Project Phase 1 Report For Hospital Management System

Prepared by: Group-7 (Manoj Kumar Thimapuram, Maulshree Verma, Naga Saiteja Chintakayala, Yogesh Nimmagadda)

University of North Texas, Denton.

10/21/2019

Table of Contents

1. Requirements

This web-application mainly focuses on four modules namely Admin, front-desk user, doctor and patient modules. Below are the major features implemented in this web-application.

1.1. PR: Patient Registration

- PRIORITY: HIGH
- The patient registration page helps the patients to create an account.
- The account for the patient can be created online or by front-desk user in the hospital.
- The patient needs to provide their details such as Full Name, Phone Number, e-mail ID, Address and Date of Birth to create the account.
- Only one account can be created with one e-mail ID.
- The user ID and password will be automatically generated after the user registration.

1.2. LP: Login Page

- PRIORITY: HIGH
- The login page can be used by doctors, patients and employees.
- The users need to provide the login details so that the user will be redirected to their respective account.
- Based on the username, the user will be redirected to their respective portal e.g. The doctor ID starts with Doc, whereas the employee ID starts with Emp.

1.3. AD: Adding Doctor

- PRIORITY: HIGH
- The doctors will be added by the Admin.
- The Doctors needs to provide their details such as Full Name, Phone Number, e-mail ID, Address, Date of Birth and also the specialty such as Cardiologist to create their account.

1.4. AE: Adding Employee

- PRIORITY: HIGH
- The admin will add the employee such as front-desk user.
- The employees need to provide their details such as Full Name, Phone Number, e-mail ID, Address and Date of Birth to create their account by the admin.

1.5. RA: Request for an Appointment

- PRIORITY: HIGH
- The patient will select the specialty so that the list of doctors of that particular specialty will be displayed.
- The patient can select the doctor from the list of doctors.
- The patient then needs to select the date, time and the reason of the visit.

1.6. ACRA: Approve or Cancel the Request for an Appointment

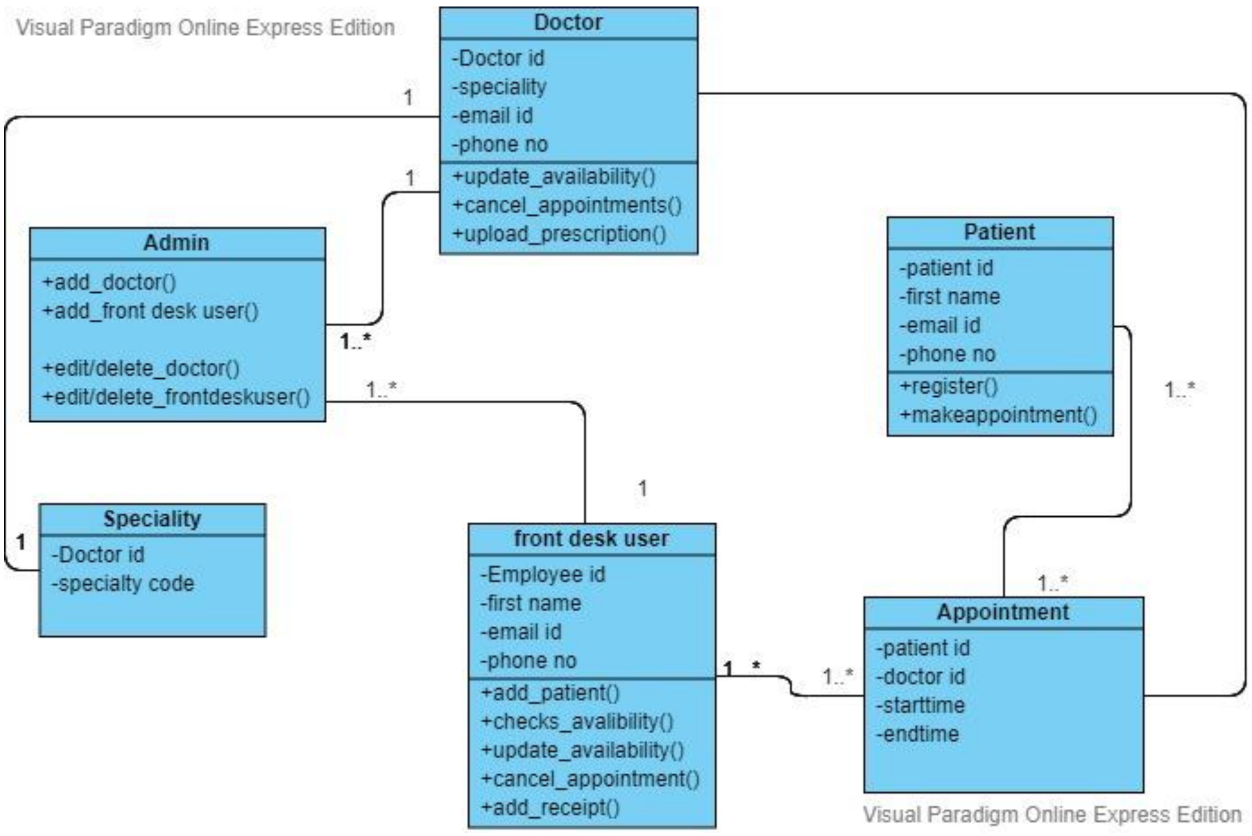
- PRIORITY: HIGH
- Once the patient completes requesting for an appointment, the front-desk user can view the list of appointment requests.
- The front-desk user checks the availability of the doctor for the requested date and time.
- The Front-desk user approves or cancels the requested appointment based on doctor's availability.

1.7. Updated Requirements

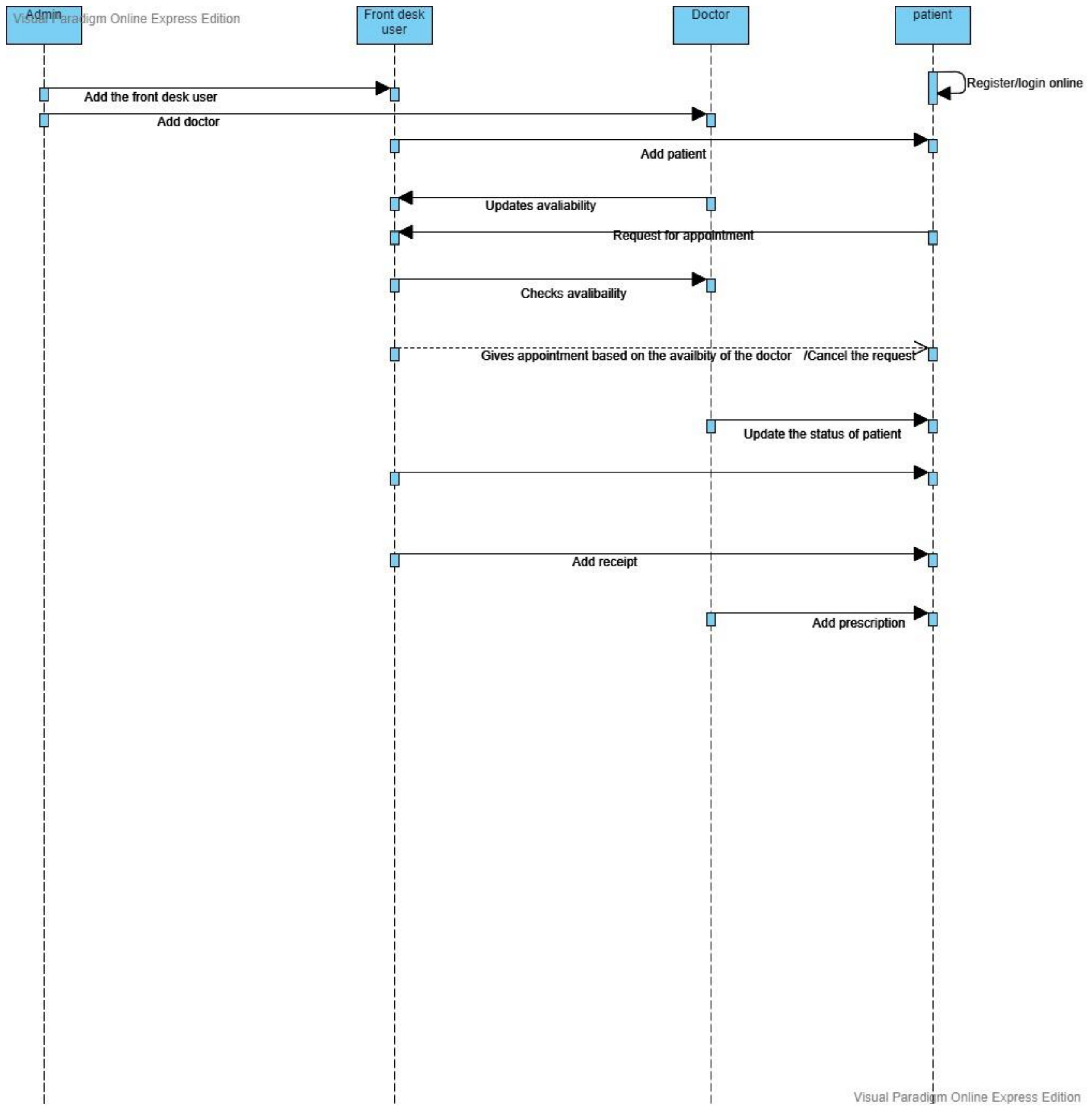
- We made one update to the requirements in the registration page. In the registration page the patient can be able to add his/her picture.

