Project Phase 3 Report For Hospital Management System

Version 3.0

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

University of North Texas, Denton.

11/18/2019

Table of Contents

1. Requirements

This web-application mainly focuses on four modules namely Admin, front-desk user, doctor and patient modules. 100% of the project is completed and below given features are the major features implemented in the development phase-3 of this web-application.

1.1. ABF: Appointment Booking by Front-Desk User

- PRIORITY: HIGH
- The Front-desk user can book appointment for the patients.
- In case if the patient visits the hospital and wants his appointment to get booked at the front desk, the front desk user can create an appointment for the patient based on the doctor's availability.

1.2. LD: List of Doctors

- PRIORITY: MEDIUM
- The list of doctors will be available in the home page.
- The users can view the list of doctors based on the specialty.
- The users can also sort the doctors based on the feedback.

1.3. UPP: Uploading Profile Picture

- PRIORITY: LOW
- The users can upload the profile picture while doing the registration of their account.
- The users can upload their profile picture in their welcome page of the user's account.

1.4. RFD: Rating and Feedback about Doctors

- PRIORITY: LOW
- The patients can rate their doctors once their appointment is completed.
- The rating is on the scale of 1 to 5 and they can also describe their experience.

1.5. CU: Contact-Us Page

- PRIORITY: LOW
- The Contact-Us link will be available on the home page.
- If the user clicks on the Contact-Us option it will redirect to the Contact-Us page which will have the hospital's address, email ID and telephone number etc.

1.6. AU: About-Us Page

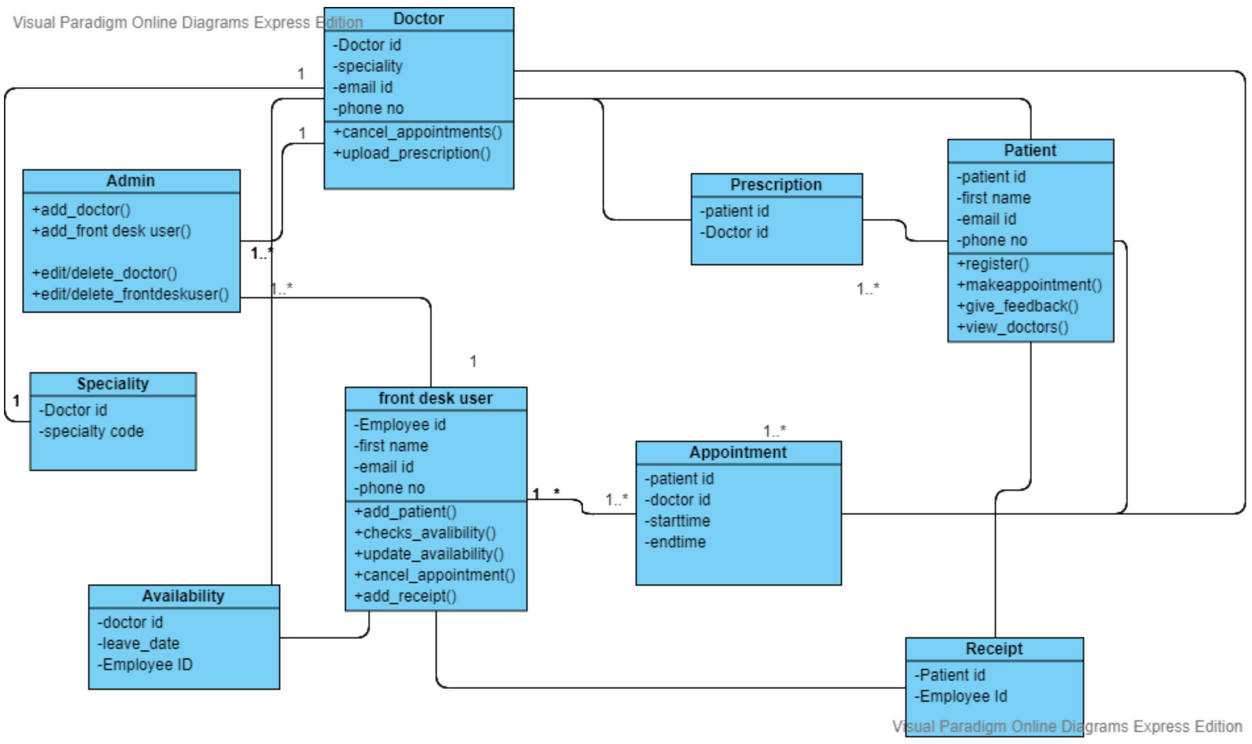
- PRIORITY: LOW
- The About-Us link will be available on the home page.
- If the user clicks on the About-Us option it will redirect to the About-Us page which will have the hospital's history, trustees and recognition etc.

1.7. Updated Requirements

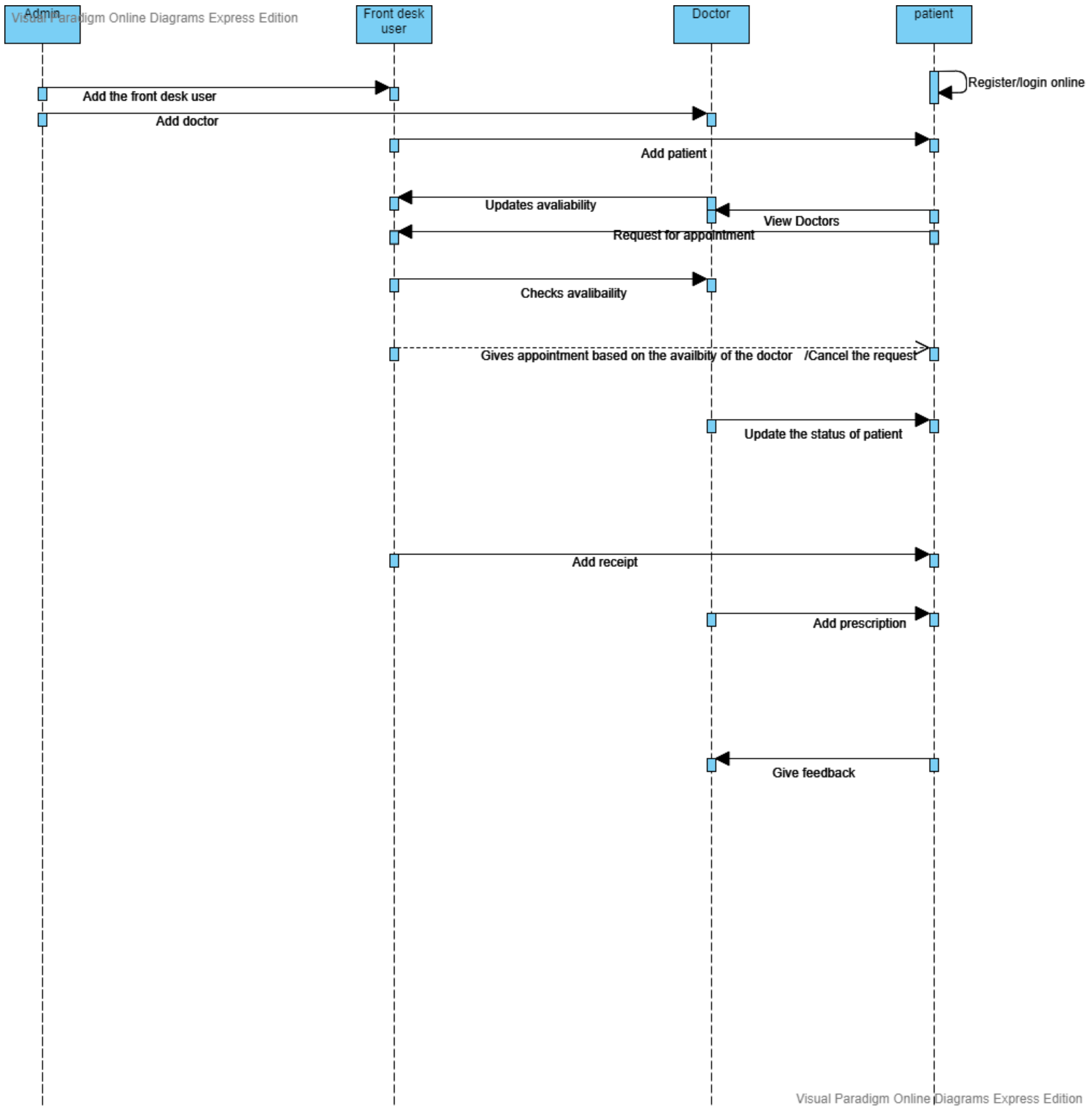
- We have updated the admin module.
- The admin can now check the leaves applied by the doctor and can delete those leaves when the end date is over. This can help the front-desk user in checking only the future availability of the doctors and not his past appointments and leaves.

2. UML Design

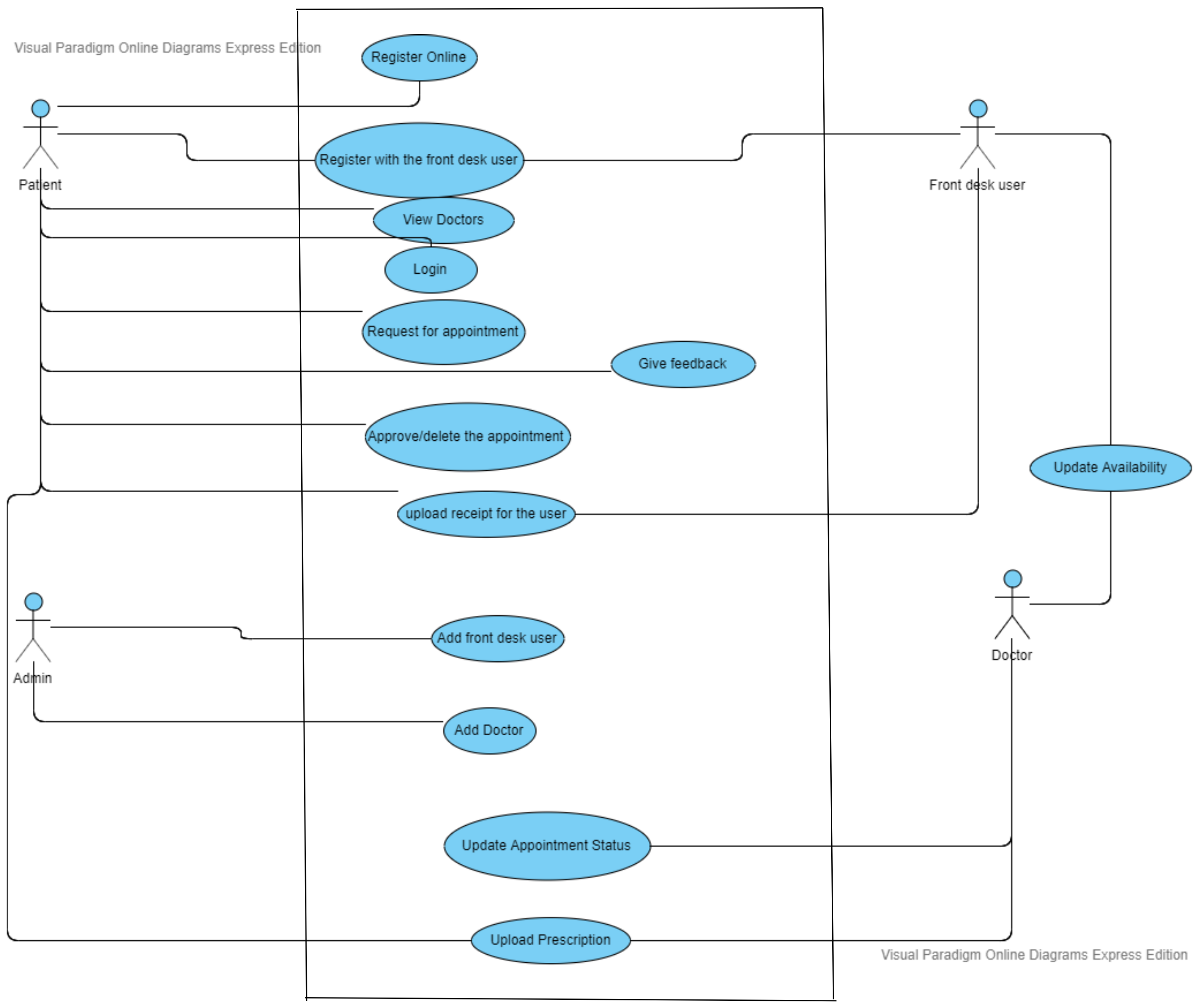
2.1. Class Diagram



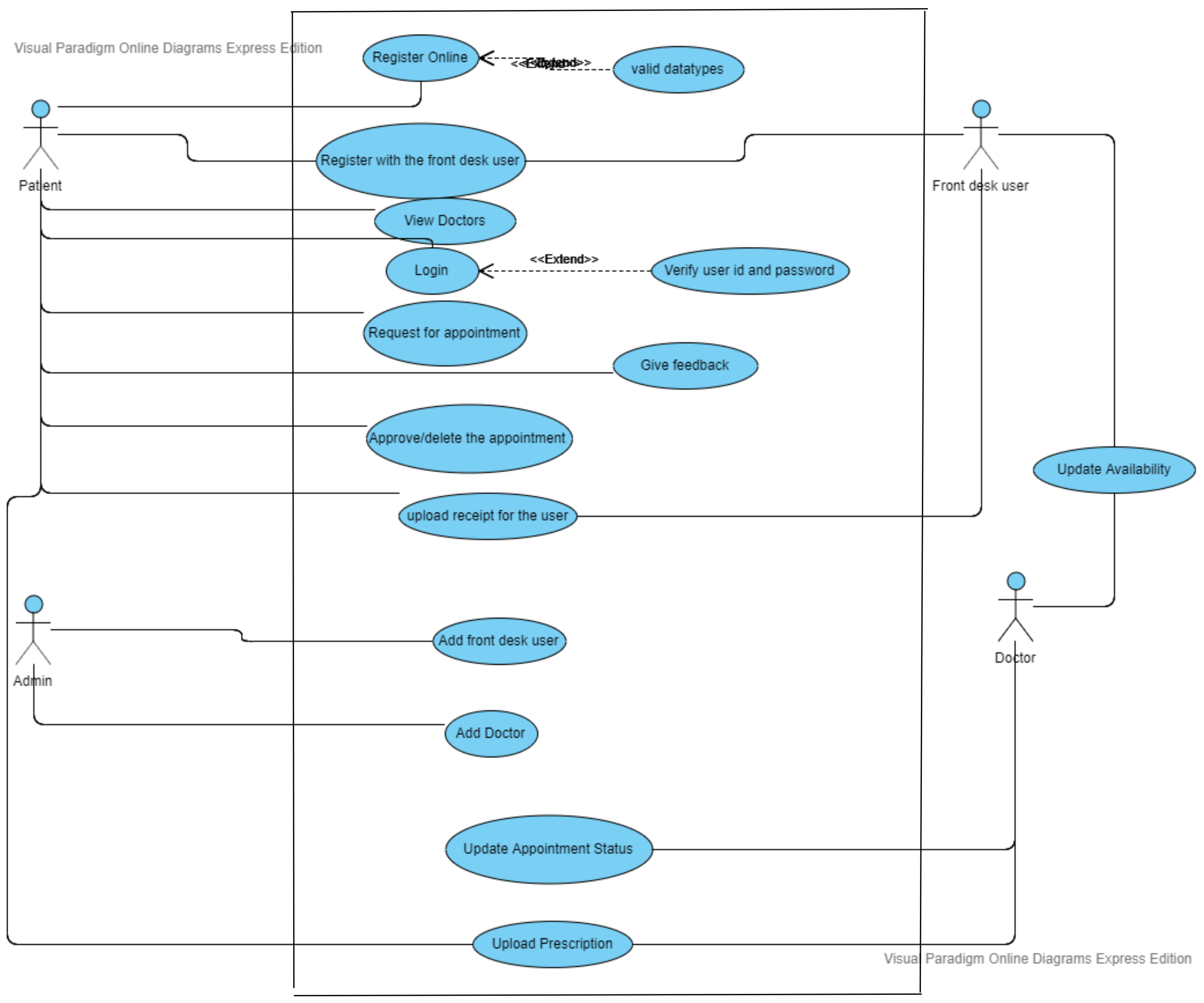
2.2. Sequence Diagram



2.3. Use case Diagram (Normal Case)


Visual Paradigm Online Diagrams Express Edition
Patient
Front desk user
Admin
Doctor
Register Online
Register with the front desk user
View Doctors
Login
Request for appointment
Give feedback
Approve/delete the appointment
upload receipt for the user
Update Availability
Add front desk user
Add Doctor
Update Appointment Status
Upload Prescription
Visual Paradigm Online Diagrams Express Edition

2.4. Use case Diagram (Error Case)


Visual Paradigm Online Diagrams Express Edition
Register Online
<<Extend>>
valid datatypes
Patient
Register with the front desk user
Front desk user
View Doctors
Login
<<Extend>>
Verify user id and password
Request for appointment
Give feedback
Approve/delete the appointment
Update Availability
upload receipt for the user
Admin
Add front desk user
Doctor
Add Doctor
Update Appointment Status
Upload Prescription
Visual Paradigm Online Diagrams Express Edition

3. Test Cases

3.1. Unit test-cases for the deliverable-5 requirements

Test Class1:

Number of Test Cases = 14

@ PatientRegisterUnittest.cs:

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

namespace HospitalSystem
{
    public class PatientRegisterUnittest
    {
        //unit test for testing registration page
        public string FirstName;
        public string LastName;
        public int Age;
        public string DOB;
        public string Email;
        public long phone;
        public string Address;
        public string gender;
        public bool tnc;

        public RegisterUnittest()
        {
            //unit test case when terms and conditions are not checked

            this.FirstName = "Sam";
            this.LastName = "Dean";
            this.Age = 30;
            this.DOB = "28/08/1989";
            this.Email = "sam@gmail.com";
            this.phone = 9898989898;
            this.Address = "123 Avenue A";
            this.gender = "male";
            this.tnc = true;
        }
    }
}
```

UnitTest4.cs:

//Description: This class is used to test whether the front-desk user is giving all the valid data fields of the registration page such as name, address, gender, date of birth etc. We did **14 tests** for the test class.

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

namespace UnitTestProject1
{
    [TestClass]
    public class UnitTest4
    {
        [TestMethod]
        public void TestMethod1()
        {
            //unit test for testing if the terms and condition are not checked
            RegisterUnittest ob = new RegisterUnittest();
            Assert.AreEqual("Sam", ob.FirstName);
            Assert.AreEqual("Dean", ob.LastName);
            Assert.AreEqual(9898989898, ob.phone);
            Assert.AreEqual(false, ob.tnc);

        }

        [TestMethod]
        public void TestMethod2()
        {
            //unit test for testing if the terms and condition are checked
            RegisterUnittest ob = new RegisterUnittest();
            Assert.AreEqual("Sam", ob.FirstName);
            Assert.AreEqual("Dean", ob.LastName);
            Assert.AreEqual(9898989898, ob.phone);
            Assert.AreEqual(true, ob.tnc);

        }

        [TestMethod]
```

```
public void TestMethod3()
{
   //unit test for testing if the phone number is not in correct format
   RegisterUnittest ob = new RegisterUnittest();
   Assert.AreEqual("Sam", ob.FirstName);
   Assert.AreEqual("Dean", ob.LastName);
   Assert.AreEqual(98989898, ob.phone);
   Assert.AreEqual("123 Avenue A", ob.Address);
   Assert.AreEqual(true, ob.tnc);

}

[TestMethod]
public void TestMethod4()
{
   //unit test for testing if the phone number is in correct format
   RegisterUnittest ob = new RegisterUnittest();
   Assert.AreEqual("Sam", ob.FirstName);
   Assert.AreEqual("Dean", ob.LastName);
   Assert.AreEqual(9898989898, ob.phone);
   Assert.AreEqual("123 Avenue A", ob.Address);
   Assert.AreEqual(true, ob.tnc);

}

[TestMethod]
public void TestMethod5()
{
   //unit test for testing if the email is not in correct format
   RegisterUnittest ob = new RegisterUnittest();
   Assert.AreEqual("Sam", ob.FirstName);
   Assert.AreEqual("Dean", ob.LastName);
   Assert.AreEqual("sam", ob.Email);
   Assert.AreEqual(9898989898, ob.phone);
   Assert.AreEqual("123 Avenue A", ob.Address);
   Assert.AreEqual(true, ob.tnc);

}

[TestMethod]
public void TestMethod6()
{
   //unit test for testing if the email is in correct format
   RegisterUnittest ob = new RegisterUnittest();
   Assert.AreEqual("Sam", ob.FirstName);
   Assert.AreEqual("Dean", ob.LastName);
```

```
             Assert.AreEqual("sam@gmail.com", ob.Email);
            Assert.AreEqual(9898989898, ob.phone);
            Assert.AreEqual("123 Avenue A", ob.Address);
            Assert.AreEqual(true, ob.tnc);

        }

        [TestMethod]
        public void TestMethod7()
        {
            //unit test for testing if the email is left blank correct format
            RegisterUnittest ob = new RegisterUnittest();
            Assert.AreEqual("Sam", ob.FirstName);
            Assert.AreEqual("Dean", ob.LastName);
            Assert.AreEqual(" ", ob.Email);
            Assert.AreEqual(9898989898, ob.phone);
            Assert.AreEqual("123 Avenue A", ob.Address);
            Assert.AreEqual(true, ob.tnc);

        }

        [TestMethod]
        public void TestMethod8()
        {
            //unit test for testing if the email already exists
            RegisterUnittest ob = new RegisterUnittest();
            Assert.AreEqual("Sam", ob.FirstName);
            Assert.AreEqual("Dean", ob.LastName);
            Assert.AreEqual("dave@gmail.com", ob.Email);
            Assert.AreEqual(9898989898, ob.phone);
            Assert.AreEqual("123 Avenue A", ob.Address);
            Assert.AreEqual(true, ob.tnc);
        }

        [TestMethod]
        public void TestMethod9()
        {
            //unit test for testing if the Date of birth is not in correct format
            RegisterUnittest ob = new RegisterUnittest();
            Assert.AreEqual("Sam", ob.FirstName);
            Assert.AreEqual("Dean", ob.LastName);
            Assert.AreEqual("28/15/1989", ob.DOB);
            Assert.AreEqual("sam@gmail.com", ob.Email);
            Assert.AreEqual(9898989898, ob.phone);
            Assert.AreEqual("123 Avenue A", ob.Address);
            Assert.AreEqual(true, ob.tnc);
```

```
}

[TestMethod]
public void TestMethod10()
{
   //unit test for testing if the Date of birth is in correct format
   RegisterUnittest ob = new RegisterUnittest();
   Assert.AreEqual("Sam", ob.FirstName);
   Assert.AreEqual("Dean", ob.LastName);
   Assert.AreEqual("28/08/1989", ob.DOB);
   Assert.AreEqual("sam@gmail.com", ob.Email);
   Assert.AreEqual(9898989898, ob.phone);
   Assert.AreEqual("123 Avenue A", ob.Address);
   Assert.AreEqual(true, ob.tnc);
}

[TestMethod]
public void TestMethod11()
{
   //unit test for testing if the age is in integer format
   RegisterUnittest ob = new RegisterUnittest();
   Assert.AreEqual("Sam", ob.FirstName);
   Assert.AreEqual("Dean", ob.LastName);
   Assert.AreEqual("28/08/1989", ob.DOB);
   Assert.AreEqual("sam@gmail.com", ob.Email);
   Assert.AreEqual(9898989898, ob.phone);
   Assert.AreEqual("10i", ob.Age);
   Assert.AreEqual("123 Avenue A", ob.Address);
   Assert.AreEqual(true, ob.tnc);
}

[TestMethod]
public void TestMethod12()
{
   //unit test for testing if the age is in correct format
   RegisterUnittest ob = new RegisterUnittest();
   Assert.AreEqual("Sam", ob.FirstName);
   Assert.AreEqual("Dean", ob.LastName);
   Assert.AreEqual("28/08/1989", ob.DOB);
   Assert.AreEqual("sam@gmail.com", ob.Email);
   Assert.AreEqual(9898989898, ob.phone);
   Assert.AreEqual(30, ob.Age);
   Assert.AreEqual("123 Avenue A", ob.Address);
   Assert.AreEqual(true, ob.tnc);
}
```

```
        [TestMethod]
        public void TestMethod13()
        {
            //unit test for testing if the gender field is left blank
            RegisterUnittest ob = new RegisterUnittest();
            Assert.AreEqual("Sam", ob.FirstName);
            Assert.AreEqual("Dean", ob.LastName);
            Assert.AreEqual("28/08/1989", ob.DOB);
            Assert.AreEqual("sam@gmail.com", ob.Email);
            Assert.AreEqual(9898989898, ob.phone);
            Assert.AreEqual(30, ob.Age);
            Assert.AreEqual("123 Avenue A", ob.Address);
            Assert.AreEqual(" ", ob.gender);
            Assert.AreEqual(true, ob.tnc);
        }


        [TestMethod]
        public void TestMethod14()
        {
            //unit test for testing if the gender field is selected
            RegisterUnittest ob = new RegisterUnittest();
            Assert.AreEqual("Sam", ob.FirstName);
            Assert.AreEqual("Dean", ob.LastName);
            Assert.AreEqual("28/08/1989", ob.DOB);
            Assert.AreEqual("sam@gmail.com", ob.Email);
            Assert.AreEqual(9898989898, ob.phone);
            Assert.AreEqual(30, ob.Age);
            Assert.AreEqual("123 Avenue A", ob.Address);
            Assert.AreEqual("male", ob.gender);
            Assert.AreEqual(true, ob.tnc);
        }
    }
}
```

Test Class 2:

Number of Test Cases = 7

@Ratingunittest.cs:

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;
```

```
namespace HospitalSystem
{
    public class Ratingunittest
    {
        public string docid;
        public string patientid;
        public int rating;

        public Ratingunittest()
        {
            this.patientid = "pt10";
            this.docid = "doc1";
            this.rating = 3;
        }
    }
}
```

UnitTest6.cs:

//Description: This class is used to test whether the rating and feedback details given is correct or not. This checks whether the doctor and patient’s id are valid or not. It also checks whether the rating is between 1 to 5 inclusive. We did **7 tests** for the test class.

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

namespace UnitTestProject1
{
    [TestClass]
    public class UnitTest6
    {
        [TestMethod]
        public void TestMethod1()
        {
            //when doctor id is not selected
            Ratingunittest ob = new Ratingunittest();
            Assert.AreEqual("", ob.docid);
        }

        [TestMethod]
        public void TestMethod2()
        {
            //when doctor id is selected
            Ratingunittest ob = new Ratingunittest();
            Assert.AreEqual("doc1", ob.docid);
        }

        [TestMethod]
        public void TestMethod3()
        {
            //when patient id is null
            Ratingunittest ob = new Ratingunittest();
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("", ob.patientid);

        }
        [TestMethod]
        public void TestMethod4()
        {
            //when patient id is selected
            Ratingunittest ob = new Ratingunittest();
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("pt10", ob.patientid);
        }

        [TestMethod]
        public void TestMethod5()
        {
            //when rating is greater than 5
            Ratingunittest ob = new Ratingunittest();
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("pt10", ob.patientid);
            Assert.AreEqual(6, ob.rating);

        }

        [TestMethod]
        public void TestMethod6()
        {
            //when no rating is given
            Ratingunittest ob = new Ratingunittest();
```

```
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("pt10", ob.patientid);
            Assert.AreEqual(0 , ob.rating);

        }

        [TestMethod]
        public void TestMethod7()
        {
            //when rating is greater than 5
            Ratingunittest ob = new Ratingunittest();
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("pt10", ob.patientid);
            Assert.AreEqual(3, ob.rating);

        }
    }
}
```

Test Class 3:

Number of Test Cases = 20

@AppointmentUnittest.cs:

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

namespace HospitalSystem
{
    public class AppointmentUnittest
    {
        public string patientid;
        public string docid;
        public string fduid;
        public string date;
        public string time;
        public bool isdeleted;
        public bool isaccepted;
        public bool istreated;

        public AppointmentUnittest()
        {
            this.patientid = "pt2";
            this.docid = "doc1";
            this.fduid = "emp2";
            this.date = "30/11/2019";
            this.time = "9:00 am";
            this.isaccepted = true;
            this.isdeleted = false;
            this.istreated = false;
        }


    }
}
```

Unittest7.cs:

//Description: This class is used to test whether the front-desk user is giving all the valid data fields for requesting an appointment such as doctor id, patient id etc. It also checks whether the date and time of the appointment is entered in correct format or not. We did **20 tests** for the test class. It checks that the isapproved check should only be true and isdeleted and istreated functionality should be false.

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

namespace UnitTestProject1
{
    [TestClass]
    public class UnitTest7
    {
        //unit test for create new appointment by front desk user
        [TestMethod]
        public void TestMethod1()
        {
            //unit test when docid is not selected
            AppointmentUnittest ob = new AppointmentUnittest();
            Assert.AreEqual("", ob.docid);
        }

        [TestMethod]
        public void TestMethod2()
        {
            //unit test when docid is selected
            AppointmentUnittest ob = new AppointmentUnittest();
            Assert.AreEqual("doc1", ob.docid);
        }

        [TestMethod]
        public void TestMethod3()
        {
            //unit test when patient is not selected
            AppointmentUnittest ob = new AppointmentUnittest();
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("", ob.patientid);
        }
        [TestMethod]
        public void TestMethod4()
        {
            //unit test when patient is selected
            AppointmentUnittest ob = new AppointmentUnittest();
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("pt2", ob.patientid);
        }

        [TestMethod]
        public void TestMethod5()
        {
            //unit test when fdu is empty
            AppointmentUnittest ob = new AppointmentUnittest();
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("pt2", ob.patientid);
            Assert.AreEqual("", ob.fduid);
        }

        [TestMethod]
        public void TestMethod6()
        {
            //unit test when fdu is selected
            AppointmentUnittest ob = new AppointmentUnittest();
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("pt2", ob.patientid);
            Assert.AreEqual("emp2", ob.fduid);
        }

        [TestMethod]
        public void TestMethod7()
        {
            //unit test when date is not selected
            AppointmentUnittest ob = new AppointmentUnittest();
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("pt2", ob.patientid);
            Assert.AreEqual("emp2", ob.fduid);
            Assert.AreEqual("", ob.date);
        }

        [TestMethod]
        public void TestMethod8()
        {
            //unit test when date is not in correct format
            AppointmentUnittest ob = new AppointmentUnittest();
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("pt2", ob.patientid);
            Assert.AreEqual("emp2", ob.fduid);
            Assert.AreEqual("11-30-2019", ob.date);
        }

        [TestMethod]
        public void TestMethod9()
        {
            //unit test when date is in correct format
            AppointmentUnittest ob = new AppointmentUnittest();
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("pt2", ob.patientid);
            Assert.AreEqual("emp2", ob.fduid);
            Assert.AreEqual("30/11/2019", ob.date);
        }

        [TestMethod]
        public void TestMethod10()
        {
            //unit test when time is not selected
            AppointmentUnittest ob = new AppointmentUnittest();
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("pt2", ob.patientid);
            Assert.AreEqual("emp2", ob.fduid);
            Assert.AreEqual("30/11/2019", ob.date);
            Assert.AreEqual("", ob.time);
        }
        [TestMethod]
        public void TestMethod11()
        {
            //unit test when time is not in correct format
            AppointmentUnittest ob = new AppointmentUnittest();
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("pt2", ob.patientid);
            Assert.AreEqual("emp2", ob.fduid);
            Assert.AreEqual("30/11/2019", ob.date);
            Assert.AreEqual("9:00:00", ob.time);
        }

        [TestMethod]
        public void TestMethod12()
        {
            //unit test when time is in correct format
            AppointmentUnittest ob = new AppointmentUnittest();
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("pt2", ob.patientid);
            Assert.AreEqual("emp2", ob.fduid);
            Assert.AreEqual("30/11/2019", ob.date);
            Assert.AreEqual("9:00 am", ob.time);
        }


        [TestMethod]
        public void TestMethod13()
        {
            //unit test when isdeleted is false and is accepted and istreated is false
            AppointmentUnittest ob = new AppointmentUnittest();
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("pt2", ob.patientid);
            Assert.AreEqual("emp2", ob.fduid);
            Assert.AreEqual("30/11/2019", ob.date);
            Assert.AreEqual("9:00 am", ob.time);
            Assert.AreEqual(false, ob.isdeleted);
            Assert.AreEqual(false, ob.isaccepted);
            Assert.AreEqual(false, ob.istreated);


        }

        [TestMethod]
        public void TestMethod14()
        {
            //unit test when isaccepted is false and is deleted is true and istreated is false
            AppointmentUnittest ob = new AppointmentUnittest();
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("pt2", ob.patientid);
            Assert.AreEqual("emp2", ob.fduid);
            Assert.AreEqual("30/11/2019", ob.date);
            Assert.AreEqual("9:00 am", ob.time);
            Assert.AreEqual(true, ob.isdeleted);
            Assert.AreEqual(false, ob.isaccepted);
            Assert.AreEqual(false, ob.istreated);
```

```
}
[TestMethod]
public void TestMethod15()
{
   //unit test when isaccepted is false and is deleted is false and istreated is true
   AppointmentUnittest ob = new AppointmentUnittest();
   Assert.AreEqual("doc1", ob.docid);
   Assert.AreEqual("pt2", ob.patientid);
   Assert.AreEqual("emp2", ob.fduid);
   Assert.AreEqual("30/11/2019", ob.date);
   Assert.AreEqual("9:00 am", ob.time);
   Assert.AreEqual(false, ob.isdeleted);
   Assert.AreEqual(false, ob.isaccepted);
   Assert.AreEqual(true, ob.istreated);
}
[TestMethod]
public void TestMethod16()
{
   //unit test when isaccepted is false and is deleted is true and istreated is true
   AppointmentUnittest ob = new AppointmentUnittest();
   Assert.AreEqual("doc1", ob.docid);
   Assert.AreEqual("pt2", ob.patientid);
   Assert.AreEqual("emp2", ob.fduid);
   Assert.AreEqual("30/11/2019", ob.date);
   Assert.AreEqual("9:00 am", ob.time);
   Assert.AreEqual(true, ob.isdeleted);
   Assert.AreEqual(false, ob.isaccepted);
   Assert.AreEqual(true, ob.istreated);
}
[TestMethod]
public void TestMethod17()
{
   //unit test when isaccepted is true and is deleted is true and istreated is false
   AppointmentUnittest ob = new AppointmentUnittest();
   Assert.AreEqual("doc1", ob.docid);
   Assert.AreEqual("pt2", ob.patientid);
   Assert.AreEqual("emp2", ob.fduid);
   Assert.AreEqual("30/11/2019", ob.date);
   Assert.AreEqual("9:00 am", ob.time);
   Assert.AreEqual(true, ob.isdeleted);
   Assert.AreEqual(true, ob.isaccepted);
   Assert.AreEqual(false, ob.istreated);
}

[TestMethod]
public void TestMethod18()
```

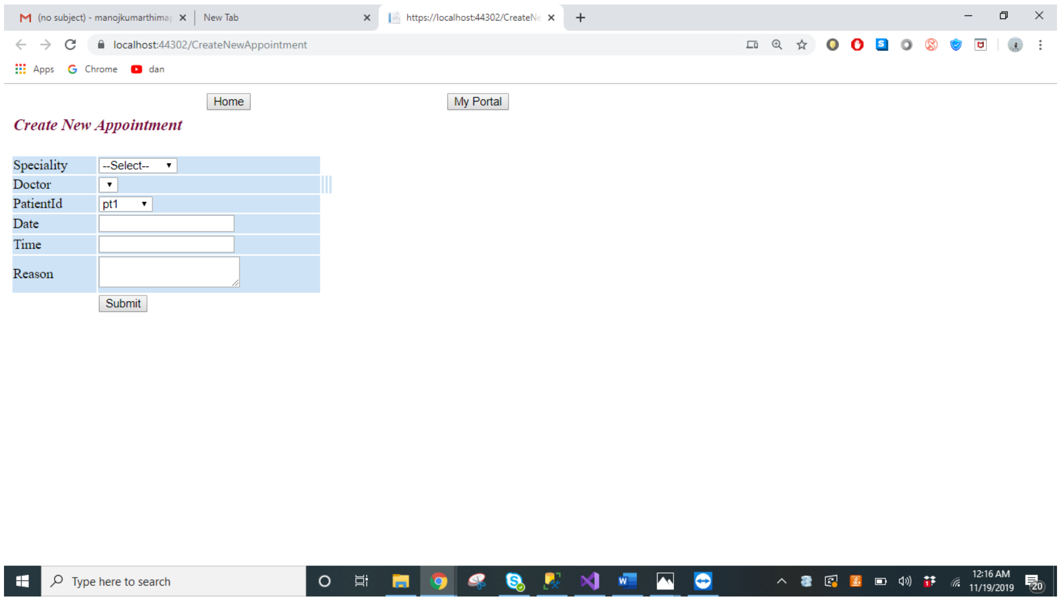
```
        {
            //unit test when isaccepted is true and is deleted is false and istreated is true
            AppointmentUnittest ob = new AppointmentUnittest();
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("pt2", ob.patientid);
            Assert.AreEqual("emp2", ob.fduid);
            Assert.AreEqual("30/11/2019", ob.date);
            Assert.AreEqual("9:00 am", ob.time);
            Assert.AreEqual(false, ob.isdeleted);
            Assert.AreEqual(true, ob.isaccepted);
            Assert.AreEqual(true, ob.istreated);
        }

        [TestMethod]
        public void TestMethod19()
        {
            //unit test when isaccepted is true and is deleted is true and istreated is true
            AppointmentUnittest ob = new AppointmentUnittest();
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("pt2", ob.patientid);
            Assert.AreEqual("emp2", ob.fduid);
            Assert.AreEqual("30/11/2019", ob.date);
            Assert.AreEqual("9:00 am", ob.time);
            Assert.AreEqual(true, ob.isdeleted);
            Assert.AreEqual(true, ob.isaccepted);
            Assert.AreEqual(true, ob.istreated);
        }

        [TestMethod]
        public void TestMethod20()
        {
            //unit test when isaccepted is true and is deleted is false and istreated is false
            AppointmentUnittest ob = new AppointmentUnittest();
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("pt2", ob.patientid);
            Assert.AreEqual("emp2", ob.fduid);
            Assert.AreEqual("30/11/2019", ob.date);
            Assert.AreEqual("9:00 am", ob.time);
            Assert.AreEqual(false, ob.isdeleted);
            Assert.AreEqual(true, ob.isaccepted);
            Assert.AreEqual(false, ob.istreated);
        }
    }
}
```

3.2. System test-cases for the deliverable-5 requirements

3.1 System Test-cases for requirement Appointment creation by Front-desk User for patient:

The front-desk user can create a new confirmed appointment by selecting the doctor id, patient id, date, time and reason for the appointment.

