n): ";
    cin >> note;

    if (note == 'y' || note == 'Y') {
        string addnote;
        cout << "Enter your note: ";
        cin.ignore();
        getline(cin, addnote);
        cout << "Note added!!" << endl;
    } else {
        cout << "No note added!!" << endl;
    }

    cout << "Thank you for providing your ratings!" << endl;
}

void createPrescription() {
    string diagnosis, medication, name;

    cout << "Enter patient name: ";
    cin >> name;
    cout << "Enter patient diagnosis: ";
    cin >> diagnosis;
    cout << "Enter medication: ";
    cin >> medication;
    cout << "The Prescription has been successfully recorded." << endl;

    // Save prescription to file
    try {
        ofstream prescription_file("Pharmacy_file.txt", ios::app);
        if (prescription_file.is_open()) {
            prescription_file << "\nPatient Name: " << name << endl;
            prescription_file << "Patient Diagnosis: " << diagnosis << endl;
            prescription_file << "Medication: " << medication << "." <<
             endl;
            prescription_file.close();
        } else {
            throw runtime_error("Failed to open the file");
        }
    } catch (const exception& e) {
        cout << e.what() << endl;
    }
}

void displayPrescription() {
    try {
        ifstream prescription_file("Pharmacy_file.txt", ios::in);
        if (prescription_file.is_open()) {
            string data;
            while (getline(prescription_file, data)) {
                cout << data << endl;
            }
            prescription_file.close();
        } else {
            throw runtime_error("Failed to open the file");
        }
    } catch (const exception& e) {
        cout << e.what() << endl;
    }
}

void inputDoctorData(DoctorData& doctor) {
    cout << "Enter doctor's name: ";
    cin.ignore();
    getline(cin, doctor.name);
    cout << "Enter doctor's specialty: ";
    getline(cin, doctor.specialty);

    cout << "Enter doctor's age: ";
    cin >> doctor.age;
    while (cin.fail() || doctor.age <= 0) {
        cout << "Invalid input. Please enter a valid age: ";
        cin.clear();
        cin.ignore(10000, '\n');
        cin >> doctor.age;
    }
    cin.ignore(); // To avoid input issues with getline

    cout << "Enter doctor's phone number: ";
    getline(cin, doctor.phone);
}

void printDoctorData(const DoctorData& doctor) {
    cout << "\nDoctor's Information:\n";
    cout << "Name: " << doctor.name << "\n";
    cout << "Specialty: " << doctor.specialty << "\n";
    cout << "Age: " << doctor.age << "\n";
    cout << "Phone: " << doctor.phone << "\n";
}

void inputDoctorAppointments(string* appointments, int numAppointments) {
    cout << "\nEnter doctor's appointments:\n";
    for (int i = 0; i < numAppointments; ++i) {
        cout << "Appointment " << (i + 1) << ": ";
        getline(cin, appointments[i]);
    }
}

void printDoctorAppointments(const string* appointments, int
 numAppointments) {
    cout << "\nDoctor's Appointments:\n";
    for (int i = 0; i < numAppointments; ++i) {
        cout << "Appointment " << (i + 1) << ": " << appointments[i] <<
         "\n";
    }
}

void saveToFile() {
    try {
        ofstream file("patients.txt");
        if (!file.is_open()) {
            throw runtime_error("Failed to open the file for saving
             patients.");
        }
        for (int i = 0; i < patientCount; ++i) {
            file << patients[i].id << ',' << patients[i].name << ',' <<
             patients[i].age << ','
                 << patients[i].gender << ',' << patients[i].diagnosis <<
                  ',' << patients[i].contact << '\n';
        }
    } catch (const exception& e) {
        cout << e.what() << endl;
    }
}

void loadFromFile() {
    try {
        ifstream file("patients.txt");
        if (!file.is_open()) {
            throw runtime_error("Failed to open the file for loading
             patients.");
        }
        string line;
        while (getline(file, line) && patientCount < MAX_PATIENTS) {
            Patient p;
            stringstream ss(line);
            string token;

            getline(ss, token, ',');
            p.id = stoi(token);
            getline(ss, p.name, ',');
            getline(ss, token, ',');
            p.age = stoi(token);
            getline(ss, p.gender, ',');
            getline(ss, p.diagnosis, ',');
            getline(ss, p.contact, ',');

            patients[patientCount++] = p;
        }
    } catch (const exception& e) {
        cout << e.what() << endl;
    }
}

void addPatient() {
    if (patientCount >= MAX_PATIENTS) {
        cout << "Patient limit reached. Cannot add more patients.\n";
        return;
    }
    Patient p;
    cout << "Enter ID: ";
    cin >> p.id;
    cout << "Enter Name: ";
    cin >> p.name;
```