2. UML Design

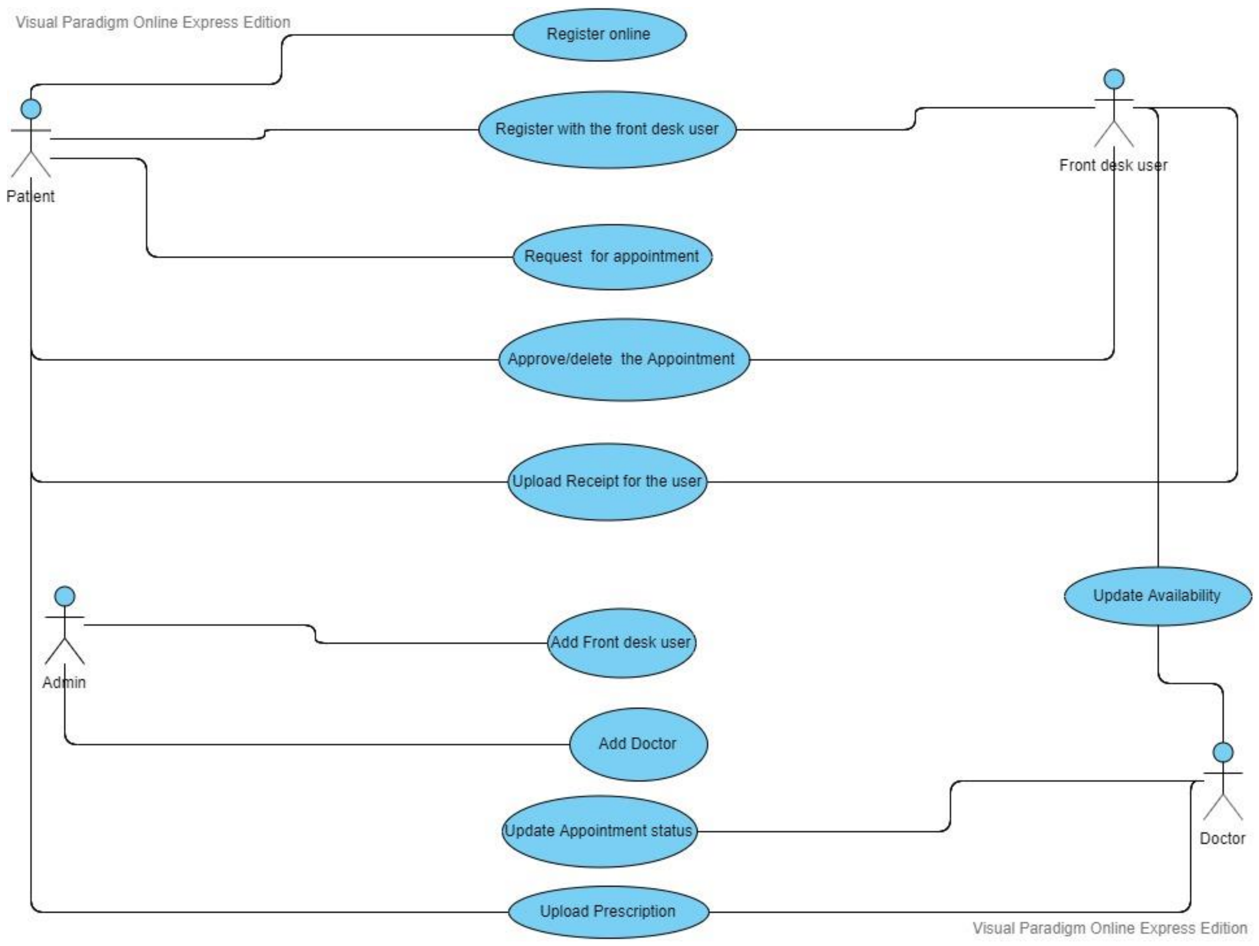
2.1. Class Diagram



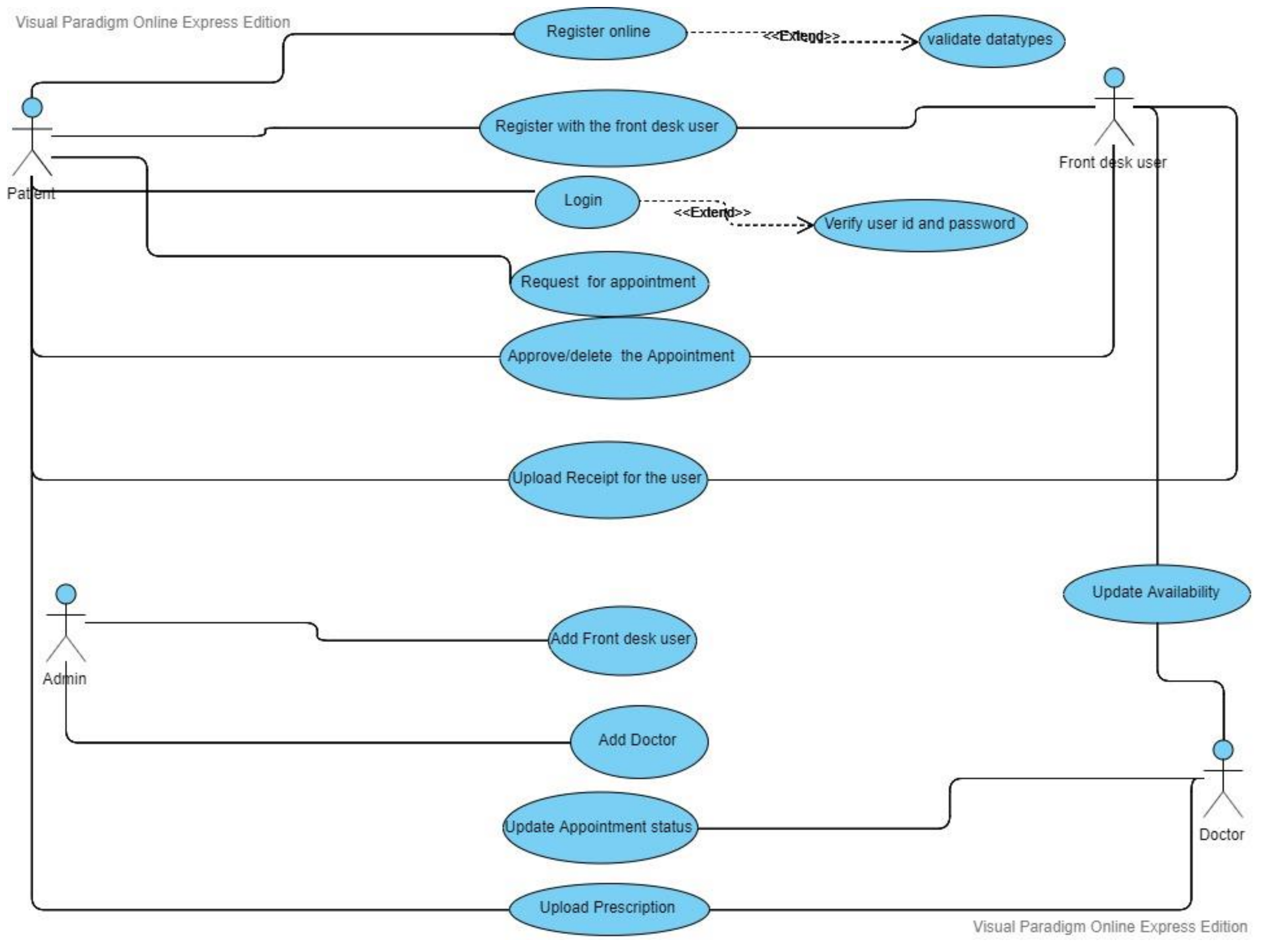
2.2. Sequence Diagram



2.3. Use case Diagram (Normal Case)


Visual Paradigm Online Express Edition
Register online
Register with the front desk user
Patient
Front desk user
Request for appointment
Approve/delete the Appointment
Upload Receipt for the user
Update Availability
Admin
Add Front desk user
Add Doctor
Update Appointment status
Doctor
Upload Prescription
Visual Paradigm Online Express Edition

2.4. Use case Diagram (Error Case)


Visual Paradigm Online Express Edition
Register online
<<Extend>>
validate datatypes
Register with the front desk user
Front desk user
Patient
Login
<<Extend>>
Verify user id and password
Request for appointment
Approve/delete the Appointment
Upload Receipt for the user
Update Availability
Admin
Add Front desk user
Add Doctor
Update Appointment status
Doctor
Upload Prescription
Visual Paradigm Online Express Edition

3. Test Cases

Registration testcase:

//Description: This class is used to test all the input fields in the registration page.
@Register.aspx.cs:

```
using System;
using System.Text;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
using HospitalSystem.App_Code;
//using System.Data;

namespace HospitalSystem
{

   public partial class Register : System.Web.UI.Page
   {
      Class1 c = new Class1();
      protected void Page_Load(object sender, EventArgs e)
      {

         if(!IsPostBack)
         {
            autoid();
             }
         // unit test
         // input: User Details

         Registrationunittest reg = new Registrationunittest();
         c.cmd.CommandText = "select * from Patient1";
         c.ds.Clear();
         c.adp.Fill(c.ds, "vt");
         c.dr = c.ds.Tables["vt"].NewRow();
         //c.dr["username"] = usernameTextBox.Text.Trim();
         c.dr["pswd"] = reg.patientpswd;
         c.dr["LastName"] = reg.lname;
         c.dr["FirstName"] = reg.fname;
         c.dr["phoneno"] = reg.phone;
         c.dr["EmailID"] = reg.emailid;
         c.dr["Age"] = reg.age;
         c.dr["PatientID"] = reg.id;
         c.dr["DOB"] = reg.dob;
         //c.dr["emailid"] = emailTxt.Text.Trim();
         //c.dr["sq"] = spList.SelectedValue.Trim();
         // c.dr["ans"] = yourAnsTxt.Text;
         c.dr["Gender"] = reg.gender;
         //c.dr["joiningdate"] = DateTime.Now;
         c.dr["Addrs"] = reg.address;


         //unit test over
         c.ds.Tables["vt"].Rows.Add(c.dr);

         c.scb.DataAdapter = c.adp;
```

