

Project Phase 2 Report For Hospital Management System

Version 2.0

**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	
1.1 DA: Doctor Availability.....	3
1.2 UR: Uploading Receipt.....	3
1.3 UP: Uploading Prescription	3
1.4 CP: Change Password.....	3
1.5 Modified Requirements.....	3
2 UML Design	
2.1 Class diagram.....	4
2.2 Sequence diagram	5
2.3 Use case diagram	6
2.4 Use case diagram (Error Case).....	7
3 Test Cases	8
4. Who wrote what components/classes of the system	22
5. User Manual	23
6. To compile/run the program and test cases.....	35
7. Feedback received during peer reivew Session	37
7.1 Actions need to taken based on the feedback.....	37

1. Requirements

This web-application mainly focuses on four modules namely Admin, front-desk user, doctor and patient modules. 75% of the project is completed and below given features are the major features implemented in the development phase-2 of this web-application.

1.1. DA: Doctor Availability

- PRIORITY: MEDIUM
- The doctors can upload their availability to the front desk user.
- While uploading the availability of doctors, the doctors can choose any of the following 3 options.
 - **Half-day**
 - **Full-day**
 - **Greater than 1 day**
- If the doctor chooses greater than 1-day option, then the doctor should select the number of days he wanted to take the leave.

1.2. UR: Uploading Receipt

- PRIORITY: MEDIUM
- The front-desk user can upload the receipt of the payment done by patient.
- Only Front-desk user can upload the payment receipt.
- The patient can see the uploaded payment receipt in his account with the help of a grid view.

1.3. UP: Uploading Prescription

- PRIORITY: MEDIUM
- Only doctor have the permissions to upload the prescription.
- Once the doctor finishes the patient's appointment, he can upload the prescription for that patient in the patient's account.
- The patient can see the uploaded prescription in his account with the help of a grid view.
- The prescription also includes the timestamp of the uploaded prescription.

1.4. CP: Change Password

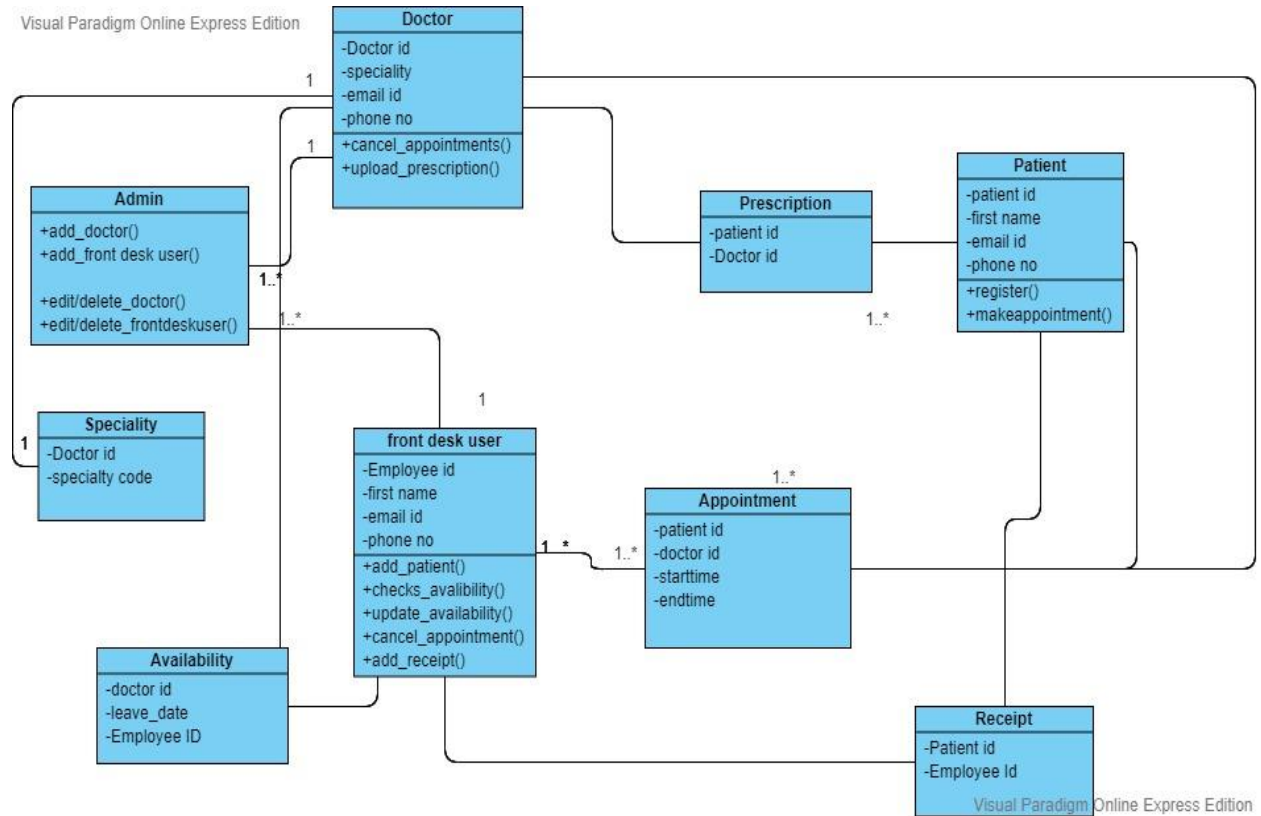
- PRIORITY: MEDIUM
- The users have the option to change the password at any time they want.
- But the password should meet the standard requirements (e.g. Minimum 8 characters (including 1 Uppercase letter and one should be a digit)).

1.5. Updated Requirements

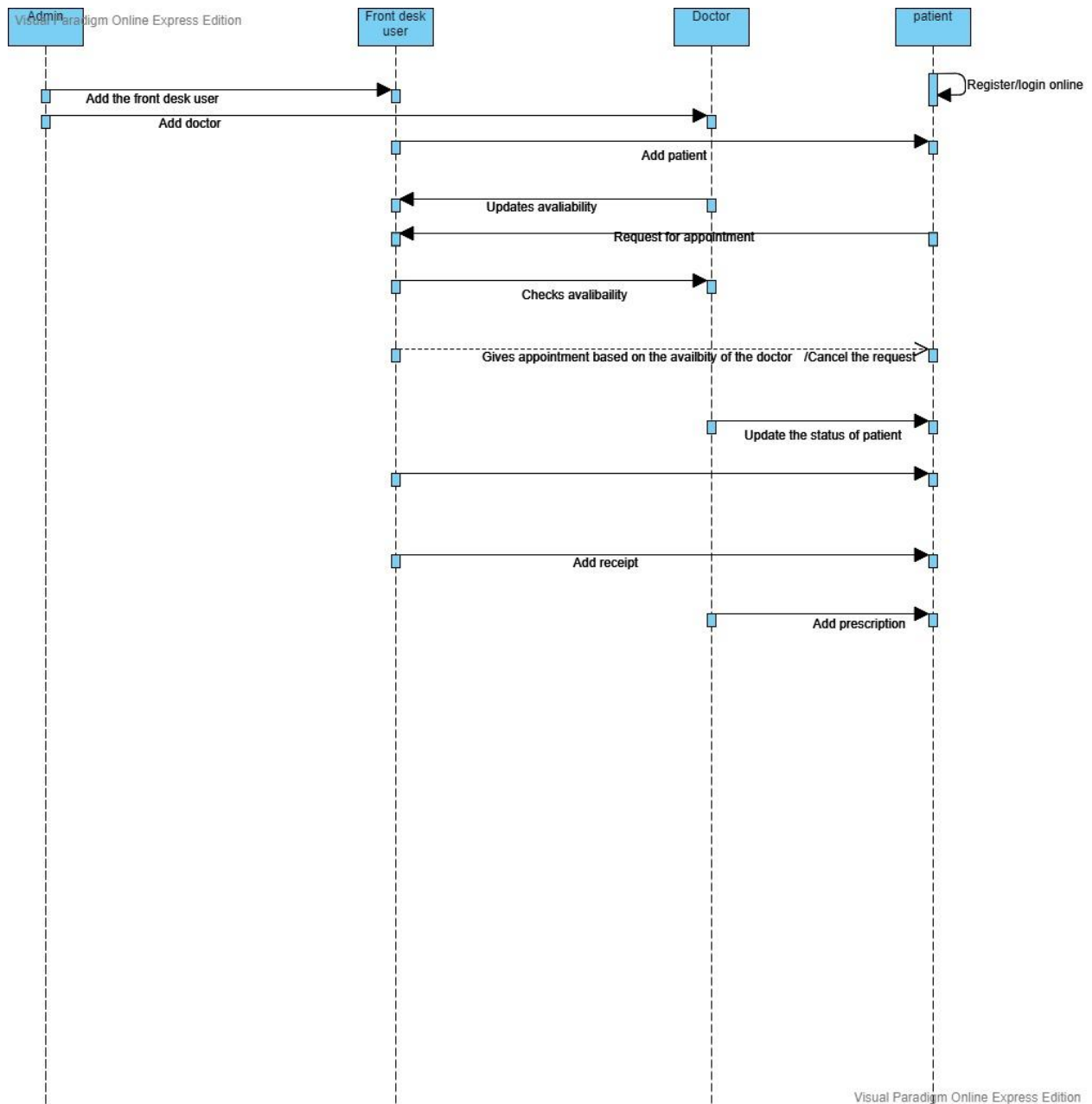
- We made one update to the requirements in the registration page. In the registration page the patient can be able to add his/her picture.

