Fatima Jinnah Women University

Opening Portals of Excellence Through Higher Education

SEMESTER PROJECT

OBJECT ORIENTED PROGRAMMING

Submitted TO:

DR. Sobia Khalid

SUBMITTED BY:

MARYAM WASEEM (2022 –BSE -059)

HOSPITAL MANAGEMENT SYSTEM

DESCRIPTIONS

Hospital management system has been made by using C++ languages and also using object oriented languages.

Hospital management system including:

Classes:

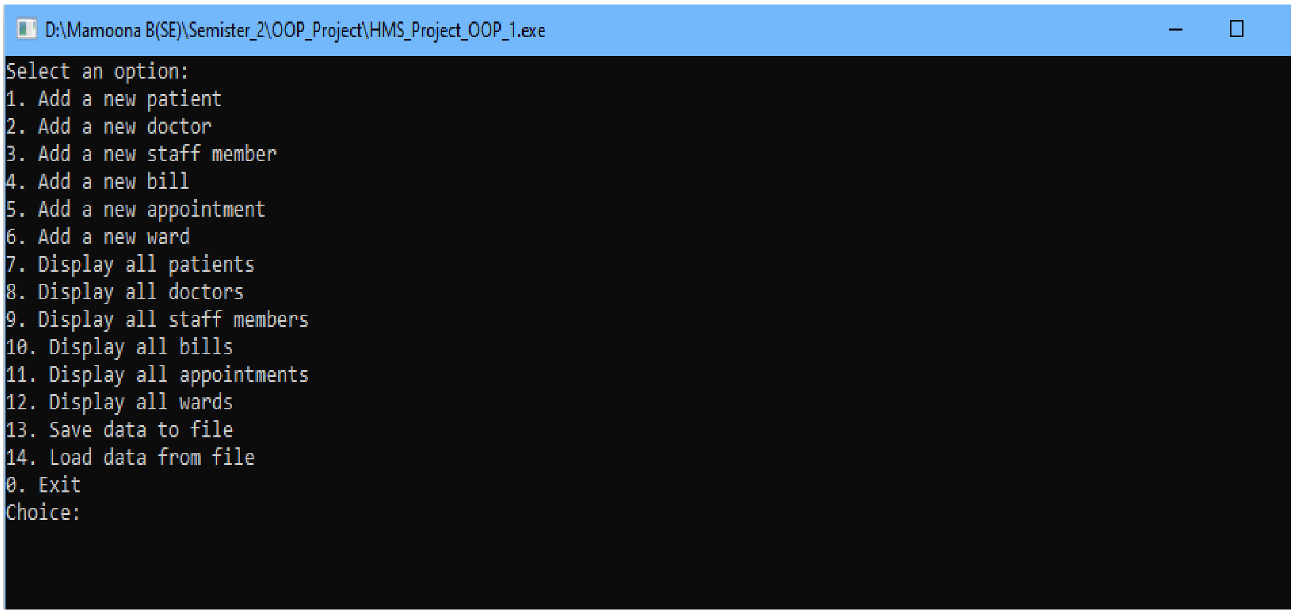
Patient, Doctor, Staff, Bill, Appointment, Ward, GeneralWard, and ICU.

Functions:

The main function provides a menu-based interface to add and display various entities such as patients, doctors, staff members, bills, appointments, and wards. It also includes functionality to save and load data to/from a file. Each option in the menu corresponds to a specific case in a switch statement, where the corresponding actions are performed based on user input.

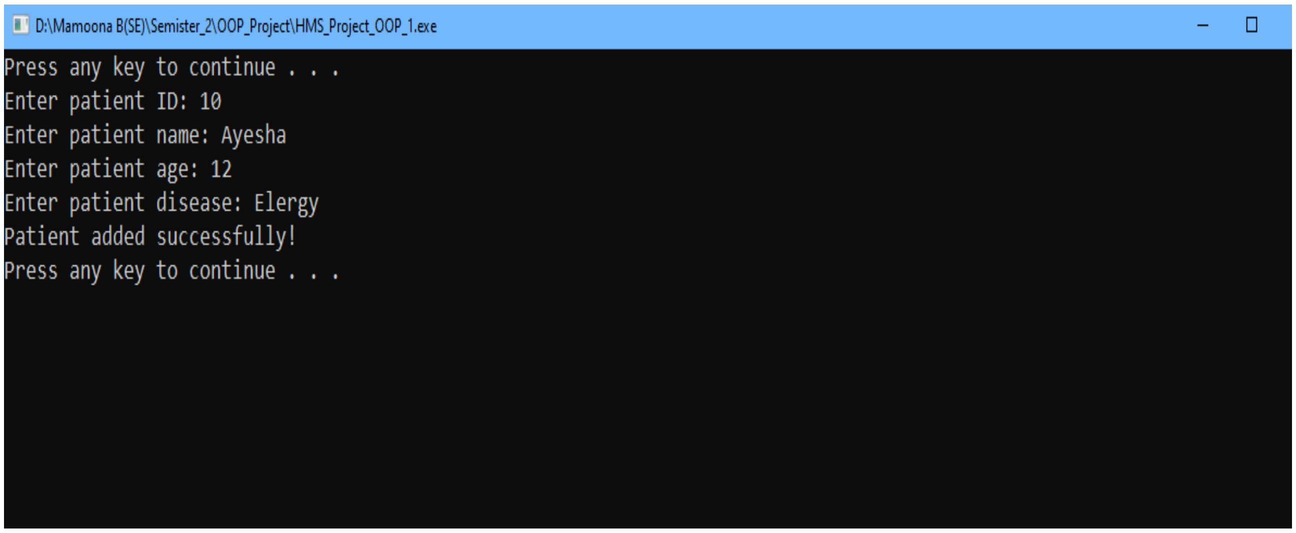
OUTPUT CONSOLE:

The MENU is displaying on choice the enter



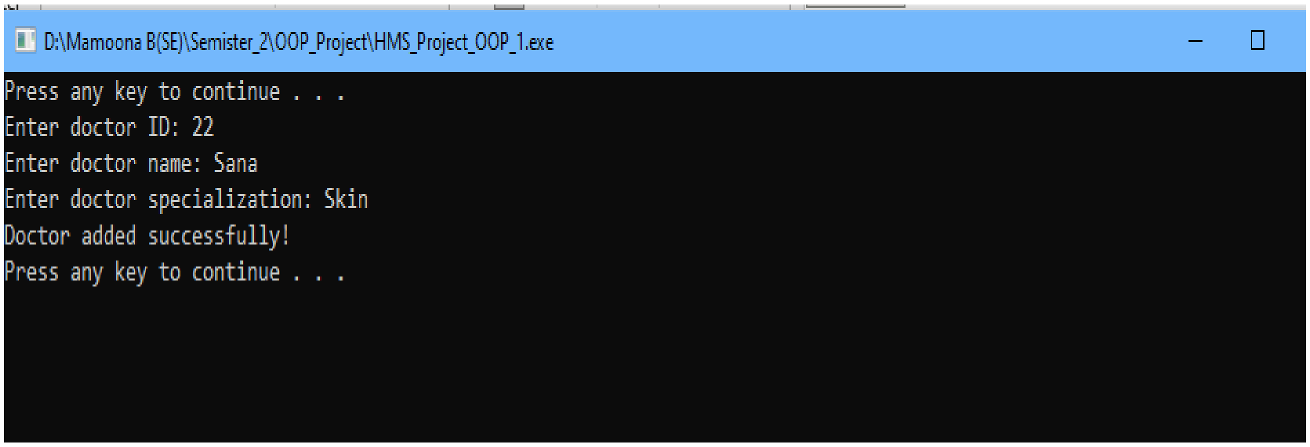
```
D:\Mamoona B(SE)\Semister_2\OOP_Project\HMS_Project_OOP_1.exe
Select an option:
1. Add a new patient
2. Add a new doctor
3. Add a new staff member
4. Add a new bill
5. Add a new appointment
6. Add a new ward
7. Display all patients
8. Display all doctors
9. Display all staff members
10. Display all bills
11. Display all appointments
12. Display all wards
13. Save data to file
14. Load data from file
0. Exit
Choice:
```

Choice 1 user is asked to enter the Patient data



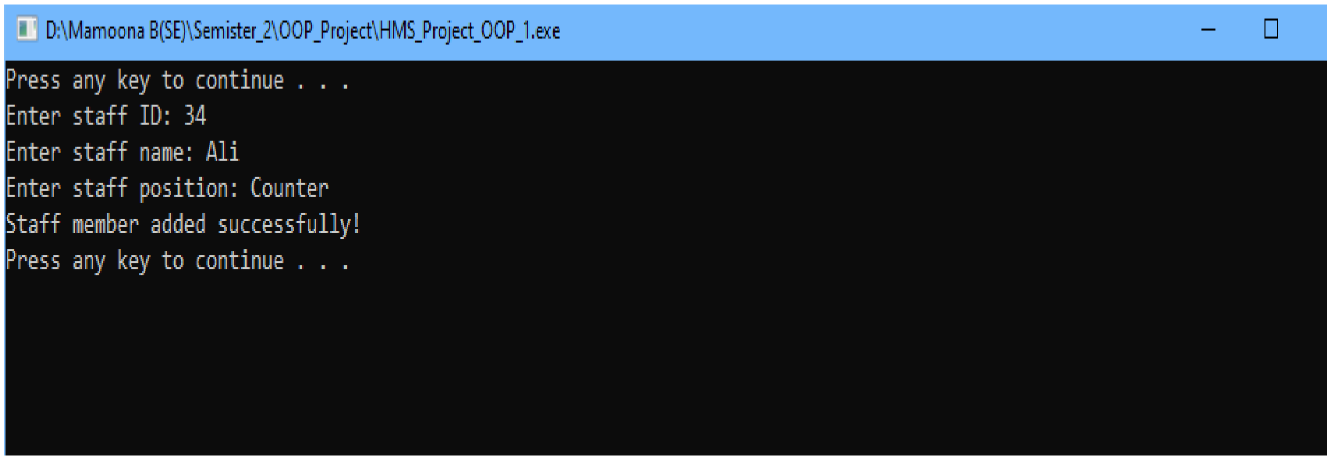
```
D:\Mamoona B(SE)\Semister_2\OOP_Project\HMS_Project_OOP_1.exe
Press any key to continue . . .
Enter patient ID: 10
Enter patient name: Ayesha
Enter patient age: 12
Enter patient disease: Elergy
Patient added successfully!
Press any key to continue . . .
```

Choice 2 user is asked to enter the Doctor data


D:\Mamoona B(SE)\Semister_2\OOP_Project\HMS_Project_OOP_1.exe

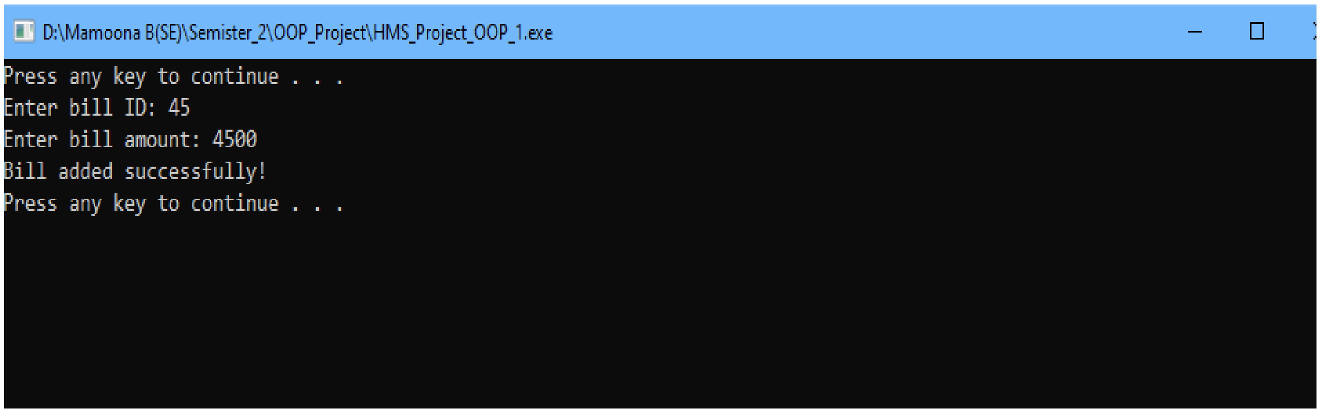
```
Press any key to continue . . .
Enter doctor ID: 22
Enter doctor name: Sana
Enter doctor specialization: Skin
Doctor added successfully!
Press any key to continue . . .
```