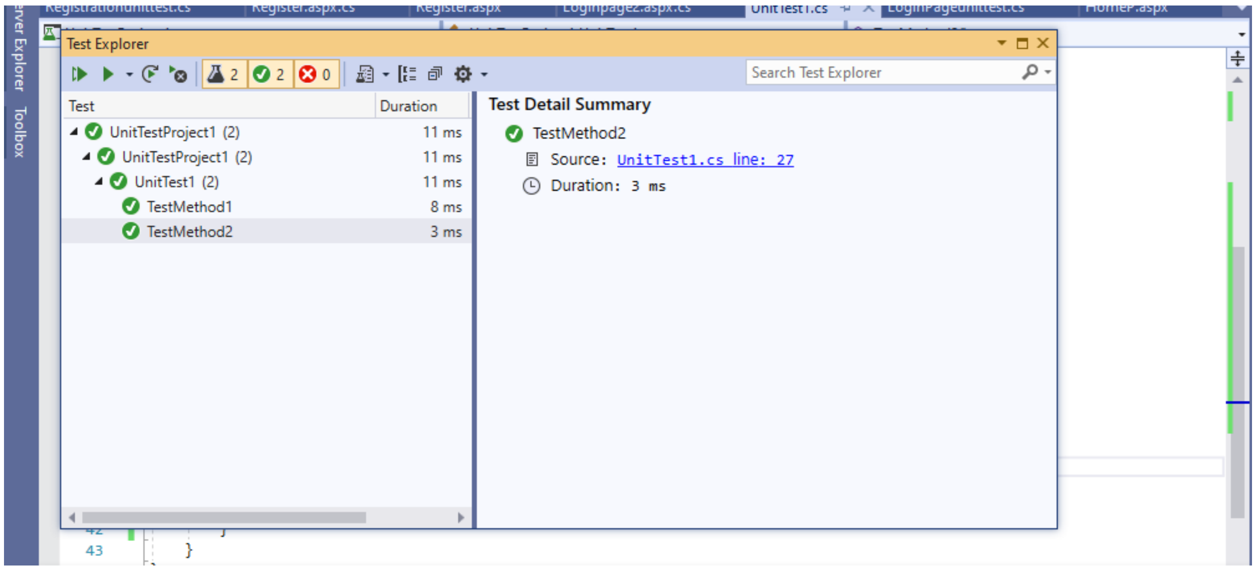
```
            c.adp.Update(c.ds.Tables["vt"]);


             }

/*
 * Description: testing whether given user details are valid or not
 * input: User Details
 * output: checking whether user details given are valid
 */
```

@Registrationunittest.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace HospitalSystem
{
    public class Registrationunittest
    {
        public string fname;
        public string lname;
        public string emailid;
        public long phone;
        public string address;
        public string gender;
        public int age;
        public DateTime dob;
        public string id;
        public string patientpswd;
        public Registrationunittest()
        {
            this.fname = "Mark";
            this.lname = "Ray";
            this.emailid = "mark@gmail.com";
            this.address = "100 Avenue D  Apt 1";
            this.phone = Convert.ToInt64("9405940316");
            this.gender = "male";
            this.age = 40;
            this.dob = Convert.ToDateTime("1979-10-19");
            this.id = "pt6";
            this.patientpswd = "MVMATIRXRK";
        }
    }
}
```

This is the unit test case for registration page.

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using HospitalSystem;

namespace UnitTestProject1
{
    [TestClass]
    public class UnitTest1
    {

    }
        [TestMethod]
        public void TestMethod2()
        {
            //Registration Test Case
            Registrationunittest obj = new Registrationunittest();
            Assert.AreEqual("pt6", obj.id);
            Assert.AreEqual("MVMATIRXRK", obj.patientpswd);
            Assert.AreEqual("Mark", obj.fname);
            Assert.AreEqual("Ray", obj.lname);
            Assert.AreEqual("mark@gmail.com", obj.emailid);
            Assert.AreEqual(9405940316, obj.phone);
            Assert.AreEqual(40, obj.age);
            Assert.AreEqual("male", obj.gender);
            Assert.AreEqual(Convert.ToDateTime("1979-10-19"), obj.dob);


        }
    }
}
```



Test	Duration
UnitTestProject1 (2)	11 ms
UnitTestProject1 (2)	11 ms
UnitTest1 (2)	11 ms
TestMethod1	8 ms
TestMethod2	3 ms

Test Detail Summary

TestMethod2

Source: UnitTest1.cs line: 27

Duration: 3 ms

Login Testcase:

//Description: This class is used to test the login details in the login page.

@Loginpage2.aspx:

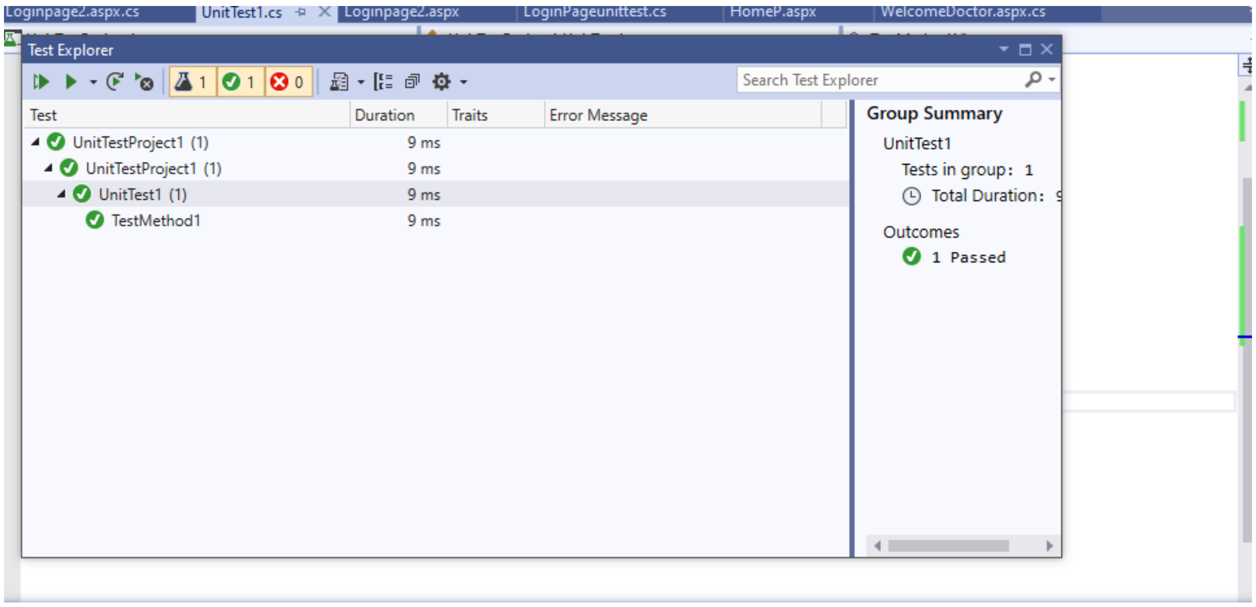
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using HospitalSystem.App_Code;

namespace HospitalSystem
{
    public partial class Loginpage2 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            LoginPageunittest lp = new LoginPageunittest();
            Response.Write(lp.Userid+ " " + lp.password) ;
            Response.Write(lp.docid + " " + lp.docpassword);
            Response.Write(lp.fduid + " " + lp.fdupassword);
            Response.Write(lp.patientid + " " + lp.patientpswd);
        }

    }

}
```

```
/*
 * Description: testing whether the given login details are valid or not
 * input: Login Details
 * output: checking whether the login details given are valid or not
 */
```

@LoginPageunittest.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace HospitalSystem
{
    public class LoginPageunittest
    {
        public string Userid;
        public string password;
        public string docid;
        public string docpassword;
        public string fduid;
        public string fdupassword;
        public string patientid;
        public string patientpswd;

        public LoginPageunittest()
        {
            this.Userid = "admin1234";
            this.password = "passwordqwerty";
            this.docid = "doc1";
            this.docpassword = "ZWFTXCJZBM";
            this.fduid = "emp1";
            this.fdupassword = "MQAVHFSLLS";
            this.patientid = "pt7";
            this.patientpswd = "TDBLCHNBVX";

    }
  }
}

using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using HospitalSystem;

namespace UnitTestProject1
```

```
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void TestMethod1()
        {
            //Login unit test
            LoginPageunittest ob = new LoginPageunittest();
            Assert.AreEqual("admin1234", ob.Userid);
            Assert.AreEqual("passwordqwerty" , ob.password);
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("ZWFTXCJZBM", ob.docpassword);
            Assert.AreEqual("emp1", ob.fduid);
            Assert.AreEqual("MQAVHFSLLS", ob.fdupassword);
            Assert.AreEqual("pt7", ob.patientid);
            Assert.AreEqual("TDBLCHNBVX", ob.patientpswd);
        }

        [TestMethod]
        public void TestMethod2()
        {
            //Registration Test Case
            Registrationunittest obj = new Registrationunittest();
            Assert.AreEqual("pt6", obj.id);
            Assert.AreEqual("MVMATIRXRK", obj.patientpswd);
            Assert.AreEqual("Mark", obj.fname);
            Assert.AreEqual("Ray", obj.lname);
            Assert.AreEqual("mark@gmail.com", obj.emailid);
            Assert.AreEqual(9405940316, obj.phone);
            Assert.AreEqual(40, obj.age);
            Assert.AreEqual("male", obj.gender);
            Assert.AreEqual(Convert.ToDateTime("1979-10-19"), obj.dob);
        }
    }
}
```



Loginpage2.aspx.cs UnitTest1.cs Loginpage2.aspx LoginPageunittest.cs HomeP.aspx WelcomeDoctor.aspx.cs

Test Explorer

Test	Duration	Traits	Error Message
✓ UnitTestProject1 (1)	9 ms		
✓ UnitTestProject1 (1)	9 ms		
✓ UnitTest1 (1)	9 ms		
✓ TestMethod1	9 ms		

Group Summary
UnitTest1
Tests in group: 1
Total Duration: 9
Outcomes
✓ 1 Passed

Add Doctor:

//Description: This class is used to test whether doctor has been added into the database or not.