2. UML Design

2.1. Class Diagram



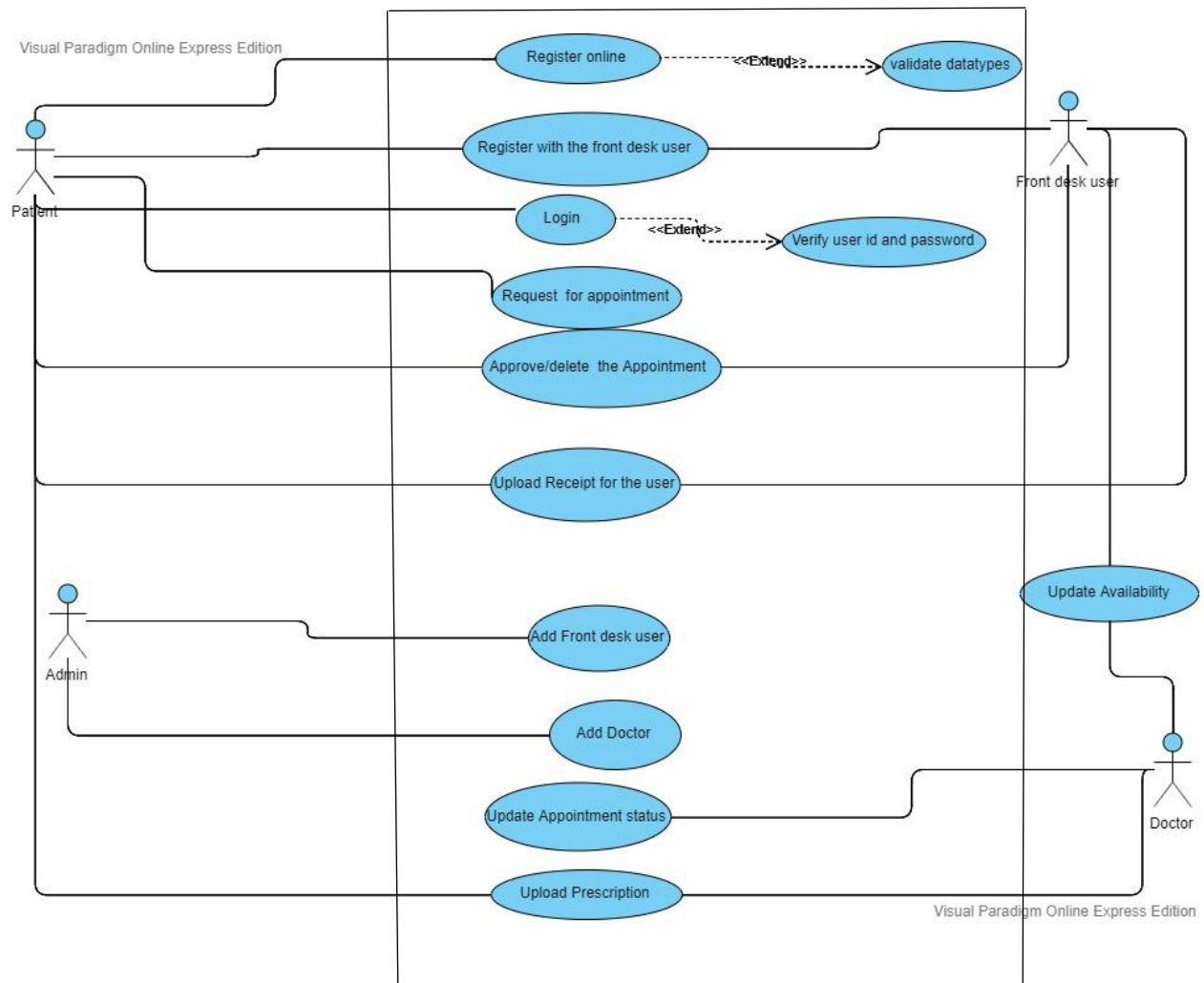
2.2. Sequence Diagram



2.3. Use case Diagram (Normal Case)



2.4. Use case Diagram (Error Case)



3. Test Cases

3.1. Unit test-cases for the deliverable-4 requirements

Test Case 1:

WelcomeDoctorUnittest.cs:

//Description: This class is used to test all the availability of the doctors, date format, uploading the leaves and uploading the prescription.

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

namespace HospitalSystem
{
    public class WelcomeDoctorUnittest
    {
        //Unit test to apply for leave

        public string docid;
        public string startdate;
        public string enddate;
        public string starttime;
        public string endtime;
        public WelcomeDoctorUnittest()
        {
            //Unit test for Upload Availability.
            this.docid = "doc1";
            this.startdate = "11/27/2019";
            this.enddate = "12/27/2019";
            this.starttime = "00:00:00";
            this.endtime = "00:00:00";

            //WelcomeDoctor Unit test for uploading Leaves
            //unit test for testing the format of time
            WelcomeDoctorUnittest obj = new WelcomeDoctorUnittest();
            this.docid = "doc1";
            this.startdate = "11/27/2019";
            this.enddate = "12/27/2019";
            this.starttime = "00:00";
            this.endtime = "00:00";

            //unit test for testing the format of date
            WelcomeDoctorUnittest obj = new WelcomeDoctorUnittest();
```



```

        this.docid = "doc1";
        this.startdate = "2019-11-27";
        this.enddate = "2019-12-27";
        this.starttime = "00:00:00";
        this.endtime = "00:00:00";
    }
}
}

```

Test Methods for testing the doctor's availability functionality.

UnitTest1.cs:

```

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

namespace UnitTestProject1
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void TestMethod1()
        {
            //WelcomeDoctor Unit test for uploading Leaves
            //unit test for testing the format of time
            WelcomeDoctorUnittest ob = new WelcomeDoctorUnittest();
            Assert.AreEqual("doc1", ob.docid);
            Assert.AreEqual("11/27/2019", ob.startdate);
            Assert.AreEqual("12/27/2019", ob.enddate);
            Assert.AreEqual("00:00:00", ob.starttime);
            Assert.AreEqual("00:00:00", ob.endtime);
        }
    }
}

```

Test Case 2:

WelcomeDoctorUnittest.cs

Unit testing for testing the application for leave by Doctor

WelcomeDoctorUnittest.cs:

```

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

```

```
namespace HospitalSystem
{
    public class WelcomeDoctorUnittest
    {
        //Unt test for apply for leave

        public string docid;
        public string startdate;
        public string enddate;
        public string starttime;
        public string endtime;

        //unit test for upload prescription
        public string doctid;
        public string patientId;
        public DateTime day;
        public string imagepath;

        public WelcomeDoctorUnittest()
        {

            //unit test for testing date format in upload prescription
            this.doctid = "doc1";
            this.patientId = "pt1";
            this.day = Convert.ToDateTime("11-04-2019 01:00:00");
            this.imagepath = "~/ images / Screenshot(108).png";

            //unit test for testing foldername in upload prescription
            this.doctid = "doc1";
            this.patientId = "pt1";
            this.day = Convert.ToDateTime("11-04-2019 01:00:00");
            this.imagepath = "~/ image / Screenshot(108).png";*/
        }
    }
}
```

Test Method for testing the inputs for uploading the prescription

UnitTest1.cs:

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

```

namespace UnitTestProject1
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void TestMethod2()
        {
            WelcomeDoctorUnittest ob = new WelcomeDoctorUnittest();
            Assert.AreEqual("doc1",ob.doctid);
            Assert.AreEqual("pt1",ob.patientId);
            Assert.AreEqual(Convert.ToDateTime("11-04-2019 01:00:00"), ob.day);
            Assert.AreEqual("~/ images / Screenshot(108).png", ob.imagepath);
        }
    }
}

```

Test Case 3:

Unit testing to test the format for receipt

WelcomeEmployeeUnittest.cs:

```

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

namespace HospitalSystem
{
    public class WelcomeEmployeeUnittest
    {
        //unit test for upload receipt
        public string Employeeid;
        public string patientId;
        public DateTime day;
        public string imagepath;
        public string imagepath2;
        public WelcomeEmployeeUnittest()
        {
            //unit test for testing date format in upload prescription
            this.Employeeid = "emp1";
            this.patientId = "pt1";
            this.day = Convert.ToDateTime("11-04-2019 01:00:00");
            this.imagepath = "~/ image / Screenshot(108).png";
            this.imagepath2 = "~/ images / Screenshot(108).png";
        }
    }
}

```

```

    }
}

```

UnitTest2.cs:

```

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

namespace UnitTestProject1
{
    [TestClass]
    public class UnitTest2
    {
        [TestMethod]
        public void TestMethod1()
        {
            //Checking format of Datetime
            WelcomeEmployeeUnittest ob = new WelcomeEmployeeUnittest();
            Assert.AreEqual("emp1", ob.Employeeid);
            Assert.AreEqual("pt1", ob.patientId);
            Assert.AreEqual("11-04-2019 01:00:00", ob.day);
        }
    }
}

```

Test case 4:

Test case for making datetime conversions:

UnitTest2.cs:

```

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

namespace UnitTestProject1
{
    [TestClass]
    public class UnitTest2
    {
        [TestMethod]
        public void TestMethod2()
        {
        }
    }
}

```

```

{
    //Checking format of Datetime
    WelcomeEmployeeUnittest ob = new WelcomeEmployeeUnittest();
    Assert.AreEqual("emp1", ob.Employeeid);
    Assert.AreEqual("pt1", ob.patientId);
    Assert.AreEqual(Convert.ToDateTime("11-04-2019 01:00:00"), ob.day);
}

```

Test case 5:

Test case for testing the format of imagepath:

WelcomeEmployeeUnittest.cs:

```

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

namespace HospitalSystem
{
    public class WelcomeEmployeeUnittest
    {
        //unit test for upload receipt
        public string Employeeid;
        public string patientId;
        public DateTime day;
        public string imagepath;
        public string imagepath2;
        public WelcomeEmployeeUnittest()
        {

            //unit test for testing foldername in upload prescription
            this.doctid = "doc1";
            this.patientId = "pt1";
            this.day = Convert.ToDateTime("11-04-2019 01:00:00");
            this.imagepath = "~/ image / Screenshot(108).png";*/
        }
    }
}

```

Test method for checking the correct type of image path

UnitTest2.cs:

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

namespace UnitTestProject1
{
    [TestClass]
    public class UnitTest2
    {
        [TestMethod]

    }

    [TestMethod]
    public void TestMethod4()
    {
        //Checking format of imagepath
        WelcomeEmployeeUnittest ob = new WelcomeEmployeeUnittest();
        Assert.AreEqual("emp1", ob.Employeeid);
        Assert.AreEqual("pt1", ob.patientId);
        Assert.AreEqual(Convert.ToDateTime("11-04-2019 01:00:00"), ob.day);
        Assert.AreEqual("~/images / Screenshot(108).png", ob.imagepath2);

    }

}
}
```

Test case 6:**Unit test cases for changing password by the user****Changepassword.cs:**

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

namespace HospitalSystem
{
    public class ChangePassUnittest
    {
        //unit test for change password by patient
        public string patientid;
        public string oldpswd;
        public string newpswd;
```

```

        public string cnfrmpswd;

        public ChangePassUnittest()
        {
            //unit test case when old password does not match
            this.patientid = "pt10";
            this.oldpswd = "abcd1234";
        }
    }
}

```

Test Method for testing the old password inputted by the user

Unittest3.cs:

```

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

namespace UnitTestProject1
{
    [TestClass]
    public class UnitTest3
    {
        [TestMethod]
        public void TestMethod1()
        {
            ChangePassUnittest ob = new ChangePassUnittest();
            Assert.AreEqual("pt10", ob.patientid);
            Assert.AreEqual("abcde1234", ob.oldpswd);
        }
    }
}

```

Test case 7:

Test case when the old password is correct but the new password does not match with the confirm password field

Changepassword.cs:

```

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

namespace HospitalSystem

```

```
{
    public class ChangePassUnittest
    {
        //unit test for change password by patient
        public string patientid;
        public string oldpswd;
        public string newpswd;
        public string cnfrmpswd;

        public ChangePassUnittest()
        {

            //unit test when old password matches but new passwords do not match
            this.patientid = "pt10";
            this.oldpswd = "abcde1234";
            this.newpswd = "1234";
            this.cnfrmpswd = "12345";*/
        }
    }
}
```

Unittest3.cs:

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

namespace UnitTestProject1
{
    [TestClass]
    public class UnitTest3
    {

        [TestMethod]
        public void TestMethod2()
        {
            ChangePassUnittest ob = new ChangePassUnittest();
            Assert.AreEqual("pt10", ob.patientid);
            Assert.AreEqual("abcde1234", ob.oldpswd);
            Assert.AreEqual("1234", ob.newpswd);
            Assert.AreEqual("1234", ob.cnfrmpswd);
        }
    }
}
```



```
}
```

Test case 8:

Unit testing of change password when the new password matches with the confirm password field and the old password is correct.

Changepassword.cs:

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

namespace HospitalSystem
{
    public class ChangePassUnittest
    {
        //unit test for change password by patient
        public string patientid;
        public string oldpswd;
        public string newpswd;
        public string cnfrmpswd;

        public ChangePassUnittest()
        {

            //unit test when new password and confirm passwords match
            this.patientid = "pt10";
            this.oldpswd = "abcde1234";
            this.newpswd = "1234";
            this.cnfrmpswd = "1234";
        }
    }
}
```

Unittest3.cs:

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

namespace UnitTestProject1
{
    [TestClass]
```

```
public class UnitTest3
{
    [TestMethod]

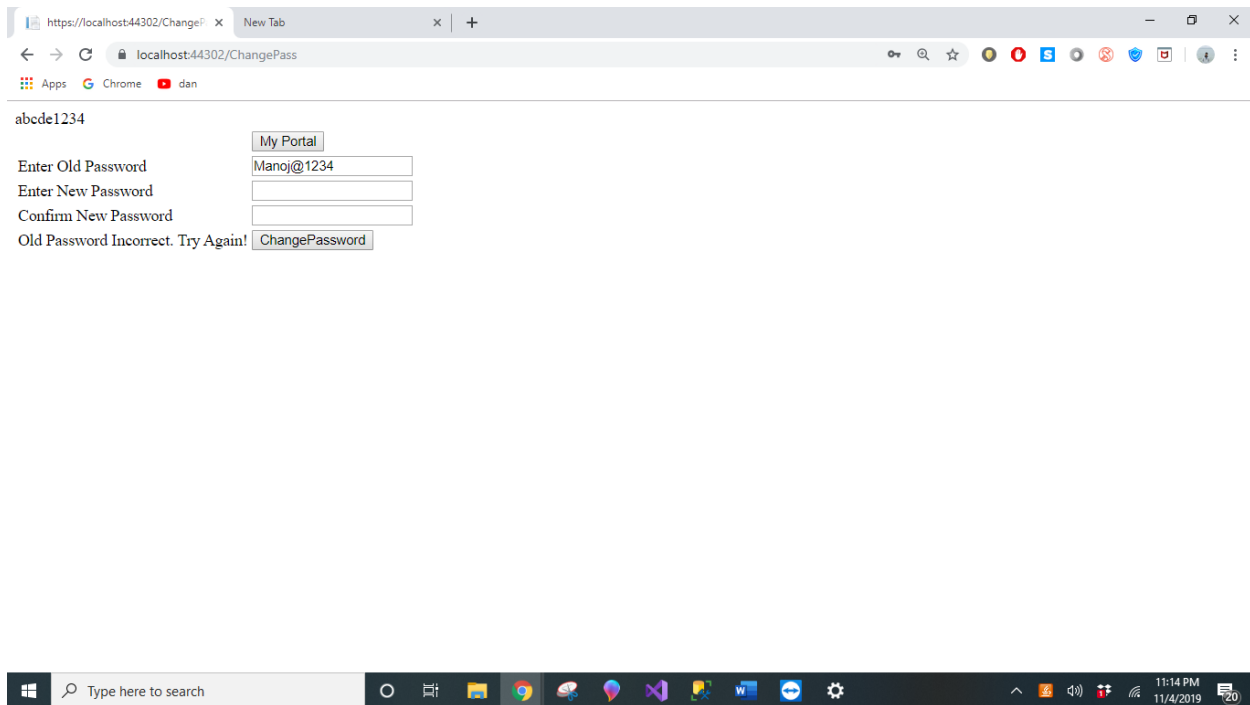
    public void TestMethod3()
    {
        ChangePassUnittest ob = new ChangePassUnittest();
        Assert.AreEqual("pt10", ob.patientid);
        Assert.AreEqual("abcde1234", ob.oldpswd);
        Assert.AreEqual("1234", ob.newpswd);
        Assert.AreEqual("1234", ob.cnfrmpswd);
    }
}
```

3.2. Functional test-cases for the deliverable-4 requirements

3.1 Functional Test-cases for requirement change password

Case1:

If the user input for the old password, then an error message will be displayed on the screen.

