UNIVERSITY OF LIBERAL ARTS BANGLADESH

“C++” PROJECT REPORT ON

“COVID-19 AFFECTED PATIENTS

MANAGEMENT SYSTEM

FOR BANGLADESH (CAPMSB)”

Project Title	**:**	COVID-19 Affected Patients Management System for Bangladesh(CAPMSB)
Course Title	**:**	Object Oriented Programming-1 Lab
Course Code	**:**	CSE 202
Section	**:**	01

Submitted by :

Student Name : MD.Mutasim Billah Abu Noman Akanda

Student ID : 192 014 038

Submitted to :

Mr. Mohammad Rifat Ahmed Rashid

TABLE OF CONTENTS

Overview

We all are known about the current pandemic situation due to **COVID-19** which is also known as Novel Corona Virus. The most dangerous virus till now named **Corona** has spread out all over the world including our motherland Bangladesh. At the time of writing this report, according to **IEDCR**, almost 803 people are affected by this virus in Bangladesh. The numbers of affected patients are increasing acutely day by day. I have noticed that to keep the records of these patients, government authorities, hospitals authorities, journalists and even IEDCR are using pen and paper or Microsoft excel which is quite backward comparing to the modern world because in this modern era of the world, everything is technology based. Under these circumstances, I have made this project so that the IT sector of Bangladesh government can build an application using the source code which can be used smartly and easily to keep the patients records for Bangladesh. This project has 3 major features-

- ➢ **User can see the whole patients records of Bangladesh,**
- ➢ **User can search a patient by his or her name/email/phone number/address etc. &**
- ➢ **User can give input of new multiple patient records.**

All the features are done carefully with **file handling** as a result all the data of users and patients will be recorded in two different text files. However, these records can be used to survey or to make articles about corona as well as if Bangladesh government wants to help corona affected patients or their family, they can use this records as every details of the patients are recorded in file.

Object-Oriented Programming Features

First of all, I would like to give a short definition what **Object-oriented programming (OPP)** is. Basically, Object-oriented programming (OOP) is a programming language model that organizes software design around data or objects rather than functions and logic. An **object** can be defined as a data field that has unique attributes and behavior. There are mainly **4 (four)** features of Object-oriented programming (OPP) -

- ➢ **Encapsulation,**
- ➢ **Inheritance,**
- ➢ **Polymorphism &**
- ➢ **Abstraction.**

You will be glad to know that I have tried my best to implement all of the OPP features in my project. Here is the short description of my implementation of OPP features-

Encapsulation

Encapsulation is an Object Oriented Programming concept that binds together the data and functions that manipulate the data, and that keeps both safe from outside interference and misuse. Data encapsulation led to the important OOP concept of **“data hiding**”. Data encapsulation is a mechanism of bundling the data and the functions that use them. C++ supports the properties of **encapsulation** through the creation of user-defined types called **classes**.

Here is a piece of my source code where I have implemented the concept of **“Encapsulation”-**

```
//creating class new_patients_records which perform the task of taking input of new patients records and display them

class new_patients_records

{

private:

//new_patient_name,new_patient_profession,new_patient_address,new_patient_age,new_patient_phone_number,new_patient_email,new_patient_district class attribute to store patients informations

  string new_patient_name,new_patient_profession,new_patient_address,new_patient_age,new_patient_phone_number,new_patient_email,new_patient_district,new_patient_sex,new_patient_date;

public:

   //setter method of new_patient_name declaration

   void set_new_patient_name(string npn);

   //getter method of new_patient_name declaration

   string get_new_patient_name();

   //setter method of new_patient_district declaration

   void set_new_patient_district(string npd);

   //getter method of new_patient_district declaration

   string get_new_patient_district();
```

```
//setter method of new_patient_address declaration

void set_new_patient_address(string npad);

//getter method of new_patient_address declaration

string get_new_patient_address();

//setter method of new_patient_email declaration

void set_new_patient_email(string npe);

//getter method of new_patient_email declaration

string get_new_patient_email();

//setter method of new_patient_profession declaration

void set_new_patient_profession(string npp);

//getter method of new_patient_profession declaration

string get_new_patient_profession();

//setter method of new_patient_phone_number declaration

void set_new_patient_phone_number(string nppn);

//getter method of new_patient_phone_number declaration

string get_new_patient_phone_number();

//setter method of new_patient_age declaration

void set_new_patient_age(string npa);

//getter method of new_patient_age declaration

string get_new_patient_age();

//setter method of new_patient_sex declaration

void set_new_patient_sex(string nps);

//getter method of new_patient_sex declaration

string get_new_patient_sex();
```

```
    //setter method of new_patient_date declaration
    void set_new_patient_date(string npdt);
    //getter method of new_patient_date declaration
    string get_new_patient_date();
    //using pure virtual function display() declaration
    void display();
};
//setter method of new_patient_name under new_patients_records definition using scope resolution operator
void new_patients_records :: set_new_patient_name(string npn)
{
    new_patient_name = npn;
}
//getter method of new_patient_name under new_patients_records definition using scope resolution operator
string new_patients_records :: get_new_patient_name()
{
    return new_patient_name;
}
//setter method of new_patient_district under new_patients_records definition using scope resolution operator
void new_patients_records :: set_new_patient_district(string npd)
{
    new_patient_district = npd;
}
```

```
//getter method of new_patient_district under new_patients_records definition using scope resolution operator

string new_patients_records :: get_new_patient_district()

{

   return new_patient_district;

}

//setter method of new_patient_address under new_patients_records definition using scope resolution operator

void new_patients_records :: set_new_patient_address(string npad)

{

   new_patient_address = npad;

}

//getter method of new_patient_address under new_patients_records definition using scope resolution operator

string new_patients_records :: get_new_patient_address()

{

   return new_patient_address;

}

//setter method of new_patient_email under new_patients_records definition using scope resolution operator

void new_patients_records :: set_new_patient_email(string npe)

{

   new_patient_email = npe;

}

//getter method of new_patient_email under new_patients_records definition using scope resolution operator
```

```
string new_patients_records :: get_new_patient_email()
{
    return new_patient_email;
}
//setter method of new_patient_profession under new_patients_records definition using scope resolution operator
void new_patients_records :: set_new_patient_profession(string npp)
{
    new_patient_profession = npp;
}
//getter method of new_patient_profession under new_patients_records definition using scope resolution operator
string new_patients_records :: get_new_patient_profession()
{
    return new_patient_profession;
}
//setter method of new_patient_phone_number under new_patients_records definition using scope resolution operator
void new_patients_records :: set_new_patient_phone_number(string nppn)
{
    new_patient_phone_number = nppn;
}
//getter method of new_patient_phone_number under new_patients_records definition using scope resolution operator
string new_patients_records :: get_new_patient_phone_number()
{
    return new_patient_phone_number;
}
//setter method of new_patient_age under new_patients_records definition using scope resolution operator
void new_patients_records :: set_new_patient_age(string npa)
{
    new_patient_age = npa;
}
//getter method of new_patient_age under new_patients_records definition using scope resolution operator
string new_patients_records :: get_new_patient_age()
{
    return new_patient_age;
}
//setter method of new_patient_sex under new_patients_records definition using scope resolution operator
void new_patients_records :: set_new_patient_sex(string nps)
{
    new_patient_sex = nps;
}
//getter method of new_patient_sex under new_patients_records definition using scope resolution operator
string new_patients_records :: get_new_patient_sex()
{
    return new_patient_sex;
}
```

```
//setter method of new_patient_date under new_patients_records definition using scope resolution operator

void new_patients_records :: set_new_patient_date(string npdt)

{

    new_patient_date = npdt;

}

//getter method of new_patient_date under new_patients_records definition using scope resolution operator

string new_patients_records :: get_new_patient_date()

{

    return new_patient_date;

}
```

