Topic : Online Medical Portal

Group no : MET_WD_07

Campus : Metro

Submission Date :

We declare that this is our own work and this Assignment does not incorporate without acknowledgment any material previously submitted by anyone else in SLIIT or any other university/Institute. And we declare that each one of us equally contributed to the completion of this Assignment.

Registration No	**Name**	**Contact Number**
IT21365928	Basnayaka D.S.B.M	0702833605
IT21249198	Peiris H.N.K	0715757598

Part 1

1.User Requirements

- Unregistered Users can view the system and to use the system they have to register to the system by providing Name, NIC, Address, Contact number and an Email Address.

- Registered Users can log into the system by entering a valid username and password.

- There are different types of registered users. They are doctors, patient and staff.

- Patients can select a time schedule details by selecting a date and the type of health issue.

- Patients can get appointments by entering Name, Age, type of health issue, doctor, NIC, date and payment.

- Patient must include payment details like payment type and card details.

- Doctors should be able to update date and time schedules.

- Database Administrator should be confirmed the details.

- Staff and administrator can update or delete the patient details.

- After the payment done system confirming appointment and generates a unique ID and payment status for the patients.

- Confirmed patients' appointment details such as name date and ID are generated to the doctor.

- Administrator staff can provide reports like lab report, payment details, appointment deatils and medical report.

2. Noun/Verb Analysis

(Nouns)

- Unregistered Users can view the system and to use the system they have to register to the system by providing Name, NIC, Address, Contact number and an Email Address.

- Registered Users can log into the system by entering a valid username and password.

- There are different types of registered users. They are doctors, patient and staff.

- Patients can select a time schedule details by selecting a date and the type of health issue.

- Patients can request an appointment by entering Name, Age, type of health issue, doctor, NIC, date and payment.

- Patient must include payment details like payment type and card details.

- Doctors should be able to update date and time schedules.

- Database administrator should be confirmed the details.

- Staff and administrator can update or delete the patient details.

- After the payment done, system confirming appointment and generates a unique ID and payment status for the patients.

- Confirmed patients' appointment details such as name, date and ID are generated to the doctor.

- Administrator staff can provide reports like lab report, payment details, appointment details and medical report.

Identified Classes

- Unregistered User
- Registered User
- Patient
- Doctor
- Staff
- Administrator
- Appointment
- Payment

Reasons for rejecting other nouns

Redundant: Database Administrator

Administrator staff

Out of the scope: system, lab report, medical report

Attributes: Name, NIC, Address, Contact number, Email address,

Username, Password,

Time schedule details (date, type of health issue),

Patient details (Name, Age, type of health issue, doctor, NIC, date,

Payment),

Payment details (payment type, card details),

Unique ID, Payment status,

Appointment details (name, date, ID),

Noun/Verb Analysis

(Verbs)

- Unregistered Users can view the system and to use the system they have to register to the system by providing Name, NIC, Address, Contact number and an Email Address.
- Registered Users can log into the system by entering a valid username and password.
- There are different types of registered users. They are doctors, patient and staff.
- Patients can select a time schedule details by selecting a date and the type of health issue.
- Patients can request an appointment by entering Name, Age, type of health issue, doctor, NIC, date and payment.
- Patient must include payment details like payment type and card details.
- Doctors should be able to update date and time schedules.
- Database administrator should be confirmed the details.
- Staff and administrator can update or delete the patient details.
- After the payment done, system confirming appointment and generates a unique ID and payment status for the patients.
- Confirmed patients' appointment details such as name, date and ID are generated to the doctor.
- Administrator staff can provide reports like lab report, payment details, appointment details and medical report.