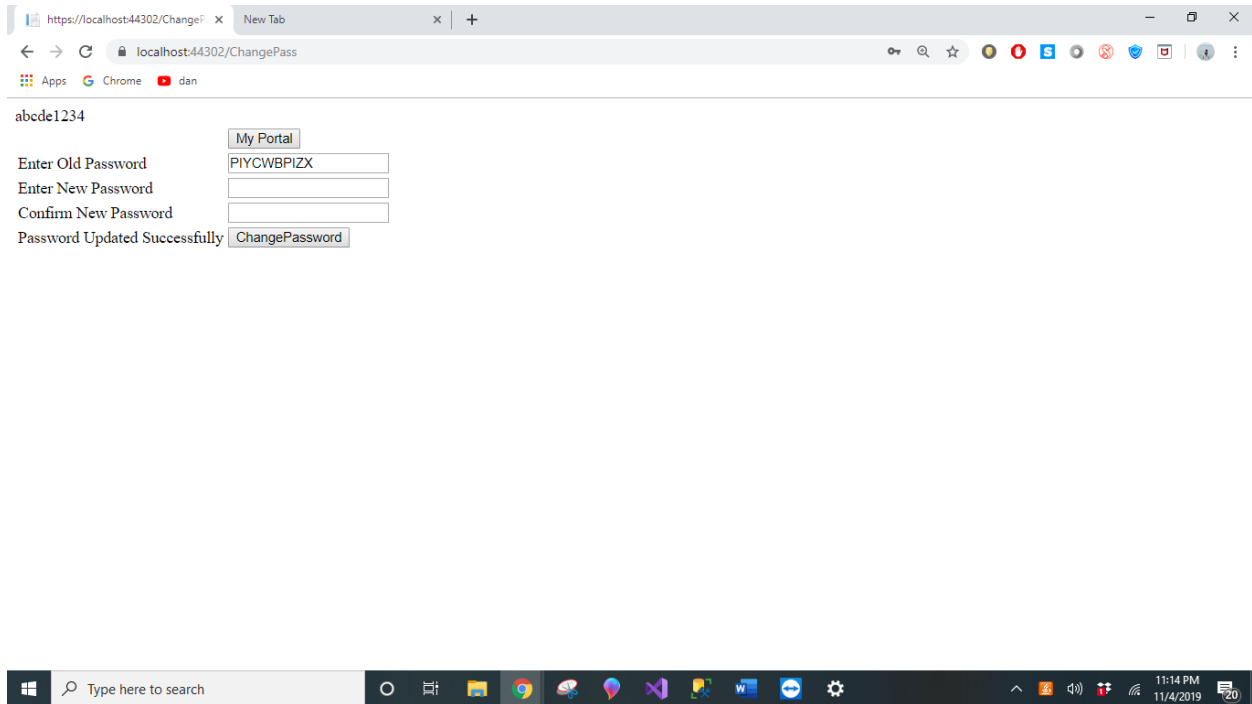


Case2:

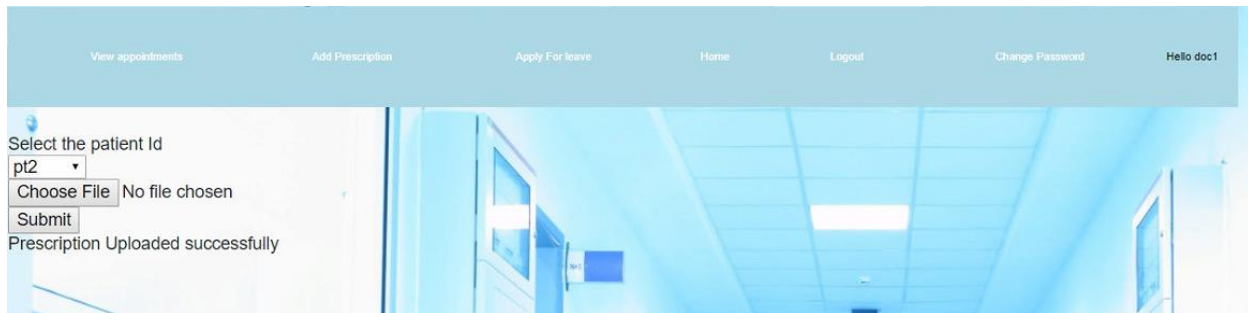
If the user input for the new password and repeat new password doesn't match, then an error message will be displayed on the screen.

Case3:

If the user input for the old password, new password and repeat new-password is correct then a successful message will be displayed on the screen.

**3.2 Functional Test-case for the Upload prescription****Case1:**

Once the doctor uploads the prescription then the patient is able to view the prescription when he clicks the see prescription button.



3.3 Functional Test-case for the Upload Receipt

Case1:

Once the front-desk user uploads the receipt then the patient is able to view the receipt when he clicks the see receipt button.

3.4 Functional Test-case for the Upload Doctor's availability

Case1:

If the doctor selects the half-day leave, then he should be able to select first-half or the second half of the day from the drop-down list. Once he clicks the submit button, a confirmation message should be displayed on the screen.

Case2:

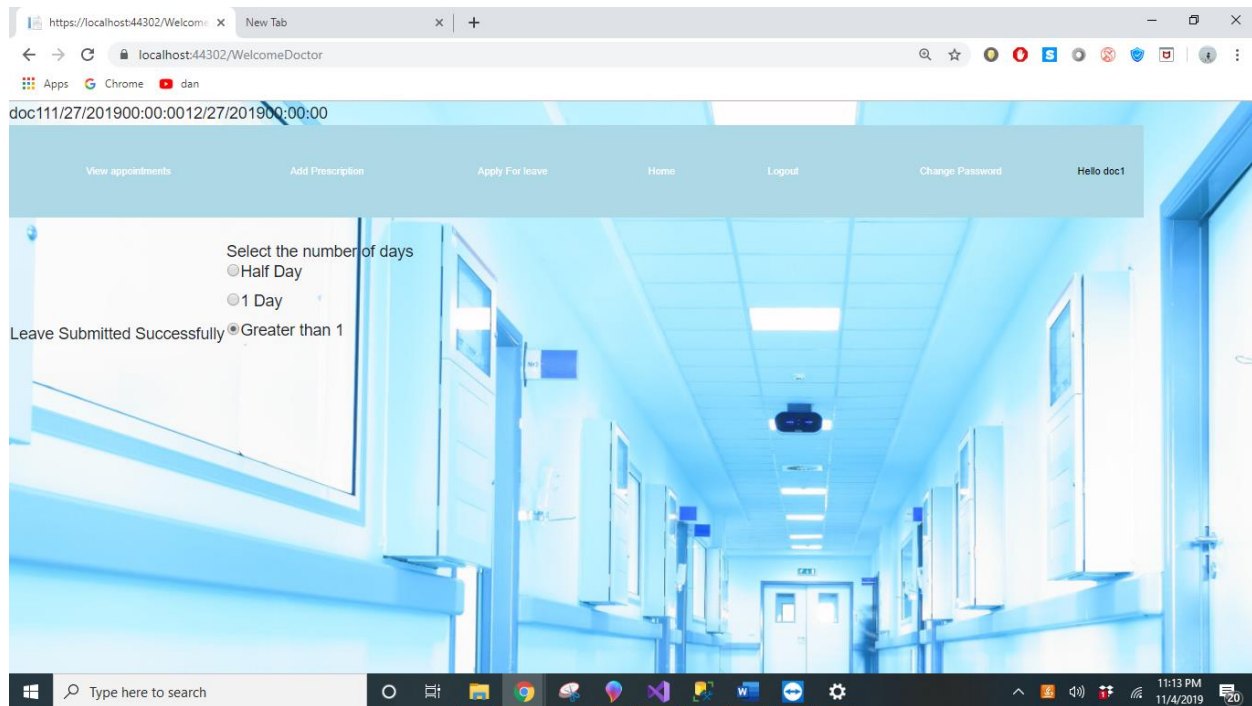
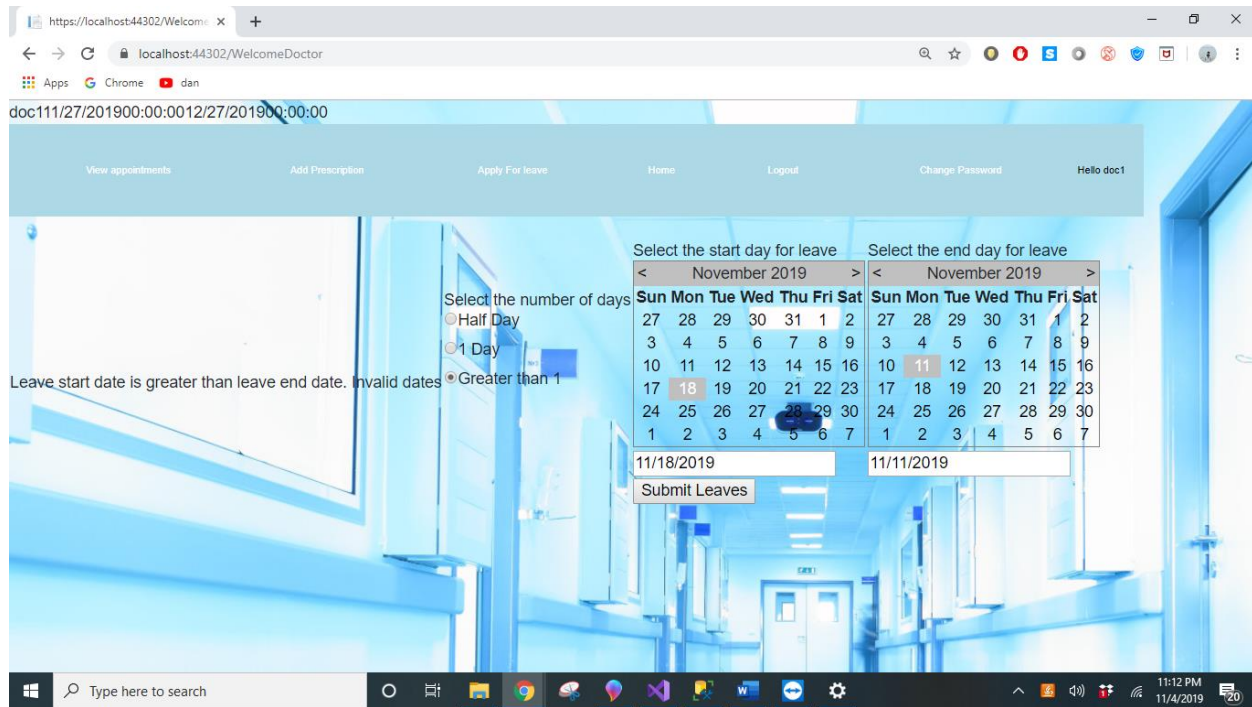
If the doctor selects the one-day leave, then he should be able to select the date from the calendar control. Once he clicks the submit button, a confirmation message should be displayed on the screen.

