“VACCINATION SLOT BOOKING SYSTEM”

INTRODUCTION :

we've all seen the horrors that covid-19 brought upon us. The disease has been engulfing the world from past 2 years.

vaccination is the only existing method to safeguard ourselves against this widespread pandemic, but in a country like india where population is in billions ,it's equally important to create a sophisticated system to deal with this huge number and data.

We have created a model of Vaccination Slot Booking System where the user inputs his/her credentials like name, phone, age, gender etc and books a slot by selecting the date, time and center.

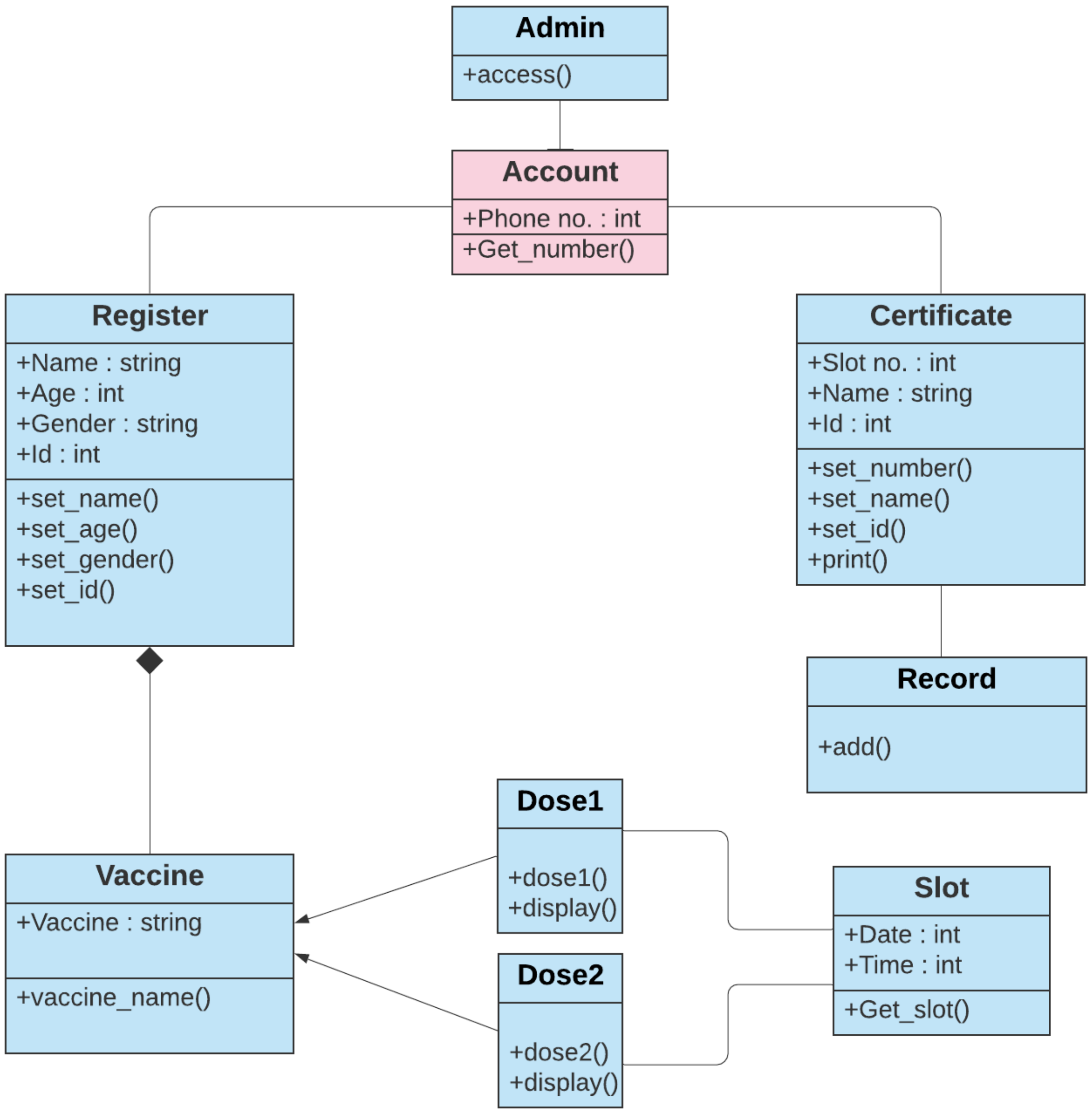
We have used Object Oriented Methodology to capture the aspects of real world entity.

REQUIREMENTS :

Our vaccination system requires the user provide the following information:-

1. User needs to provide his/her phone number.
2. User shall provide name, age and gender.
3. Slots will be allocated according to time and date provided.
4. User will be needed to select the favorable type of vaccine.
5. Centre will be provided by the system.