Inheritance

One of the most important concepts in object-oriented programming is that of **inheritance**. Inheritance allows us to define a class in terms of another class, which makes it easier to create and maintain an application. This also provides an opportunity to reuse the code functionality and fast implementation time.

When creating a class, instead of writing completely new data members and member functions, the programmer can designate that the new class should inherit the members of an existing class. This existing class is called the **base class** and the new class is referred to as the **derived class.**

Here is a piece of my source code where I have implemented the concept of **“Inheritance”-**

```
//derived class derived_user_details from the abstract base class base_user_details

class derived_user_details : public base_user_details

{

public:
    //using pure virtual function display() declaration

    void display();

};

//pure virtual function under derived_user_details class, display() definition using scope resolution operator

void derived_user_details :: display()

{

    //using ofstream mode to create a file stored in user_file

    ofstream user_file;

    //opening the file "User_details.txt" in append mood to write data

    user_file.open("User_details.txt",ios::out|ios::app);

    cout << "User Informations Successfully Taken : " << endl;

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

    //calling getter method of user_name

    cout << "Name : " << get_user_name() << endl;

    //writing user_name in file

    user_file << "Name : " << get_user_name() << endl;

    //calling getter method of user_address

    cout << "Address : " << get_user_address() << endl;

    //writing user_address in file

    user_file << "Address : " << get_user_address() << endl;

    //calling getter method of user_district

    cout << "District : " << get_user_district() << endl;
```

```
//writing user_district in file

user_file << "District : " << get_user_district() << endl;

//calling getter method of user_email

cout << "Email : " << get_user_email() << endl;

//writing user_email in file

user_file << "Email : " << get_user_email() << endl;

//calling getter method of user_proffession

cout << "Profession : " << get_user_proffession() << endl;

//writing user_proffession in file

user_file << "Profession : " << get_user_proffession() << endl;

//calling getter method of user_phone_number

cout << "Phone Number : " << "+880" << get_user_phone_number() << endl;

//writing user_phone_number in file

user_file << "Phone Number : " << "+880" << get_user_phone_number() << endl;

//calling getter method of user_age

cout << "Age : " << get_user_age() << endl;

//writing user_age in file

user_file << "Age : " << get_user_age() << endl;

//calling getter method of user_sex

cout << "Sex : " << get_user_sex() << endl;

//writing user_sex in file

user_file << "Sex : " << get_user_sex() << endl;
```

```
    //calling getter method of user_date

    cout << "Date : " << get_user_date() << endl << endl;

    //writing user_date in file

    user_file << "Date : " << get_user_date() << endl;

    user_file << "**************************************************" << endl;

    //closing the file "User_details.txt"

    user_file.close();

}
```

Abstraction

Data abstraction refers to providing only essential information to the outside world and hiding their background details to represent the needed information in program without presenting the details. It is a programming (and design) technique that relies on the separation of interface and implementation. In C++, classes provide great level of **data abstraction**. They provide sufficient public methods to the outside world to play with the functionality of the object and to manipulate object data state without actually knowing how class has been implemented internally.