Case3:

If the doctor selects greater than 1-day option and selects the leave and date which is earlier than the leave start date, then an error message will be displayed on the screen.

Case4:

If the doctor selects greater than 1-day option and selects the leave and date which is greater than the leave start date, then he should be able to submit the leave successfully.



4. Who wrote what components/classes of the system

@Testing

File Name	Person
WelcomeDoctorUnittest.cs	Maulshree Verma
ChangePassunitetest.cs	Maulshree Verma
WelcomeEmployeeUnittest.cs	Maulshree Verma
Unittest1.cs	Maulshree Verma
Unittest2.cs	Maulshree Verma
Unittest3.cs	Maulshree Verma

@Changepassword

File Name	Person
Changepassword.aspx	Yogesh

@Profilepicture

File Name	Person
WelcomePatient.aspx	NagaSaiTeja

@Doctorsavailability

File Name	Person
WelcomeDoctor.aspx	Maulshree Verma

@Prescription

File Name	Person
WelcomeDoctor.aspx	Manoj

@Receipt

File Name	Person
WelcomeEmployee.aspx	Yogesh

@AppintmentRequests.cs

Filename	Person
AppointmentRequests.aspx	Maulshree Verma
AppointmentsRequests.Designer	Maulshree Verma
AppointmentRequests.aspx.cs	Maulshree Verma

Database Connectivity:**@App_Code:**

File Name	Person
Class1.cs	Maulshree Verma

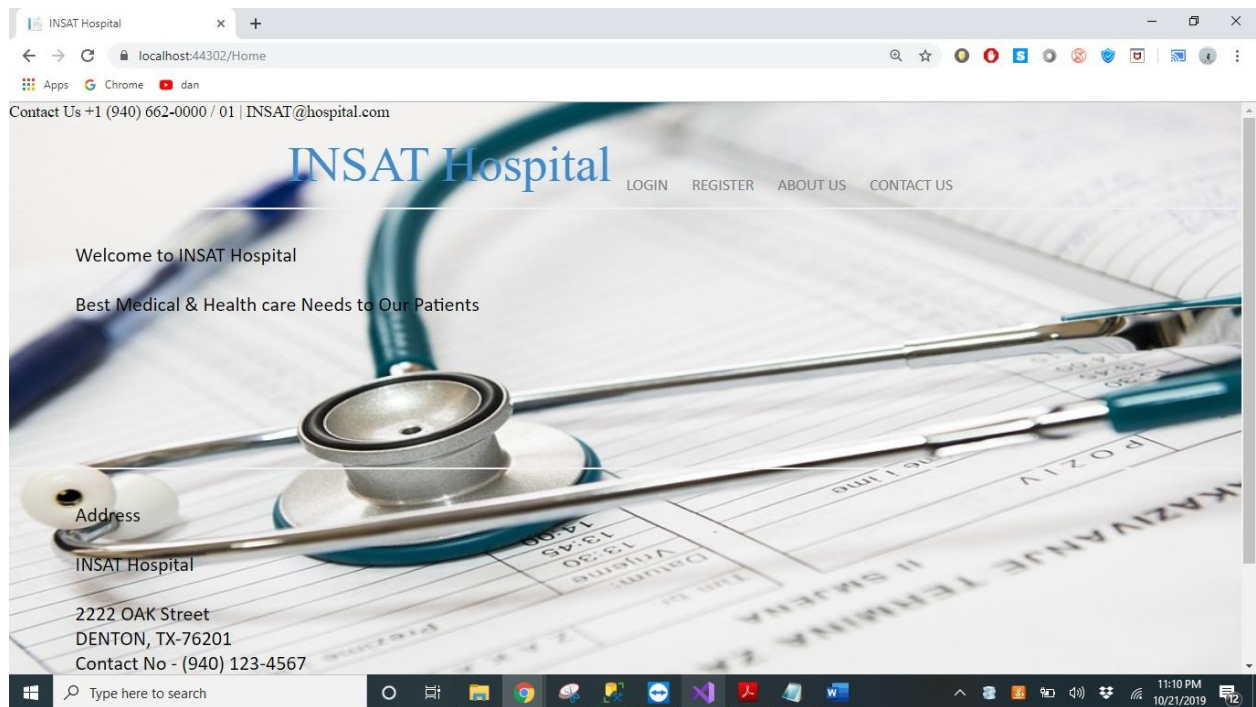
5. User Manual**5.1. Summary:**

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" 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".



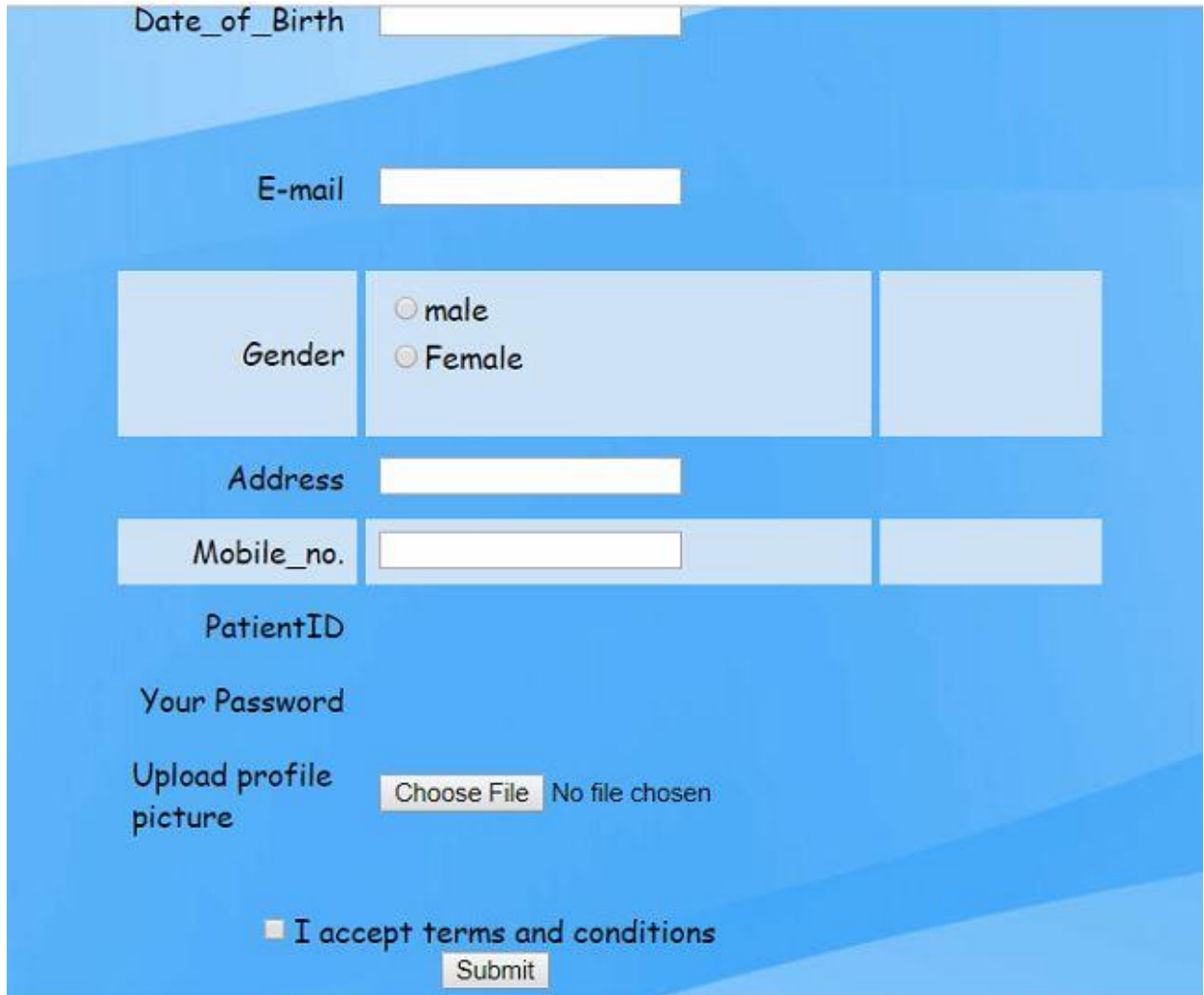
5.3. Patient with no account:

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.