@AddDoc.aspx.cs:

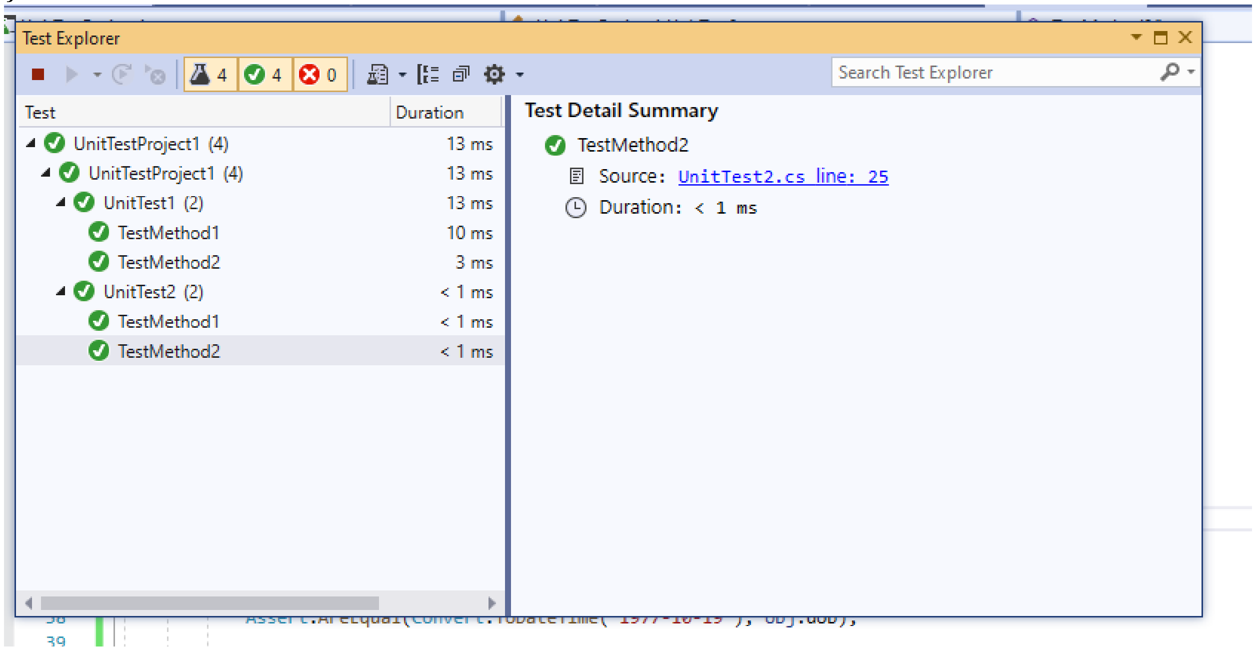
```
using System;
using System.Text;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
using HospitalSystem.App_Code;

namespace HospitalSystem
{

   public partial class AddDoc : System.Web.UI.Page
   {
      Class1 c = new Class1();
      protected void Page_Load(object sender, EventArgs e)
      {


         if (!IsPostBack)
         {
                autoid();
            }
            // unit test

            adddocunittest obj = new adddocunittest();
            c.cmd.CommandText = "select * from Doctor1";
            c.ds.Clear();
            c.adp.Fill(c.ds, "vt");
            c.dr = c.ds.Tables["vt"].NewRow();
            //c.dr["username"] = usernameTextBox.Text.Trim();
            c.dr["pswd"] = obj.docpwd;
            c.dr["LastName"] = obj.lname;
            c.dr["FirstName"] = obj.fname;
            c.dr["phoneno"] = obj.phone;
            c.dr["Speciality"] = obj.speciality;
            c.dr["Age"] = obj.age;
            c.dr["DoctorID"] = obj.id;
            c.dr["DOB"] = obj.dob;
            c.dr["Experience"] = obj.experience;
            // c.dr["ans"] = yourAnsTxt.Text;
            c.dr["Gender"] = obj.gender;
            //c.dr["joiningdate"] = DateTime.Now;
            c.dr["Addrs"] = obj.address;


            //unit test over

        }
/*
 * Description: Testing whether doctor has been added into the database or not.
 * input: Doctors Details and specialty
 * output: checking whether doctor has been added into the database or not.
 */
```

@Adddocunittest.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace HospitalSystem
{
    public class adddocunittest
    {
        public string fname;
        public string lname;
        public string experience;
        public long phone;
        public string address;
        public string gender;
        public int age;
        public DateTime dob;
        public string id;
        public string docpwd;
        public string speciality;
        public adddocunittest()
        {
            this.fname = "Dean";
            this.lname = "Sam";
            this.experience = "10 years";
            this.address = "100 Avenue D  Apt 1";
            this.phone = Convert.ToInt64("9415940316");
            this.gender = "male";
            this.age = 42;
            this.dob = Convert.ToDateTime("1977-10-19");
            this.id = "doc2";
            this.speciality = "Pediatrician";
            this.docpwd = "MVMXLIRXRK";
        }
    }
}
```

UnitTest2.cs:

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using HospitalSystem;

namespace UnitTestProject1
{
    [TestClass]
    public class UnitTest2
    {



        [TestMethod]
        public void TestMethod2()
        {
            //Registration Test Case
            adddocunittest obj = new adddocunittest();
            Assert.AreEqual("doc2", obj.id);
            Assert.AreEqual("MVMXLIRXRK", obj.docpwd);
            Assert.AreEqual("Dean", obj.fname);
            Assert.AreEqual("Sam", obj.lname);
            Assert.AreEqual("Pediatrician", obj.speciality);
            Assert.AreEqual(9415940316, obj.phone);
            Assert.AreEqual(42, obj.age);
            Assert.AreEqual("10 years", obj.experience);
            Assert.AreEqual("male", obj.gender);
            Assert.AreEqual(Convert.ToDateTime("1977-10-19"), obj.dob);


        }
    }
}
```



```
Test Explorer
4  4  0
Test                              Duration
UnitTestProject1 (4)              13 ms
  UnitTestProject1 (4)            13 ms
    UnitTest1 (2)                 13 ms
      TestMethod1                 10 ms
      TestMethod2                  3 ms
    UnitTest2 (2)                < 1 ms
      TestMethod1                < 1 ms
      TestMethod2                < 1 ms

Test Detail Summary
TestMethod2
  Source: UnitTest2.cs line: 25
  Duration: < 1 ms
```

Request for appointment Testcase:

//Description: This class is used to test whether the requested appointment by the patient is display to the front desk user and the doctor

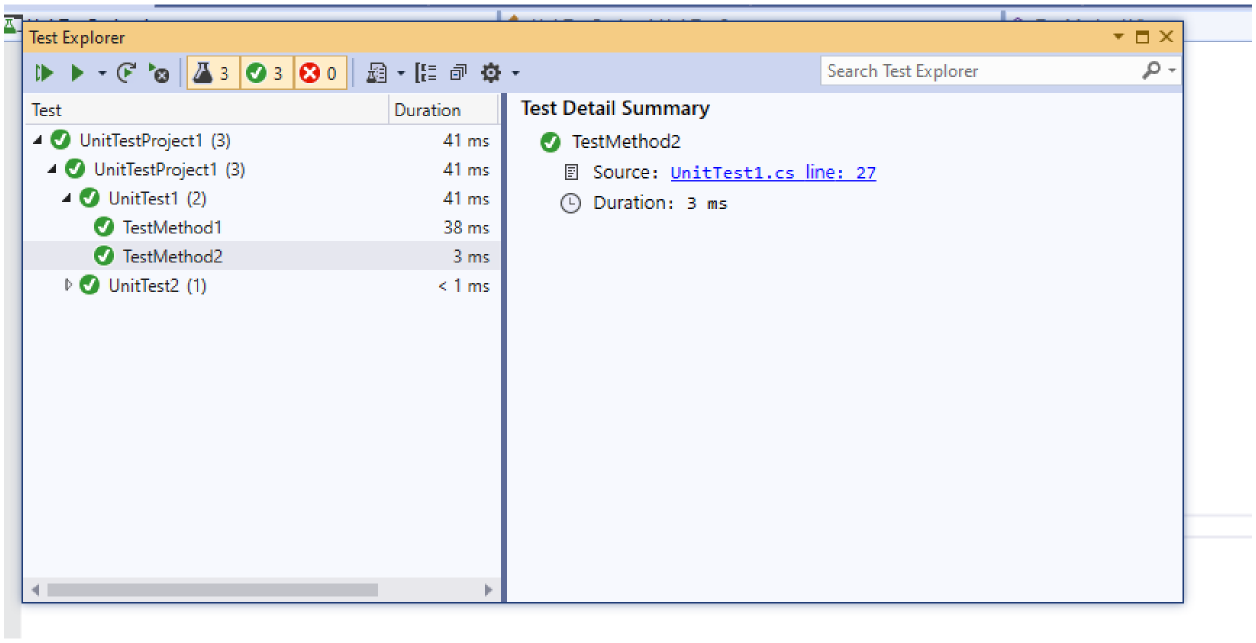
@Request for appointment.aspx.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using HospitalSystem.App_Code;
using System.Data;
using System.Data.SqlClient;

namespace HospitalSystem
{
    public partial class Request_for_appointment : System.Web.UI.Page
    {
        Class1 c = new Class1();
        protected void Page_Load(object sender, EventArgs e)
        {

            BindDropDownList();
            //unit test

            rqstunittest req = new rqstunittest();
            c.cmd.CommandText = "select * from Appointments";
            c.ds.Clear();
            c.adp.Fill(c.ds, "vt");
            c.dr = c.ds.Tables["vt"].NewRow();
            c.dr["Reason"] = req.reason;
            c.dr["PID"] = req.ptid;
            string Date1 = Convert.ToString(req.Date1);
            string Time1 = req.Time1;
            DateTime result = DateTime.Parse(Date1 + " " + Time1);
            c.dr["starttime"] = result;
            c.dr["DocID"] = req.docid;
            c.ds.Tables["vt"].Rows.Add(c.dr);
            c.scb.DataAdapter = c.adp;
            c.adp.Update(c.ds.Tables["vt"]);

            //unit test over

        }

/*
 * Description: testing whether the requested appointment for the patient is success or not
 * input: User ID, doctor ID, time and reason for the appointment
 * output: checks whether the requested appointment is displayed to the front desk user an
doctor
 */
```

@Requestforappointmentunittest.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