Data abstraction provides two important advantages-

- **Class internals are protected from inadvertent user-level errors, which might corrupt the state of the object &**
- **The class implementation may evolve over time in response to changing requirements or bug reports without requiring change in user-level code.**

Here is a piece of my source code where I have implemented the concept of **“Abstraction”-**

```
//abstract base class base_user_details which perform the indirect input taking from user

class base_user_details

{

private:
```

```
//user_name,user_address,user_email,user_proffesion,user_phone_number,user_age,user_district class attributes to store accordingly name,address,email,profession,phone number,age & district of user

    string
user_name,user_address,user_email,user_proffesion,user_phone_number,user_age,user_district,
user_sex,user_date;

public:

    //constructor base_user_details() declaration

    base_user_details();

    //setter method of user_name declaration

    void set_user_name(string un);

    //getter method of user_name declaration

    string get_user_name();

    //setter method of user_district declaration

    void set_user_district(string udt);

    //getter method of user_district declaration

    string get_user_district();

    //setter method of user_address declaration

    void set_user_address(string uad);

    //getter method of user_address declaration

    string get_user_address();

    //setter method of user_email declaration

    void set_user_email(string ue);

    //getter method of user_email declaration

    string get_user_email();
    //setter method of user_proffession declaration
    void set_user_proffession(string up);
    //getter method of user_proffession declaration
    string get_user_proffession();
    //setter method of user_phone_number declaration
    void set_user_phone_number(string upn);
    //getter method of user_phone_number declaration
    string get_user_phone_number();
    //setter method of user_age declaration
    void set_user_age(string ua);
    //getter method of user_age declaration
    string get_user_age();
    //setter method of user_sex declaration
    void set_user_sex(string us);
    //getter method of user_age declaration
    string get_user_sex();
    //setter method of user_date declaration
    void set_user_date(string usd);
    //getter method of user_age declaration
    string get_user_date();

    //pure virtual function display() declaration
    virtual void display() = 0;
};
```

And I have access this **abstract class** from the main function using **pointer**. Here is the piece of the main function from my source code where I access the abstract class from the main function-

```
//main() function to run the program

int main()

{

    //declaring the pointer of abstract class base_user_details

    base_user_details *bud;

    //creating an object of derived class derived_user_details from abstract class base_user_details
class

    derived_user_details ud;

    //passing the address of derived_user_details object to the pointer of base_user_details

    bud = &ud;

    //calling input() function

    input();

    //calling setter method of user_name by pointer of base_user_details

    bud -> set_user_name(name);

    //calling setter method of user_address by pointer of base_user_details

    bud -> set_user_address(address);

    //calling setter method of user_district by pointer of base_user_details

    bud -> set_user_district(district);

    //calling setter method of user_email by pointer of base_user_details

    bud -> set_user_email(email);

    //calling setter method of user_proffession by pointer of base_user_details

    bud -> set_user_proffession(proffesion);

    //calling setter method of user_age by pointer of base_user_details
    bud -> set_user_age(age);

    //calling setter method of user_phone_number by pointer of base_user_details

    bud -> set_user_phone_number(phone_number);

    //calling setter method of user_sex by pointer of base_user_details

    bud -> set_user_sex(sex);

    //calling setter method of user_date by pointer of base_user_details

    bud -> set_user_date(date);

    //calling display member function under abstract base class by pointer of base_user_details

    bud -> display();

    return 0;

}
```

Polymorphism

The word **polymorphism** means having many forms. Typically, polymorphism occurs when there is a hierarchy of classes and they are related by inheritance. “C++” polymorphism means that a call to a member function will cause a different function to be executed depending on the type of object that invokes the function.

Such as I have used same member function declaration named **void display()** but different function definition which is **run time polymorphism** (Function overriding). Here is a piece of my source code where I have implemented the concept of **“Polymorphism”-**

```
//derived class derived_user_details from the abstract base class base_user_details

class derived_user_details : public base_user_details

{

public:

    void display();   //skipping the function definition

};
```

```
//creating class new_patients_records which perform the task of taking input of new patients records and display them

class new_patients_records

{

public:

   void display();    //skipping the function definition

};
```

Module Description

I am going to explain the whole source code of my project **“COVID-19 Affected Patients Management System for Bangladesh (CAPMSB)”** step by step. So here I go-

Step 1.User part:

When the program will be run by any user, First of all, the project title which is considered as the application name **“COVID-19 Affected Patients Management System for Bangladesh (CAPMSB)”** will be displayed and the application will ask some information from user such as user name, profession, address, email etc. to use the application. If user wants to skip any information input then an error message will be thrown and again ask for the information. The all user information will be saved in a text file named **“User_details.txt”.** After entering user details successfully, all information will be displayed as per user input to ensure the user that he/she gives the required information correctly. The source code of this **step 1** is given below-

```
//abstract base class base_user_details which perform the indirect input taking from user

class base_user_details

{

private:


//user_name,user_address,user_email,user_proffesion,user_phone_number,user_age,user_district class attributes to store accordingly name,address,email,profession,phone number,age & district of user
```

```
    string
user_name,user_address,user_email,user_proffesion,user_phone_number,user_age,user_district,
user_sex,user_date;

public:

    //constructor base_user_details() declaration

    base_user_details();

    //setter method of user_name declaration

    void set_user_name(string un);

    //getter method of user_name declaration

    string get_user_name();

    //setter method of user_district declaration

    void set_user_district(string udt);

    //getter method of user_district declaration

    string get_user_district();

    //setter method of user_address declaration

    void set_user_address(string uad);

    //getter method of user_address declaration

    string get_user_address();

    //setter method of user_email declaration

    void set_user_email(string ue);

    //getter method of user_email declaration

    string get_user_email();

    //setter method of user_proffession declaration

    void set_user_proffession(string up);

    //getter method of user_proffession declaration
    string get_user_proffession();
    //setter method of user_phone_number declaration
    void set_user_phone_number(string upn);
    //getter method of user_phone_number declaration
    string get_user_phone_number();
    //setter method of user_age declaration
    void set_user_age(string ua);
    //getter method of user_age declaration
    string get_user_age();
    //setter method of user_sex declaration
    void set_user_sex(string us);
    //getter method of user_age declaration
    string get_user_sex();
    //setter method of user_date declaration
    void set_user_date(string usd);
    //getter method of user_age declaration
    string get_user_date();
    //pure virtual function display() declaration
    virtual void display() = 0;
};
//constructor base_user_details() under base_user_details definition using scope resolution
operator
base_user_details :: base_user_details()
{
    cout << "                    ****************************************" << endl;
    cout << "                    * COVID-19 AFFECTED PATIENTS MANAGEMENT *" << endl;
    cout << "                    *     SYSTEM FOR BANGLADESH (CAPMSB)    *" << endl;
    cout << "                    ****************************************" << endl << endl << endl;
    cout << "To use this application,you need to give your required informations given below : " << endl;
    cout << "---------------------------------------------------------------------------" << endl << endl;
}
//setter method of user_name under base_user_details definition using scope resolution operator
void base_user_details :: set_user_name(string un)
{
    user_name = un;
}
//getter method of user_name under base_user_details definition using scope resolution operator
string base_user_details :: get_user_name()
{
    return user_name;
}
//setter method of user_district under base_user_details definition using scope resolution operator
void base_user_details :: set_user_district(string udt)
{
    user_district = udt;
}
```

```
//getter method of user_district under base_user_details definition using scope resolution operator

string base_user_details :: get_user_district()

{

   return user_district;

}


//setter method of user_address under base_user_details definition using scope resolution operator

void base_user_details :: set_user_address(string uad)

{

   user_address = uad;

}

//getter method of user_address under base_user_details definition using scope resolution operator

string base_user_details :: get_user_address()

{

   return user_address;

}

//setter method of user_email under base_user_details definition using scope resolution operator

void base_user_details :: set_user_email(string ue)

{

   user_email = ue;

}

//getter method of user_email under base_user_details definition using scope resolution operator
```

```
string base_user_details :: get_user_email()

{

    return user_email;

}

//setter method of user_proffesion under base_user_details definition using scope resolution operator

void base_user_details :: set_user_proffession(string up)

{

    user_proffesion = up;

}

//getter method of user_proffesion under base_user_details definition using scope resolution operator

string base_user_details :: get_user_proffession()

{

    return user_proffesion;

}

//setter method of user_phone_number under base_user_details definition using scope resolution operator

void base_user_details :: set_user_phone_number(string upn)

{

    user_phone_number = upn;

}

//getter method of user_phone_number under base_user_details definition using scope resolution operator

string base_user_details :: get_user_phone_number()

{
    return user_phone_number;
}
//setter method of user_age under base_user_details definition using scope resolution operator
void base_user_details :: set_user_age(string ua)
{
    user_age = ua;
}
//getter method of user_age under base_user_details definition using scope resolution operator
string base_user_details :: get_user_age()
{
    return user_age;
}
//setter method of user_sex under base_user_details definition using scope resolution operator
void base_user_details :: set_user_sex(string us)
{
    user_sex = us;
}
//getter method of user_sex under base_user_details definition using scope resolution operator
string base_user_details :: get_user_sex()
{
    return user_sex;
}
//setter method of user_date under base_user_details definition using scope resolution operator
void base_user_details :: set_user_date(string usd)
```

```
{
    user_date = usd;
}
//getter method of user_date under base_user_details definition using scope resolution operator
string base_user_details :: get_user_date()
{
    return user_date;
}
//derived class derived_user_details from the abstract base class base_user_details
class derived_user_details : public base_user_details
{
public:
    //using pure virtual function display() declaration
    void display();
};
//pure virtual function under derived_user_details class, display() definition using scope resolution operator
void derived_user_details :: display()
{
    //using ofstream mode to create a file stored in user_file
    ofstream user_file;
    //opening the file "User_details.txt" in append mood to write data
    user_file.open("User_details.txt",ios::out|ios::app);
    cout << "User Informations Successfully Taken : " << endl;
    cout << "-------------------------------------"  << endl << endl;

    //calling getter method of user_name

    cout << "Name : " << get_user_name() << endl;

    //writing user_name in file

    user_file << "Name : " << get_user_name() << endl;

    //calling getter method of user_address

    cout << "Address : " << get_user_address() << endl;

    //writing user_address in file

    user_file << "Address : " << get_user_address() << endl;

    //calling getter method of user_district

    cout << "District : " << get_user_district() << endl;

    //writing user_district in file

    user_file << "District : " << get_user_district() << endl;

    //calling getter method of user_email

    cout << "Email : " << get_user_email() << endl;

    //writing user_email in file

    user_file << "Email : " << get_user_email() << endl;

    //calling getter method of user_proffession

    cout << "Profession : " << get_user_proffession() << endl;

    //writing user_proffession in file

    user_file << "Profession : " << get_user_proffession() << endl;

    //calling getter method of user_phone_number

    cout << "Phone Number : " << "+880" << get_user_phone_number() << endl;

    //writing user_phone_number in file
```

```
    user_file << "Phone Number : " << "+880" << get_user_phone_number() << endl;
    //calling getter method of user_age
    cout << "Age : " << get_user_age() << endl;
    //writing user_age in file
    user_file << "Age : " << get_user_age() << endl;
    //calling getter method of user_sex
    cout << "Sex : " << get_user_sex() << endl;
    //writing user_sex in file
    user_file << "Sex : " << get_user_sex() << endl;
    //calling getter method of user_date
    cout << "Date : " << get_user_date() << endl << endl;
    //writing user_date in file
    user_file << "Date : " << get_user_date() << endl;
    user_file << "****************************************************" << endl;
    //closing the file "User_details.txt"
    user_file.close();
}
//global function input() definition to take user's informations
void input()
{
    cout << "Enter name(according to the NID CARD/BIRTH CERTIFICATE) : ";
    getline(cin,name);
    //while loop if user skips to fill the name
    while(name=="")
```

```
{
   //calling show_error() function
   show_error();
   cout << "Enter name : ";
   getline(cin,name);
}
cout << "Enter profession : ";
getline(cin,proffesion);
//while loop if user skips to fill the proffesion
while(proffesion=="")
{
   //calling show_error() function
   show_error();
   cout << "Enter profession : ";
   getline(cin,proffesion);
}
cout << "Enter address : ";
getline(cin,address);
//while loop if user skips to fill the address
while(address=="")
{
   //calling show_error() function
   show_error();
   cout << "Enter address : ";
```

```
    getline(cin,address);
}
cout << "Enter district : ";
getline(cin,district);
//while loop if user skips to fill the district
while(district=="")
{
    //calling show_error() function
    show_error();
    cout << "Enter district : ";
    getline(cin,district);
}
cout << "Enter your email : ";
getline(cin,email);
//while loop if user skips to fill the email
while(email=="")
{
    //calling show_error() function
    show_error();
    cout << "Enter email : ";
    getline(cin,email);
}
cout << "Enter age : ";
getline(cin,age);
```

```
//while loop if user skips to fill the age or assign age as 0
while(age<="0" || age=="")
{
   //calling show_error() function
   show_error();
   cout << "Enter age(years) : ";
   getline(cin,age);
}
cout << "Enter phone number : ";
getline(cin,phone_number);
//while loop if user skips to fill the phone_number
while(phone_number=="")
{
   //calling show_error() function
   show_error();
   cout << "Enter phone number : ";
   getline(cin,phone_number);
}
cout << "Enter Sex (Male/Female/3rd gender) : ";
getline(cin,sex);
//while loop if user skips to fill the phone_number
while(sex=="")
{
   //calling show_error() function
```

```
        show_error();
        cout << "Enter Sex (Male/Female/3rd gender) : ";
        getline(cin,sex);
    }
    cout << "Enter date(dd/mm/yy) : ";
    getline(cin,date);
    //while loop if user skips to fill the phone_number
    while(date=="")
    {
        //calling show_error() function
        show_error();
        cout << "Enter date(dd/mm/yy) : ";
        getline(cin,date);
    }
    cout << endl << endl;
}

//global function show_error() definition to display error message to user
void show_error()
{
    cout << endl << "Please fill the required information!!!" << endl;
}
```