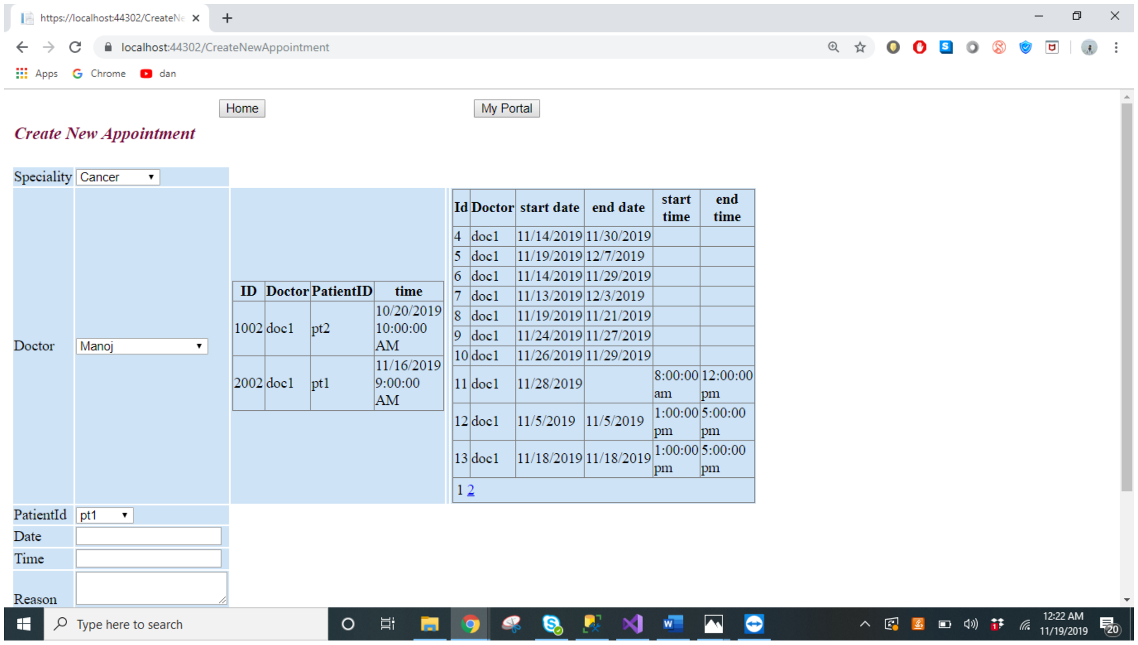
Case 1:


localhost:44302/CreateNewAppointment

Create New Appointment

Speciality	--Select--
Doctor	
PatientId	pt1
Date	
Time	
Reason	

Case 2:


localhost:44302/CreateNewAppointment

Create New Appointment

Speciality	Cancer
Doctor	Manoj
PatientId	pt1
Date	
Time	
Reason	

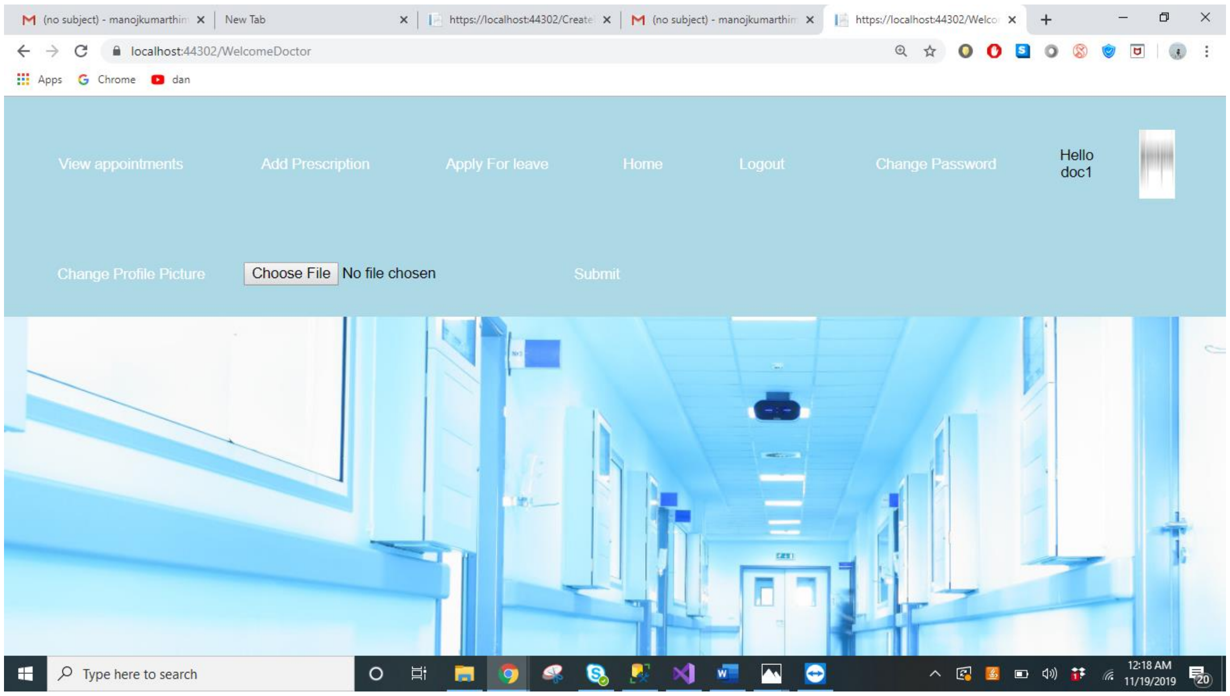
ID	Doctor	PatientID	time
1002	doc1	pt2	10/20/2019 10:00:00 AM
2002	doc1	pt1	11/16/2019 9:00:00 AM

Id	Doctor	start date	end date	start time	end time
4	doc1	11/14/2019	11/30/2019		
5	doc1	11/19/2019	12/7/2019		
6	doc1	11/14/2019	11/29/2019		
7	doc1	11/13/2019	12/3/2019		
8	doc1	11/19/2019	11/21/2019		
9	doc1	11/24/2019	11/27/2019		
10	doc1	11/26/2019	11/29/2019		
11	doc1	11/28/2019		8:00:00 am	12:00:00 pm
12	doc1	11/5/2019	11/5/2019	1:00:00 pm	5:00:00 pm
13	doc1	11/18/2019	11/18/2019	1:00:00 pm	5:00:00 pm
1 2					

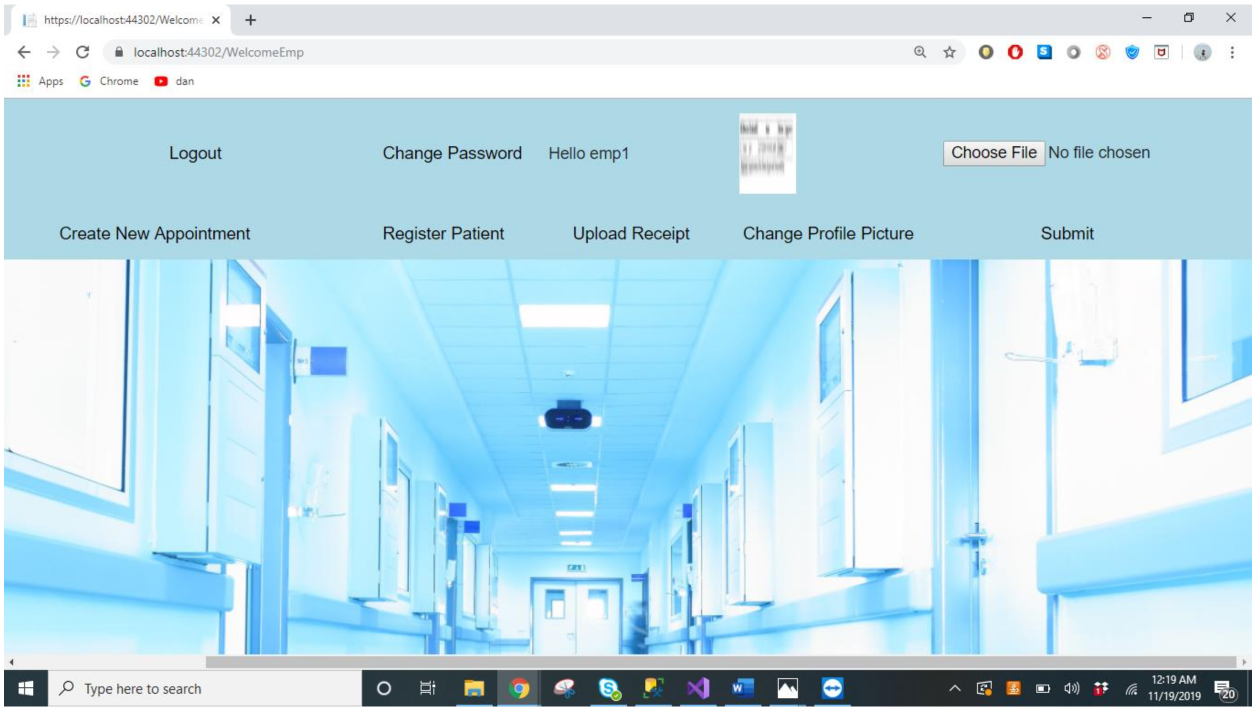
3.2 System Test-case for the Upload Profile Picture:

The doctor and Front-Desk User can change their profile pictures

Case1: Doctor's Profile Picture

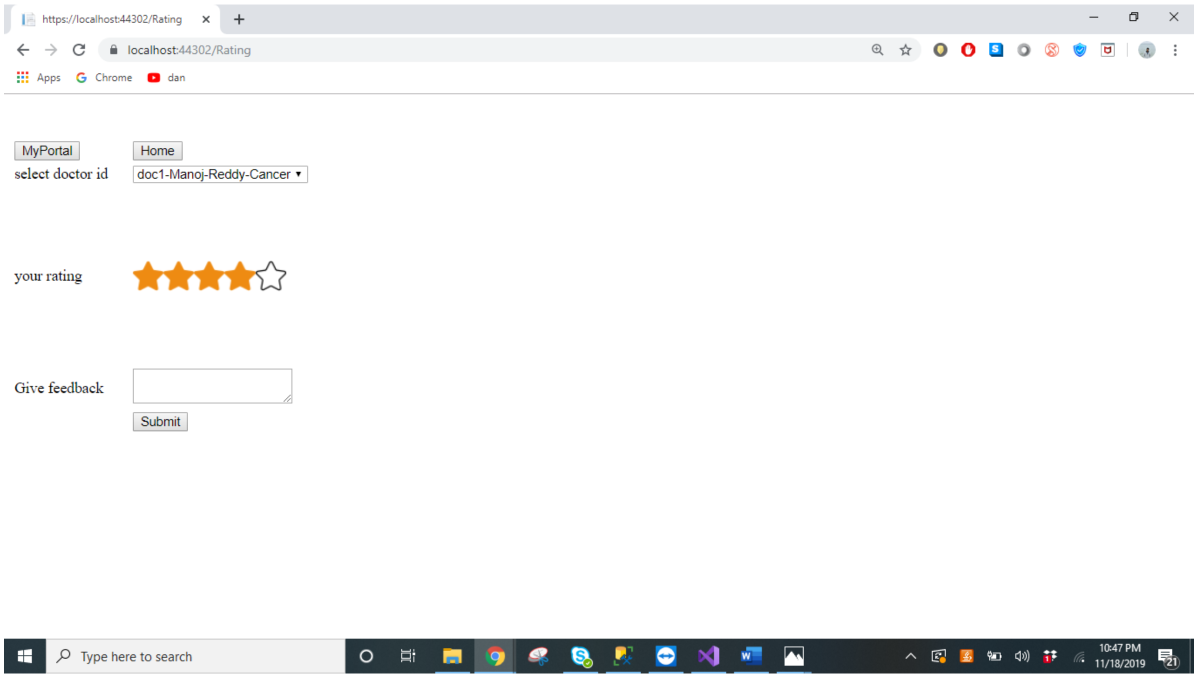


Case 2: Front-Desk User Profile Picture



3.3 System Test-case for Doctor’s Rating and Review

The Patient can rate only the doctor who has treated him/her.



select doctor id doc1-Manoj-Reddy-Cancer

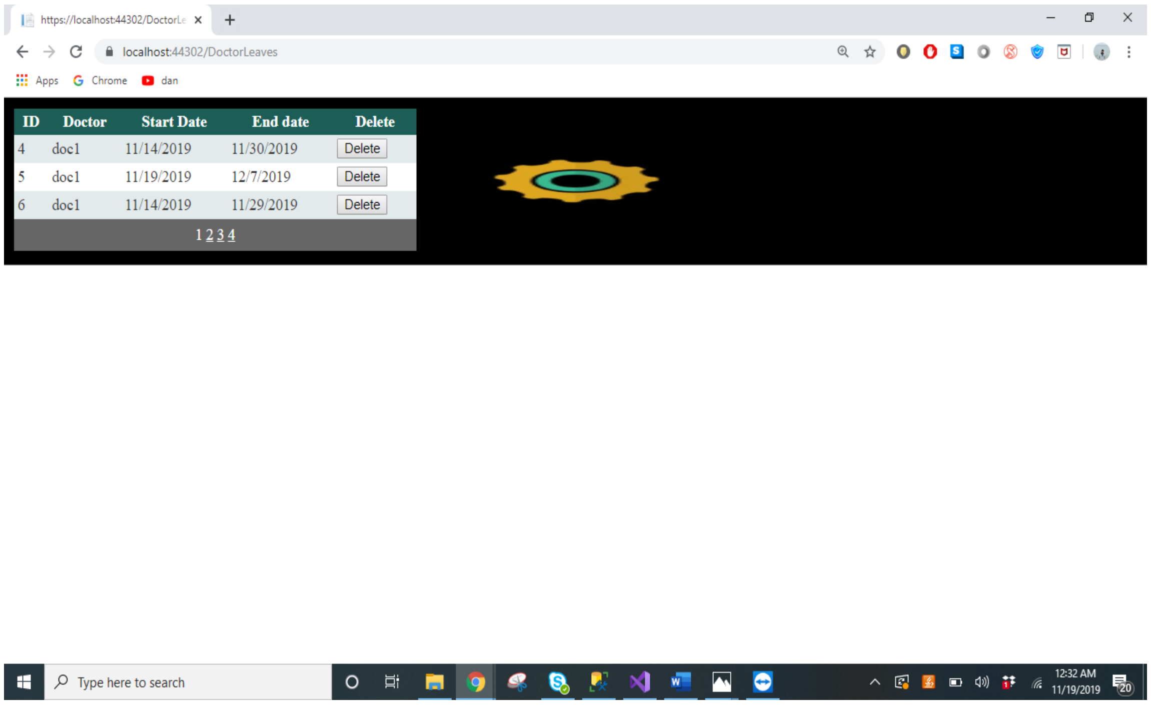
your rating ★★★★☆

Give feedback

Submit

3.4 System Test-case Deletion of Leaves by Admin

The Admin can delete the leaves of the doctor.