Choice 3 user is asked to enter the Staff data


D:\Mamoona B(SE)\Semister_2\OOP_Project\HMS_Project_OOP_1.exe

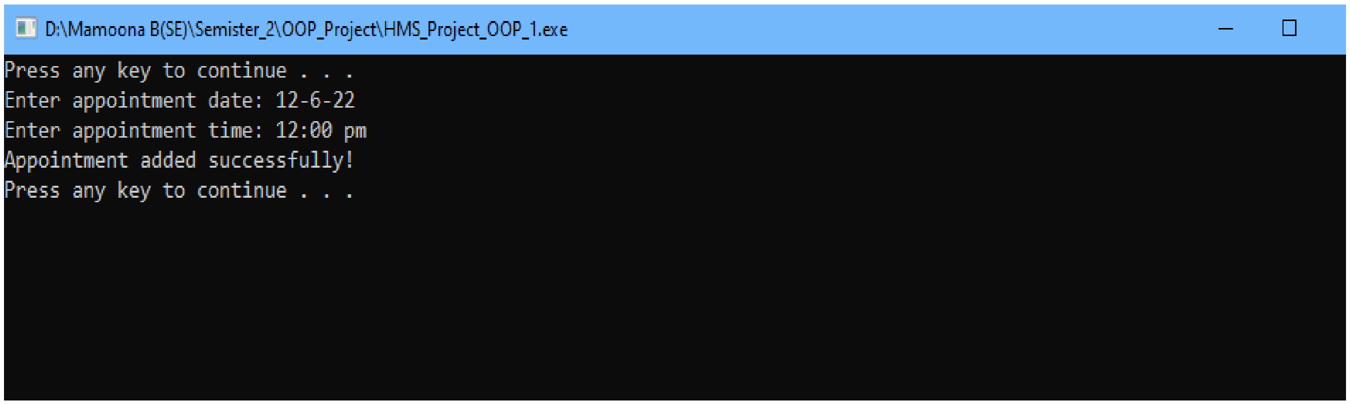
```
Press any key to continue . . .
Enter staff ID: 34
Enter staff name: Ali
Enter staff position: Counter
Staff member added successfully!
Press any key to continue . . .
```

Choice 4 user is asked to enter the Bill data


D:\Mamoona B(SE)\Semister_2\OOP_Project\HMS_Project_OOP_1.exe

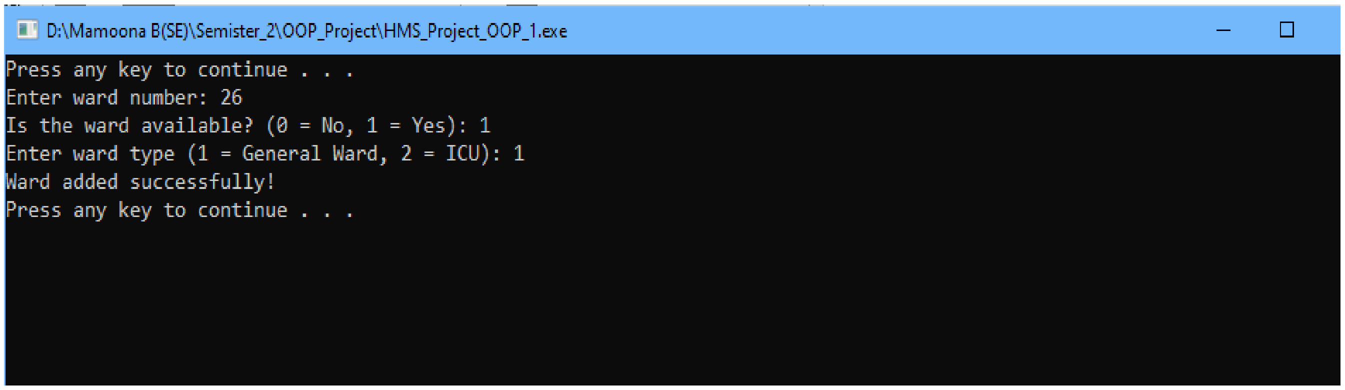
```
Press any key to continue . . .
Enter bill ID: 45
Enter bill amount: 4500
Bill added successfully!
Press any key to continue . . .
```

Choice 5 user is asked to enter the Appointment detial

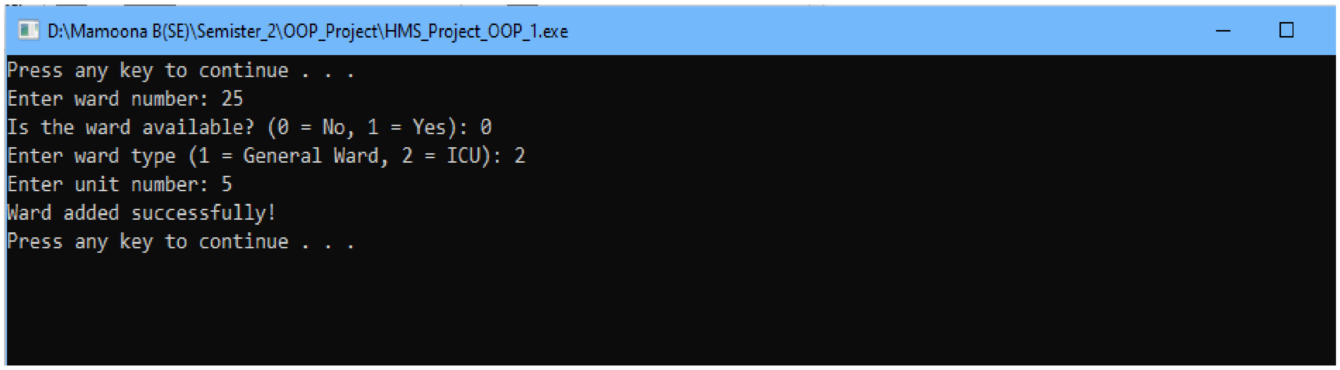

D:\Mamoona B(SE)\Semister_2\OOP_Project\HMS_Project_OOP_1.exe

```
Press any key to continue . . .
Enter appointment date: 12-6-22
Enter appointment time: 12:00 pm
Appointment added successfully!
Press any key to continue . . .
```