Methods

- **Unregistered User-** View the system

  Registering to the system by providing details

- **Registered User** – Login to the system

  Use and view the system

- **Patient -** Login to the system

  Request an appointment

  Search date and a doctor

  Paying for the appointment

- **Doctor-** Login to the system

  Update date and time

  Checking appointments

- **Staff –** Login to the system

  Checking patient details

  Confirm done appointment details to the system

- **Administrator –** Login to the system

  Manage patient details

  Manage appointments

- **Appointment –** Generate an appointment ID

  Check time schedules

  Checking payment status

- **Payment –** Generate a payment ID

  Manage payment details

  Confirming payments

- **Report -** Generate lab reports and medicine details

  Generate appointment details

  Generate payment details

CRC Cards

Unregistered User	
Responsibility	**Collaborators**
Allow to view the appointment time schedules	Appointment
Register to the system	

Registered User	
Responsibility	**Collaborators**
Login to the system	
Allow to view and use appointment time schedules	Appointment

Patient	
Responsibility	**Collaborators**
Login to the system	Registered User
Allow to request appointments	Appointment
Allow to select dates and doctors	
Do payments for the appointment	Appointment

Doctor	
Responsibility	**Collaborators**
Login to the system	
Update available dates and time	
Checking appointments	Appointment

Staff	
Responsibility	**Collaborators**
Login to the system	
Confirming done appointments to the system	Appointment
Checking patient details	

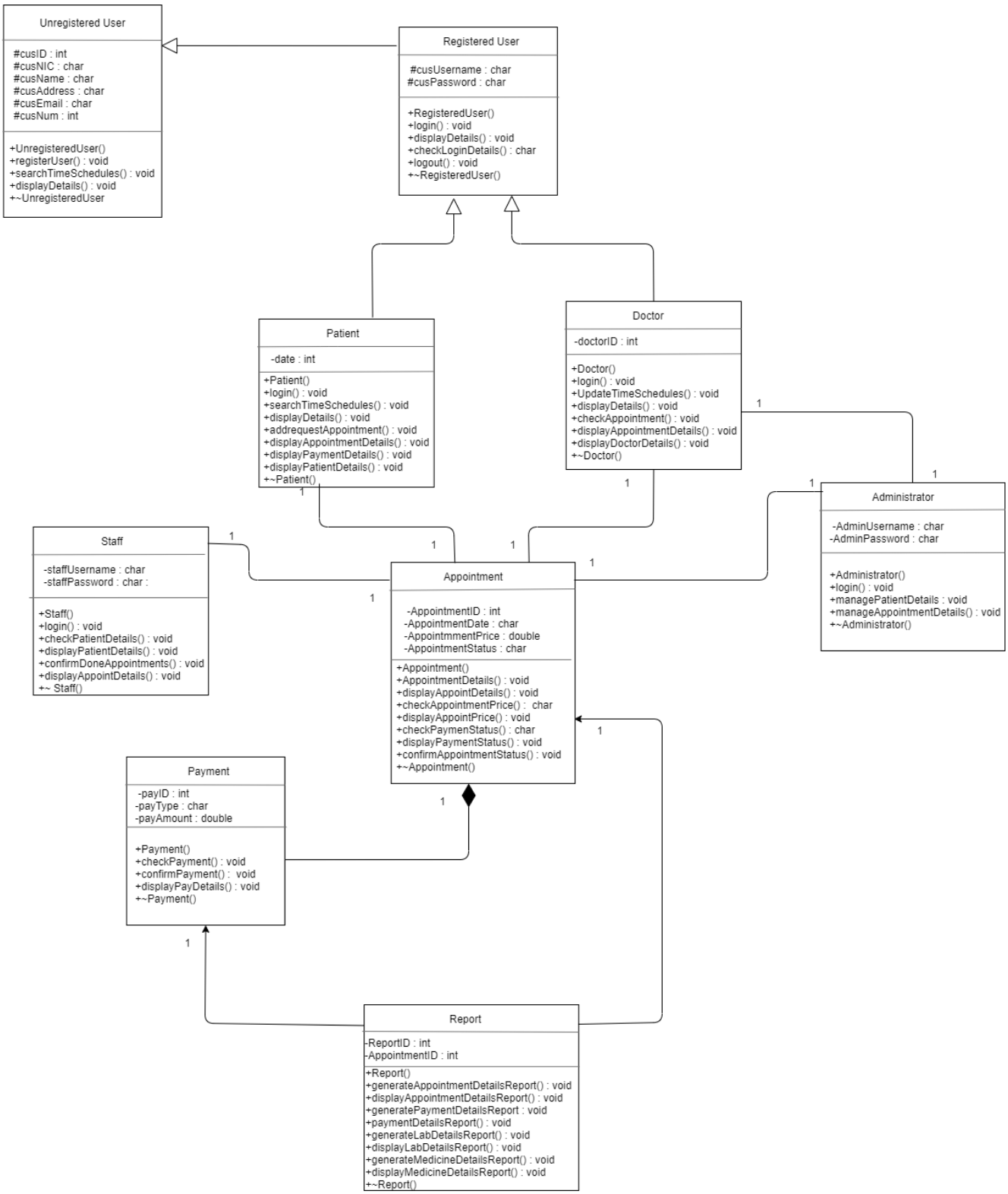
Administrator	
Responsibility	**Collaborators**
Login to the system	
Manage patient details	
Manage appointments	Appointments