Step 2.Menu section:

After completing the step 1 successfully, a **menu bar** will be displayed and the application will ask user which feature he/she wants to use. If user gives wrong input without the mentioned key, an error will be thrown and again user will be asked for the desired feature key. There are 4 options in the menu section-

- User can see the whole patient records of Bangladesh,
- User can search any patient by their name/phone number/email etc.,
- User can enter new patient records as many he/she wants &
- Exit option.

The source code of this **step 2** is given below-

```
//global function features() definition to display menu section & take input from user which feature is need to use

void features()

{

   cout << "                              |------|" << endl;

   cout << "                              | MENU |" << endl;

   cout << "                              |------|" << endl << endl;

   cout << "       ***************************************************************" << endl;

   cout << "       *  => To see the patients records of Bangladesh : enter 1 *" << endl;

   cout << "       *  => To search any patient : enter 2                    *" << endl;

   cout << "       *  => To enter a new patient record : enter 3            *" << endl;

   cout << "       *  => To exit : enter 4                                  *" << endl;

   cout << "       ***************************************************************" << endl << endl;

   cout << "Enter your choice : ";

   //fflush() which is defined in header file <stdio.h> is used to clear the output buffer

   fflush(stdin);
```

```
    cin >> features_counter;

    //while loop to correct user wrong input of choosing feature

    while((features_counter!="1" && features_counter!="2" && features_counter!="3" &&
features_counter!="4") || (features_counter==""))

    {

        cout << endl << "ERROR! Invalid choice!! Try again!!!" << endl;

        cout << "Enter your choice : ";

        fflush(stdin);

        cin >> features_counter;

    }

}
```

Step 3.Option 1 from menu section:

If user wants to use option 1, he/she will enter 1 as input and then compiler will read every single line from the text file **"Patient_details.txt"** and display the whole data of affected patients. After showing the whole data, the menu bar will again appear before the user so that he/she can use the other options as well. The source code of this **step 3** is given below-

```
//class show_patient_records to perform the task of displaying the patients records of Bangladesh

class show_patient_records

{

public:

    //record class attribute

    string record;

    //constructor show_patient_records declaration

    show_patient_records();

};
```

```
//constructor show_patient_records definition under class show_patient_records using scope resolution operator

show_patient_records :: show_patient_records()

{

   //opening "Patient_details.txt" file in reading mode stored in file variable

   ifstream file("Patient_details.txt");

   //while loop until the lines saved in the file are not taken as input

   while(getline(file,record))

   {

      cout << record << endl;

   }

   //closing "Patient_details.txt" file

   file.close();

}
```