Choice 6 user is asked to enter the Ward detial

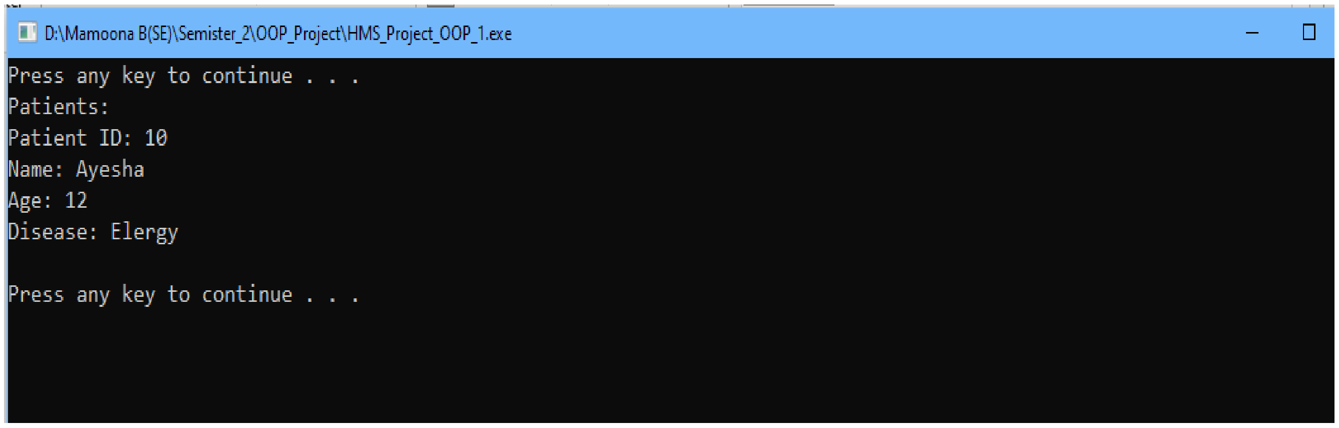

D:\Mamoona B(SE)\Semister_2\OOP_Project\HMS_Project_OOP_1.exe

```
Press any key to continue . . .
Enter ward number: 26
Is the ward available? (0 = No, 1 = Yes): 1
Enter ward type (1 = General Ward, 2 = ICU): 1
Ward added successfully!
Press any key to continue . . .
```


D:\Mamoona B(SE)\Semister_2\OOP_Project\HMS_Project_OOP_1.exe

```
Press any key to continue . . .
Enter ward number: 25
Is the ward available? (0 = No, 1 = Yes): 0
Enter ward type (1 = General Ward, 2 = ICU): 2
Enter unit number: 5
Ward added successfully!
Press any key to continue . . .
```

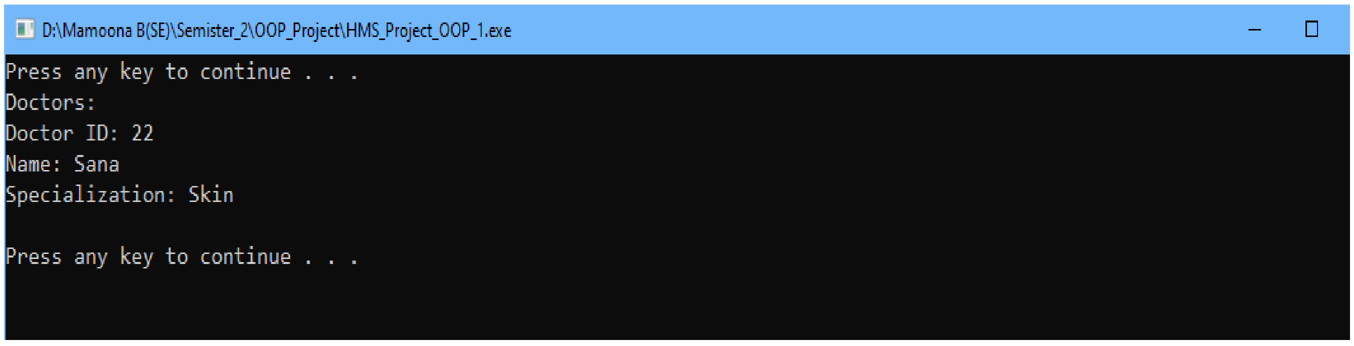
Choice 7 the Patient data is displayed


D:\Mamoona B(SE)\Semister_2\OOP_Project\HMS_Project_OOP_1.exe

```
Press any key to continue . . .
Patients:
Patient ID: 10
Name: Ayesha
Age: 12
Disease: Elergy

Press any key to continue . . .
```

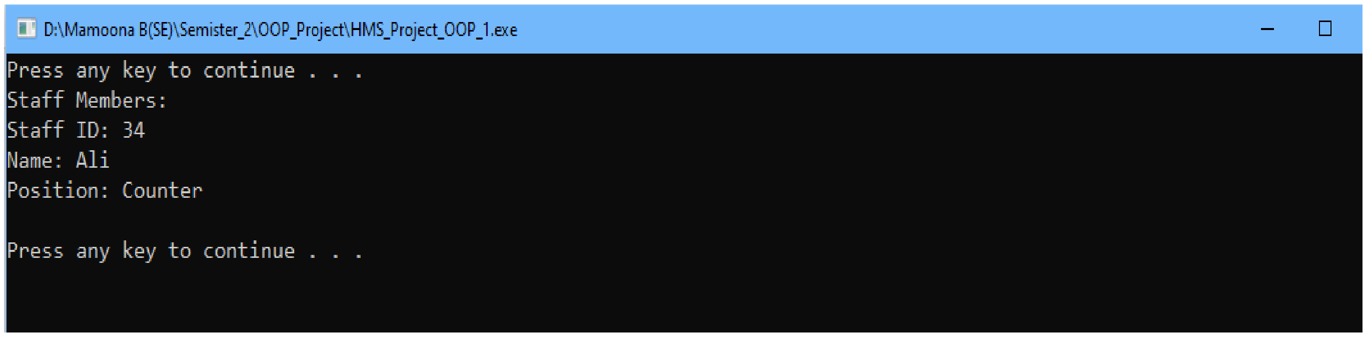
Choice 8 the Doctor data is displayed


D:\Mamoona B(SE)\Semister_2\OOP_Project\HMS_Project_OOP_1.exe

```
Press any key to continue . . .
Doctors:
Doctor ID: 22
Name: Sana
Specialization: Skin

Press any key to continue . . .
```

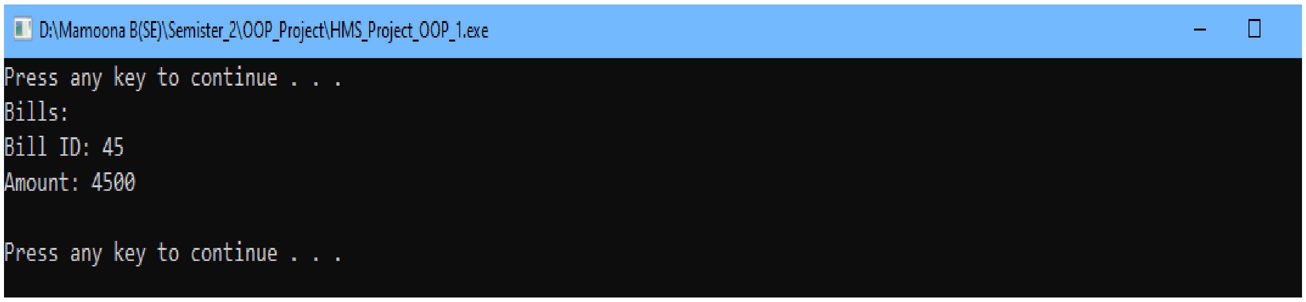
Choice 9 the Staff data is displayed


D:\Mamoona B(SE)\Semister_2\OOP_Project\HMS_Project_OOP_1.exe

```
Press any key to continue . . .
Staff Members:
Staff ID: 34
Name: Ali
Position: Counter

Press any key to continue . . .
```