```
namespace HospitalSystem
{
    public class Requestforappointmentunittest
    {
        public string speciality;
        //public string Doctor;
        public DateTime Date1;
        public string Time1;
        public string reason;
        public string docid;
        public string ptid;
        public Requestforappointmentunittest()
        {
            this.speciality = "Cancer";
            this.docid = "doc2";
            this.ptid = "pt6";
            this.Date1 = Convert.ToDateTime("10 - 19 - 2019");
            this.Time1 = "9:00 am";
            this.reason = "headache";
        }
    }
}
```



```
Test Explorer
3  3  0                                   Search Test Explorer
Test                              Duration   Test Detail Summary
UnitTestProject1 (3)               41 ms       TestMethod2
  UnitTestProject1 (3)             41 ms         Source: UnitTest1.cs line: 27
    UnitTest1 (2)                  41 ms         Duration: 3 ms
      TestMethod1                  38 ms
      TestMethod2                   3 ms
    UnitTest2 (1)                < 1 ms
```

4. Who wrote what components/classes of the system

@HomePage

File Name	Person
Home.aspx	Yogesh
Home.aspx.cs	Yogesh
Style.css	Yogesh

@LoginPage

File Name	Person
Loginpage2.aspx	Yogesh
Loginpage2.aspx.cs	Yogesh
Loginpage2.aspx.designer.cs	Yogesh

@RegistrationPage

File Name	Person
Register.aspx	Manoj
Register.aspx.cs	Manoj
Register.aspx.designer.cs	Manoj

@AddDoctor

File Name	Person
AddDoc.aspx	Manoj
AddDoc.aspx.cs	Manoj
AddDoc.aspx.designer.cs	Manoj

@AddEmployee

File Name	Person
AddEmp.aspx	Manoj
AddEmp.aspx.cs	Manoj
AddEmp.aspx.designer.cs	Manoj

@WelcomeEmployee

File Name	Person
WelcomeEmp.aspx	NagaSaiTeja
WelcomeEmp.aspx.cs	NagaSaiTeja
WelcomeEmp.aspx.designer.cs	NagaSaiTeja

@WelcomeDoctor

File Name	Person
WelcomeDoctor.aspx	NagaSaiTeja
WelocmeDoctor.aspx.cs	NagaSaiTeja
WelcomeDoctor.aspx.designer.cs	NagaSaiTeja

@WelcomePatient

File Name	Person
WelcomePatient.aspx	NagaSaiTeja
WelcomePatient.aspx.cs	NagaSaiTeja
WelcomePatient.aspx.designer.cs	NagaSaiTeja

@RequestForAppointment

File Name	Person
Request for appointment.aspx	Maulshree Verma
Request for appointment.aspx.cs	Maulshree Verma
Request for appointment.aspx.designer.cs	Maulshree Verma

@AppointmentRequests (FrontDeskUser)

File Name	Person
appointmentrequests.aspx	Maulshree Verma
Appointmentrequests.aspx.cs	Maulshree Verma
Appointmentrequests.aspx.designer.cs	Maulshree Verma

@FrontEnd

File Name	Person
Home.aspx	Yogesh
Loginpage2.aspx	Yogesh
WelcomeDoctor.aspx	NagaSaiTeja

@DoctorAppointment

File Name	Person
DoctorAppointment.aspx	Manoj
DoctorAppointment.aspx.cs	Manoj
DoctorAppointment.aspx.designer.cs	Manoj

Database Connectivity:

@App_Code:

File Name	Person
Class1.cs	Maulshree Verma

5. User Manual