<u>Step 4.Option 2 from menu section:</u>

After appearing the menu bar again, if user wants to use option 2, he/she will enter 2 and then the program will ask the desired patient name/phone number/email whomever user wants to search from the patient records. After entering the desired patient name/phone number/email, compiler will read the whole data line by line from the text file **"Patient_details.txt"** and compare with the entered data by user. Before comparing the data, compiler will convert the both data into uppercase because as we know that programming language is case sensitive. Comparing the both data, if any data from the file is matched with the entered data by user, the compiler will show that the patient is found in the records. Otherwise, compiler will show that the patient is not found in the records. After searching a patient successfully, the menu bar will again appear before the user so that he/she can use the other options as well. The source code of this **step 4** is given below-

```
//class search_patient_records to do the task of searching the inputted patient from user

class search_patient_records

{

private:

    //search_patient1 class attribute

    string search_patient1;

public:

    //constructor search_patient_records declaration

    search_patient_records();

    //setter method of search_patient1 declaration

    void set_search_patient(string sp);

    //getter method of search_patient1 declaration

    string get_search_patient();

    //check_records() member function declaration  to search the inputted patient from user

    void check_records();

};

//constructor search_patient_records definition under class search_patient_records using scope
resolution operator

search_patient_records :: search_patient_records()

{

    cout << endl << "To search a patient,enter his/her name or phone number(including country
code +880) or email id : " << endl;

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

}
```

```
//setter method of search_patient1 definition under class search_patient_records using scope
resolution operator

void search_patient_records :: set_search_patient(string sp)

{

    search_patient1 = sp;

}

//getter method of search_patient1 definition under class search_patient_records using scope
resolution operator

string search_patient_records :: get_search_patient()

{

    return search_patient1;

}

//check_records() member function definition under class search_patient_records using scope
resolution operator

void search_patient_records :: check_records()

{

    //opening "Patient_details.txt" file in reading mode stored in FILE

    ifstream FILE("Patient_details.txt");

    //local variables data to store the string read from the file,string type array name[1000] to store
the values of data serially & temp to store the values of name[1000] for temporary

    string data,temp,name[1000];

    //declaring i & j to continue the for loop

    int i=0,j;

    //while loop to read the data from the file until the all data are taken as input

    while(FILE)

    {
        //taking the data line by line

        getline(FILE,data);

        //storing the values of data serially in string type array name

        name[i++] = data;

    }

    //for loop to convert the whole data in upper case

    for(j=0 ; j<i ; j++)

    {

        //storing the array in temp individually

        temp = name[j];

        //converting the data stored in temp character by character,auto type is used so that compiler
can detect any kind of data type

        for(auto &c : temp)

        {

            c = toupper(c);

        }

        //converting the data stored in search_patient1 character by character,auto type is used so
that compiler can detect any kind of data type

        for(auto &c : search_patient1)

        {

            c = toupper(c);

        }

        //finding the index number where search_patient1 is found comparing with temp and index
number is stored in found

        auto found = temp.find(search_patient1);
        //this will execute when index number stored in found will be smaller than the length of temp

        if(found < temp.length())

        {

            cout << endl << "Search Result : Patient is found in the records" << endl;

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

            //breaking the loop when the patient will be found

            break;

        }

        else

            cout << endl << "Search Result : Patient is not found in the records" << endl;

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

    }

    //closing the file "Patient_details.txt"

    FILE.close();

}
```