Choice 10 the Bill detail is displayed

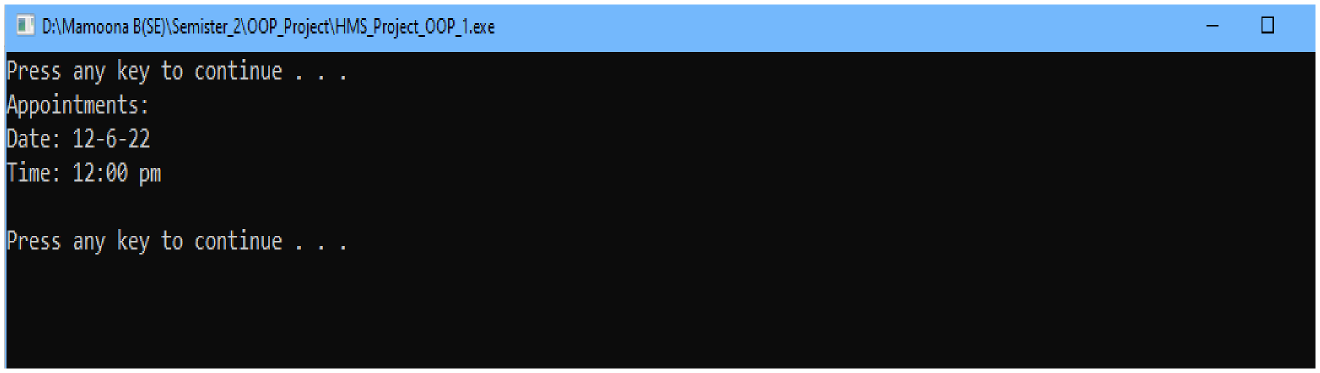


D:\Mamoona B(SE)\Semister_2\OOP_Project\HMS_Project_OOP_1.exe

```
Press any key to continue . . .
Bills:
Bill ID: 45
Amount: 4500

Press any key to continue . . .
```

Choice 11 the Appointment detail is displayed

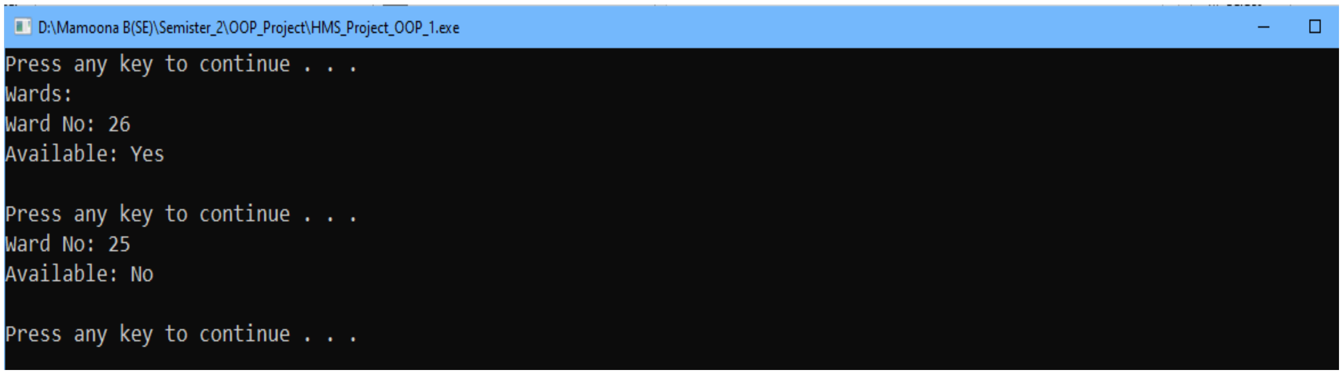


D:\Mamoona B(SE)\Semister_2\OOP_Project\HMS_Project_OOP_1.exe

```
Press any key to continue . . .
Appointments:
Date: 12-6-22
Time: 12:00 pm

Press any key to continue . . .
```

Choice 12 the Ward detail is displayed



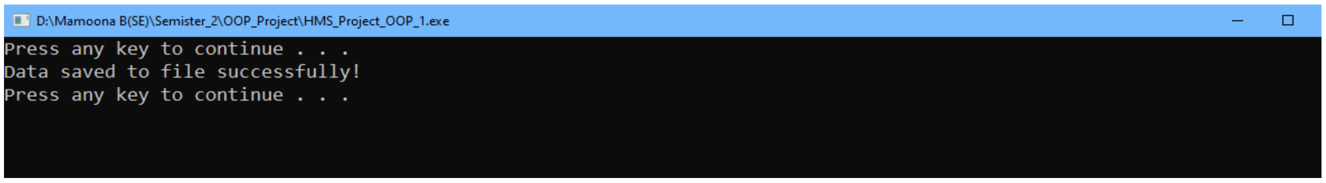
D:\Mamoona B(SE)\Semister_2\OOP_Project\HMS_Project_OOP_1.exe

```
Press any key to continue . . .
Wards:
Ward No: 26
Available: Yes

Press any key to continue . . .
Ward No: 25
Available: No

Press any key to continue . . .
```

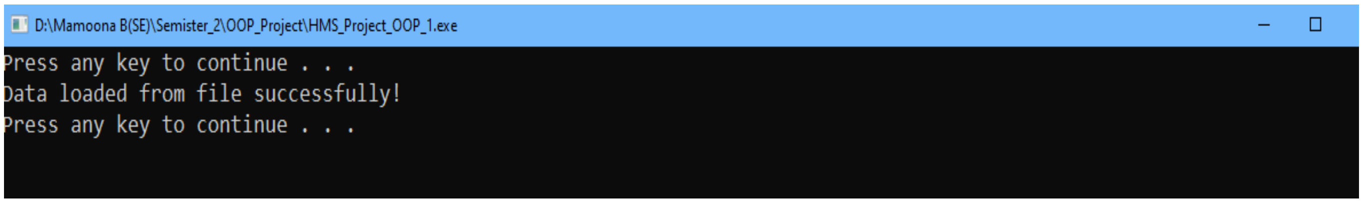
Choice 13 the Save data to file



D:\Mamoona B(SE)\Semister_2\OOP_Project\HMS_Project_OOP_1.exe

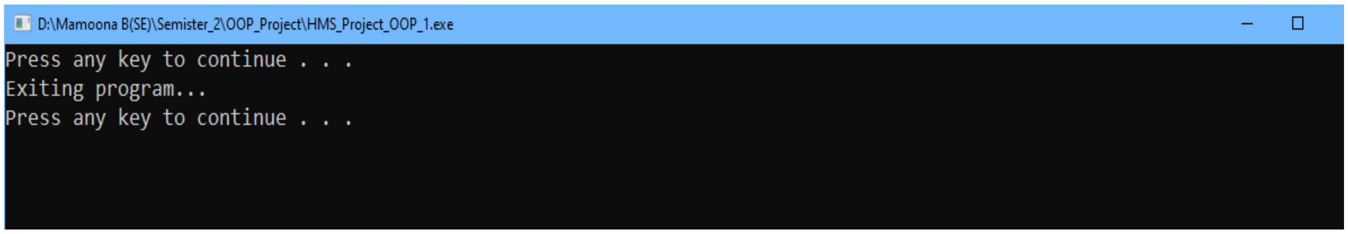
```
Press any key to continue . . .
Data saved to file successfully!
Press any key to continue . . .
```