Appointment	
Responsibility	**Collaborators**
Generate an appointment ID	Patient, Doctor
Check time schedules	Administrator
Checking payment status	Patient, Doctor, Administrator

Payment	
Responsibility	**Collaborators**
Generate a payment ID	Patient
Manage payment details	Patient
Confirming payments	

Report	
Responsibility	**Collaborators**
Generate payment details	Payment
Generate appointment details	Appointment
Generate lab reports and medicine details	

Class Diagram (UML Notation)



Class Header Files

UnregisteredUser.h

```
class UnregisteredUser
{
protected:
     int cusID;
     char cusNIC[12];
     char cusName[20];
     char cusAddress[30];
     char cusEmail[20];
     int cusNum[10];
public:
      UnregisteredUser( );
      UnregisteredUser(int pcusID, const char pcusNIC[], const char pcusName[], const
char pcusAddress[], const char pcusEmail[], int pcusNum[]);
       void registerUser( );
       void searchTimeSchedules( );
       virtual void displayDetails( TimeSchedules* pTi);
       ~UnregisteredUser( );
};
```

RegisteredUser.h

```
#include "UnregisteredUser.h"
class RegisteredUser  :public UnregisteredUser
{
protected:
     char cusUsername[12];
     char cusPassword[10];

public:
      RegisteredUser( );
      RegisteredUser(const char pcusUsername[], const char pcusPassword[], int pcusID,
const char pcusNIC[], const char pcusName[], const char pcusAddress[], const char
pcusEmail[], int pcusNum[]);
       void displayDetails( );
       void login( );
       char checkLoginDetails( );
       void logout( );
       ~RegisteredUser( );
};
```

Patient.h

```
#include "RegisteredUser.h"
#include "Appointment.h"

class Patient  :public RegisteredUser
{
private:
     int date[8];

public:
      Patient( );
      Patient(const char pcusUsername[], const char pcusPassword[], int pcusID, const
char pcusNIC[], const char pcusName[], const char pcusAddress[], const char
pcusEmail[], int pcusNum[], int pDate);
      void login( );
       void searchTimeSchedules( );
       void displayDetails(TimeSchedules* pTi );
       void addrequestAppointment( );
       void displayAppointmentDetails(Appointment *appD);
       void displayPaymentDetails(Payment* p);
       void displayPatientDetails( );

       ~Patient( );
};
```

Doctor.h

```
#include "RegisteredUser.h"
#include "Appointment.h"
class Doctor  :public RegisteredUser
{
private:
     int doctorID;
public:
      Doctor( );
      Doctor(const char pcusUsername[], const char pcusPassword[], int pcusID, const
char pcusNIC[], const char pcusName[], const char pcusAddress[], const char
pcusEmail[], int pcusNum[], int docID);
      void login( );
       void UpdateTimeSchedules( );
       void displayDetails( );
       void checkAppointment( );
       void displayAppointmentDetails(Appointment* appD );
       void displayDoctorDetails( );

       ~Doctor( );
};
```

Staff.h

```
#include "RegisteredUser.h"
#include "Patient.h"
#include "Appointment.h"
class Staff : public RegisteredUser
{
private:
    char staffUsername;
    char staffPassword;
public:
     Staff();
     Staff(const char pcusUsername[], const char pcusPassword[], int pcusID, const
char pcusNIC[], const char pcusName[], const char pcusAddress[], const char
pcusEmail[], int pcusNum[] );
          void login();
          void checkPatientDetails();
          void displayPatientDetails( );
          void confirmDoneAppointments();
          void displayAppointDetails(Appointment* appD);
       ~Staff();
```