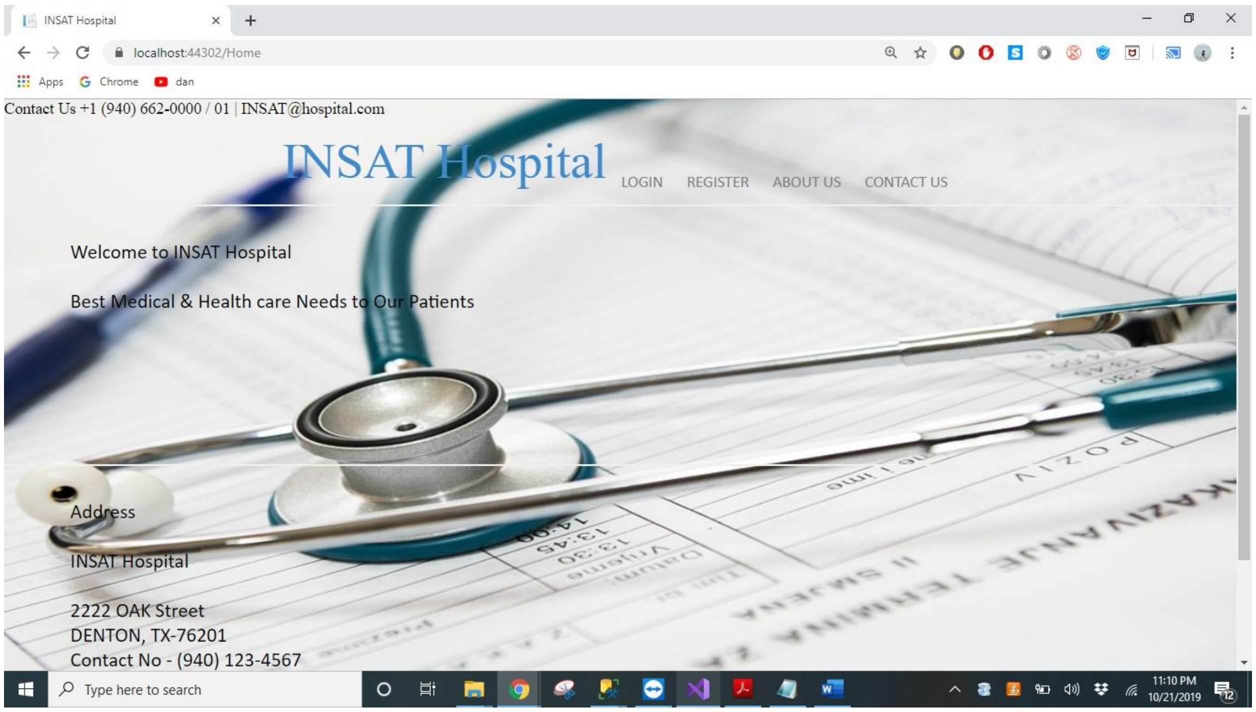
5.1. Summary:

Hospital Management System is a web application which provides a platform for patients to check with the doctor regarding their health-related issues.

This document contains detailed steps indicating its reader on how to use this application.

5.2. Home Page

This is the main page when any user enters the website’s URL. It displays basic options such as “Login”, “Register”, “Contact Us” and “About Us” and displays a list of featured properties. If the user doesn’t have the account, then he/she should click the “Register” button. If the user has an account, then he/she should click “Login” button. If the user needs any contact information about the hospital, then he/she can go to the “Contact Us”. If the user needs to know about the hospital, then he/she can go to the “About Us”.


Contact Us +1 (940) 662-0000 / 01 | INSAT@hospital.com

INSAT Hospital

LOGIN REGISTER ABOUT US CONTACT US

Welcome to INSAT Hospital

Best Medical & Health care Needs to Our Patients

Address

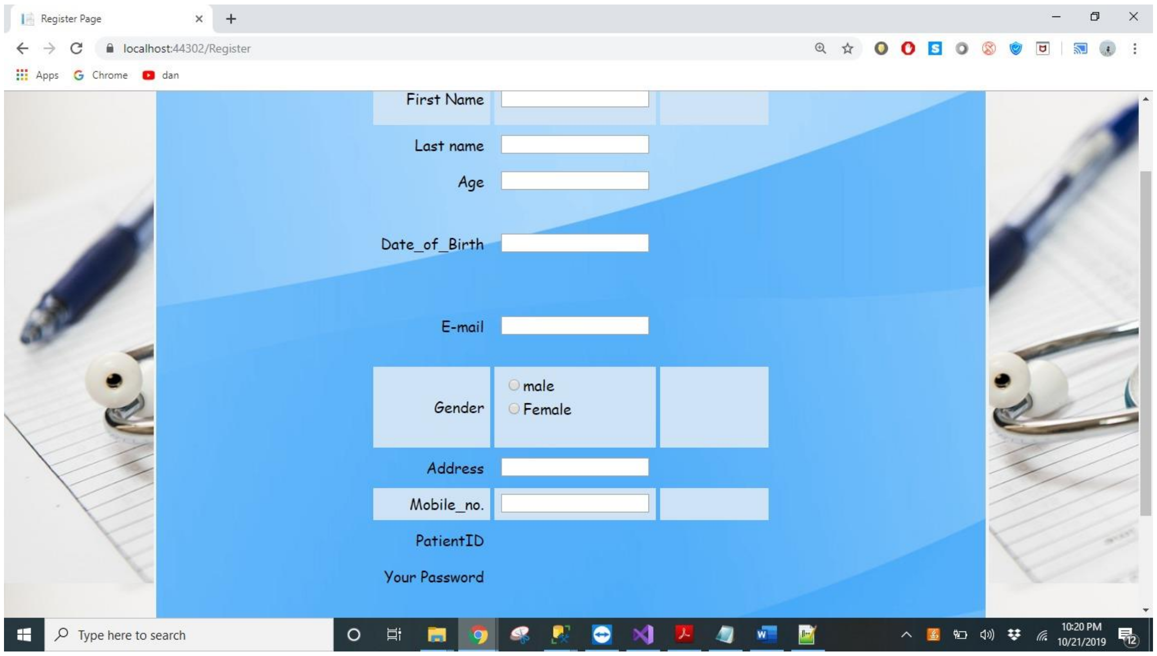
INSAT Hospital

2222 OAK Street
DENTON, TX-76201
Contact No - (940) 123-4567

5.3. Patient with no account:

A customer without an account in the system cannot be able to book an appointment.

- First, the user needs to create an account to book an appointment.
- For creating an account, the user should first go to register page and then needs to fill all the details like First Name, Last Name, Email-ID etc.
- Once the registration is successful, then the user is provided with UserID and temporary password.


First Name

Last name

Age

Date_of_Birth

E-mail

Gender ○ male ○ Female

Address

Mobile_no.

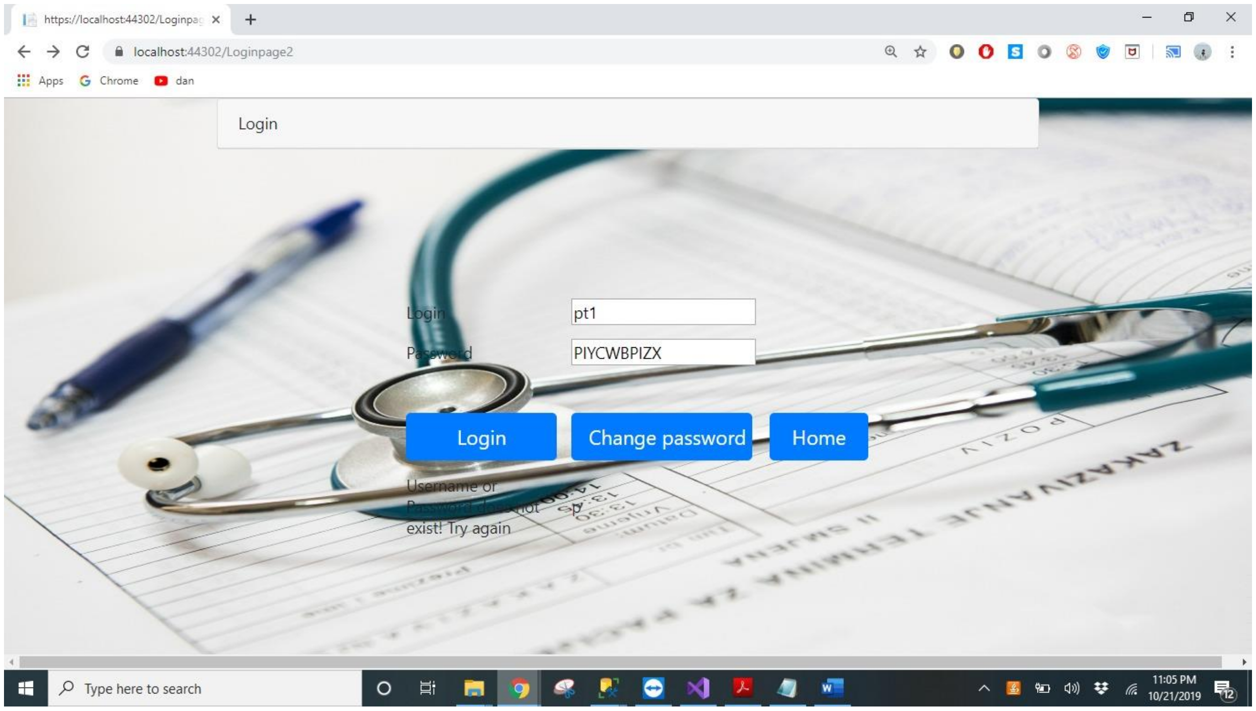
PatientID

Your Password

- With the UserID and password the user can login into his account.
- The password can be changed at any time by the user.

5.4. Patient with account:

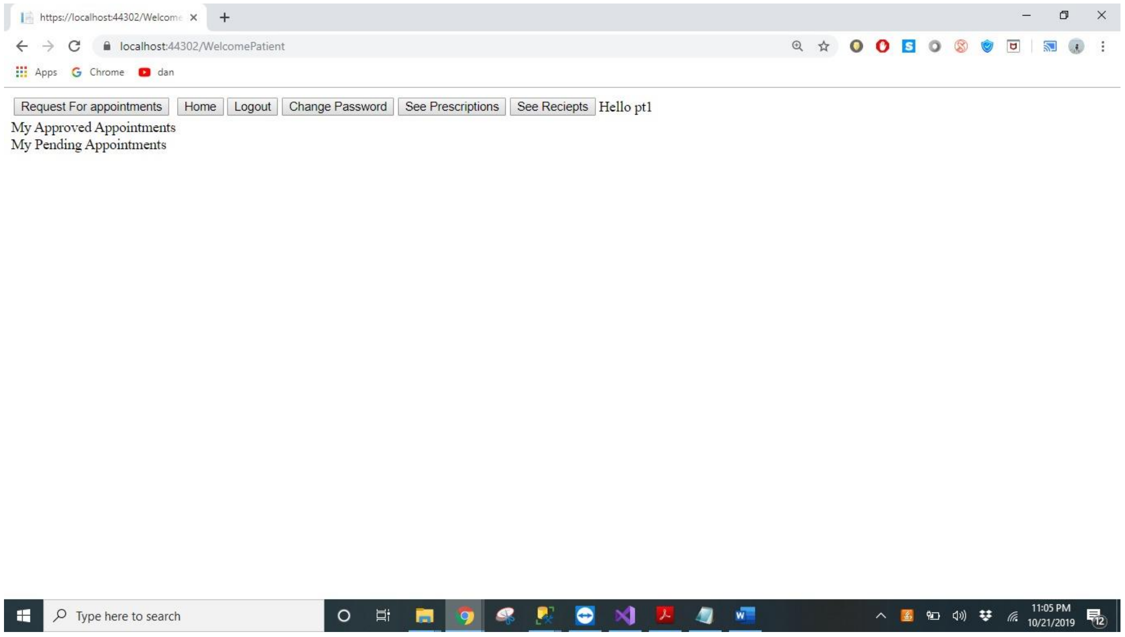
- The user with the account can login into his/her account.



Login pt1
Password PIYCWBPIZX

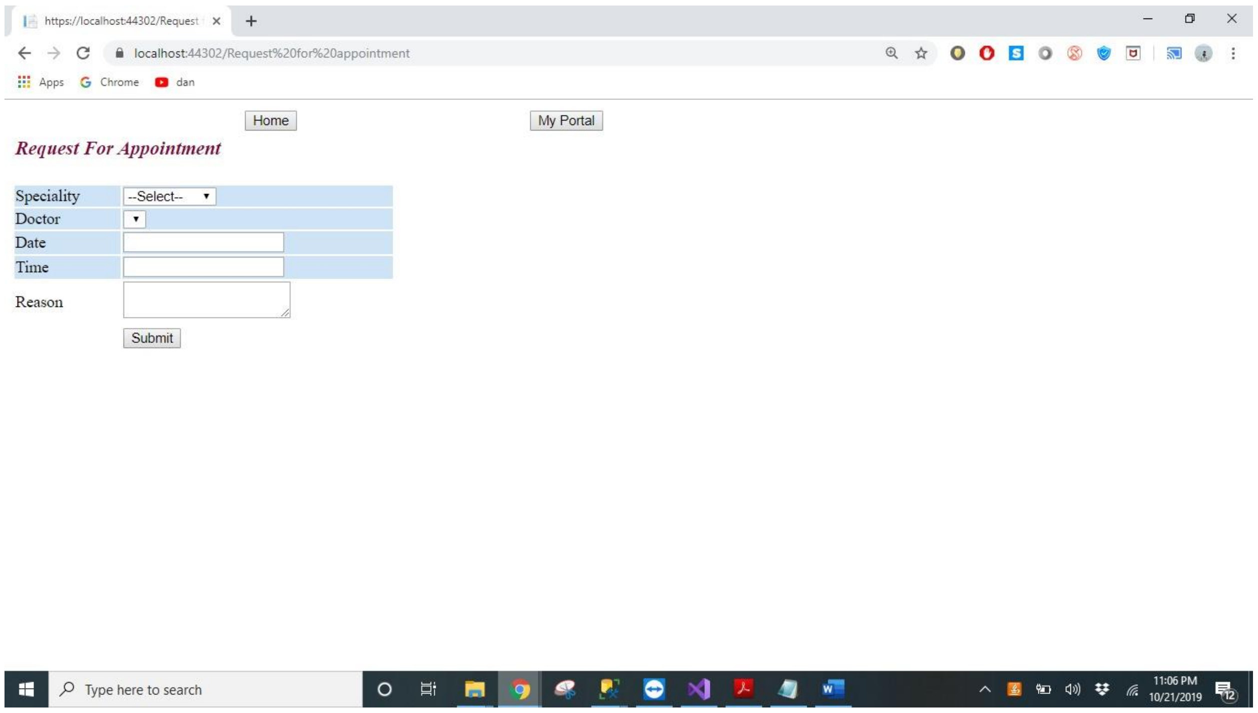
Username or Password does not exist! Try again

- Once the user is logged in, then the patient can request for an appointment.



Hello pt1
My Approved Appointments
My Pending Appointments

- While requesting for an appointment the patient can select the specialty, then the doctors with that specialty are displayed in a drop-down box.
- The patient can choose any doctor from that list, select the date and time and then request for an appointment which will be approved/declined by the front-desk user later.



Request For Appointment

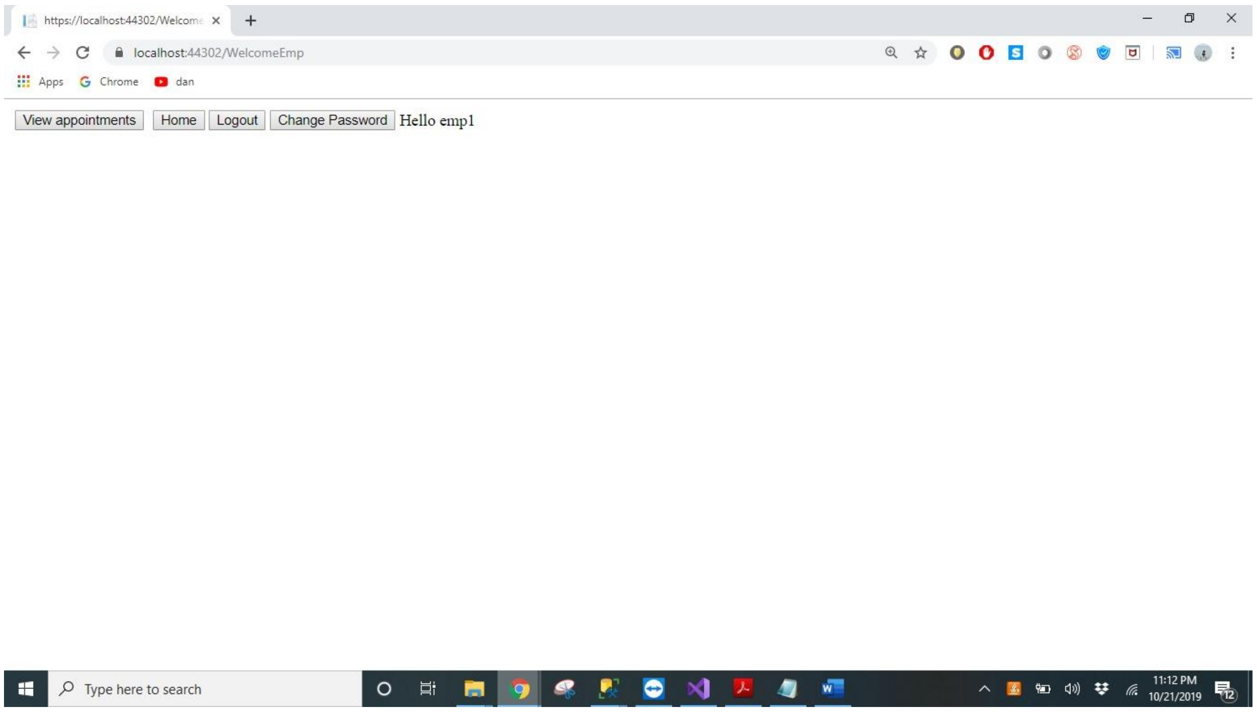
Speciality	--Select--
Doctor	
Date	
Time	
Reason	

5.5. Admin

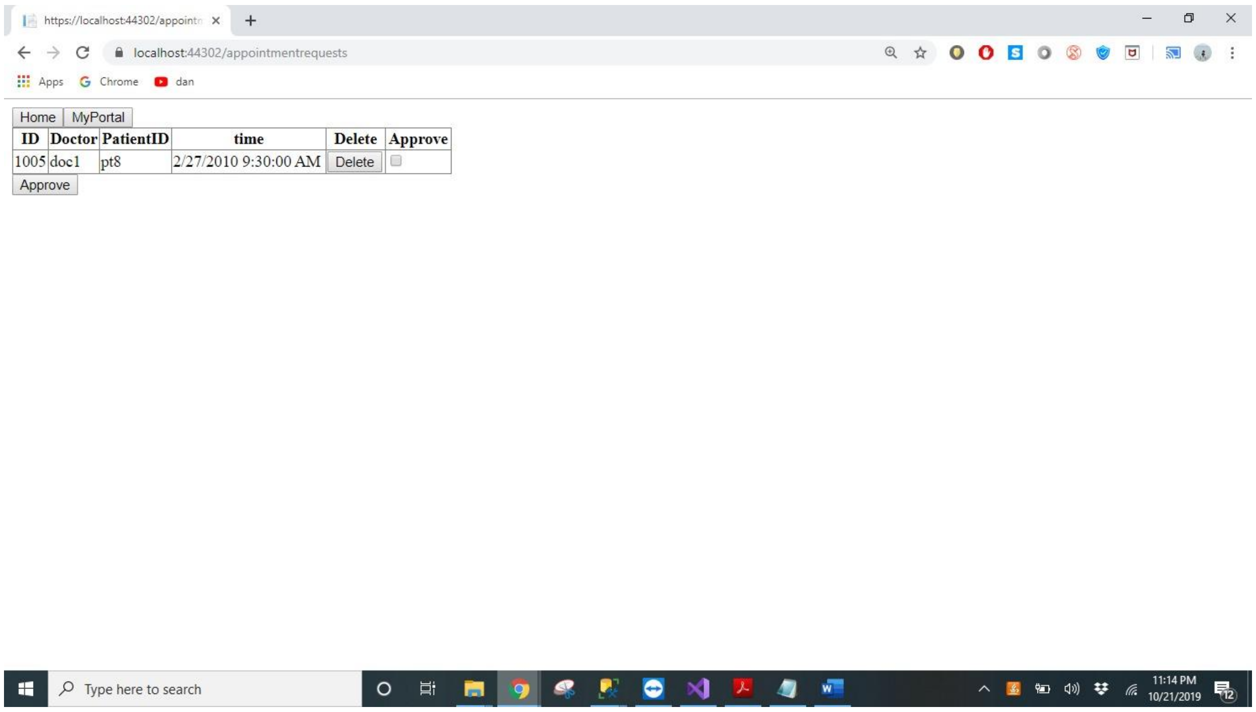