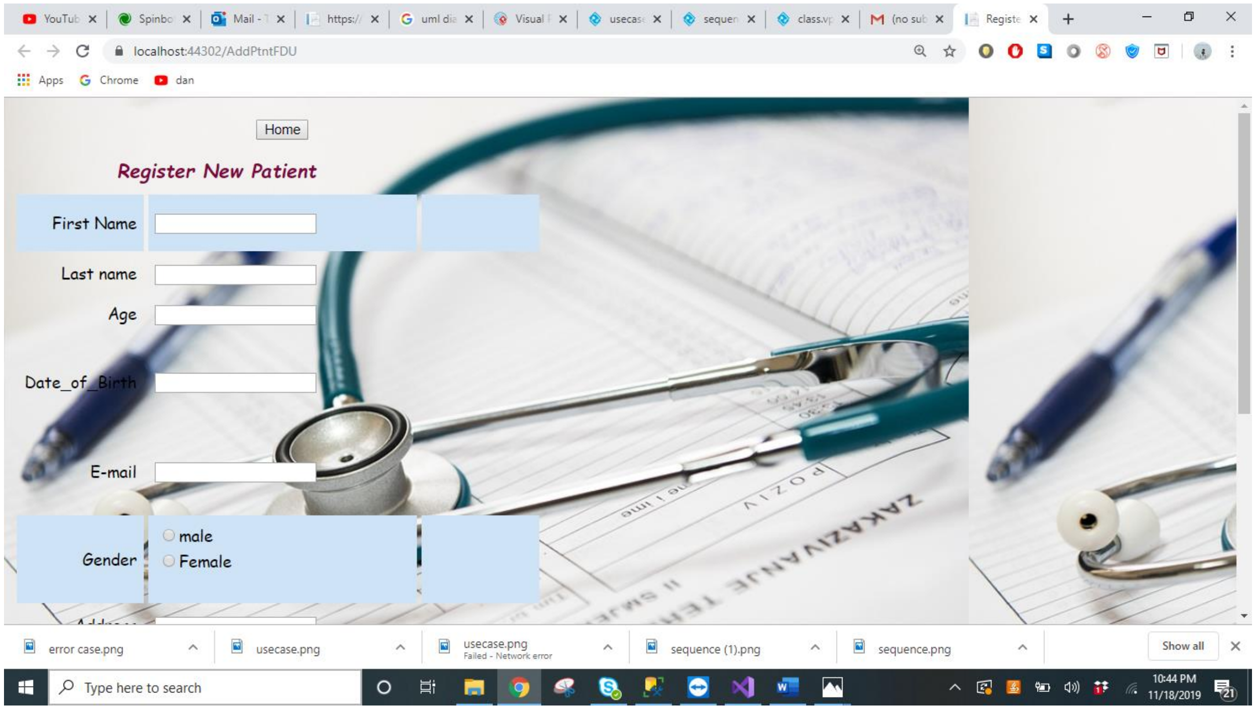
https://localhost:44302/DoctorLe
localhost:44302/DoctorLeaves
Apps
Chrome
dan
Type here to search
12:32 AM
11/19/2019

ID	Doctor	Start Date	End date	Delete
4	doc1	11/14/2019	11/30/2019	Delete
5	doc1	11/19/2019	12/7/2019	Delete
6	doc1	11/14/2019	11/29/2019	Delete

1 2 3 4

3.5 System Test-case for Patient Registration by Front-desk User

The Front-desk user can register an account for the patients.



4. Who wrote what components/classes of the system

@Testing

File Name	Person
RatingUnittest.cs	Maulshree Verma
AppointmentUnittest.cs	Maulshree Verma
PatientRegisterUnittest.cs	Manoj
Unittest5.cs	Manoj
Unittest6.cs	Yogesh
Unittest7.cs	Naga SaiTeja

@CreateNewAppointment

File Name	Person
CreateNewAppointment.aspx	Yogesh

@ListofDoctors

File Name	Person
ViewourDoctors.aspx	Manoj

@About-Us

File Name	Person
AboutUs.aspx	Naga SaiTeja

@Contact-Us

File Name	Person
ContactUs.aspx	Naga SaiTeja

@Rating.cs

Filename	Person
Rating.aspx	Maulshree Verma
Rating.Designer	Maulshree Verma
Rating.aspx.cs	Maulshree Verma

Database Connectivity:

@App_Code:

File Name	Person
Class1.cs	Maulshree Verma

5. User Manual

5.1. Summary:

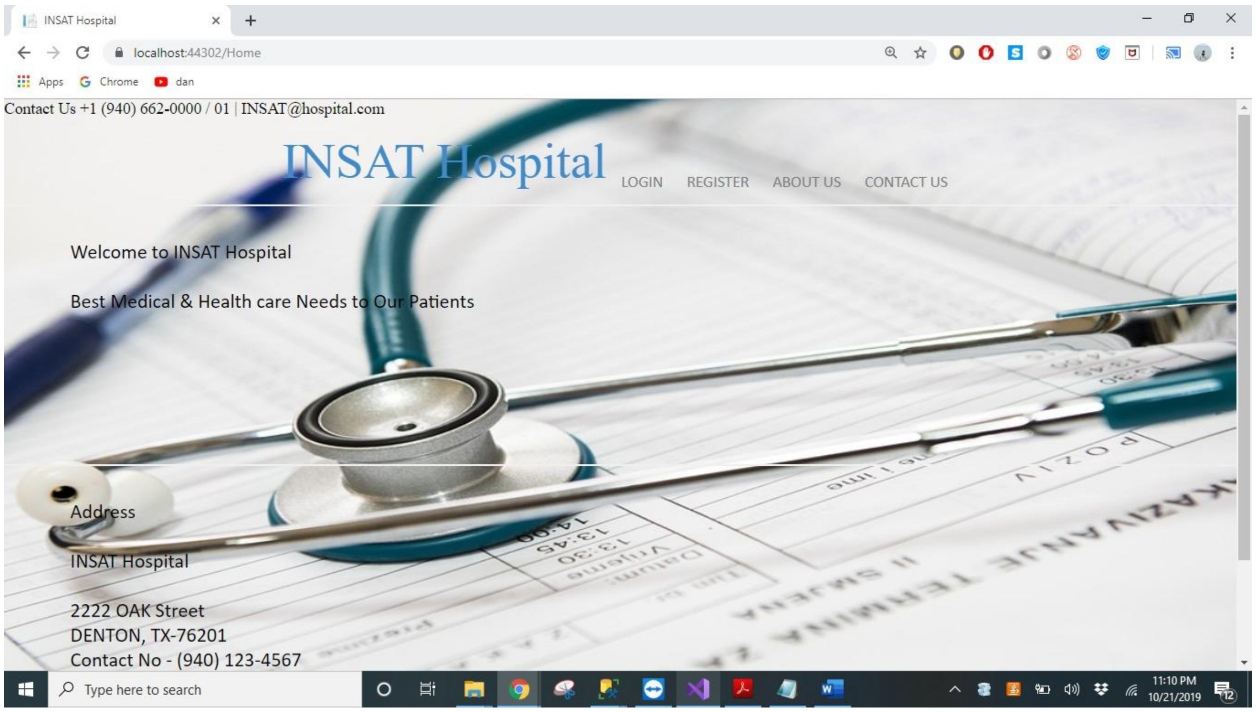
Hospital Management System is a web application which provides a platform for patients to check with the doctor regarding their health-related issues.

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", "About Us" and "View our Doctors" 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".
- The users can go to "View our Doctors" option and get the doctors available for different specialties. They can also get the reviews of the doctors.



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

INSAT Hospital

Welcome to INSAT Hospital

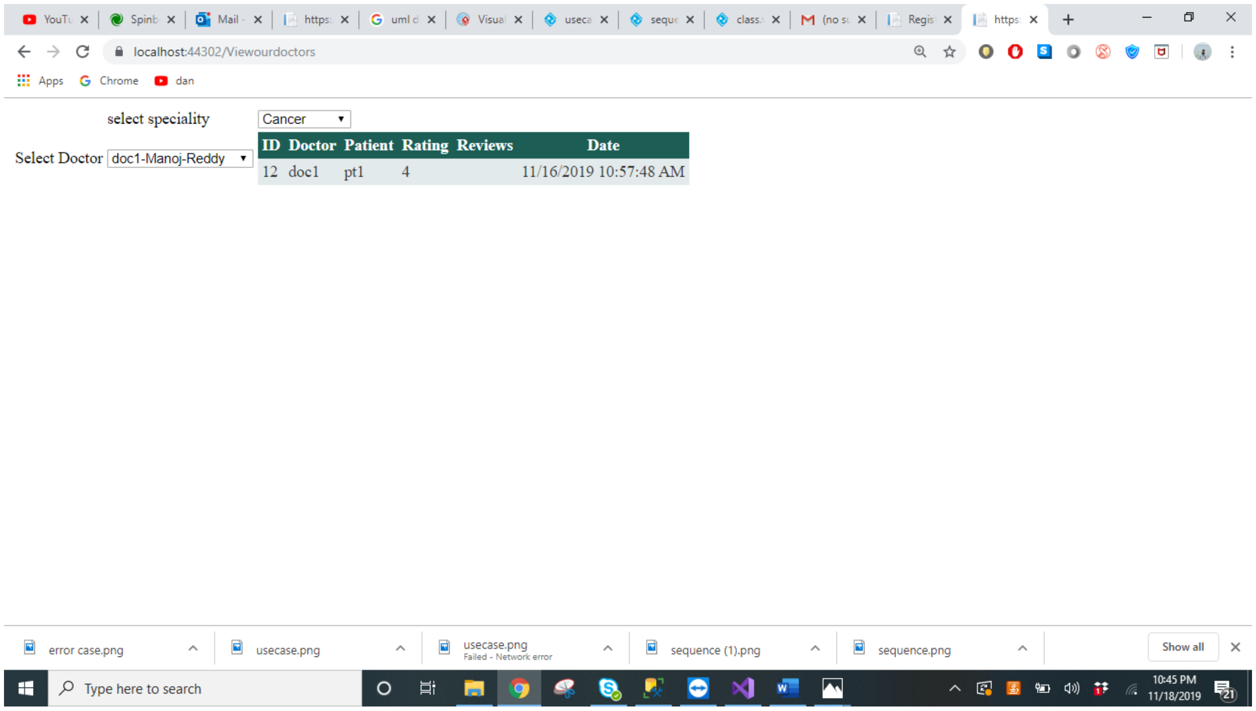
Best Medical & Health care Needs to Our Patients

Address

INSAT Hospital

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

- **View our Doctors page:**



select speciality: Cancer

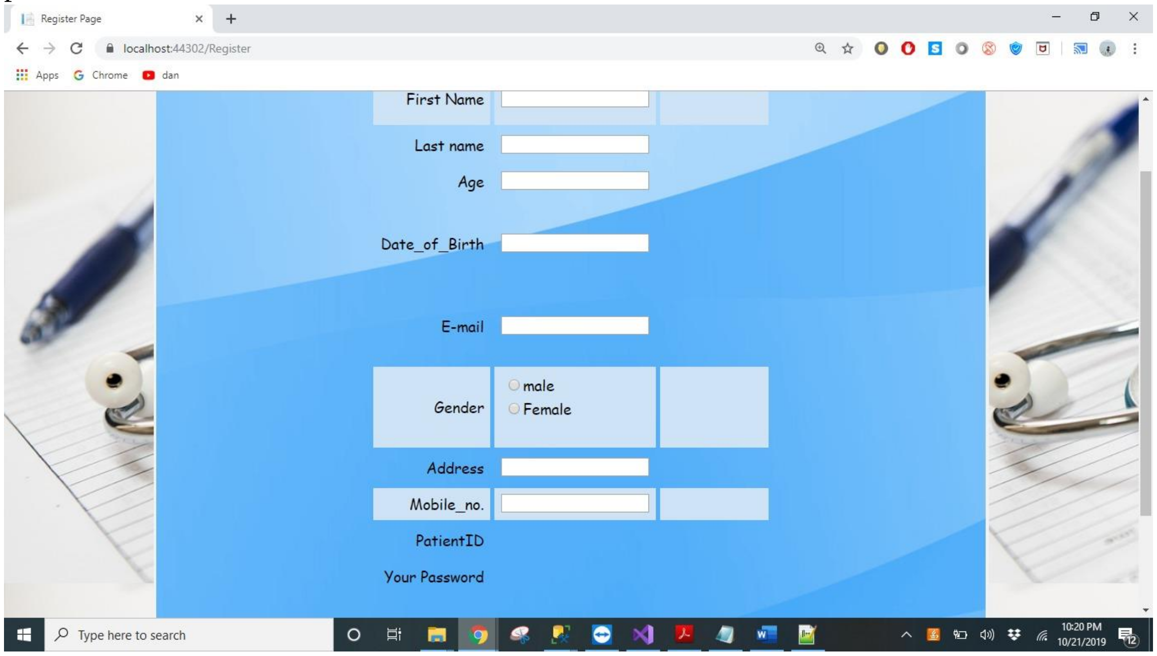
Select Doctor: doc1-Manoj-Reddy

ID	Doctor	Patient	Rating	Reviews	Date
12	doc1	pt1	4		11/16/2019 10:57:48 AM

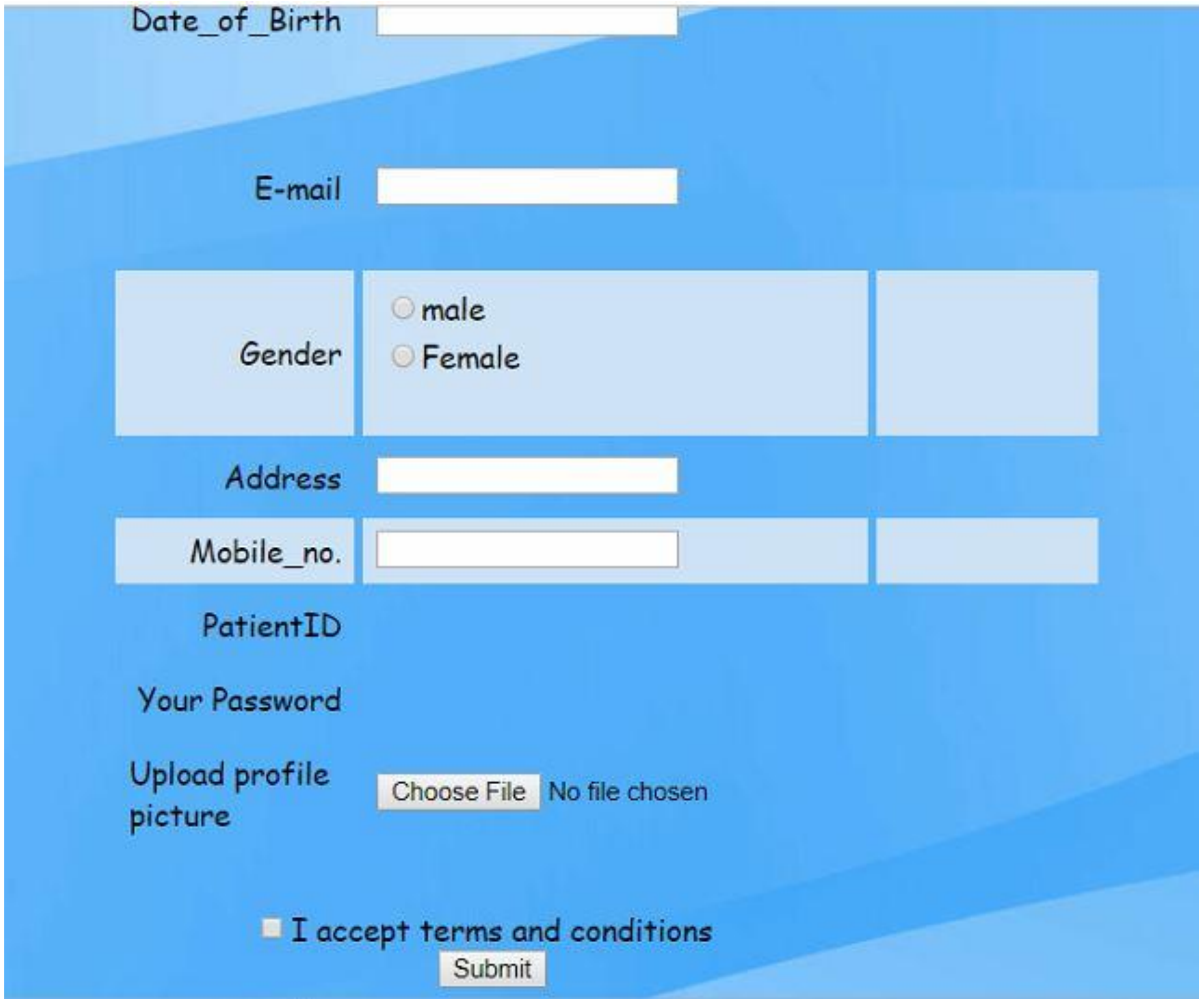
5.3. Patient with no account:

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.