Administrator.h

```
#include "RegisteredUser.h"
#include "Patient.h"
class Administrator: public RegisteredUser
{
private:
    char AdminUsername;
    char AdminPassword;

public:
     Administrator();
     Administrator (const char pcusUsername[], const char pcusPassword[], int
pcusID, const char pcusNIC[], const char pcusName[], constchar pcusAddress[], const
char pcusEmail[],  int pcusNum[]);
          void login();
          void managePatientDetails();
           void manageAppointmentDetails();
           void managePaymentDetails( );
     ~Administrator ();
```

Payment.h

```
#include "Patient.h"
#include "Administrator.h"
class Payment
    {
    private:
        int payID;
       char payType;
       double payAmount;
    public:
          Payment();
          Payment(int payId[], const char payTy[], double payAm[]);
           void checkPayment();
           void confirmPayment();
           void displayPayDetails()

          ~Payment ();
```

Appointment.h

```
#include "Doctor.h"
#include "Patient.h"
#include "Administrator.h"
#include "Staff.h"

class Appointment
{
private:
     int AppointmentID;
     char AppointmentDate[8];
     double AppointmentPrice;
     char AppointmentStatus[10];
     int count =0;
    Doctor* doc;
    Patient* patient;
    Staff* staff;

public:
     Appointment( );
     Appointment(Doctor* cdoc, Patient* cpatient, Staff* cstaff);
     void AppointmentDetails(int appID, const char appDate[], double appPrice[], const
char appStatus[], Doctor* cdoc, Patient* cpatient, Staff* cstaff);
     void displayAppointDetails(Appointment* appD );
     char checkAppointmentPrice( );
     void displayAppointPrice(Payment* p );
      char checkPaymentStatus( );
      void displayPaymentStatus(Status* st );
      void confirmAppointmentStatus( );

      ~Appointment( );
};
```

Report.h

```
#include "Payment.h"
#include "Appointment.h"
class Report
{
private:
     int ReportID;
     int AppointmentID;

public:
      Report( );
      Report(int RepID, int AppointID);
      void generateAppointmentDetailsReport(Appointment* appD);
      void displayAppointmentDetailsReport( );
      void generatePaymentDetailsReport(Payment* p);
      void paymentDetailsReport( );
      void generateLabReport( );
      void displayLabReport( );
      void generateMedicineDetailsReport( );
      void displayMedicineDetailsReport( );

       ~Report( );
};
```

Class cpp Files

UnregisteredUser.cpp

```
#include "UnregisteredUser.h"
#include <cstring>


UnregisteredUser :: UnregisteredUser()
{
     cusID=0;
    strcpy(cusNIC,  " ");
    strcpy(cusName, " ");
    strcpy(cusAddress, " ");
    strcpy(cusEmail, " ");
    cusNum=0;
}
     UnregisteredUser::UnregisteredUser(int pcusID, const char pcusNIC[], const char
pcusName[], const char pcusAddress[], const char pcusEmail[], int pcusNum[])
{
    cusID =pcusID;
    strcpy(cusNIC,  pcusNIC);
    strcpy(cusName, pcusName);
    strcpy(cusAddress, pcusAddress);
    strcpy(cusEmail, pcusEmail);
    cusNum=pcusNum;
}
     void UnregisteredUser :: registerUser( )
     {
     }
     void UnregisteredUser :: searchTimeSchedules( )
     {
     }
    void UnregisteredUser :: displayDetails( TimeSchedules* pTi)
     {
     }
     UnregisteredUser :: ~UnregisteredUser( )
      {
      }
```

RegisteredUser.cpp

```
#include "RegisteredUser.h"
#include <cstring>


RegisteredUser :: RegisteredUser()
{
    strcpy(cusUsername, " ");
    strcpy(cusPassword, " ");
}
     RegisteredUser :: RegisteredUser(const char pcusUsername[ ], const char
pcusPassword[ ], int pcusID, const char pcusNIC[], const char pcusName[], const char
pcusAddress[], const char pcusEmail[], int pcusNum[]) : UnregisteredUser(pcusID,
pcusNIC, pcusName, pcusAddress, pcusEmail,pcusNum)
{
    strcpy(cusUsername, pcusUsername);
    strcpy(cusPassword, pcusPassword);

}
     void RegisteredUser :: displayDetails( )
     {
     }
     void RegisteredUser :: login( )
     {
     }
     char RegisteredUser :: checkLoginDetails( )
     {
          return 0;
     }
     void RegisteredUser :: logout( )
     {
     }
     RegisteredUser :: ~RegisteredUser( )
     {
     }
```