- Admin views a similar website as the customer with having additional options for him.
- The admin will add the doctors and the front-desk users into the system.
- The admin will have all the privileges.
- The admin generates the UserID and passwords for the doctors and Front-desk User.

5.6. Front-Desk User

- Once the Front-desk user logs into the account the Front-desk user page looks as follows:



- The Front-desk user will approve/decline the appointments requested by the patients based on the availability of the doctor.

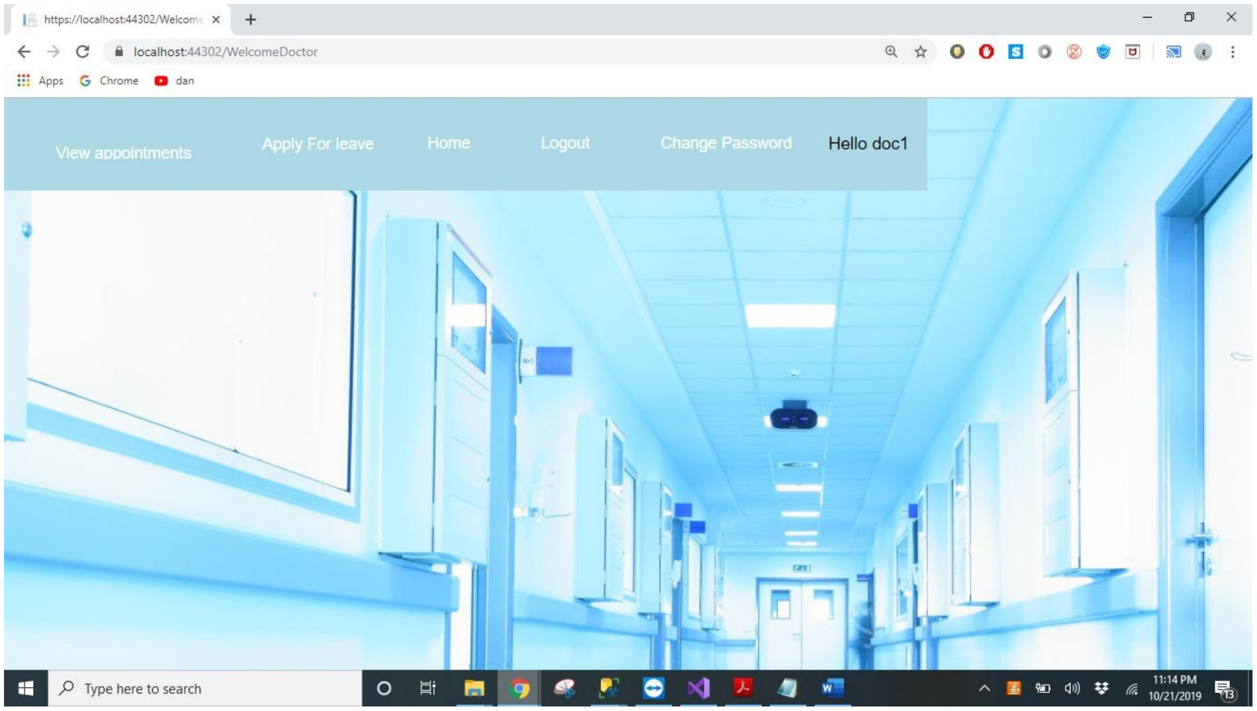


ID	Doctor	PatientID	time	Delete	Approve
1005	doc1	pt8	2/27/2010 9:30:00 AM	Delete	☐

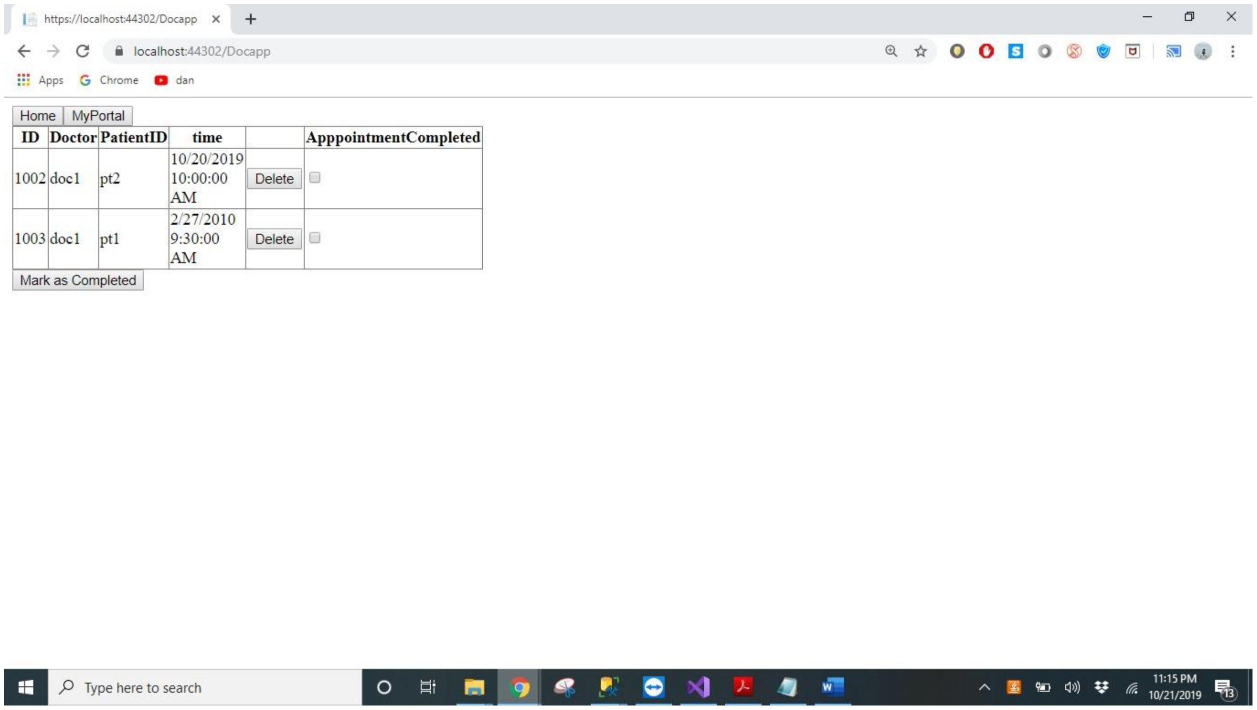
- The Front-desk user can add the patients into the systems when they walk-in into the hospital.

5.7. Doctor

- Once the doctor logs into the account the doctor's page looks as follows:



- Doctor checks the patient and updates the appointment status to completed if the treatment is done.



ID	Doctor	PatientID	time		ApppointmentCompleted
1002	doc1	pt2	10/20/2019 10:00:00 AM	Delete	☐
1003	doc1	pt1	2/27/2010 9:30:00 AM	Delete	☐

- Doctor updates his availability to the front-desk user.

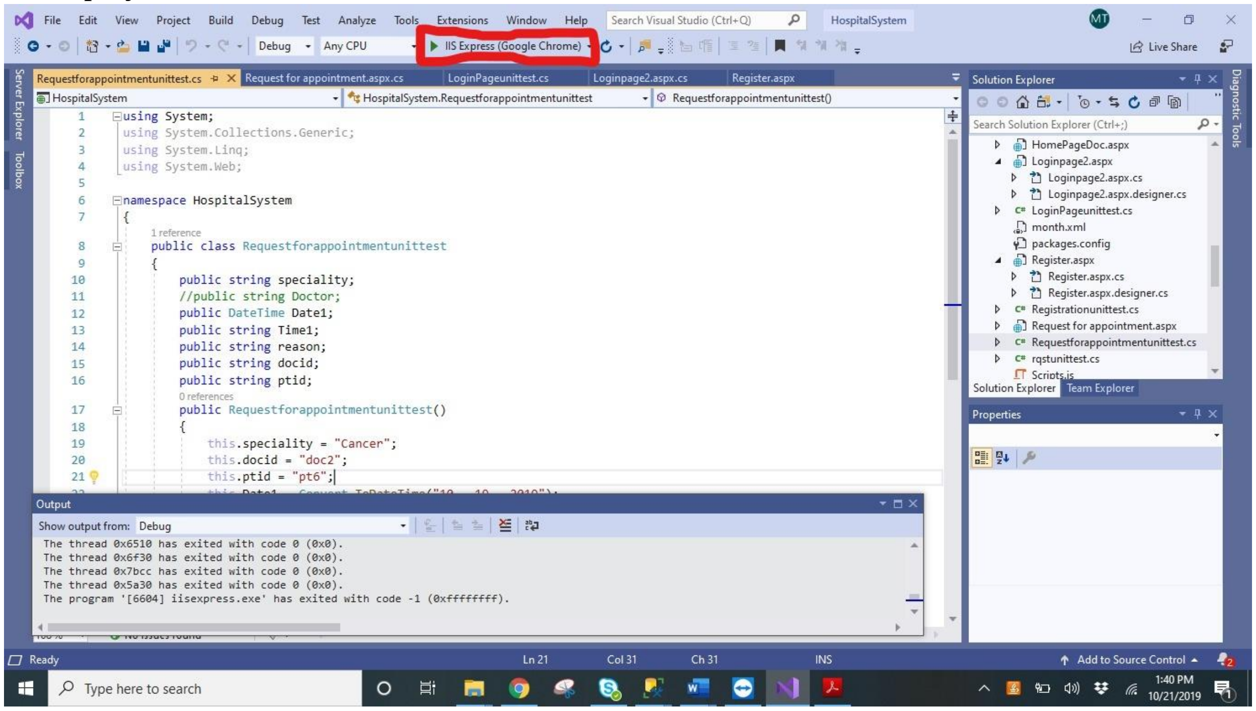
6. To compile/run the program and test cases

To compile/run the program and test cases:

- Install Visual Studio 2019
- Install Microsoft asp.net framework with C# for web development.
- Install SQL Server Management Studio (SSMS) for database.
- Install IIS server.

Compile/Run the program

- Open the Visual Studio software and create new web project.
- Open SQL Server Management Studio and create new account type "se".
- Create the database and required tables
- Create the web forms in the asp.net empty web application.
- Compile the web application by right click on the project in solution explorer and clicking on the option build solution.
- Set up database connectivity in the website using ADO.Net by adding a Connection Class in the project.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace HospitalSystem
{
    1 reference
    public class Requestforappointmentunittest
    {
        public string speciality;
        //public string Doctor;
        public DateTime Date1;
        public string Time1;
        public string reason;
        public string docid;
        public string ptid;
        0 references
        public Requestforappointmentunittest()
        {
            this.speciality = "Cancer";
            this.docid = "doc2";
            this.ptid = "pt6";
```

```
The thread 0x6510 has exited with code 0 (0x0).
The thread 0x6f30 has exited with code 0 (0x0).
The thread 0x7bcc has exited with code 0 (0x0).
The thread 0x5a30 has exited with code 0 (0x0).
The program '[6604] iisexpress.exe' has exited with code -1 (0xffffffff).
```