CLASS DIAGRAM :



EXPLANATION OF CODE :

Main.cpp

The program starts with whether the user is an admin(who can access the database) or a client(who can book a slot).

If it’s admin, then he has to provide a password for security and can access the database which contains all the details of slots booked by users.

Admin.cpp/admin.h (Admin)

When it’s object is created and it’s ‘access’ function is called. It asks for the password for security,

(the password is ‘password’) and if entered correctly, you can access the data as an admin.

Database is created using file handling which is used later in the program after the slot has been booked.

If it’s a client, then he/she is asked to create an account using his/her phone number.

Account.cpp/account.h (Account)

Account is created using class ‘account’ which takes phone no. as the parameter of its constructor.

After inserting the phone no. , it asks us to provide the captcha which is done by calling the ‘captcha’ function.

If captcha is entered wrong, it asks us to re-enter the captcha.

Captcha.h

This contains a function which when called generates a random string and returns it back.

Reg.cpp/reg.h (Register)

The account object contains the object ‘r’ of class reg which registers the user for booking a slot.

In the main function, it will ask for your credentials such as name, age and gender as input.

Name, age and gender is given as parameters to its function setdetails().

The class ‘reg’ also contains the object of classes ‘dose1’ and ‘dose2’ which signifies the First dose and Second Dose.

Dose1.cpp/dose1.h/dose2.cpp/dose2.h (Dose1/Dose2)

These classes of First dose and second dose inherits the properties of another class ‘vaccine’ which is explained below.

These classes contains the details of the vaccine(name of the vaccine).

In the display(), depending if you want first dose or second, you can register accordingly.

You can choose between Covishield and Covaxin as your dose.

Vaccine.cpp/vaccine.h (Vaccine)

This class contains the string variable which is used to store the name of the vaccine.

This class is inherited by the classes ‘dose1’ and ‘dose2’.

Vaccine.h has virtual function ‘display()’ which is overridden by the class ‘dose1’ and ‘dose2’.

slot.cpp/slot.h (slot)

The classes dose1 and dose2 contains the class ‘slot’.

When its function ‘getslot’ is called from the main(), it does the following –

For booking a slot, it asks for us to choose a center ranging from A to Z.

Then, you have to enter a valid date in the format dd/mm/yyyy.

Date.h

This contains isValidDate().

To check if the data is valid or not , isValidDate() is called.

Assuming it takes only a minute to receive the dose, we enter the time slot in the format hh:mm (24hr format).

Time.h

This contains isValidTime()

To check if the time is valid or not, isValidTime() is called.

Certificate.cpp/certificate.h (certificate)

The certificate takes the user details as parameters and displays it.

Depending which dose(‘dose1’ or ‘dose2’) you have slotted, the certificate is displayed.

This class contains the class ‘record’ and the function ‘loading’ which are executed parallely using threads.

The program stores all the data in the database(text file) via file handling.

Sometimes, the server is crowded in an actual case and takes time.

So using multi-threading, we store the data in the database ‘record.txt’ meanwhile loading screen is displayed on the screen to keep the user waiting.

Record.h

‘record’ class is connected to thread ‘t2’.

Data is stored in the database by calling the class ‘record’ through threading which takes the user details as parameters and outputs it into the text file “record.txt”

Loading.h

‘Loading’ function is connected to the thread ‘t1’.

Loading screen is shown by calling the function ‘loading’ which lasts long for around 10 sec. This is done because if there’s any server traffic while the data is transferring to the database which might take a long time, then showing the loading screen can hold the user’s patience.

After that, slot certificate is provided at the end for booking the slot.

Object Creation Heirarchy

Main -> admin

Main -> Account -> reg -> dose1 / dose2 -> slot

Main -> certificate -> record

File Handling

Using fstream library, the user’s data is stored in a text file ‘record.txt’.

Multithreading

Using thread library, storing the data in the database and displaying the loading screen is done by using two different threads.

Inheritance

Class ‘dose1’ and class ‘dose2’ inherits from the class ‘vaccine’.

Polymorphism

The base class ‘vaccine’ has a virtual function ‘display()’ which is overridden by the ‘dose1’s and ‘dose2’s display().

Run command -

```
 g++ -pthread -std=c++11 -o output account.cpp dose1.cpp dose2.cpp main.cpp
reg.cpp slot.cpp vaccine.cpp admin.cpp certificate.cpp
```

CONCLUSION :

So coming to the end of our project we conclude that our project was highly inspired by cowin website. It is truly based on real life problem.
Based on our knowledge of OOM, we think we were able to get satisfactory results. We were able to incorporate the concepts of inheritance, polymorphism, file handling, constructor overloading and multi-threading.

Submitted by -
Aakash Jha (LIT2020019)
Nirmal Minz(LCI2020076)
Pallavi Chauhan(LCI2020073)
Rohan Bhardwaj(LIT2020003)

Guided by -
Saurabh Srivastava Sir