Choice 14 the Load data to file


D:\Mamoona B(SE)\Semister_2\OOP_Project\HMS_Project_OOP_1.exe

```
Press any key to continue . . .
Data loaded from file successfully!
Press any key to continue . . .
```

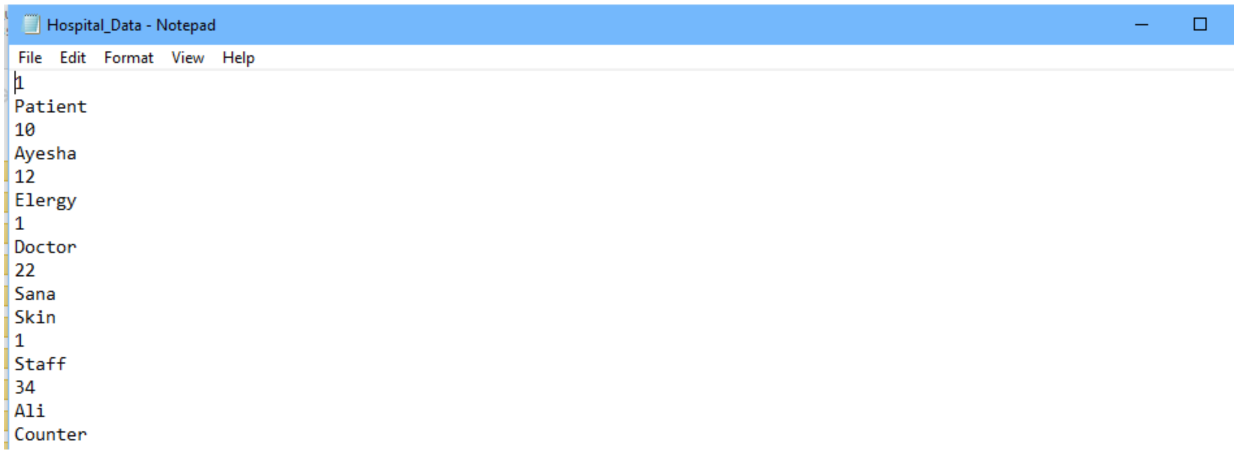
Choice 0 to Exist the Program


D:\Mamoona B(SE)\Semister_2\OOP_Project\HMS_Project_OOP_1.exe

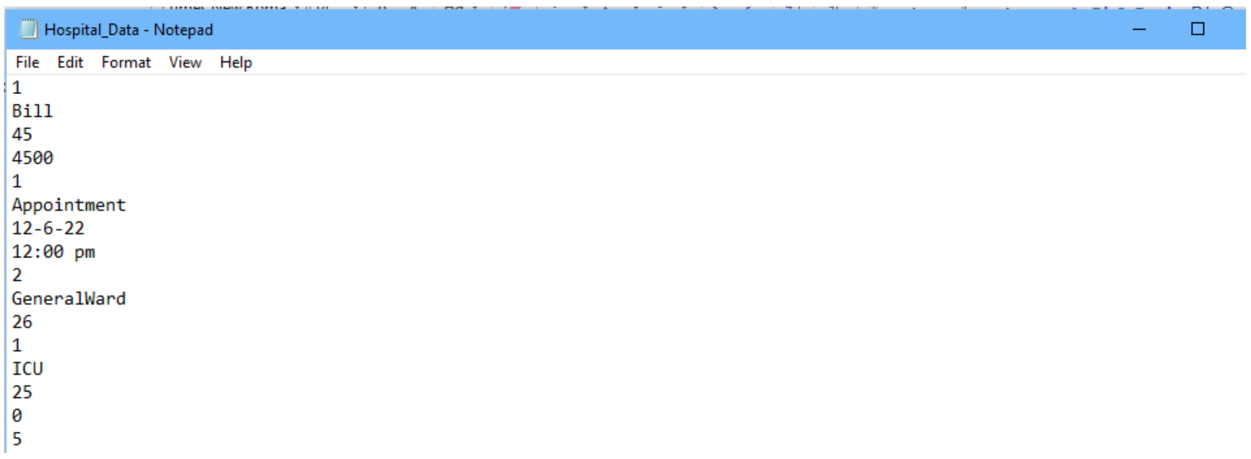
```
Press any key to continue . . .
Exiting program...
Press any key to continue . . .
```

<u>Save Data To File</u>

The entered data is saved in the form of a text file in the computer data.



```
1
Patient
10
Ayesha
12
Elergy
1
Doctor
22
Sana
Skin
1
Staff
34
Ali
Counter
```



Hospital_Data - Notepad

```
1
Bill
45
4500
1
Appointment
12-6-22
12:00 pm
2
GeneralWard
26
1
ICU
25
0
5
```

HOSPITAL MANAGEMENT SYSTEM SOURCE CODE:

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
class Patient {
public:
    int id;
    string name;
    int age;
    string disease;

    Patient(int id, const string& name, int age, const string& disease)
        : id(id), name(name), age(age), disease(disease) {}

    void saveToFile(ofstream& file) {
        file << "Patient" << endl;
        file << id << endl;
        file << name << endl;
        file << age << endl;
        file << disease << endl;
    }

    void loadFromFile(ifstream& file) {
        file >> id;
        file.ignore();
        getline(file, name);
        file >> age;
        file.ignore();
        getline(file, disease);
    }
};

class Doctor {
public:
    int id;
    string name;
    string specialization;

    Doctor(int id, const string& name, const string& specialization)
        : id(id), name(name), specialization(specialization) {}

    void saveToFile(ofstream& file) {
        file << "Doctor" << endl;
        file << id << endl;
        file << name << endl;
        file << specialization << endl;
    }

    void loadFromFile(ifstream& file) {
        file >> id;
        file.ignore();
        getline(file, name);
        getline(file, specialization);
    }
};

class Staff {
public:
    int id;
    string name;
    string position;