```cpp
    cout << "Enter Age: ";
    cin >> p.age;
    cout << "Enter Gender: ";
    cin >> p.gender;
    cout << "Enter Diagnosis: ";
    cin >> p.diagnosis;
    cout << "Enter Contact: ";
    cin >> p.contact;
    patients[patientCount++] = p;
}

void deletePatient() {
    int id;
    cout << "Enter Patient ID to delete: ";
    cin >> id;
    bool found = false;
    for (int i = 0; i < patientCount; ++i) {
        if (patients[i].id == id) {
            found = true;
            for (int j = i; j < patientCount - 1; ++j) {
                patients[j] = patients[j + 1];
            }
            --patientCount;
            cout << "Patient record deleted successfully.\n";
            break;
        }
    }
    if (!found) {
        cout << "Patient record not found.\n";
    }
}

void updatePatient() {
    int id;
    cout << "Enter Patient ID to update: ";
    cin >> id;
    for (int i = 0; i < patientCount; ++i) {
        if (patients[i].id == id) {
            cout << "Enter new Name: ";
            cin >> patients[i].name;
            cout << "Enter new Age: ";
            cin >> patients[i].age;
            cout << "Enter new Gender: ";
            cin >> patients[i].gender;
            cout << "Enter new Diagnosis: ";
            cin >> patients[i].diagnosis;
            cout << "Enter new Contact: ";
            cin >> patients[i].contact;
            cout << "Patient record updated successfully.\n";
            return;
        }
    }
    cout << "Patient record not found.\n";
}
```

```cpp
void searchPatient() {
    int id;
    cout << "Enter Patient ID to search: ";
    cin >> id;
    for (int i = 0; i < patientCount; ++i) {
        if (patients[i].id == id) {
            patients[i].print();
            return;
        }
    }
    cout << "Patient record not found.\n";
}

void billing() {
    int id;
    double amount;
    cout << "Enter Patient ID for billing: ";
    cin >> id;

    try {
        bool found = false;
        for (int i = 0; i < patientCount; ++i) {
            if (patients[i].id == id) {
                found = true;
                cout << "Enter billing amount: ";
                cin >> amount;
                if (cin.fail() || amount < 0) {
                    throw invalid_argument("Invalid amount entered.");
                }
                ofstream billing_file("billing.txt", ios::app);
                if (!billing_file.is_open()) {
                    throw runtime_error("Failed to open billing file.");
                }
                billing_file << "Patient ID: " << id << ", Amount: " <<
                 amount << "\n";
                billing_file.close();
                cout << "Billing information recorded successfully.\n";
                break;
            }
        }
        if (!found) {
            cout << "Patient record not found.\n";
        }
    } catch (const exception& e) {
        cout << e.what() << endl;
        cin.clear();
        cin.ignore(10000, '\n');
    }
}

void payment() {
    int id;
    double amount;
    cout << "Enter Patient ID for payment: ";
    cin >> id;
    try {
        bool found = false;
        for (int i = 0; i < patientCount; ++i) {
            if (patients[i].id == id) {
                found = true;
                cout << "Enter payment amount: ";
                cin >> amount;
                if (cin.fail() || amount < 0) {
                    throw invalid_argument("Invalid amount entered.");
                }
                // Save payment information to file
                ofstream payment_file("payments.txt", ios::app);
                if (!payment_file.is_open()) {
                    throw runtime_error("Failed to open payment file.");
                }
                payment_file << "Patient ID: " << id << ", Amount: " <<
                 amount << "\n";
                payment_file.close();
                cout << "Payment information recorded successfully.\n";
                break;
            }
        }
        if (!found) {
            cout << "Patient record not found.\n";
        }
    } catch (const exception& e) {
        cout << e.what() << endl;
        cin.clear();
        cin.ignore(10000, '\n');
    }
}

void addStaff() {
    Staff* temp = new Staff[staffCount + 1];
    for (int i = 0; i < staffCount; ++i) {
        temp[i] = staff[i];
    }
    delete[] staff;
    staff = temp;

    cout << "Enter ID: ";
    cin >> staff[staffCount].id;
    cout << "Enter Name: ";
    cin >> staff[staffCount].name;
    cout << "Enter Position: ";
    cin >> staff[staffCount].position;
    cout << "Enter Contact: ";
    cin >> staff[staffCount].contact;

    staffCount++;
}

void deleteStaff() {
    int id;
    cout << "Enter Staff ID to delete: ";
    cin >> id;
    bool found = false;
    for (int i = 0; i < staffCount; ++i) {
        if (staff[i].id == id) {
            found = true;
            for (int j = i; j < staffCount - 1; ++j) {
                staff[j] = staff[j + 1];
            }
            staffCount--;
            Staff* temp = new Staff[staffCount];
            for (int k = 0; k < staffCount; ++k) {
                temp[k] = staff[k];
            }
            delete[] staff;
            staff = temp;
            cout << "Staff record deleted successfully.\n";
            break;
        }
    }
    if (!found) {
        cout << "Staff record not found.\n";
    }
}

void updateStaff() {
    int id;
    cout << "Enter Staff ID to update: ";
    cin >> id;
    for (int i = 0; i < staffCount; ++i) {
        if (staff[i].id == id) {
            cout << "Enter new Name: ";
            cin >> staff[i].name;
            cout << "Enter new Position: ";
            cin >> staff[i].position;
            cout << "Enter new Contact: ";
            cin >> staff[i].contact;
            cout << "Staff record updated successfully.\n";
            return;
        }
    }
    cout << "Staff record not found.\n";
}

void searchStaff() {
    int id;
    cout << "Enter Staff ID to search: ";
    cin >> id;
    for (int i = 0; i < staffCount; ++i) {
        if (staff[i].id == id) {
            staff[i].print();
            return;
        }
    }
    cout << "Staff record not found.\n";
```