A screenshot of a web browser displaying the 'Register Page' of the hospital management system. The browser's address bar shows 'localhost:44302/Register'. The form is set against a blue background with a stethoscope image on the right. The form fields include: 'First Name', 'Last name', 'Age', 'Date_of_Birth', 'E-mail', 'Gender' (with radio buttons for 'male' and 'Female'), 'Address', 'Mobile_no.', 'PatientID', and 'Your Password'. The Windows taskbar at the bottom shows the time as 10:20 PM on 10/21/2019.

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

A user registration form with a blue background. The form contains several input fields and a submit button. The fields are: Date_of_Birth (text input), E-mail (text input), Gender (radio buttons for male and female), Address (text input), Mobile_no. (text input), PatientID (text input), Your Password (text input), and Upload profile picture (with a Choose File button and No file chosen text). At the bottom, there is a checkbox for 'I accept terms and conditions' and a Submit button.

Date_of_Birth

E-mail

Gender ☐ male ☐ Female

Address

Mobile_no.

PatientID

Your Password

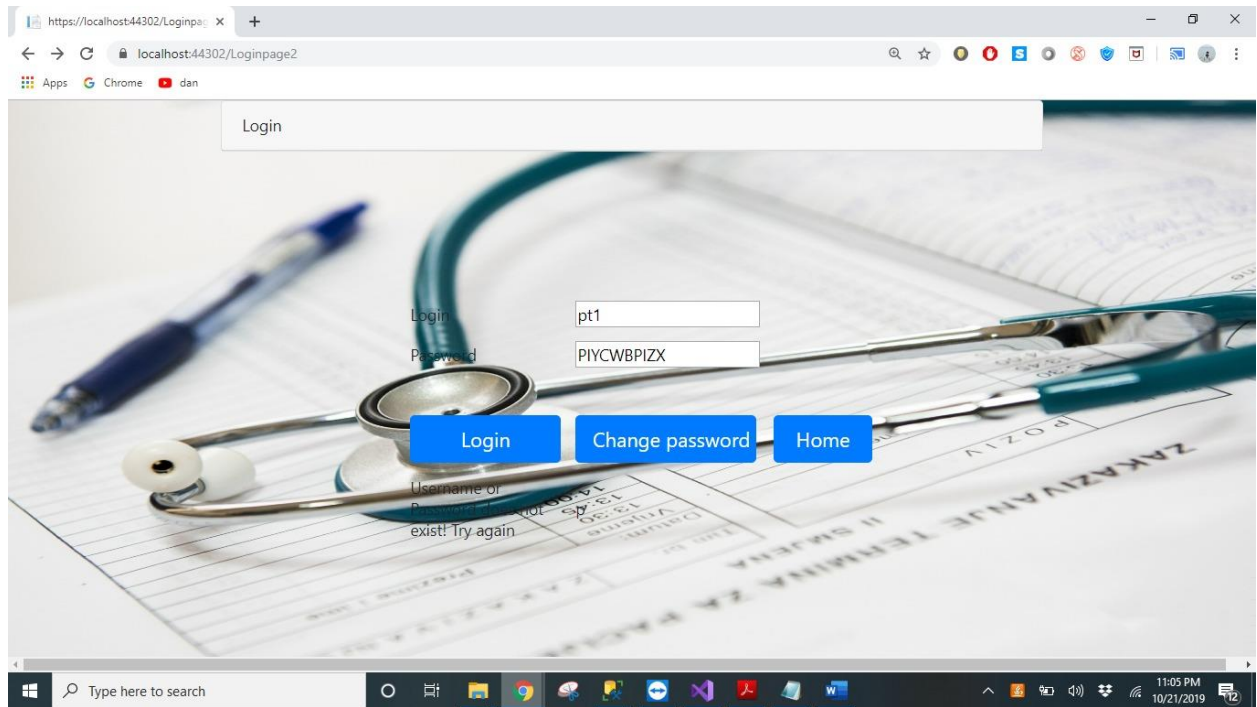
Upload profile picture No file chosen

☐ I accept terms and conditions

- 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.



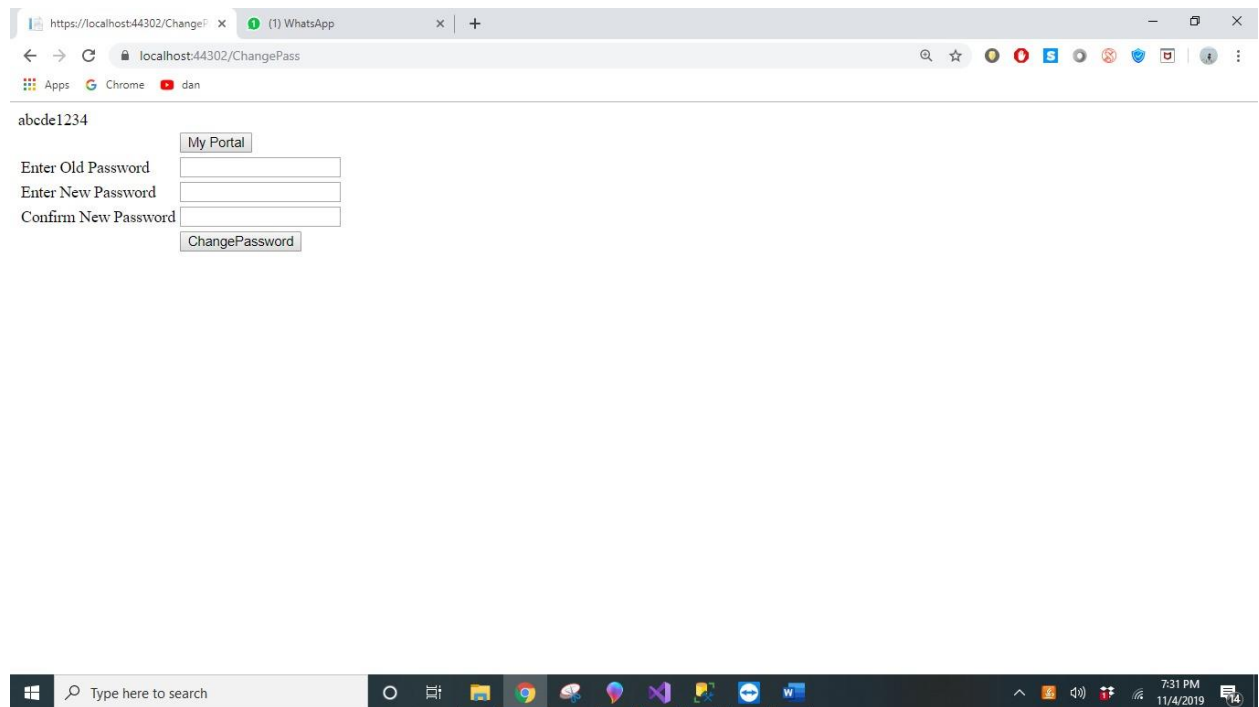
- Once the user is logged in, then the patient can request for an appointment.



- 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.

The screenshot displays a web application running on a local host. The browser window has a single tab titled 'https://localhost:44302/Request'. The address bar shows the URL 'localhost:44302/Request%20for%20appointment'. The page features a navigation bar with 'Home' and 'My Portal' links. The main heading is 'Request For Appointment'. Below this, there is a form with the following fields: 'Speciality' (a dropdown menu with '--Select--' as the current selection), 'Doctor' (a dropdown menu), 'Date' (a text input field), 'Time' (a text input field), and 'Reason' (a larger text input field). A 'Submit' button is located at the bottom of the form. The Windows taskbar at the bottom of the screen shows the search bar, several application icons, and the system clock indicating 11:06 PM on 10/21/2019.

- 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.