Patient.cpp

```
#include "Patient.h"

Patient  ::  Patient( )
{
    date = 0;
}
     Patient :: Patient(const char pcusUsername[], const char pcusPassword[], int pcusID, const char pcusNIC[], const char pcusName[], const char pcusAddress[], const char pcusEmail[], int pcusNum[], int pDate):RegisteredUser( pcusName, pcusPassword, pcusID, pcusNIC, pcusName, pcusAddress, pcusEmail, pcusNum)
{
    date=pDate;
}
     void Patient :: login( )
     {
     }
     void Patient :: searchTimeSchedules( )
     {
     }
     void Patient :: displayDetails(TimeSchedules* pTi )
     {
     }
     void Patient :: addrequestAppointment( )
     {
     }
     void Patient :: displayAppointmentDetails(Appointment* appD)
     {
     }
     void Patient :: displayPaymentDetails(Payment* p)
     {
     }
     void Patient :: displayPatientDetails( )
     {
     }
     Patient :: ~Patient( )
     {
     }
```

Payment.cpp

```
#include "Payment.h"
#include <cstring>
Payment::Payment()
   {
    payID = 0;
```

```
Strcpy(payType, " ");
oayAmount+0;

Payment::Payment(int payId[], const char payTy[], double payAm[]);
payID = payId;
strcpy(payType, appDate);
strcpy(AppointmentPrice, payTy);
payamount=payAm;

void Payment:: checkPayment()
{
}
void PaymentL:: confirmPayment()
{
}
void Payment:: displayPayDetails()
{
}
Payment::~Payment ()
{
}
```

Doctor.cpp

```
#include "Doctor.h"
Doctor :: Doctor( )
{
    doctorID = 0;
}

     Doctor :: Doctor(const char pcusUsername[], const char pcusPassword[], int pcusID,
const char pcusNIC[], const char pcusName[], const char pcusAddress[], const char
pcusEmail[], int pcusNum[], int docID) :RegisteredUser(pcusUsername, pcusPassword,
pcusNIC, pcusName, pcusAddress, pcusEmail, pcusNum)
{
  doctorID = docID;
}
      void Doctor :: login( )
      {
      }
       void Doctor :: UpdateTimeSchedules( )
      {
      }
       void Doctor :: displayDetails(TimeSchedules* pTi )
      {
       }
       void Doctor :: checkAppointment( )
       {
        }
       void Doctor :: displayAppointmentDetails(Appointment* appD )
       {
        }
       void Doctor :: displayDoctorDetails( )
       {
       }
      Doctor :: ~Doctor( )
       {
       }
```

Staff.cpp

```
#include "Staff.h"
#include <cstring>
Staff : : Staff( )
{

    strcpy(staffUsername, " ");
    strcpy(staffPassword,  " ");
     Staff :: Staff(const char pcusUsername[], const char pcusPassword[], int pcusID,
const char pcusNIC[], const char pcusName[], const char pcusAddress[], const char
pcusEmail[], int pcusNum[] )
{
     Strcpy(staffUsername, pcusUsername);
    Strcpy(staffPassword, pcusPassword);
}
          Void Staff ::  login()
         {
         }
         void Staff :: checkPatientDetails()
         {
         }
         void Staff :: displayPatientDetails( )
         {
         }
         void Staff :: confirmDoneAppointments()
          {
          }
         void Staff :: displayAppointDetails(Appointment* appD)
          {
          }

       Staff~Staff()
          {
          }
```