- Click on the green symbol (in red color box shown in the above picture) to execute the entire project.
- Execute the created web pages in the localhost by right clicking on the web forms and selecting view in browser option and validate the results in SSMS.

Sample Credentials for functional testing:

Sample login credentials:

Admin:
ID: **admin1234**
Password: **passwordqwerty**

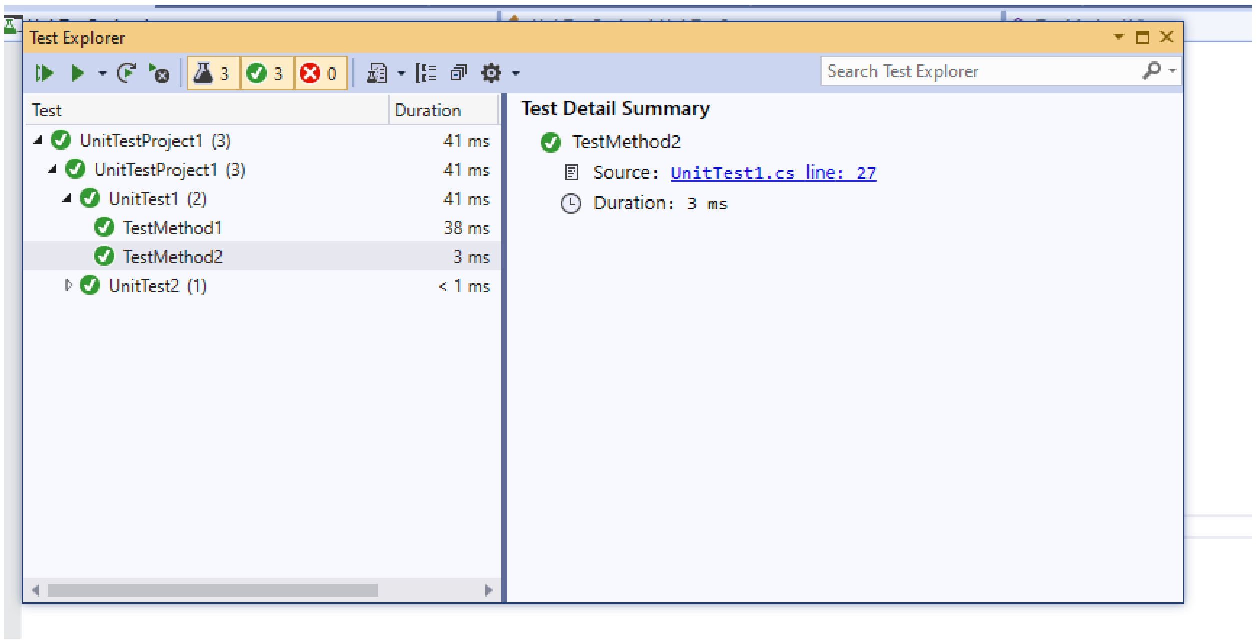
FrontDeskUser:
ID: **emp1**
Password: **MQAVHFSLLS**

Patient:
Id: **pt1**
Password: **PIYCWBIZX**

Doctor:
Id: **doc1**
Password: **ZWFTXCJZBM**

Compile/Run the test cases

- For unit test cases we add unit test project in the project solution in visual studio and create a C# class for unit testing.
- Create a demo test class in the Hospital System project for the webform we want to test. For example, for unit testing login page we create a class Loginunittest.cs
- We initialize the variables used in that web form and hard code the values in the constructor.
- Next, we check theses values in the unit test class created in the Unit Test Project in the same solution explorer. We pass these values in the “TestMethod()” function of unit test class.
- For executing all the unit test we right click on the Unit test project in the solution explorer and select the option- “run all tests”. We get the result as below:



Test	Duration
UnitTestProject1 (3)	41 ms
UnitTestProject1 (3)	41 ms
UnitTest1 (2)	41 ms
TestMethod1	38 ms
TestMethod2	3 ms
UnitTest2 (1)	< 1 ms

Test Detail Summary

TestMethod2

Source: UnitTest1.cs line: 27

Duration: 3 ms

7. Feedback received during peer review Session

- Less validations of the data fields in registration form.
- Enhance User Interface.

7.1. Actions taken based on the feedback

- For validation of the data fields in the registration form, we applied ASP.Net validation controls which checks the values entered in the data fields such as email id, etc.
- We will be using bootstrapping for User Interface enhancements.