    Staff(int id, const string& name, const string& position)
        : id(id), name(name), position(position) {}

    void saveToFile(ofstream& file) {
        file << "Staff" << endl;
        file << id << endl;
        file << name << endl;
        file << position << endl;
    }

    void loadFromFile(ifstream& file) {
        file >> id;
        file.ignore();
        getline(file, name);
        getline(file, position);
    }
};

class Bill {
public:
    int billId;
    double amount;

    Bill(int billId, double amount)
        : billId(billId), amount(amount) {}

    void saveToFile(ofstream& file) {
        file << "Bill" << endl;
        file << billId << endl;
        file << amount << endl;
    }

    void loadFromFile(ifstream& file) {
        file >> billId;
        file >> amount;
    }
};

class Appointment {
```

```
public:
    string date;
    string time;

    Appointment(const string& date, const string& time)
        : date(date), time(time) {}

    void saveToFile(ofstream& file) {
        file << "Appointment" << endl;
        file << date << endl;
        file << time << endl;
    }

    void loadFromFile(ifstream& file) {
        file.ignore();
        getline(file, date);
        getline(file, time);
    }
};

class Ward {
public:
    int wardNo;
    bool available;

    Ward(int wardNo, bool available)
        : wardNo(wardNo), available(available) {}
    virtual void saveToFile(ofstream& file) = 0;
    virtual void loadFromFile(ifstream& file) = 0;
};

class GeneralWard : public Ward {
public:
    GeneralWard(int wardNo, bool available)
        : Ward(wardNo, available) {}

    void saveToFile(ofstream& file) {
        file << "GeneralWard" << endl;
        file << wardNo << endl;
        file << available << endl;
    }

    void loadFromFile(ifstream& file) {
        file >> wardNo;
        file >> available;
    }
};

class ICU : public Ward {
public:
    int unitNo;

    ICU(int wardNo, bool available, int unitNo)
        : Ward(wardNo, available), unitNo(unitNo) {}
    void saveToFile(ofstream& file) {
        file << "ICU" << endl;
        file << wardNo << endl;
        file << available << endl;
        file << unitNo << endl;
    }

    void loadFromFile(ifstream& file) {
        file >> wardNo;
        file >> available;
        file >> unitNo;
    }
};

int main() {
            system("CLS");
    const int MAX_PATIENTS = 100;
    const int MAX_DOCTORS = 100;
    const int MAX_STAFF = 100;
    const int MAX_BILLS = 100;
    const int MAX_APPOINTMENTS = 100;
    const int MAX_WARDS = 100;

    Patient* patients[MAX_PATIENTS];
    Doctor* doctors[MAX_DOCTORS];
    Staff* staffMembers[MAX_STAFF];
    Bill* bills[MAX_BILLS];
    Appointment* appointments[MAX_APPOINTMENTS];
```

```
Ward* wards[MAX_WARDS];

int patientCount = 0;
int doctorCount = 0;
int staffCount = 0;
int billCount = 0;
int appointmentCount = 0;
int wardCount = 0;

int choice;
do {
        system("CLS");
    cout << "Select an option:" << endl;
    cout << "1. Add a new patient" << endl;
    cout << "2. Add a new doctor" << endl;
    cout << "3. Add a new staff member" << endl;
    cout << "4. Add a new bill" << endl;
    cout << "5. Add a new appointment" << endl;
    cout << "6. Add a new ward" << endl;
    cout << "7. Display all patients" << endl;
    cout << "8. Display all doctors" << endl;
    cout << "9. Display all staff members" << endl;
    cout << "10. Display all bills" << endl;
    cout << "11. Display all appointments" << endl;
    cout << "12. Display all wards" << endl;
    cout << "13. Save data to file" << endl;
    cout << "14. Load data from file" << endl;
    cout << "0. Exit" << endl;
```

```
        cout << "Choice: ";
        cin >> choice;
    system("CLS");
    system("pause");
        switch (choice) {
          case 1: {
              // Add a new patient
              int id;
              string name;
              int age;
              string disease;

              cout << "Enter patient ID: ";
              cin >> id;
              cin.ignore();

              cout << "Enter patient name: ";
              getline(cin, name);

              cout << "Enter patient age: ";
              cin >> age;
              cin.ignore();

              cout << "Enter patient disease: ";
              getline(cin, disease);

              patients[patientCount++] = new Patient(id, name, age, disease);
    cout << "Patient added successfully!" << endl;
     system("pause");
    break;
    system("pause");
}
case 2: {
    // Add a new doctor
    int id;
    string name;
    string specialization;

    cout << "Enter doctor ID: ";
    cin >> id;
    cin.ignore();

    cout << "Enter doctor name: ";
    getline(cin, name);

    cout << "Enter doctor specialization: ";
    getline(cin, specialization);

    doctors[doctorCount++] = new Doctor(id, name, specialization);

    cout << "Doctor added successfully!" << endl;
    system("pause");
    break;
            system("pause");
}
```

```
case 3: {
    // Add a new staff member
    int id;
    string name;
    string position;

    cout << "Enter staff ID: ";
    cin >> id;
    cin.ignore();

    cout << "Enter staff name: ";
    getline(cin, name);

    cout << "Enter staff position: ";
    getline(cin, position);

    staffMembers[staffCount++] = new Staff(id, name, position);

    cout << "Staff member added successfully!" << endl;
    system("pause");
    break;
    system("pause");
}
case 4: {
    // Add a new bill
    int  billId;
    double amount;
```

```
        cout << "Enter bill ID: ";
        cin >> billId;

        cout << "Enter bill amount: ";
        cin >> amount;

        bills[billCount++] = new Bill(billId, amount);

        cout << "Bill added successfully!" << endl;
        system("pause");
        break;
        system("pause");
    }
    case 5: {
        // Add a new appointment
        string date;
        string time;

        cout << "Enter appointment date: ";
        cin.ignore();
        getline(cin, date);

        cout << "Enter appointment time: ";
        getline(cin, time);

        appointments[appointmentCount++] = new Appointment(date, time);

        cout << "Appointment added successfully!" << endl;
        system("pause");
        break;
        system("pause");
    }
    case 6: {
        // Add a new ward
        int wardNo;
        bool available;
        int wardType;

        cout << "Enter ward number: ";
        cin >> wardNo;

        cout << "Is the ward available? (0 = No, 1 = Yes): ";
        cin >> available;

        cout << "Enter ward type (1 = General Ward, 2 = ICU): ";
        cin >> wardType;

        if (wardType == 1) {
            wards[wardCount++] = new GeneralWard(wardNo, available);
        } else if (wardType == 2) {
            int unitNo;
            cout << "Enter unit number: ";
            cin >> unitNo;
            wards[wardCount++] = new ICU(wardNo, available, unitNo);
        } else {
            cout << "Invalid ward type!" << endl;
    }

    cout << "Ward added successfully!" << endl;
    system("pause");
    break;
    system("pause");
}
case 7: {