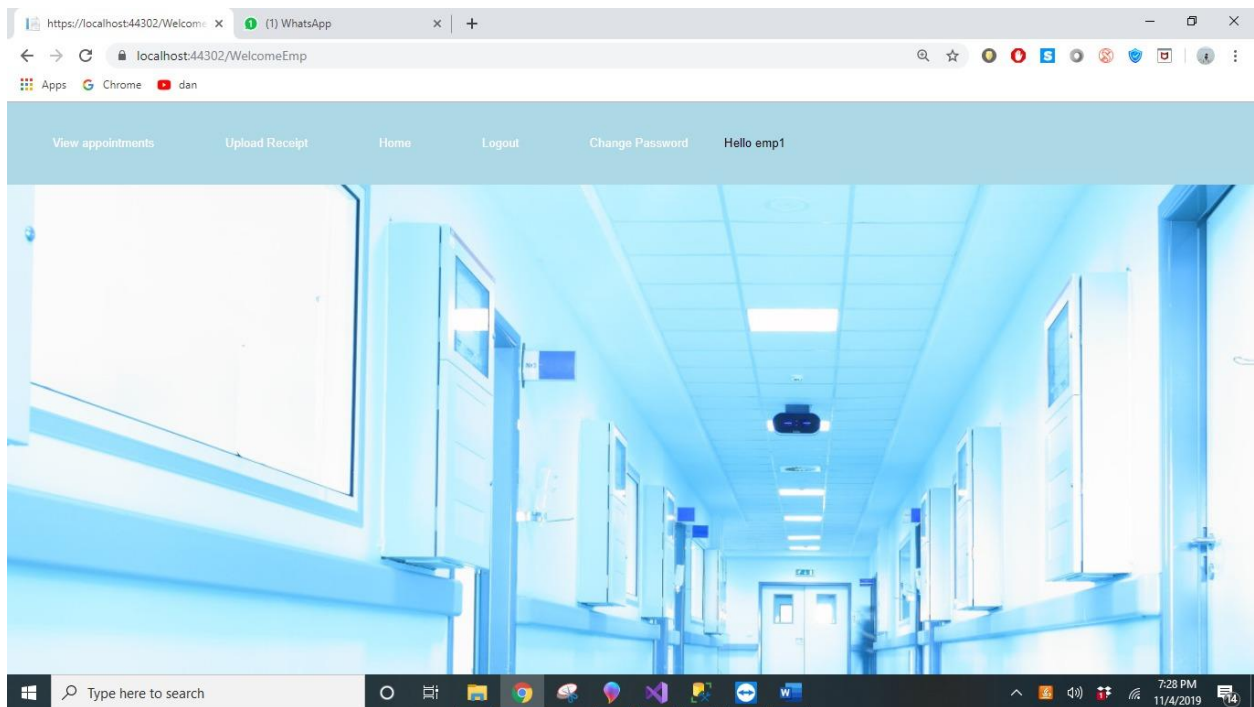


5.5. Admin

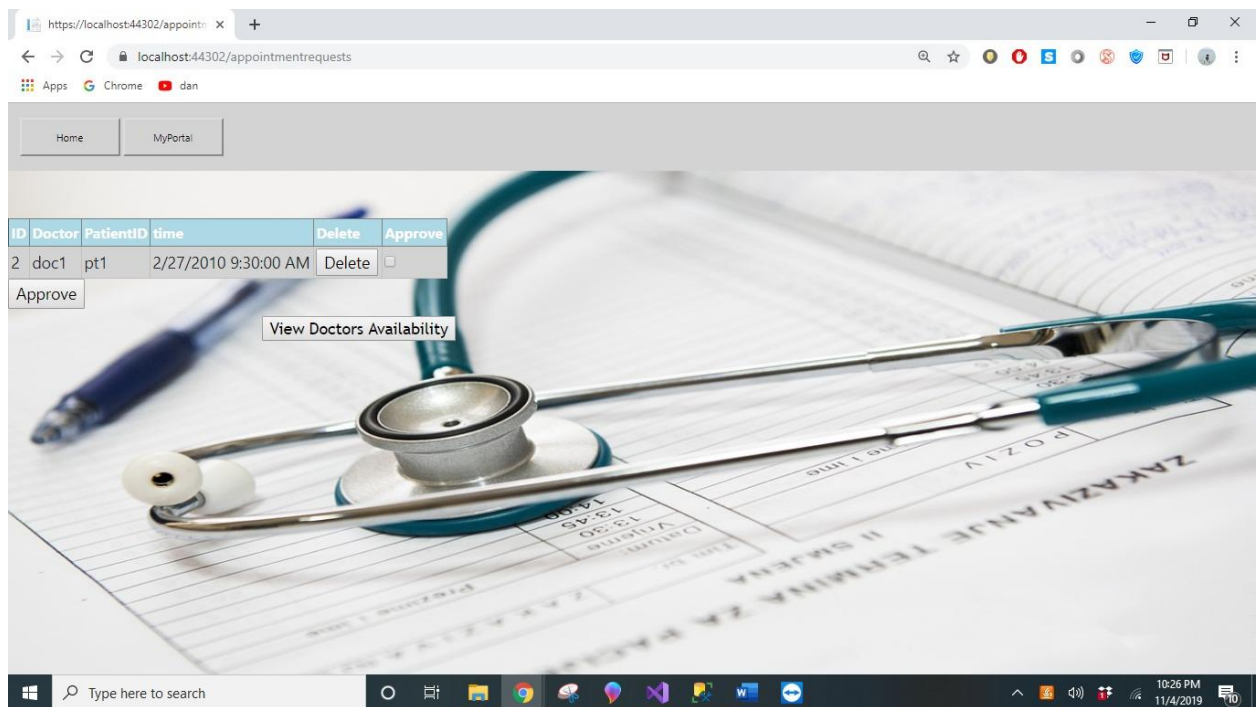
- Admin views a similar website as the customer with having additional options for him.
- The admin will add the doctors and the front-desk users into the system.
- The admin will have all the privileges.
- The admin generates the UserID and passwords for the doctors and Front-desk User.

5.6. Front-Desk User

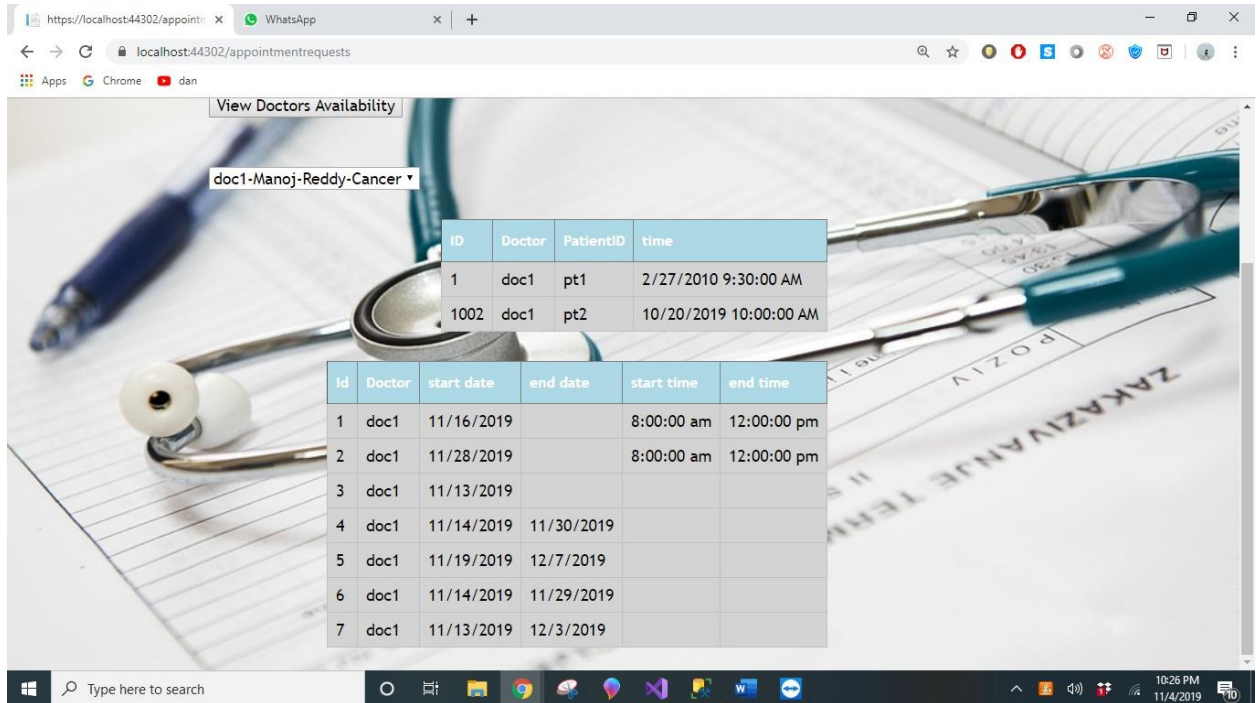
- Once the Front-desk user logs into the account the Front-desk user page looks as follows:



- The Front-desk user will approve/decline the appointments requested by the patients based on the availability of the doctor.
- The front desk user can also view the doctors availability so that based on that availability the front desk user will approve/cancel the requests of the appointments.