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



Date_of_Birth	
E-mail	
Gender	○ male ○ Female
Address	
Mobile_no.	
PatientID	
Your Password	
Upload profile picture	Choose File No file chosen

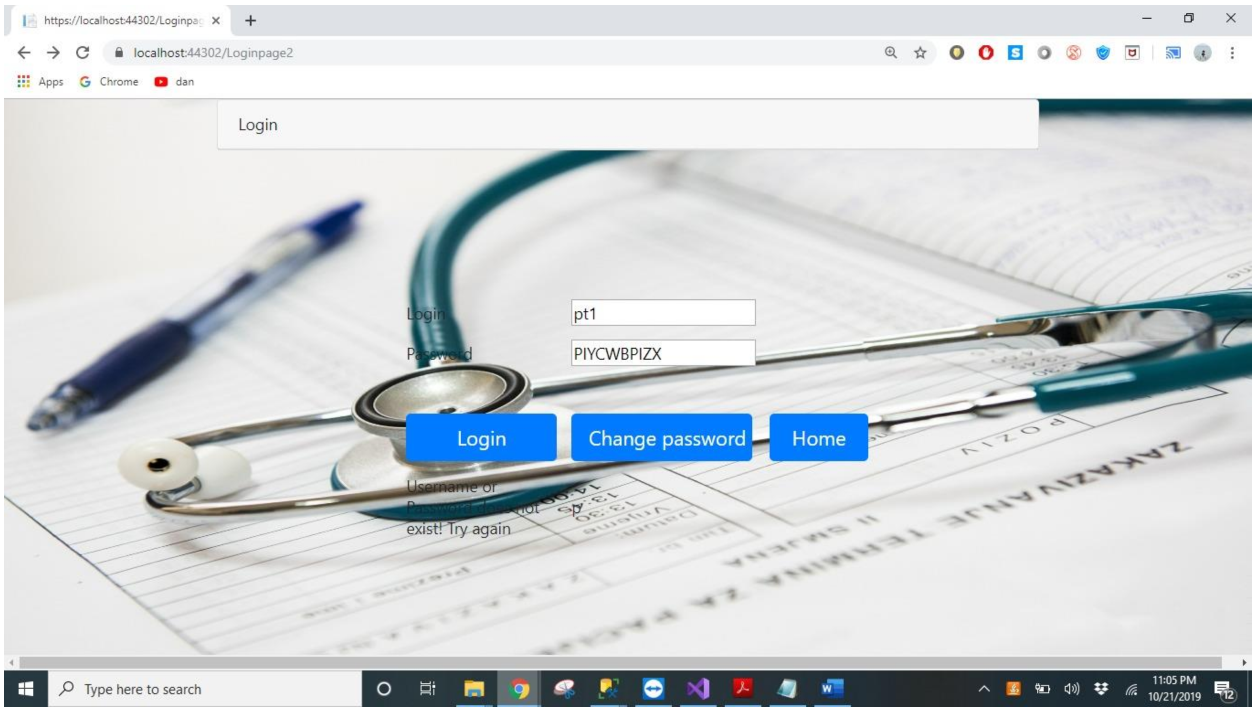
☐ I accept terms and conditions

Submit

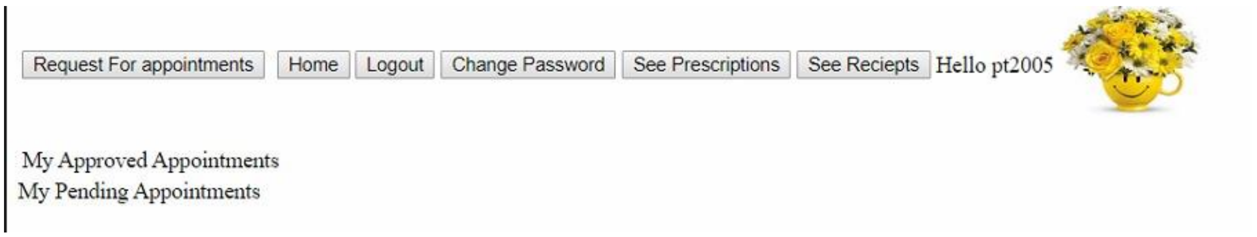
- The user can change upload his profile picture by using the choose file option.

5.4. Patient with account:

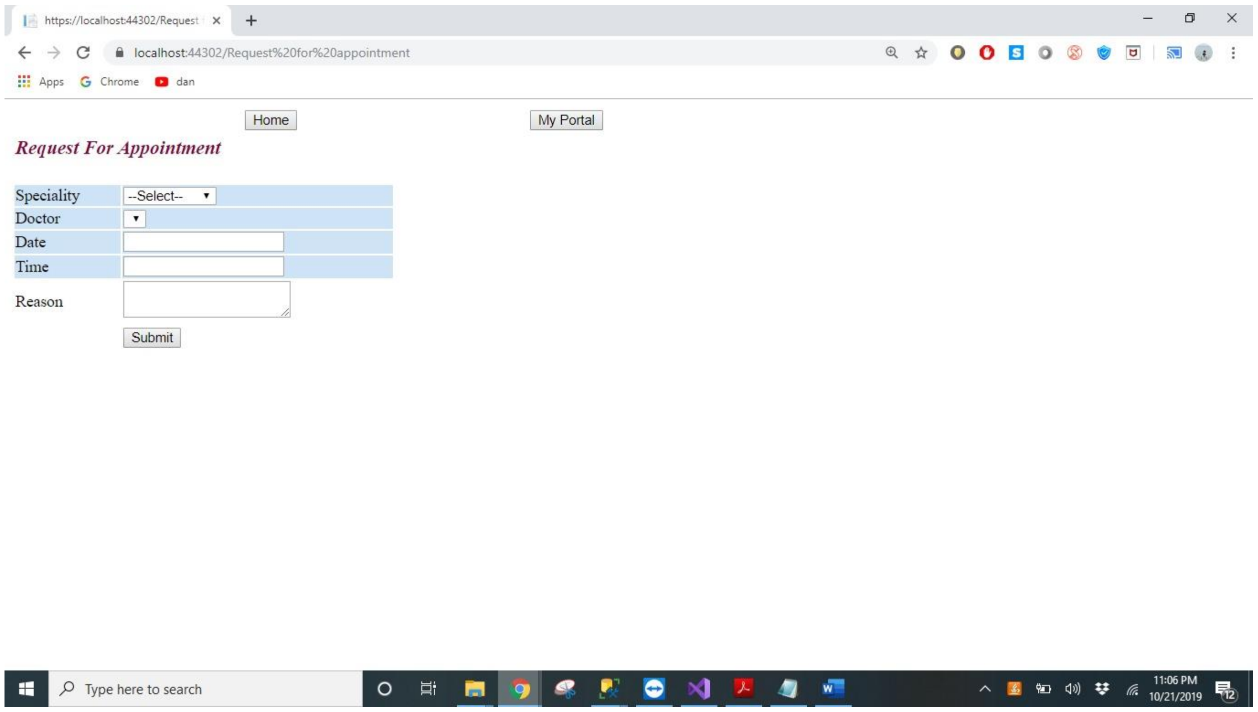
- The user with the account can login into his/her account.


https://localhost:44302/Loginpage2
localhost:44302/Loginpage2
Apps
Chrome
dan
Login
Login
pt1
Password
PIYCWBPIZX
Login
Change password
Home
Username or Password does not exist! Try again
Type here to search
11:05 PM
10/21/2019

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



- 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.

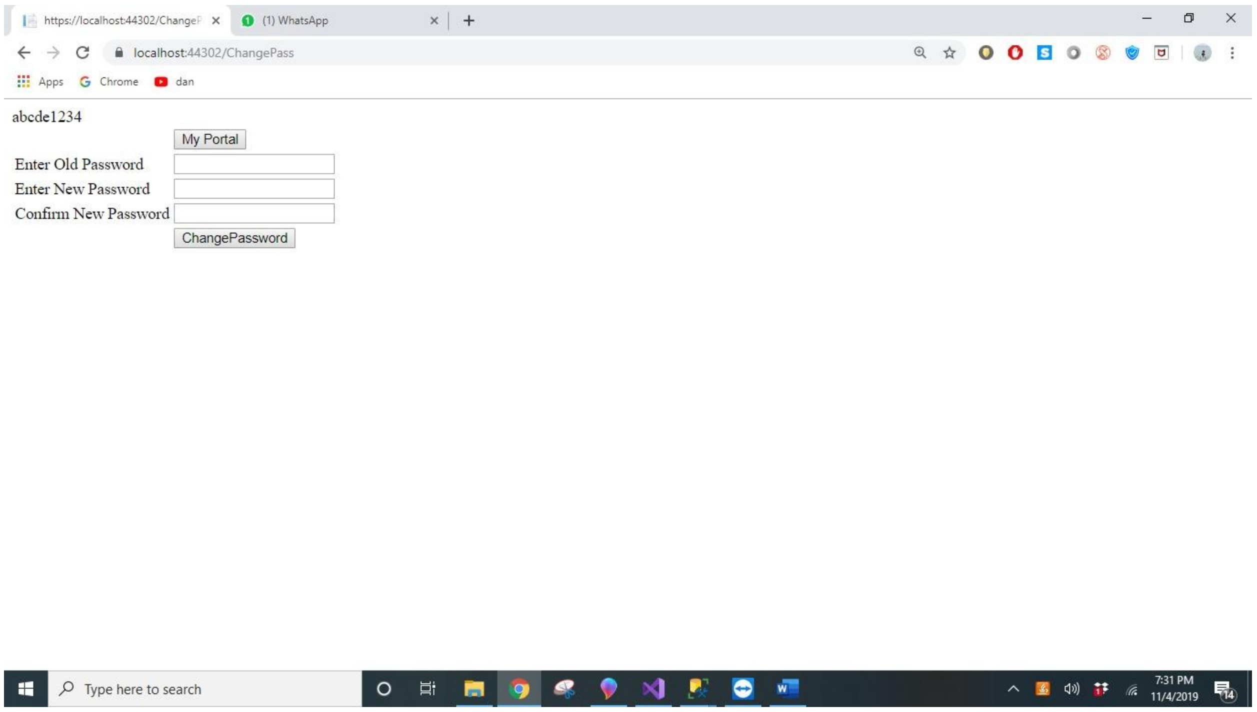

https://localhost:44302/Request
localhost:44302/Request%20for%20appointment
Apps
Chrome
dan
Home
My Portal
Type here to search
11:06 PM
10/21/2019

Request For Appointment

Speciality	--Select--
Doctor	
Date	
Time	
Reason	

Submit

- The user who has the account can change their account password at any time if they need to change the password.
- The new password that the user wants to set should meet the standard requirements.
- These requirements include minimum of 8 character in which one letter should be Uppercase, one character should be digit, one should be a special symbol and the remaining characters can be anything.



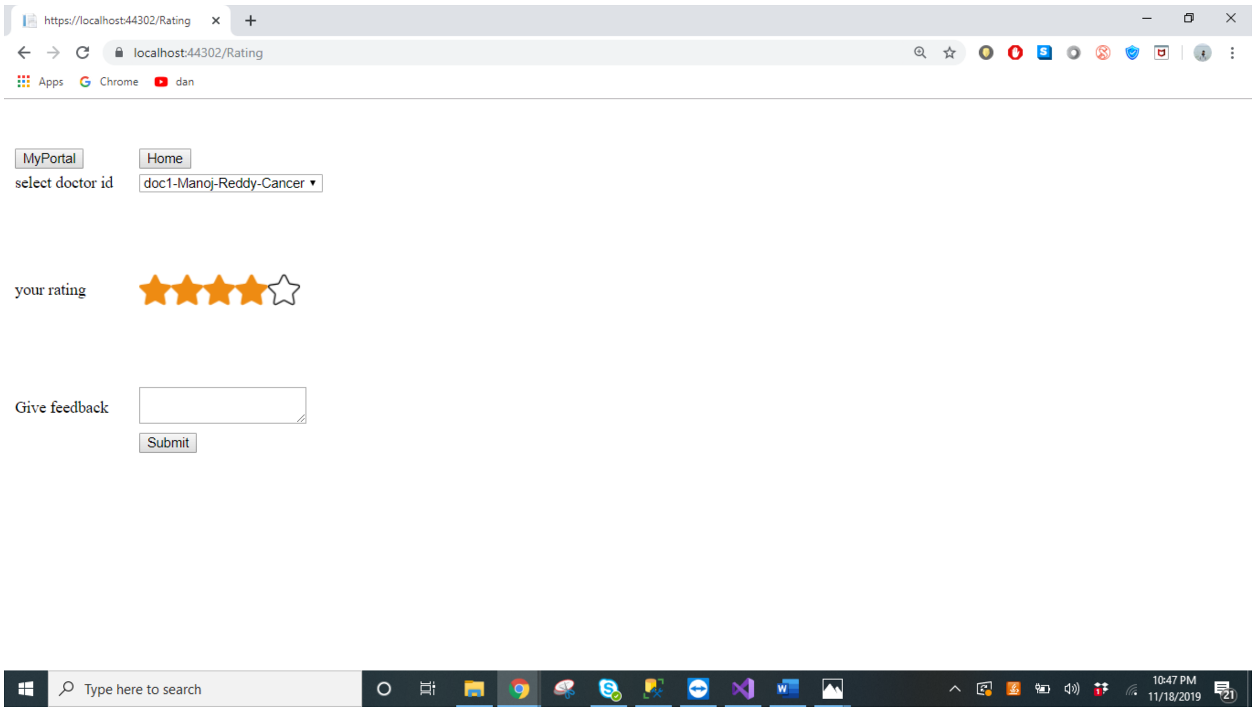
abcde1234

My Portal

Enter Old Password	
Enter New Password	
Confirm New Password	

ChangePassword

- The patients can give feedback and rating about the doctor who have treated the patient.



select doctor id: doc1-Manoj-Reddy-Cancer

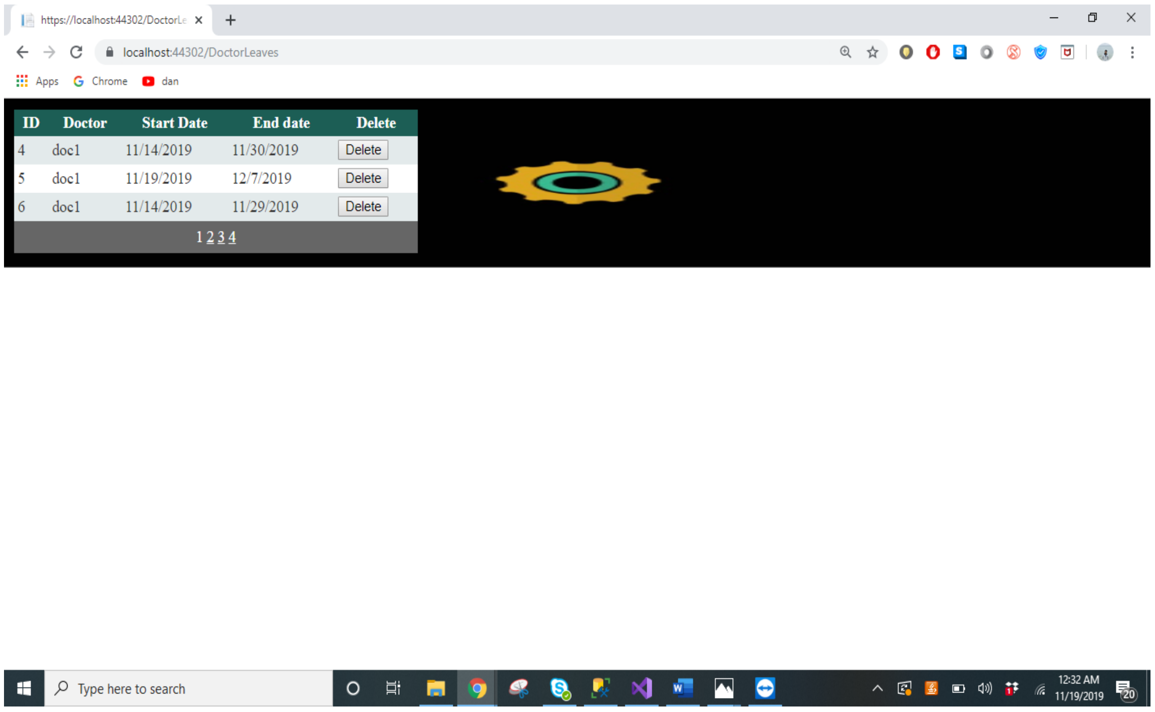
your rating: ★★★★☆

Give feedback

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.

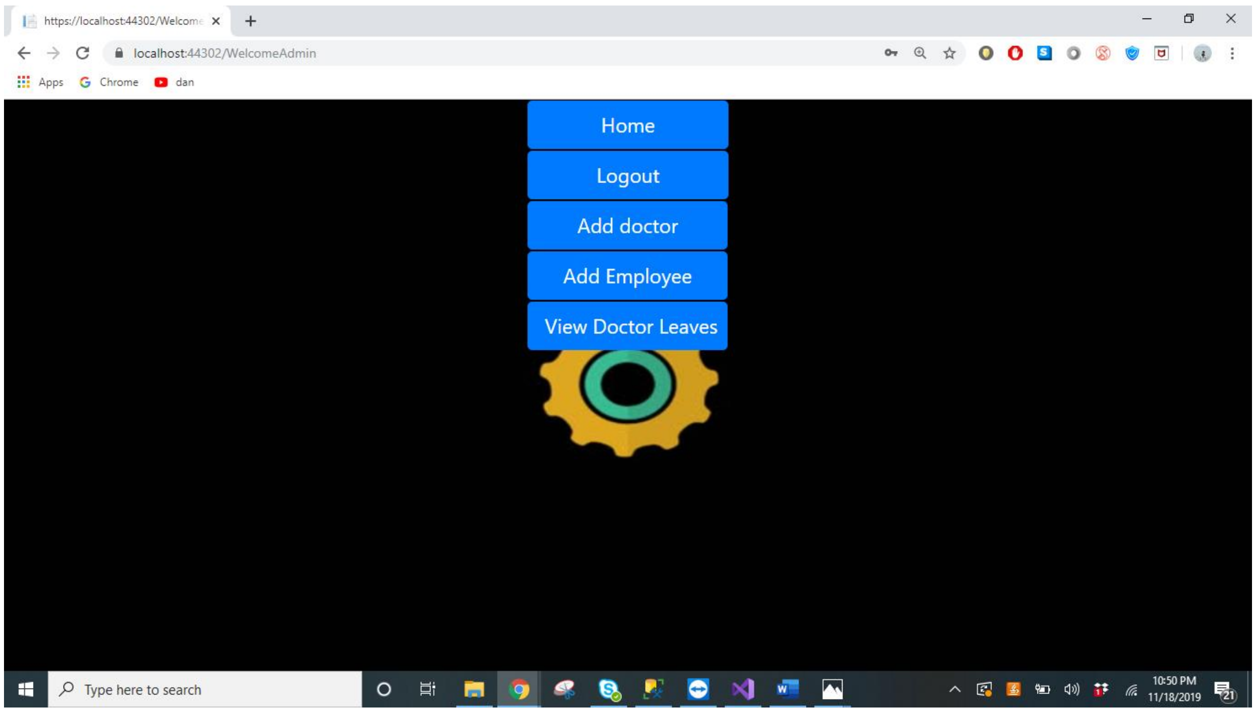
- The admin will have all the privileges.
- The admin generates the UserID and passwords for the doctors and Front-desk User.
- The admin can view the leaves of the doctors and can delete the past leaves of the doctor.