Administrator.cpp

```
#include "Administrator.h"
#include <cstring>
 Administrator::Administrator( )
    {
       strcpy(AdminUsername, " ");
        strcpy(AdminPassword, " ");
}
          Administrator:: Administrator (const char pcusUsername[], const char
pcusPassword[], int pcusID, const char pcusNIC[], const char pcusName[], constchar
pcusAddress[], const char pcusEmail[],  int pcusNum[])
{         Strcpy(AdminUsername, pcusUsername);
          Strcpy(AdminPassword, pcusPassword);
}
                void Administrator::login()
                {
                }
                void Administrator :: managePatientDetails()
                 {
                 }
                 void Administrator::manageAppointmentDetails()
                 {
                 }
                 void Administrator:: managePaymentDetails( )
                 {
                 }
               Administrator::~Administrator ()
                 {
                 }
```

Appointment.cpp

```
#include "Appointment.h"

Appointment :: Appointment( )
{
     AppointmentID = 0;
     Strcpy(AppointmentDate, " ");
     Strcpy(AppointmentPrice, " ");
     Strcpy(AppointmentStatus, " ");

      Appointment :: Appointment(Doctor* cdoc, Patient* cpatient, Staff* cstaff)
{
   Doctor = cdoc;
   Patient = cpatient;
   Staff = cstaff;
}
      void Appointment :: AppointmentDetails(int appID, const char appDate[], double
appPrice[], const char appStatus[], Doctor* cdoc, Patient* cpatient, Staff* cstaff)
{
     AppointmentID = appID;
    strcpy(AppointmentDate, appDate);
    strcpy(AppointmentPrice, appPrice);
    strcpy(AppointmentStatus, appStatus);
}
      void Appointment :: displayAppointDetails(Appointment* appD )
      {
      }
      char Appointment :: checkAppointmentPrice( )
      {
      }
      void Appointment :: displayAppointPrice(Payment* p )
      {
      }
      char Appointment :: checkPaymentStatus( )
      {
      }
      void Appointment :: displayPaymentStatus(Status* st )
      {
      }
      void Appointment :: confirmAppointmentStatus( )
      {
      }

      Appointment :: ~Appointment( )
      {
      }
```

Report.cpp

```
#include "Report.h"
Report :: Report( )
{
   ReportID = 0;
   AppointmentID = 0;
    }
     Report :: Report(int RepID, int AppointID)
{
          ReportID = RepID;
          AppointmentID = AppointID;
}

     void Report :: generateAppointmentDetailsReport(Appointment* appD)
       {
       }
     void Report :: displayAppointmentDetailsReport( )
       {
       }
     void Report :: generatePaymentDetailsReport(Payment* p)
       {
       }
     void Report :: paymentDetailsReport( )
       {
       }
     void Report :: generateLabReport( )
        {
        }
     void Report :: displayLabReport( )
        {
        }
     void Report :: generateMedicineDetailsReport( )
        {
        }
     void Report :: displayMedicineDetailsReport( )
        {
        }
      Report :: ~Report( )
        {
        }
```

Main Program

Main.cpp

```
#include "UnregisteredUser.h"
#include "RegisteredUser.h"
#include "Patient.h"
#include "Doctor.h"
#include "Appointment.h"
#include "Staff.h"
#include "Administrator.h"
#include "Payment.h"
#include "Report.h"

#include <iostream>
using namespace std;

int main( )
{
    //----Object creation----

    UnregisteredUser* reg = new RegisteredUser( ); // Object - RegisteredUser class
    RegisteredUser* patient = new Patient( ); // Object - Patient class
    RegisteredUser* doctor = new Doctor( ); // Object - Doctor class
    Appointment* appoint = new Appointment( ); // Object - Appointment class
    Staff* staff = new Staff( ); //Object-Staff class
    Administrator* admin = new Administrator( ); // Object - Administrator class
    Report* report = new Report( ); // Object - Report class


    //----Method Calling----
   reg->login( );
   reg->displayDetails( );

   patient->login( );
   patient->displayPatientDetails( );

   doctor->login( );
   doctor->displayDoctorDetails( );

   appoint->displayAppointDetails( );
   appoint->confirmAppointmentStatus( );


  report->generateAppointmentDetailsReport( );delete
  report->generatePaymentDetailsReport( );
```

```
    report->generateLabReport( );
    report->generateMedicineDetailsReport( );

    //----Delete Dynamic objects----
    delete reg;
    delete patient;
    delete doctor;
    delete appoint;
    delete report;

    return 0;
}
```