<u>Step 5.Option 3 from menu section:</u>

After appearing the menu bar again, if user wants to use option 3, he/she will enter 3 and then the program will ask how many patient records he/she wants to submit. After entering the new patient number, compiler will ask the patient details for the entered number of new patients. After completing the task of entering new patient records, the entered details will be displayed before user to ensure him/her that the records are successfully taken. Then, the menu bar will again appear before the user so that he/she can use the previous options again or simply exit. The source code of **step 5** is given below-

```
//global function input_patient_number() definition to take input from user how many patients records he/she wants to submit
```

```
void input_patient_number()
{
    cout << endl << "Enter how many patients records will you submit(in number) : ";
    cin >> new_patients_number;
    //cin.ignore() function which is used to clear one or more characters from the input buffer
    cin.ignore();
    cout << endl;
    //while loop if user give input 0
    while(new_patients_number<=0)
    {
        //calling show_error() function
        show_error();
        cout << "Enter how many patients records will you submit : ";
        cin >> new_patients_number;
        //cin.ignore() function which is used to clear one or more characters from the input buffer
        cin.ignore();
        cout << endl;
    }
}
//creating class new_patients_records which perform the task of taking input of new patients records and display them
class new_patients_records
{
private:
```

```
//new_patient_name,new_patient_profession,new_patient_address,new_patient_age,new_patient
_phone_number,new_patient_email,new_patient_district class attribute to store patients
informations

    string
new_patient_name,new_patient_profession,new_patient_address,new_patient_age,new_patient_
phone_number,new_patient_email,new_patient_district,new_patient_sex,new_patient_date;

public:

    //setter method of new_patient_name declaration

    void set_new_patient_name(string npn);

    //getter method of new_patient_name declaration

    string get_new_patient_name();

    //setter method of new_patient_district declaration

    void set_new_patient_district(string npd);

    //getter method of new_patient_district declaration

    string get_new_patient_district();

    //setter method of new_patient_address declaration

    void set_new_patient_address(string npad);

    //getter method of new_patient_address declaration

    string get_new_patient_address();

    //setter method of new_patient_email declaration

    void set_new_patient_email(string npe);

    //getter method of new_patient_email declaration

    string get_new_patient_email();

    //setter method of new_patient_profession declaration

    void set_new_patient_profession(string npp);
    //getter method of new_patient_profession declaration

    string get_new_patient_profession();

    //setter method of new_patient_phone_number declaration

    void set_new_patient_phone_number(string nppn);

    //getter method of new_patient_phone_number declaration

    string get_new_patient_phone_number();

    //setter method of new_patient_age declaration

    void set_new_patient_age(string npa);

    //getter method of new_patient_age declaration

    string get_new_patient_age();

    //setter method of new_patient_sex declaration

    void set_new_patient_sex(string nps);

    //getter method of new_patient_sex declaration

    string get_new_patient_sex();

    //setter method of new_patient_date declaration

    void set_new_patient_date(string npdt);

    //getter method of new_patient_date declaration

    string get_new_patient_date();

    //using pure virtual function display() declaration

    void display();

};

//setter method of new_patient_name under new_patients_records definition using scope
resolution operator

void new_patients_records :: set_new_patient_name(string npn)
```

```
{
    new_patient_name = npn;
}
//getter method of new_patient_name under new_patients_records definition using scope resolution operator
string new_patients_records :: get_new_patient_name()
{
    return new_patient_name;
}
//setter method of new_patient_district under new_patients_records definition using scope resolution operator
void new_patients_records :: set_new_patient_district(string npd)
{
    new_patient_district = npd;
}
//getter method of new_patient_district under new_patients_records definition using scope resolution operator
string new_patients_records :: get_new_patient_district()
{
    return new_patient_district;
}
//setter method of new_patient_address under new_patients_records definition using scope resolution operator
void new_patients_records :: set_new_patient_address(string npad)
{
    new_patient_address = npad;
}
//getter method of new_patient_address under new_patients_records definition using scope resolution operator
string new_patients_records :: get_new_patient_address()
{
   return new_patient_address;
}
//setter method of new_patient_email under new_patients_records definition using scope resolution operator
void new_patients_records :: set_new_patient_email(string npe)
{
   new_patient_email = npe;
}
//getter method of new_patient_email under new_patients_records definition using scope resolution operator
string new_patients_records :: get_new_patient_email()
{
   return new_patient_email;
}
//setter method of new_patient_profession under new_patients_records definition using scope resolution operator
void new_patients_records :: set_new_patient_profession(string npp)
{
   new_patient_profession = npp;
}
```

```
//getter method of new_patient_profession under new_patients_records definition using scope resolution operator

string new_patients_records :: get_new_patient_profession()

{

   return new_patient_profession;

}

//setter method of new_patient_phone_number under new_patients_records definition using scope resolution operator

void new_patients_records :: set_new_patient_phone_number(string nppn)

{

   new_patient_phone_number = nppn;

}

//getter method of new_patient_phone_number under new_patients_records definition using scope resolution operator

string new_patients_records :: get_new_patient_phone_number()

{

   return new_patient_phone_number;

}

//setter method of new_patient_age under new_patients_records definition using scope resolution operator

void new_patients_records :: set_new_patient_age(string npa)

{

   new_patient_age = npa;

}

//getter method of new_patient_age under new_patients_records definition using scope resolution operator
```

```
string new_patients_records :: get_new_patient_age()

{

    return new_patient_age;

}

//setter method of new_patient_sex under new_patients_records definition using scope resolution operator

void new_patients_records :: set_new_patient_sex(string nps)

{

    new_patient_sex = nps;

}

//getter method of new_patient_sex under new_patients_records definition using scope resolution operator

string new_patients_records :: get_new_patient_sex()

{

    return new_patient_sex;

}

//setter method of new_patient_date under new_patients_records definition using scope resolution operator

void new_patients_records :: set_new_patient_date(string npdt)

{

    new_patient_date = npdt;

}

//getter method of new_patient_date under new_patients_records definition using scope resolution operator

string new_patients_records :: get_new_patient_date()

{
    return new_patient_date;
}
//pure virtual function under new_patients_records class, display() definition using scope resolution operator
void new_patients_records :: display()
{
    //using ofstream mode to create a file stored in patient_file
    ofstream patient_file;
    //opening the file "Patient_details.txt" in append mood to write data
    patient_file.open("Patient_details.txt",ios::out|ios::app);
    cout << "Patient Informations Successfully Taken : " << endl;
    cout << "----------------------------------------"  << endl << endl;
    //calling getter method of new_patient_name
    cout << "Name : " << get_new_patient_name() << endl;
    //writing patient_name in file
    patient_file << get_new_patient_name() << endl;
    //calling getter method of new_patient_name
    cout << "Address : " << get_new_patient_address() << endl;
    //writing patient_address in file
    patient_file << get_new_patient_address() << endl;
    //calling getter method of new_patient_name
    cout << "District : " << get_new_patient_district() << endl;
    //writing patient_district in file
    patient_file << get_new_patient_district() << endl;
```

```
//calling getter method of new_patient_name

cout << "Email : " << get_new_patient_email() << endl;

//writing patient_email in file

patient_file << get_new_patient_email() << endl;

//calling getter method of new_patient_name

cout << "Profession : " << get_new_patient_profession() << endl;

//writing patient_profession in file

patient_file << get_new_patient_profession() << endl;

//calling getter method of new_patient_name

cout << "Phone Number : " << "+880" << get_new_patient_phone_number() << endl;

//writing patient_phone_number in file

patient_file << "+880" << get_new_patient_phone_number() << endl;

//calling getter method of new_patient_name

cout << "Age : " << get_new_patient_age() << endl;

//writing patient_age in file

patient_file << get_new_patient_age() << endl;

//calling getter method of new_patient_sex

cout << "Sex : " << get_new_patient_sex() << endl;

//writing patient_sex in file

patient_file << get_new_patient_sex() << endl;

//calling getter method of new_patient_date

cout << "Date : " << get_new_patient_date() << endl;

//writing patient_date in file

patient_file << get_new_patient_date() << endl;
```

```
    patient_file << "**************************************************" << endl;

    //closing the file "Patient_details.txt"

    patient_file.close();

}
```

Step 6.Option 4 from menu section:

If user wants to exit from the program after finishing his/her needs, he/she will have to enter 4 and then the program will successfully be exited.