ID	Doctor	Start Date	End date	Delete
4	doc1	11/14/2019	11/30/2019	Delete
5	doc1	11/19/2019	12/7/2019	Delete
6	doc1	11/14/2019	11/29/2019	Delete

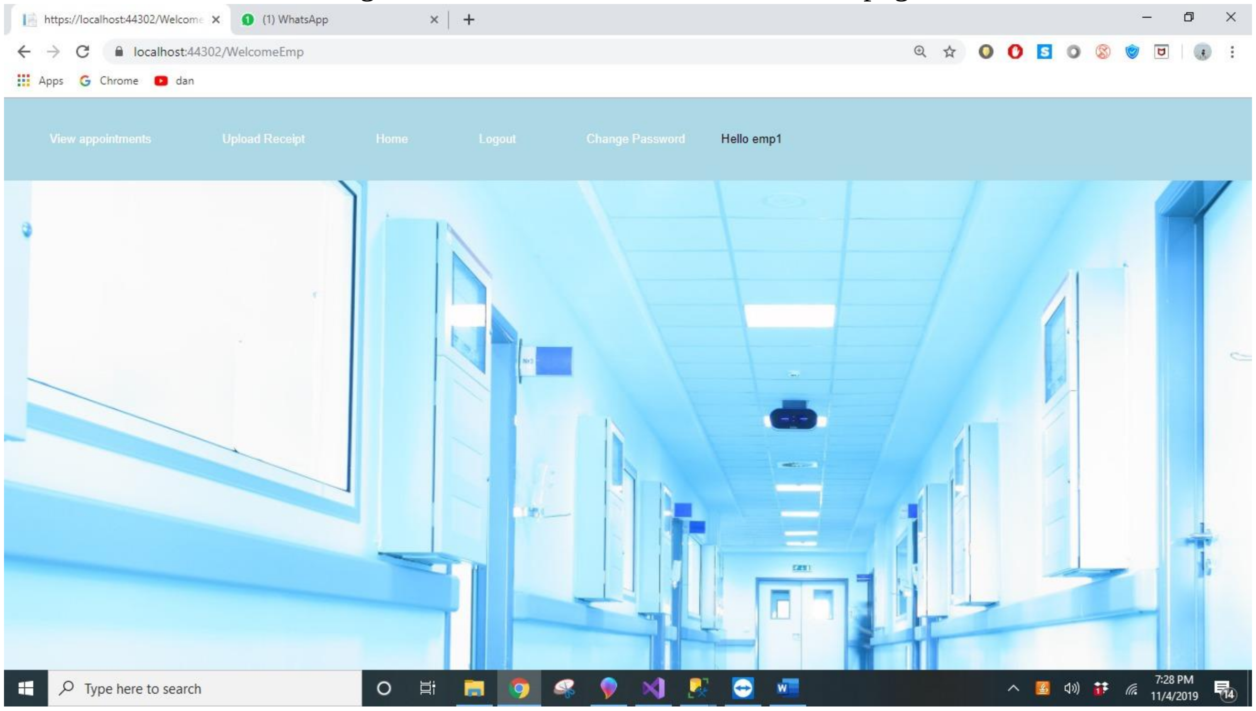
1 2 3 4

- The admin can upload the profile picture for the doctor and the employee while registration.

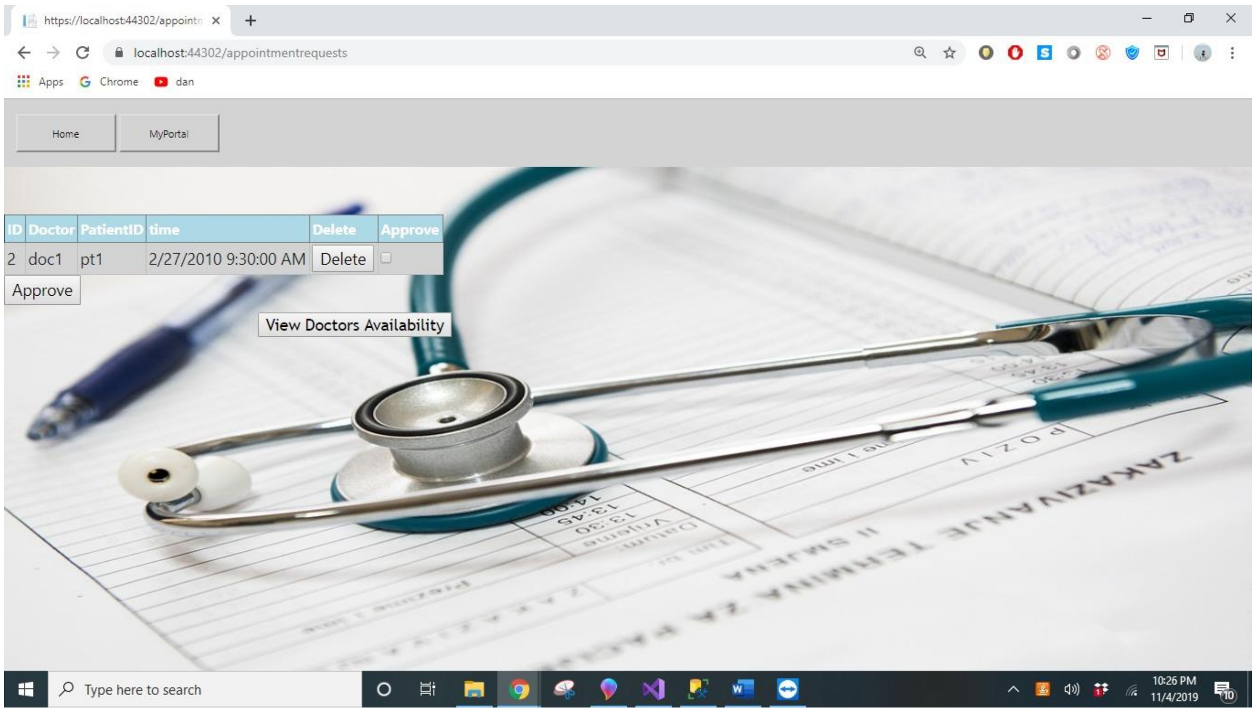


5.6. Front-Desk User

- 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.
- The front desk user can also view the doctor's availability so that based on that availability the front desk user will approve/cancel the requests of the appointments.

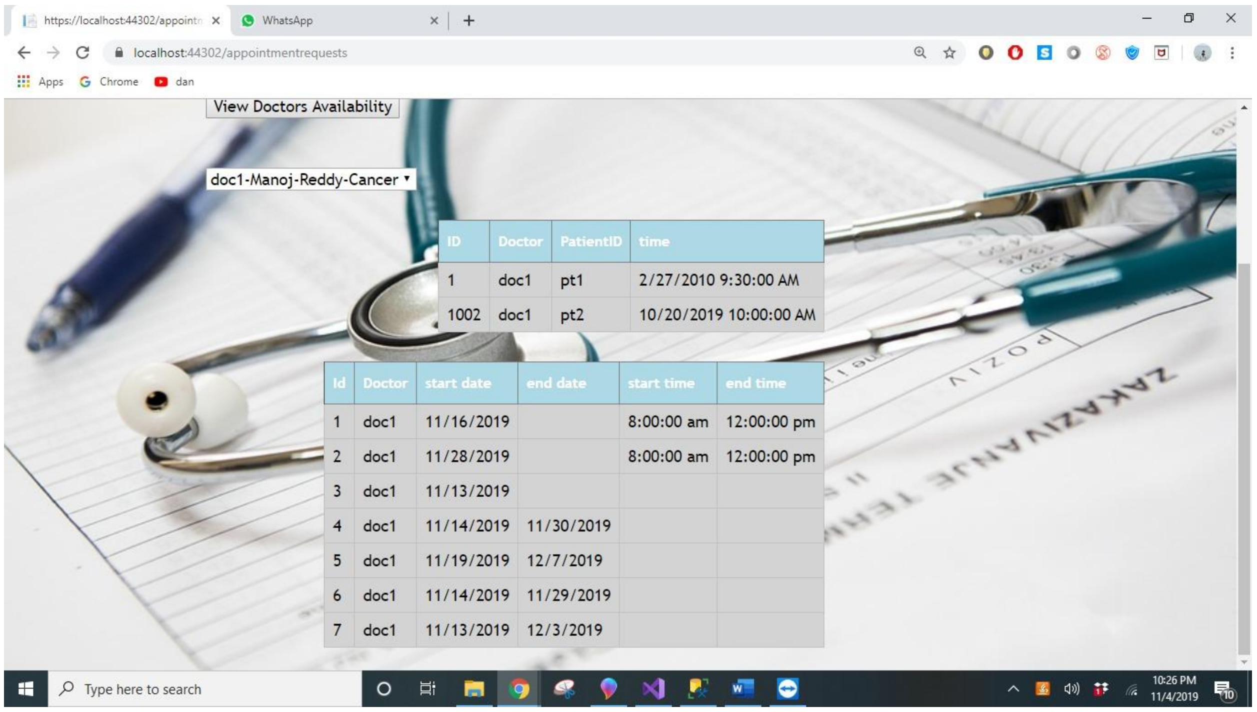

https://localhost:44302/appointm
localhost:44302/appointmentrequests
Apps
Chrome
dan
Home
MyPortal
Type here to search
10:26 PM
11/4/2019

ID	Doctor	PatientID	time	Delete	Approve
2	doc1	pt1	2/27/2010 9:30:00 AM	Delete	☐

Approve

View Doctors Availability

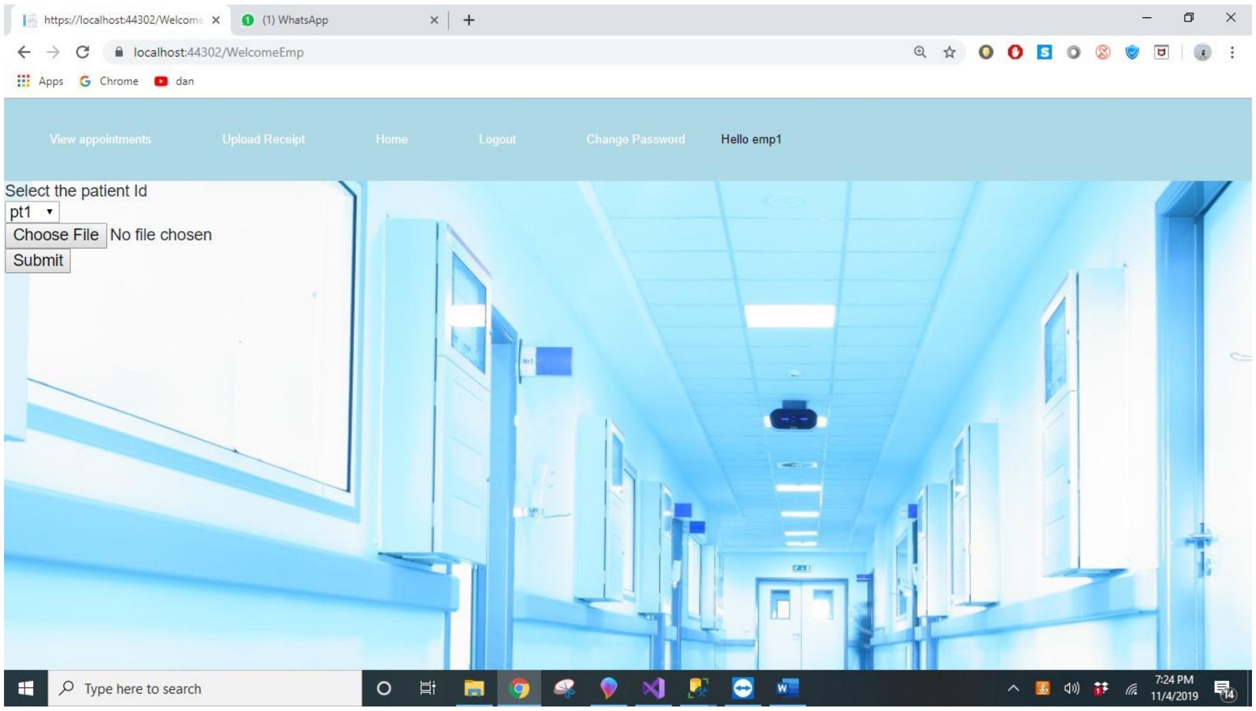
- If we click on the View Doctors Availability, then the availability page looks like as follows:


https://localhost:44302/appointm
WhatsApp
localhost:44302/appointmentrequests
Apps
Chrome
dan
View Doctors Availability
doc1-Manoj-Reddy-Cancer
Type here to search
10:26 PM
11/4/2019

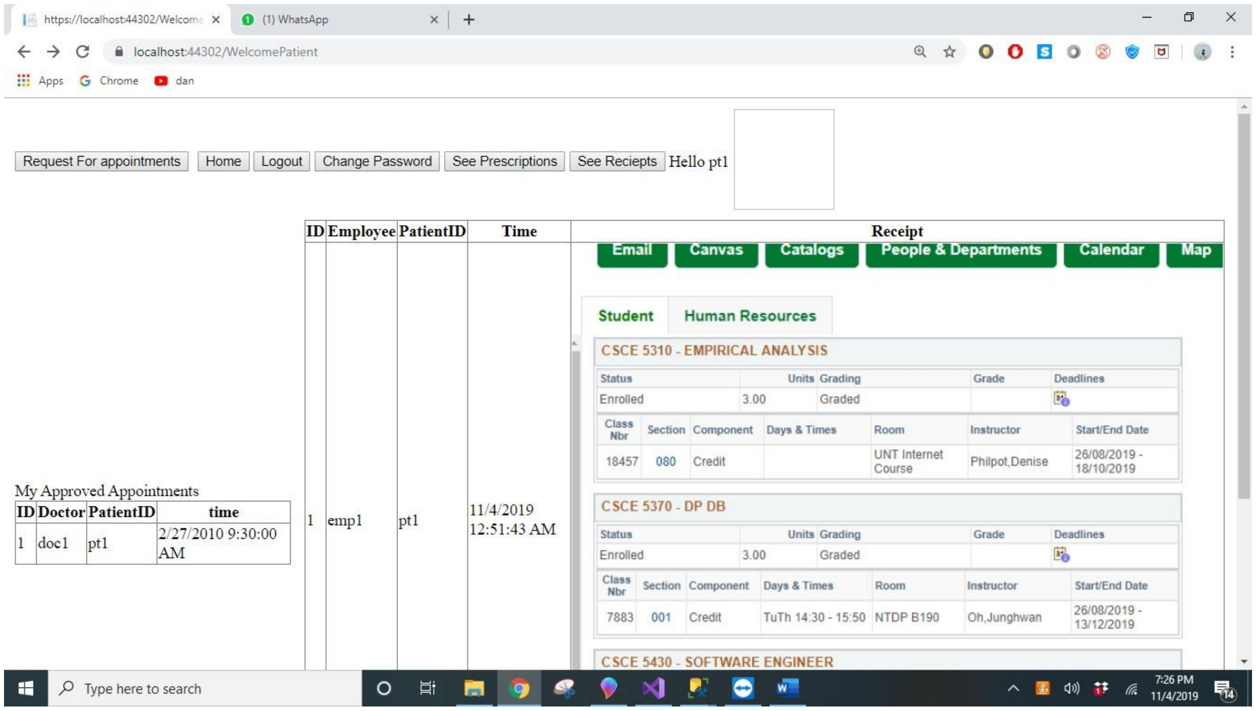
ID	Doctor	PatientID	time
1	doc1	pt1	2/27/2010 9:30:00 AM
1002	doc1	pt2	10/20/2019 10:00:00 AM

Id	Doctor	start date	end date	start time	end time
1	doc1	11/16/2019		8:00:00 am	12:00:00 pm
2	doc1	11/28/2019		8:00:00 am	12:00:00 pm
3	doc1	11/13/2019			
4	doc1	11/14/2019	11/30/2019		
5	doc1	11/19/2019	12/7/2019		
6	doc1	11/14/2019	11/29/2019		
7	doc1	11/13/2019	12/3/2019		

- The Front-desk user can add the patients into the systems when they walk-in into the hospital.
- The front-desk user selects the patient to whom he/she needs to send.



- After selecting the patient to whom the receipt should be sent, the front desk user uploads the receipt to the patients account.
- The patients can view the receipt in their account.