    // Display all patients
    cout << "Patients:" << endl;
    for (int i = 0; i < patientCount; i++) {
        cout << "Patient ID: " << patients[i]->id << endl;
        cout << "Name: " << patients[i]->name << endl;
        cout << "Age: " << patients[i]->age << endl;
        cout << "Disease: " << patients[i]->disease << endl;
        cout << endl;
        system("pause");
    }
    break;
    system("pause");
}

case 8: {
    // Display all doctors
    cout << "Doctors:" << endl;
    for (int i = 0; i < doctorCount; i++) {
        cout << "Doctor ID: " << doctors[i]->id << endl;
        cout << "Name: " << doctors[i]->name << endl;
        cout << "Specialization: " << doctors[i]->specialization << endl;
        cout << endl;
        system("pause");
    }
    break;
    system("pause");
}
case 9: {
    // Display all staff members
    cout << "Staff Members:" << endl;
    for (int i = 0; i < staffCount; i++) {
        cout << "Staff ID: " << staffMembers[i]->id << endl;
        cout << "Name: " << staffMembers[i]->name << endl;
        cout << "Position: " << staffMembers[i]->position << endl;
        cout << endl;
        system("pause");
    }
    break;
    system("pause");
}

case 10: {
    // Display all bills
    cout << "Bills:" << endl;
    for (int i = 0; i < billCount; i++) {
        cout << "Bill ID: " << bills[i]->billId << endl;
        cout << "Amount: " << bills[i]->amount << endl;
        cout << endl;
        system("pause");
    }
    break;
    system("pause");
}

case 11: {
    // Display all appointments
    cout << "Appointments:" << endl;
    for (int i = 0; i < appointmentCount; i++) {
        cout << "Date: " << appointments[i]->date << endl;
        cout << "Time: " << appointments[i]->time << endl;
        cout << endl;
        system("pause");
    }
    break;
    system("pause");
}
case 12: {
    // Display all wards
    cout << "Wards:" << endl;
    for (int i = 0; i < wardCount; i++) {
        cout << "Ward No: " << wards[i]->wardNo << endl;
        cout << "Available: " << (wards[i]->available ? "Yes" : "No") << endl;
        cout << endl;
        system("pause");
    }
```

```
        break;
        system("pause");
    }
    case 13: {
        // Save data to file
        ofstream file("hospital_data.txt");
        if (file.is_open()) {
            // Save patients
            file << patientCount << endl;
            for (int i = 0; i < patientCount; i++) {
                patients[i]->saveToFile(file);
            }

            // Save doctors
            file << doctorCount << endl;
            for (int i = 0; i < doctorCount; i++) {
                doctors[i]->saveToFile(file);
            }

            // Save staff members
            file << staffCount << endl;
            for (int i = 0; i < staffCount; i++) {
                staffMembers[i]->saveToFile(file);
            }

            // Save bills
            file << billCount << endl;
            for (int i = 0; i < billCount; i++) {
                bills[i]->saveToFile(file);
            }

            // Save appointments
            file << appointmentCount << endl;
            for (int i = 0; i < appointmentCount; i++) {
                appointments[i]->saveToFile(file);
            }

            // Save wards
            file << wardCount << endl;
            for (int i = 0; i < wardCount; i++) {
                wards[i]->saveToFile(file);
            }

            file.close();
            cout << "Data saved to file successfully!" << endl;
        } else {
            cout << "Error opening file." << endl;
        }
        system("pause");
        break;
        system("pause");
    }
    case 14: {
        // Load data from file
        ifstream file("hospital_data.txt");
        if (file.is_open()) {
```

```
// Clear existing data
for (int i = 0; i < patientCount; i++) {
    delete patients[i];
}
patientCount = 0;

for (int i = 0; i < doctorCount; i++) {
    delete doctors[i];
}
doctorCount = 0;

for (int i = 0; i < staffCount; i++) {
    delete staffMembers[i];
}
staffCount = 0;

for (int i = 0; i < billCount; i++) {
    delete bills[i];
}
billCount = 0;

for (int i = 0; i < appointmentCount; i++) {
    delete appointments[i];
}
appointmentCount = 0;

for (int i = 0; i < wardCount; i++) {
    delete wards[i];
}
wardCount = 0;

// Load patients
file >> patientCount;
file.ignore();
for (int i = 0; i < patientCount; i++) {
    patients[i] = new Patient(0, "", 0, "");
    patients[i]->loadFromFile(file);
}

// Load doctors
file >> doctorCount;
file.ignore();
for (int i = 0; i < doctorCount; i++) {
    doctors[i] = new Doctor(0, "", "");
    doctors[i]->loadFromFile(file);
}

// Load staff members
file >> staffCount;
file.ignore();
for (int i = 0; i < staffCount; i++) {
    staffMembers[i] = new Staff(0, "", "");
    staffMembers[i]->loadFromFile(file);
}

// Load bills
```

```
file >> billCount;
file.ignore();
for (int i = 0; i < billCount; i++) {
    bills[i] = new Bill(0, 0.0);
    bills[i]->loadFromFile(file);
}

// Load appointments
file >> appointmentCount;
file.ignore();
for (int i = 0; i < appointmentCount; i++) {
    appointments[i] = new Appointment("", "");
    appointments[i]->loadFromFile(file);
}

// Load wards
file >> wardCount;
file.ignore();
for (int i = 0; i < wardCount; i++) {
    string wardType;
    getline(file, wardType);

    if (wardType == "GeneralWard") {
        wards[i] = new GeneralWard(0, false);
        wards[i]->loadFromFile(file);
    } else if (wardType == "ICU") {
        wards[i] = new ICU(0, false, 0);
        wards[i]->loadFromFile(file);
```

```
                } else {
                    cout << "Invalid ward type in file." << endl;
                }
            }

            cout << "Data loaded from file successfully!" << endl;
            file.close();
        } else {
            cout << "Error opening file." << endl;
        }
        system("pause");
        break;
        system("pause");
    }
    case 0: {
        // Exit the program
        cout << "Exiting program..." << endl;
        system("pause");
        break;
        system("pause");
    }
    default:
        cout << "Invalid choice. Please try again." << endl;
}
} while (choice != 0);

// Clean up memory
for (int i = 0; i < patientCount; i++) {
        delete patients[i];
    }

    for (int i = 0; i < doctorCount; i++) {
        delete doctors[i];
    }

    for (int i = 0; i < staffCount; i++) {
        delete staffMembers[i];
    }

    for (int i = 0; i < billCount; i++) {
        delete bills[i];
    }

    for (int i = 0; i < appointmentCount; i++) {
        delete appointments[i];
    }

    for (int i = 0; i < wardCount; i++) {
        delete wards[i];
    }

    return 0;
}
```