```cpp
}

void displayStaff() {
    cout << "\n=== All Staff ===" << endl;
    for (int i = 0; i < staffCount; ++i) {
        staff[i].print();
    }
}

int main() {
    loadFromFile();
    int choice;

    while (true) {
        cout << "\nHospital Management System\n";
        cout << "1. Doctor\n";
        cout << "2. Patient\n";
        cout << "3. Reception\n";
        cout << "4. HR\n";
        cout << "5. Rate Hospital\n";
        cout << "6. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1: {
                int doctorChoice;
                cout << "\nDoctor Menu\n";
                cout << "1. Write Report\n";
                cout << "2. View Reports\n";
                cout << "3. Create Prescription\n";
                cout << "4. Exit\n";
                cout << "Enter your choice: ";
                cin >> doctorChoice;
                switch (doctorChoice) {
                    case 1:
                        writereports();
                        break;
                    case 2:
                        viewreports();
                        break;
                    case 3:
                        createPrescription();
                        break;
                    case 4:
                        break;
                    default:
                        cout << "Invalid choice. Please try again.\n";
                }
                break;
            }
            case 2: {
                int patientChoice;
                cout << "\nPatient Menu\n";
                cout << "1. Book Appointment\n";
                cout << "2. Make Payment\n";
                cout << "3. Show Reports\n";
                cout << "4. Display Prescriptions\n";
                cout << "5. Exit\n";
                cout << "Enter your choice: ";
                cin >> patientChoice;
                switch (patientChoice) {
                    case 1: {
                        int numAppointments;
                        cout << "Enter the number of appointments: ";
                        cin >> numAppointments;
                        string* appointments = new string[numAppointments];
                        cin.ignore();
                        inputDoctorAppointments(appointments,
                         numAppointments);
                        printDoctorAppointments(appointments,
                         numAppointments);
                        delete[] appointments;
                        break;
                    }
                    case 2:
                        payment();
                        break;
                    case 3:
                        viewreports();
                        break;
                    case 4:
                        displayPrescription();
                        break;
                    case 5:
                        break;
                    default:
                        cout << "Invalid choice. Please try again.\n";
                }
                break;
            }
            case 3: {
                int receptionChoice;
                cout << "\nReception Menu\n";
                cout << "1. Show Doctors Info\n";
                cout << "2. Search Patient\n";
                cout << "3. Add Patient\n";
                cout << "4. Update Patient\n";
                cout << "5. Delete Patient\n";
                cout << "6. Billing\n";
                cout << "7. Exit\n";
                cout << "Enter your choice: ";
                cin >> receptionChoice;
                switch (receptionChoice) {
                    case 1: {
                        DoctorData doctor;
                        inputDoctorData(doctor);
                        printDoctorData(doctor);
                        break;
                    }
```

```
                case 2:
                    searchPatient();
                    break;
                case 3:
                    addPatient();
                    break;
                case 4:
                    updatePatient();
                    break;
                case 5:
                    deletePatient();
                    break;
                case 6:
                    billing();
                    break;
                case 7:
                    break;
                default:
                    cout << "Invalid choice. Please try again.\n";
            }
            break;
        }
        case 4: {
            int hrChoice;
            cout << "\nHR Menu\n";
            cout << "1. Add Staff\n";
            cout << "2. Delete Staff\n";
            cout << "3. Update Staff\n";
            cout << "4. Search Staff\n";
            cout << "5. Display All Staff\n";
            cout << "6. Add Doctor Appointments\n";
            cout << "7. Exit\n";
            cout << "Enter your choice: ";
            cin >> hrChoice;
            switch (hrChoice) {
                case 1:
                    addStaff();
                    break;
                case 2:
                    deleteStaff();
                    break;
                case 3:
                    updateStaff();
                    break;
                case 4:
                    searchStaff();
                    break;
                case 5:
                    displayStaff();
                    break;
                case 6: {
                    int numAppointments;
                    cout << "Enter the number of appointments: ";
                    cin >> numAppointments;
                    string* appointments = new string[numAppointments];
                    cin.ignore();
                    inputDoctorAppointments(appointments,
                     numAppointments);
                    printDoctorAppointments(appointments,
                     numAppointments);
                    delete[] appointments;
                    break;
                }
                case 7:
                    break;
                default:
                    cout << "Invalid choice. Please try again.\n";
            }
            break;
        }
        case 5:
            rateHospital();
            break;
        case 6:
            saveToFile();
            return 0;
        default:
            cout << "Invalid choice. Please try again.\n";
    }
}

return 0;
}
```