My Approved Appointments

ID	Doctor	PatientID	time
1	doc1	pt1	2/27/2010 9:30:00 AM

ID	Employee	PatientID	Time	Receipt
1	emp1	pt1	11/4/2019 12:51:43 AM	

CSCE 5310 - EMPIRICAL ANALYSIS

Status	Units	Grading	Grade	Deadlines
Enrolled	3.00	Graded		

Class Nbr	Section	Component	Days & Times	Room	Instructor	Start/End Date
18457	080	Credit		UNT Internet Course	Philpot,Denise	26/08/2019 - 18/10/2019

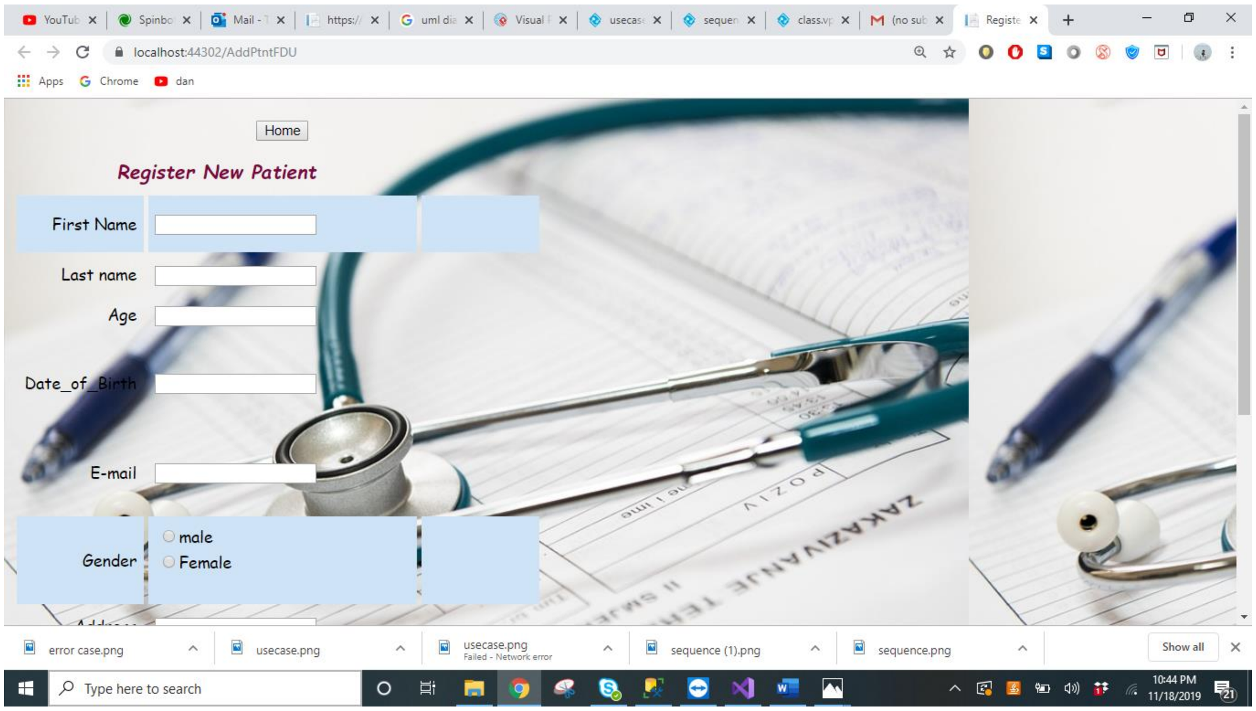
CSCE 5370 - DP DB

Status	Units	Grading	Grade	Deadlines
Enrolled	3.00	Graded		

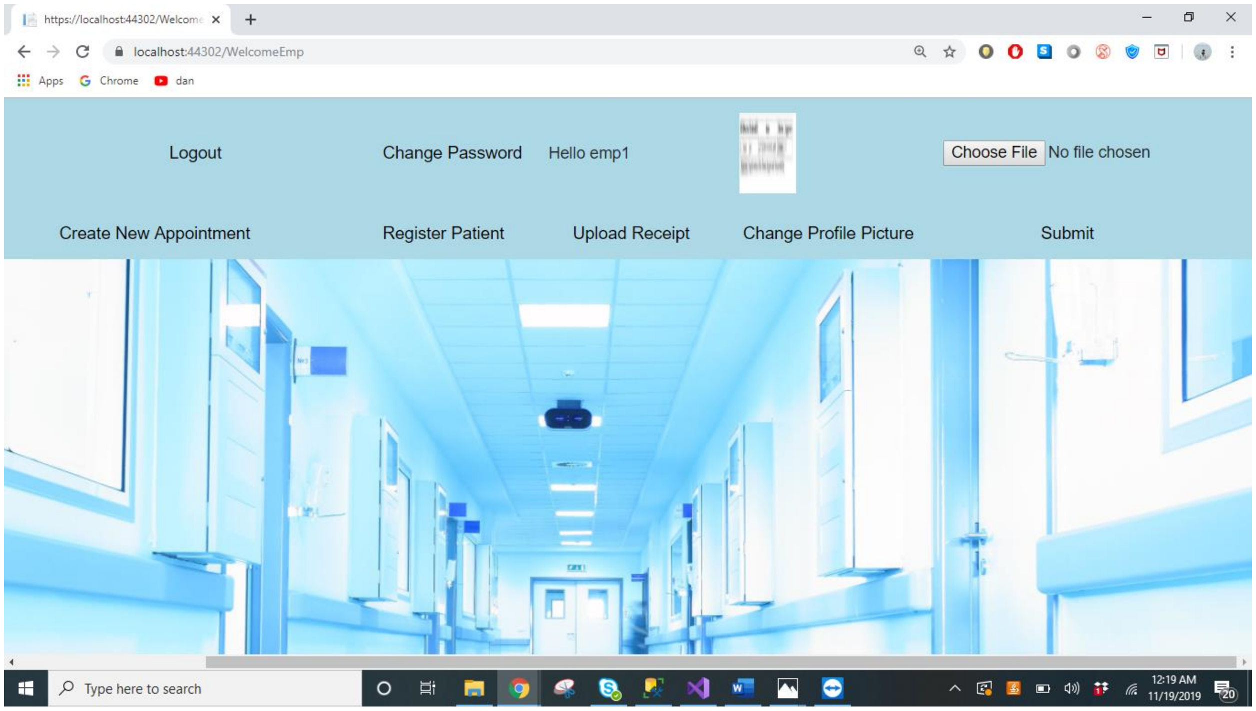
Class Nbr	Section	Component	Days & Times	Room	Instructor	Start/End Date
7883	001	Credit	TuTh 14:30 - 15:50	NTDP B190	Oh,Junghwan	26/08/2019 - 13/12/2019

CSCE 5430 - SOFTWARE ENGINEER

- The front-desk user can create new appointment for the patients.

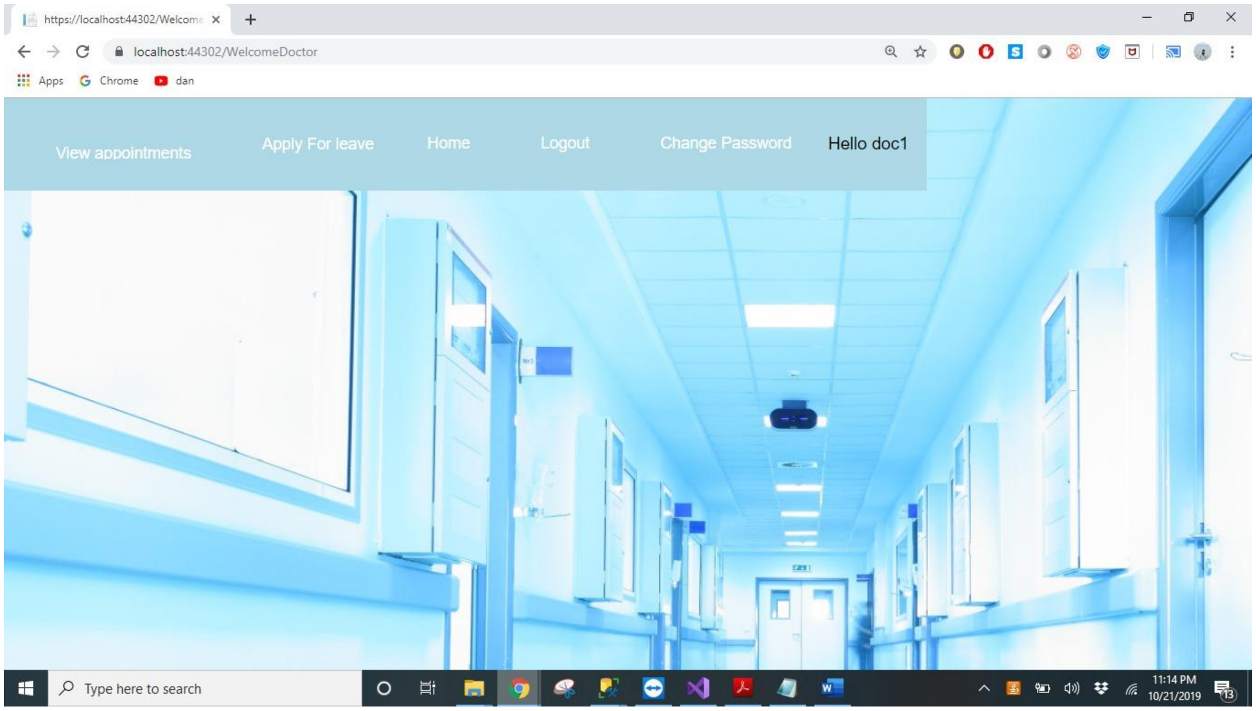


- The front-desk user can upload the profile picture in his/her account.

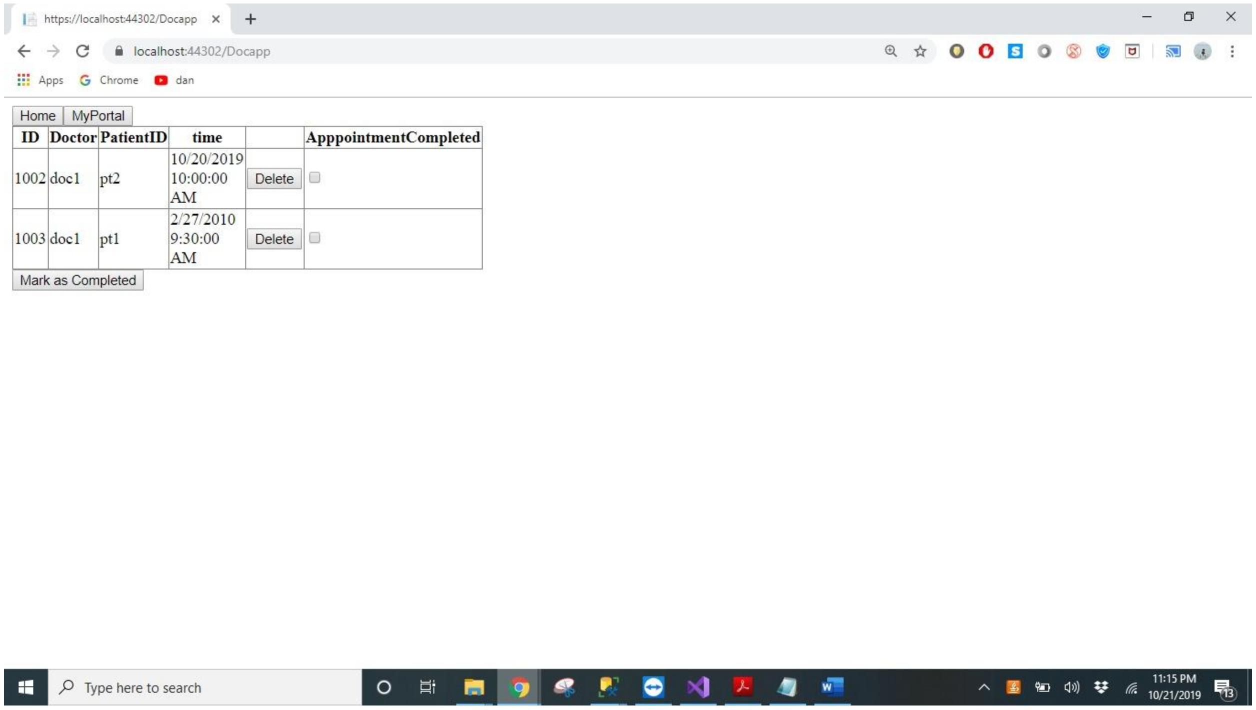


5.7. Doctor

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

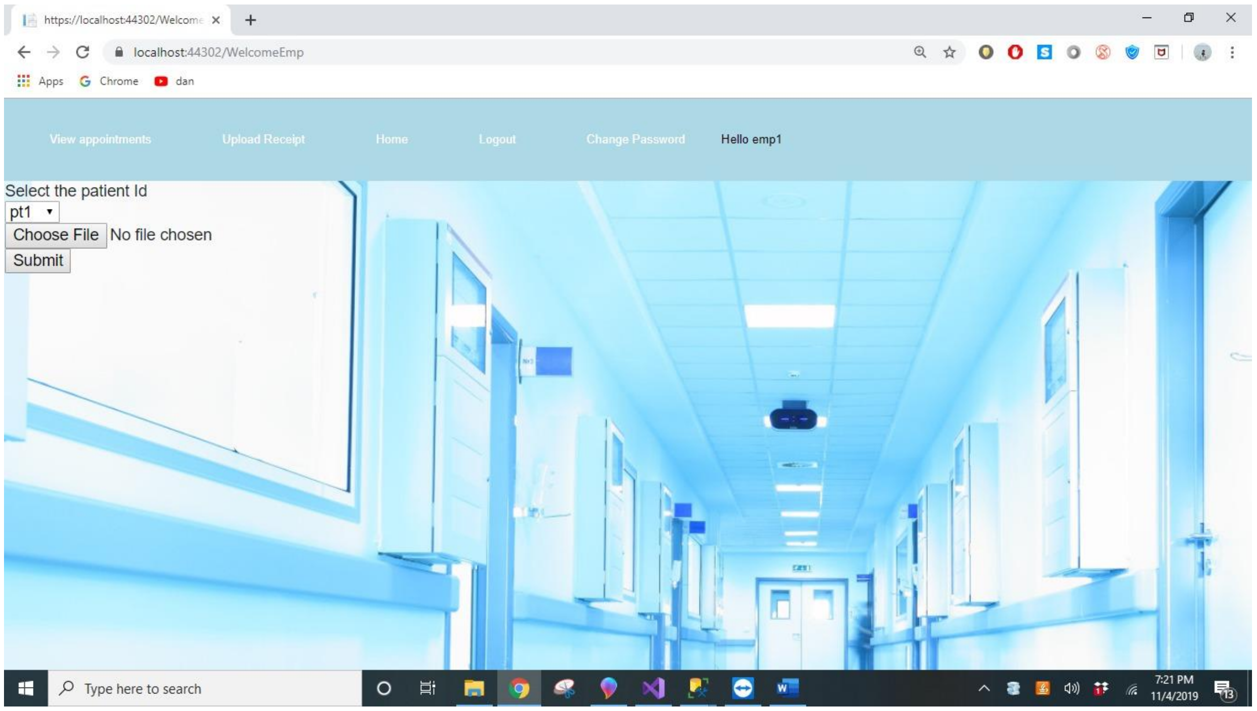


- Doctor checks the patient and updates the appointment status to completed if the treatment is done.



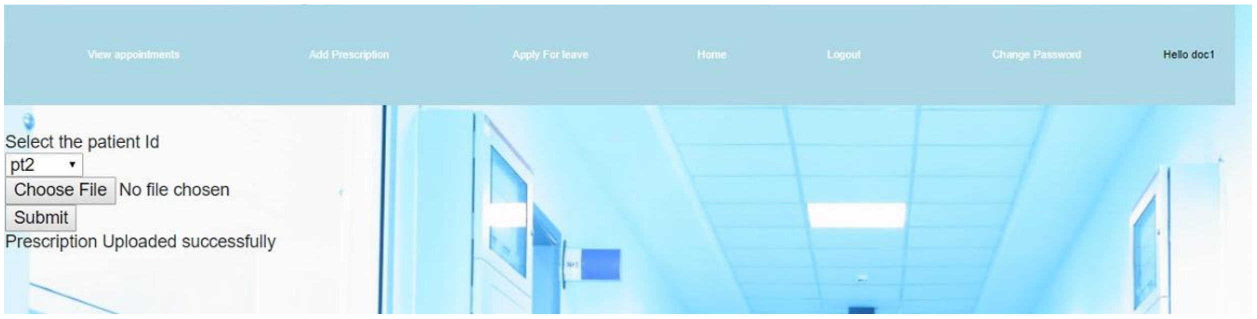
ID	Doctor	PatientID	time		ApppointmentCompleted
1002	doc1	pt2	10/20/2019 10:00:00 AM	Delete	☐
1003	doc1	pt1	2/27/2010 9:30:00 AM	Delete	☐

- The doctor will select the patient to whom he/she needs to upload the prescription.



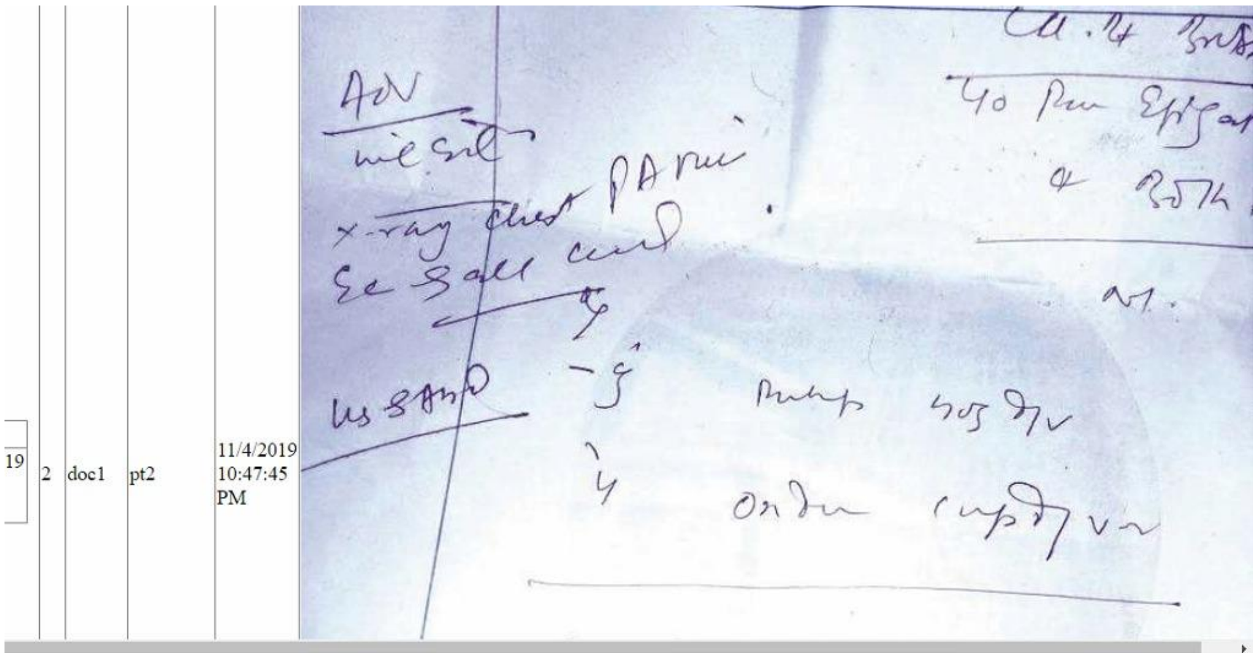
Select the patient Id
pt1
No file chosen

- After selecting the patient to whom the prescription needs to be sent, the doctor will upload the prescription to the respective patients account.
- Once the prescription is uploaded then the notification will be displayed that the prescription is uploaded successfully.