Step 7.Main function:

And finally, to access the all types of classes, variables and functions mentioned above, main function is created. The source code of main function is given below-

```
//main() function to run the program

int main()

{

    //declaring the pointer of abstract class base_user_details

    base_user_details *bud;

    //creating an object of derived class derived_user_details from abstract class base_user_details
class

    derived_user_details ud;

    //passing the address of derived_user_details object to the pointer of base_user_details

    bud = &ud;

    //calling input() function

    input();

    //calling setter method of user_name by pointer of base_user_details
```

```
bud -> set_user_name(name);

//calling setter method of user_address by pointer of base_user_details

bud -> set_user_address(address);

//calling setter method of user_district by pointer of base_user_details

bud -> set_user_district(district);

//calling setter method of user_email by pointer of base_user_details

bud -> set_user_email(email);

//calling setter method of user_proffession by pointer of base_user_details

bud -> set_user_proffession(proffesion);

//calling setter method of user_age by pointer of base_user_details

bud -> set_user_age(age);

//calling setter method of user_phone_number by pointer of base_user_details

bud -> set_user_phone_number(phone_number);

//calling setter method of user_sex by pointer of base_user_details

bud -> set_user_sex(sex);

//calling setter method of user_date by pointer of base_user_details

bud -> set_user_date(date);

//calling display member function under abstract base class by pointer of base_user_details

bud -> display();

//creating a label named tasks:

tasks:

    //calling global function features to display menu section

    features();

    //condition for the 1st choice of menu section from user
```

```
if(features_counter=="1")
{
   cout << endl << "Here is the patients records : " << endl;
   cout << "------------------------------"  << endl << endl;
   //creating an object of class show_patient_records
   show_patient_records spr;
   cout << endl << endl << "GOING BACK TO MENU SECTION : " << endl;
   cout << "----------------------------" << endl << endl << endl;
   //going back to tasks label after executing 1st section of menu
   goto tasks;
}
//condition for the 2nd choice of menu section from user
else if(features_counter=="2")
{
   //local variable search_patient to store the patient name which user wants to search
   string search_patient;
   //creating an object of class search_patient_records
   search_patient_records srpr;
   cout << "Enter patient name/phone number/email id : ";
   //taking input from user which patient name is to be searched
   getline(cin,search_patient);
   //ignore() function is used to ignore the next character
   cin.ignore();
   //calling setter method of search_patient by the object of class search_patient_records
```

```
        srpr.set_search_patient(search_patient);
        //calling check_records() member function to display the search result by the object of class search_patient_records
        srpr.check_records();
        cout << endl << endl << "GOING BACK TO MENU SECTION : " << endl;
        cout << "----------------------------" << endl << endl << endl;
        //going back to tasks label after executing 2nd section of menu
        goto tasks;
    }
    //condition for the 3rd choice of menu section from user
    else if(features_counter=="3")
    {
        //calling global function input_patient_number()
        input_patient_number();
        //calling object type array of class new_patients_records
        new_patients_records npr[new_patients_number];
        //declaring local variable i initialized with 0 to continue the for loop below
        int i = 0;
        //for loop to take the informations of new patients as input from user
        for(i=0 ; i<new_patients_number ; i++)
        {
            cout << "Enter details for patient " << i+1 << " : " << endl;
            cout << "-----------------------------"  << endl << endl;
            //calling global function input()
```

```
input();

//calling setter method of new_patient_name under new_patients_records by the array type object of new_patients_records

npr[i].set_new_patient_name(name);

//calling setter method of new_patient_district under new_patients_records by the array type object of new_patients_records

npr[i].set_new_patient_district(district);

//calling setter method of new_patient_address under new_patients_records by the array type object of new_patients_records

npr[i].set_new_patient_address(address);

//calling setter method of new_patient_email under new_patients_records by the array type object of new_patients_records

npr[i].set_new_patient_email(email);

//calling setter method of new_patient_profession under new_patients_records by the array type object of new_patients_records

npr[i].set_new_patient_profession(proffesion);

//calling setter method of new_patient_phone_number under new_patients_records by the array type object of new_patients_records

npr[i].set_new_patient_phone_number(phone_number);

//calling setter method of new_patient_age under new_patients_records by the array type object of new_patients_records

npr[i].set_new_patient_age(age);

//calling setter method of new_patient_sex under new_patients_records by the array type object of new_patients_records

npr[i].set_new_patient_sex(sex);

//calling setter method of new_patient_date under new_patients_records by the array type object of new_patients_records
```

```
                npr[i].set_new_patient_date(date);

                //calling member function display under new_patients_records by the array type object
of new_patients_records

                npr[i].display();

            }

            cout << endl << endl << "GOING BACK TO MENU SECTION : " << endl;

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

            //going back to tasks label after executing 3rd section of menu

            goto tasks;

        }

        else

            cout << endl << "Thank you for using this application!!!" << endl << "EXIT!!! ";

    getch();

}
```

Search Error

I have successfully used all the features of my project excluding the feature **Search Patient** which is **option 2** from **"Menu" section.** I have checked the code of searching patient part but do not know why search result is not coming perfectly. Every time compiler shows me that the patient is found in the records though he/she is not in the records list. Rest of the code are all fine.

Conclusion

I have tried my best to build a well-developed project so that this can be an innovative technology based solution for keeping the coronavirus affected patient records and I hope that to materialize the vision of **"Digital Bangladesh"** within 2021, this can be a role model for our us at this pandemic situation due to COVID-19.

However, I have felt some lack of features in my own project as a user and I hope that I need to update myself more to fulfill the lacking. So that’s all about my project.

THE END