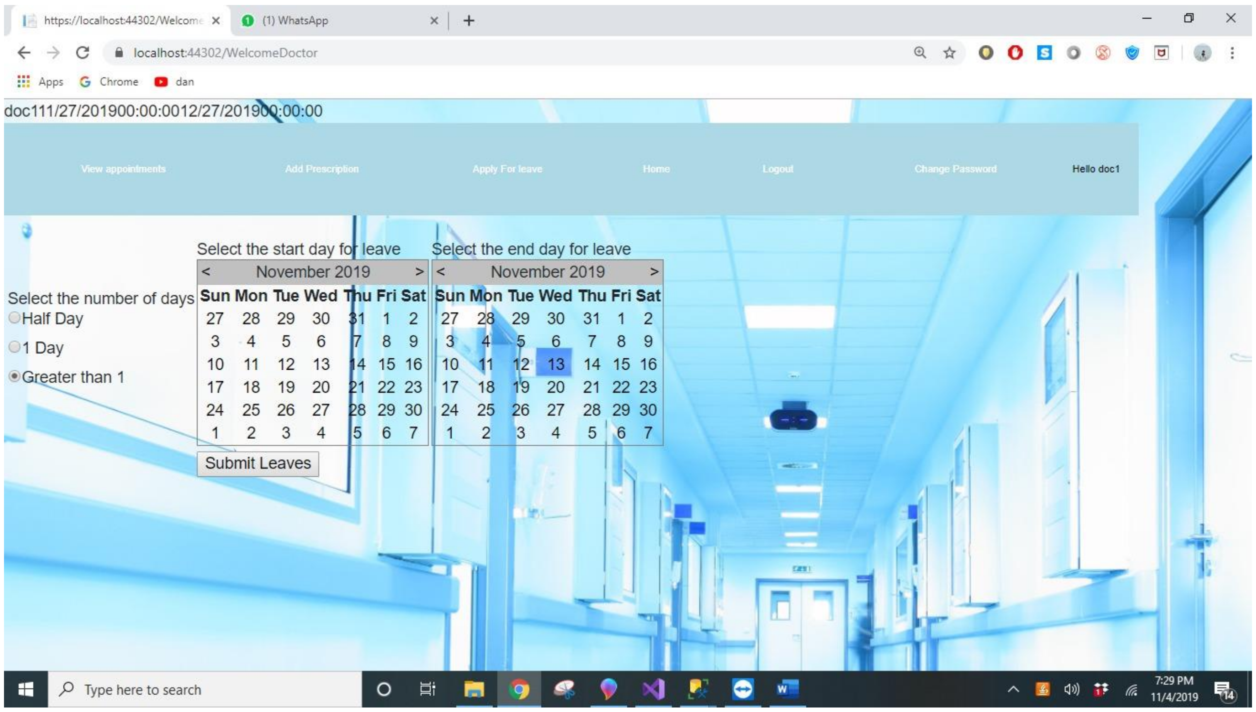
Select the patient Id
pt2
No file chosen
Prescription Uploaded successfully

- **The patients can view the prescription in their account.**



19 | 2 | doc1 | pt2 | 11/4/2019 10:47:45 PM

- Doctor updates his availability to the front-desk user.
- The doctor can select the number of leaves by selecting any one of the following three options.
- The 3 options are:
    - **Half Day**
    - **1 Day**
    - **Greater than 1 Day**
- If the doctor selects greater than 1 Day option, then he needs to select the number of days (leave) from the calendar so that it can be sent directly to the front-desk user.
- The doctor's availability will be later used by the front-desk user to approve/cancel the requests made by the patients.



doc111/27/201900:00:0012/27/201900:00:00

Select the number of days
Half Day
1 Day
[✓ Greater than 1]

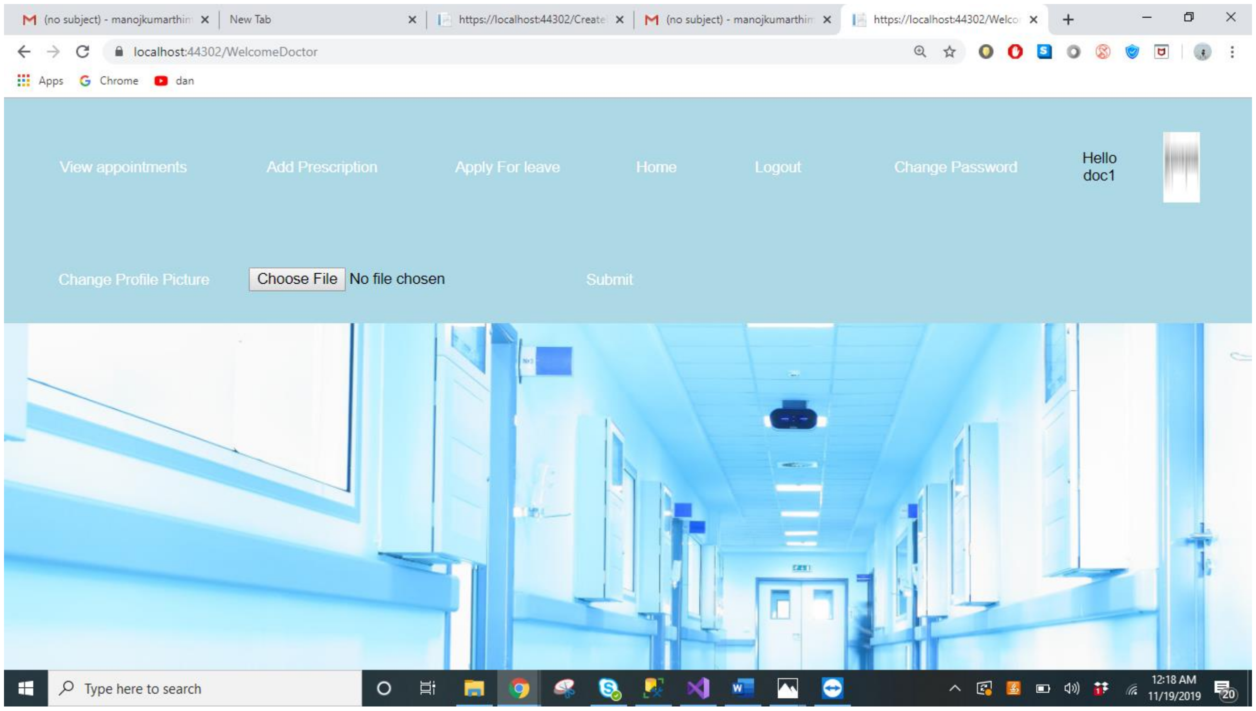
Select the start day for leave

<			November 2019			>
Sun	Mon	Tue	Wed	Thu	Fri	Sat
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

Select the end day for leave

<			November 2019			>
Sun	Mon	Tue	Wed	Thu	Fri	Sat
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

- The doctor can upload the profile picture in his/her account.



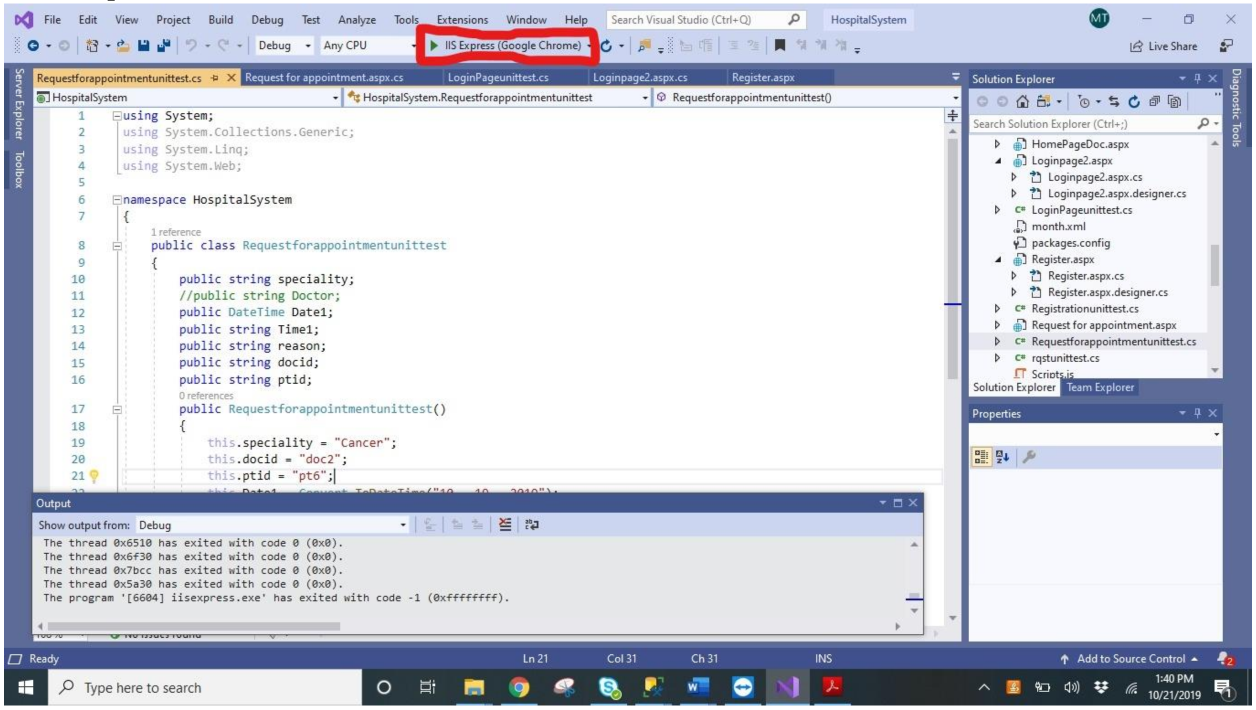
6. To compile/run the program and test cases

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" and password=pass.
- Create the database and required tables by adding the database from database backup uploaded on Github
- Click on the solution file attached on Github and open it on Visual Studios 2019
- Open SQL Server Management Studio and Right click on the database created
- In the Properties, click on ConnectionString.
- Copy the connection String.
- In the folder App_Code in Visual Studio Project, open Class1.cs
- In the Class1.cs change the connection string as per the values of connection string in SQL Server Management Studio.
- Save the changes made to the project.
- Compile the web application by right click on the project in solution explorer and clicking on the option build solution.



```
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";
```

```
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:

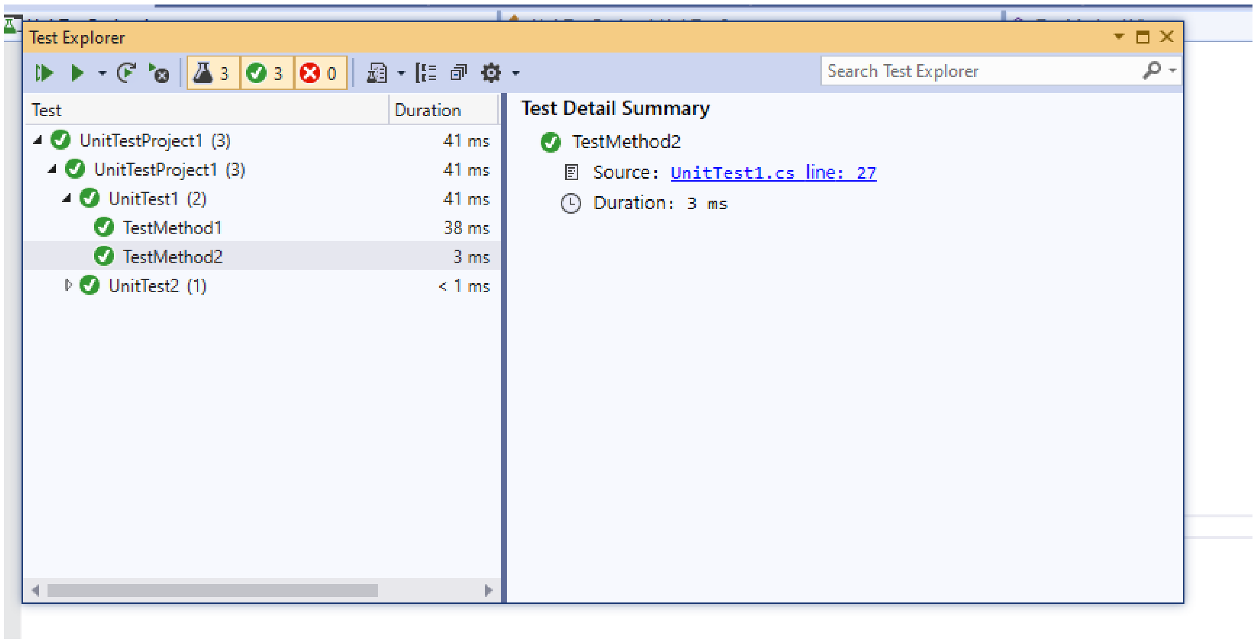
Admin:
ID: admin1234
Password: passwordqwerty

FrontDeskUser:
ID: emp1
Password: MQAVHFSLLS

Patient:
Id: pt1
Password: **Manoj@123**

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 of Rating and feedback by patient we create a class RatingUnittest.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 Explorer

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. Features Successfully Implemented/Future scope and Peer Feedback

7.1. Successfully Implemented Features

Mainly we have 4 modules namely Admin, Front-desk User, Doctor and Patient Modules. We have successfully implemented all the 4 modules. In each module we implemented multiple features. We have successfully implemented 17 features for our project without any known limitations.

1. Patient Registration (PR)

- The Patient can register an account by filling all the details such as name, address, phone number, e-mail-ID, date of birth etc.

2. Login Page (PR)

- All the users like patients, doctors, employees and administrator can login to the account using their credentials.

3. Adding Doctor (AD)

- The admin will add the doctors and creates an account for the doctors.

4. Adding Employee (AE)

- The admin will add the employees and creates an account for the employees.

5. Request for an Appointment (RA)

- The patient selects the specialty so that he can get the doctors list of that specialty. If the patient selects the doctor, then the patient can request for an appointment.

6. Approve or Cancel Request for an Appointment (ACRA)

- Once the patient completes requesting for an appointment, the front-desk user can view and approve or cancel the requested appointment.

7. Doctor’s Availability (DA)

- The doctors will upload their availability to the front-desk user. Then the front-desk user can approve or cancel the requested appointment based on the doctor’s availability.

8. Uploading Receipt (UR)

- The front-desk user uploads the patient’s receipt to the patient’s account.

9. Uploading Prescription (UP)

- The doctor uploads the patient’s prescription to the patient’s account.

10. Change Password (CP)

- The users can change their password by clicking on the “change password” option in the login page.

11. Appointment Booking by Front-Desk User (ABF)

- If the patient is not able to request for an appointment or has any issues in requesting appointment, then the front-desk user helps the patient with requesting for an appointment.

12. List of Doctors (LD)

- The users can see the list of doctors if they click on “view our doctors” in the home page.

13. Uploading Profile Picture (UPP)

- The users can upload their profile picture in their account, or they can also upload their picture wile registration.

14. Rating and Feedback about Doctors (RFD)

- Once the patient’s appointment is completed the patient can rate their doctors in a scale of 1 to 5 and can give their feedback.

15. About-Us Page (AU)

- The users can visit the About-Us page by clicking on the “About-Us” page link in the home page. The About-Us page contains the hospital’s address, email-ID and contact information.

16. Contact-Us Page (CU)

- The users can visit the Contact-Us page by clicking on the “Contact-Us” page link in the home page. The Contact-Us page contains the hospital’s trustees, history etc.

17. Deleting the Doctor’s leaves by Admin (DDA)

- The admin can delete the leaves of the doctor once the leaves end date is completed.

7.2. Feedback received during peer sessions and the actions taken based on the feedback

We received a positive feedback from our peer team. We got 4 suggestions in total from our peer team. The 2 feedbacks that we got in the requirements elicitation workshop is to do the validations for the data fields in the registration page and the other feedback is to enhance the user-interface. In the next meeting we got 2 suggestions. The 2 feedbacks that we got in this meeting is to add the image functionality into the patient’s account and the other feedback is to limit the number of accounts to 1 for each user.

The actions taken based on the feedback are:

- We used bootstrapping for User Interface enhancements.
- 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 improved the image functionality by adding profile picture, receipt and prescription uploading.
- We made sure that every user can create only one account.

7.3. Future Scope

- Payment gateway integration: This allows the customers to pay online.
- We can develop a mobile application on the hospital management system.
- We can also provide SMS alerts regarding the appointment update.
- E-mail verification code to verify the email address of the patient.

8. Member Contribution Table

Member Name	**Contribution Description**	**Overall Contribution %**
Manoj Kumar Thimapuram	Patient Registration Page	50
Maulshree Verma	Patient Registration Page	50
Yogesh Nimmagadda	Login Page	50
Naga Saiteja Chintakayala	Login Page	50
Naga Saiteja Chintakayala	Welcome Employee (Scheduling Appointment)	25
Maulshree Verma	Welcome Employee (Scheduling Appointment)	75

Manoj Kumar Thimapuram	Welcome Doctor (Scheduling Appointment)	50
Yogesh Nimmagadda	Welcome Doctor (Scheduling Appointment)	50
Maulshree Verma	Welcome Patient (Booking Appointment and Rating)	75
Yogesh Nimmagadda	Welcome Patient (Booking Appointment and Rating)	25
Naga Saiteja Chintakayala	Change Password	25
Maulshree Verma	Change Password	25
Manoj Kumar Thimapuram	Change Password	25
Yogesh Nimmagadda	Change Password	25
Maulshree Verma	Welcome Admin	50
Manoj Kumar Thimapuram	Welcome Admin	50
Maulshree Verma	Upload Profile Picture	50
Naga Saiteja	Upload Profile Picture	50
Maulshree Verma	Upload Receipt	50
Yogesh Nimmagadda	Upload Receipt	50
Maulshree Verma	Upload Prescription	60
Manoj Kumar Thimapuram	Upload Prescription	40
Maulshree Verma	List of Doctors	100
Manoj Kumar Thimapuram	Contact-Us Page	100
Naga Saiteja	About-Us Page	100
Maulshree Verma	Database Connectivity	50
Manoj Kumar Thimapuram	Database Connectivity	50

Documentation:

Yogesh Nimmagadda	User Report	25
Manoj Kumar Thimapuram	User Report	25
Naga Saiteja Chintakayala	User Report	25
Maulshree Verma	User Report	25
Yogesh Nimmagadda	Uml Diagrams	25
Manoj Kumar Thimapuram	Uml Diagrams	25
Naga Saiteja Chintakayala	Uml Diagrams	25
Maulshree Verma	Uml Diagrams	25
Manoj Kumar Thimapuram	TestCases	50
Maulshree Verma	TestCases	50
Naga Saiteja Chintakayala	Other Parts of Documentation	50
Yogesh Nimmagadda	Other Parts of Documentation	50
Maulshree Verma	PPT’s	25
Naga Saiteja Chintakayala	PPT’s	25
Yogesh Nimmagadda	PPT’s	25
Manoj Kumar Thimapuram	PPT’s	25