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

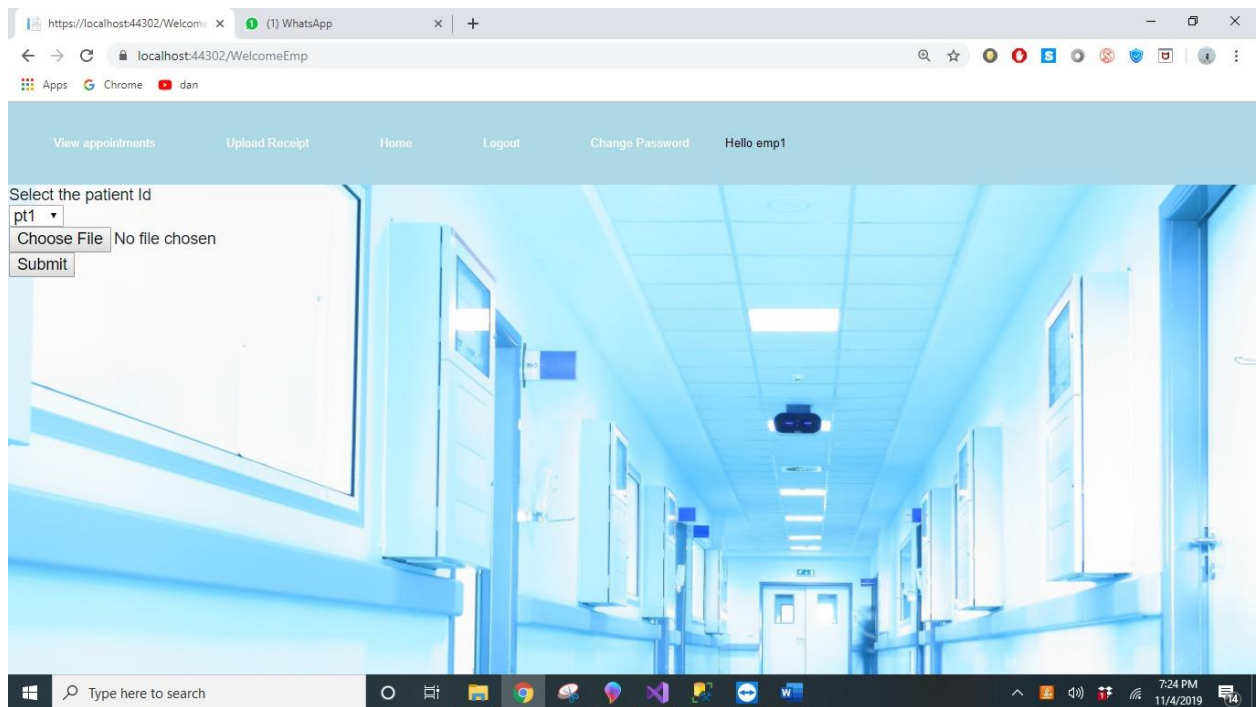


The screenshot shows a web browser window with the URL `localhost:44302/appointmentrequests`. The page title is "View Doctors Availability". A dropdown menu shows "doc1-Manoj-Reddy-Cancer". Below the dropdown are two tables.

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.

Request For appointments Home Logout Change Password See Prescriptions See Receipts Hello pt1

My Approved Appointments

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

1 emp1 pt1 11/4/2019 12:51:43 AM

Receipt

Email Canvas Catalogs People & Departments Calendar Map

Student Human Resources

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

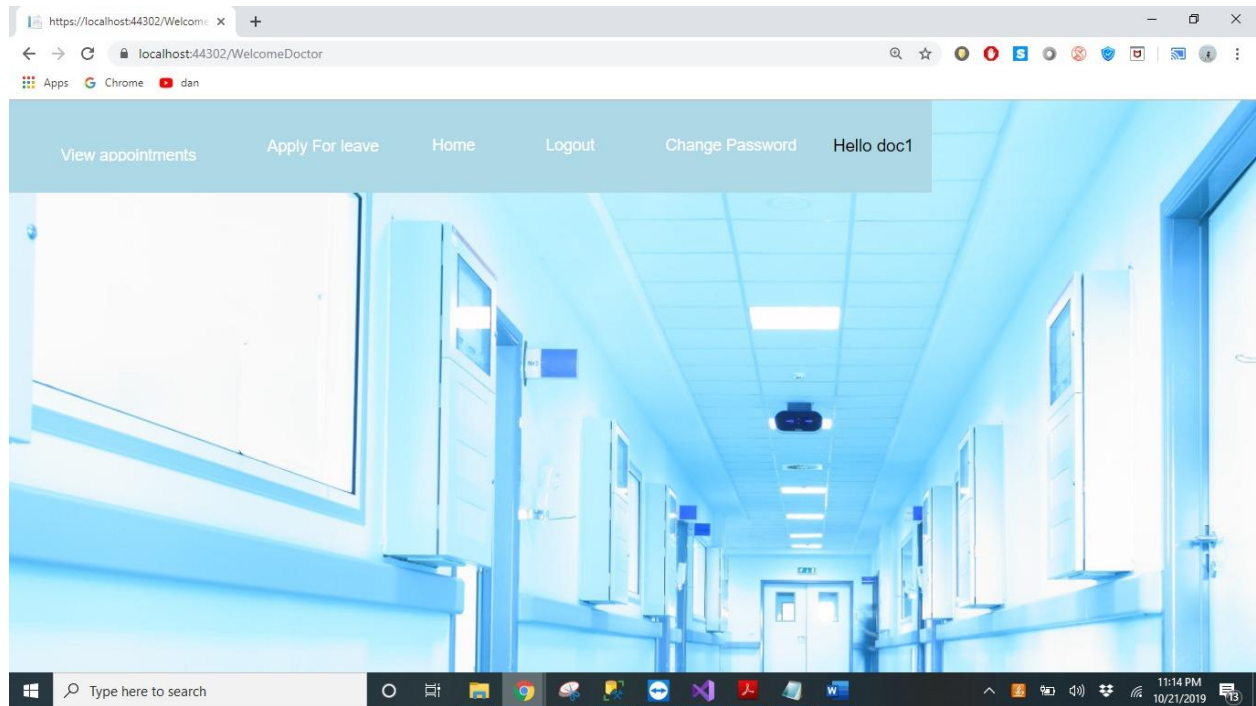
Status	Units	Grading	Grade	Deadlines
Enrolled	3.00	Graded		

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

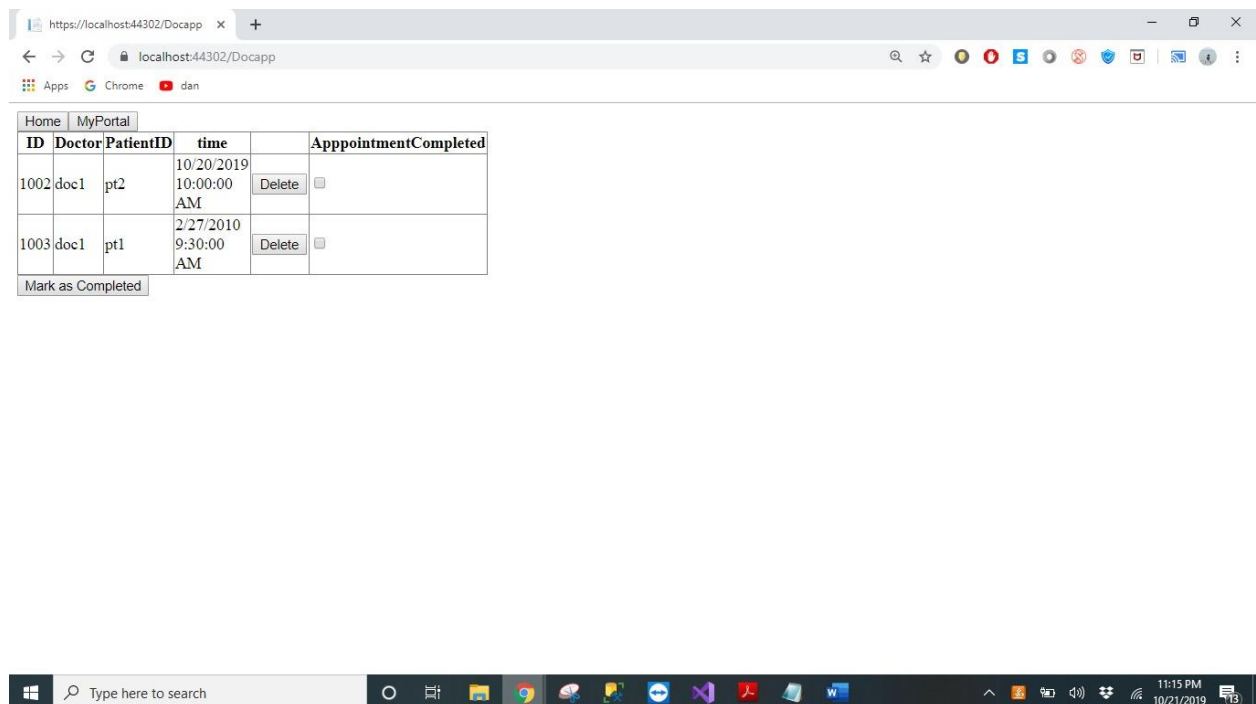
CSCE 5430 - SOFTWARE ENGINEER

5.7. Doctor

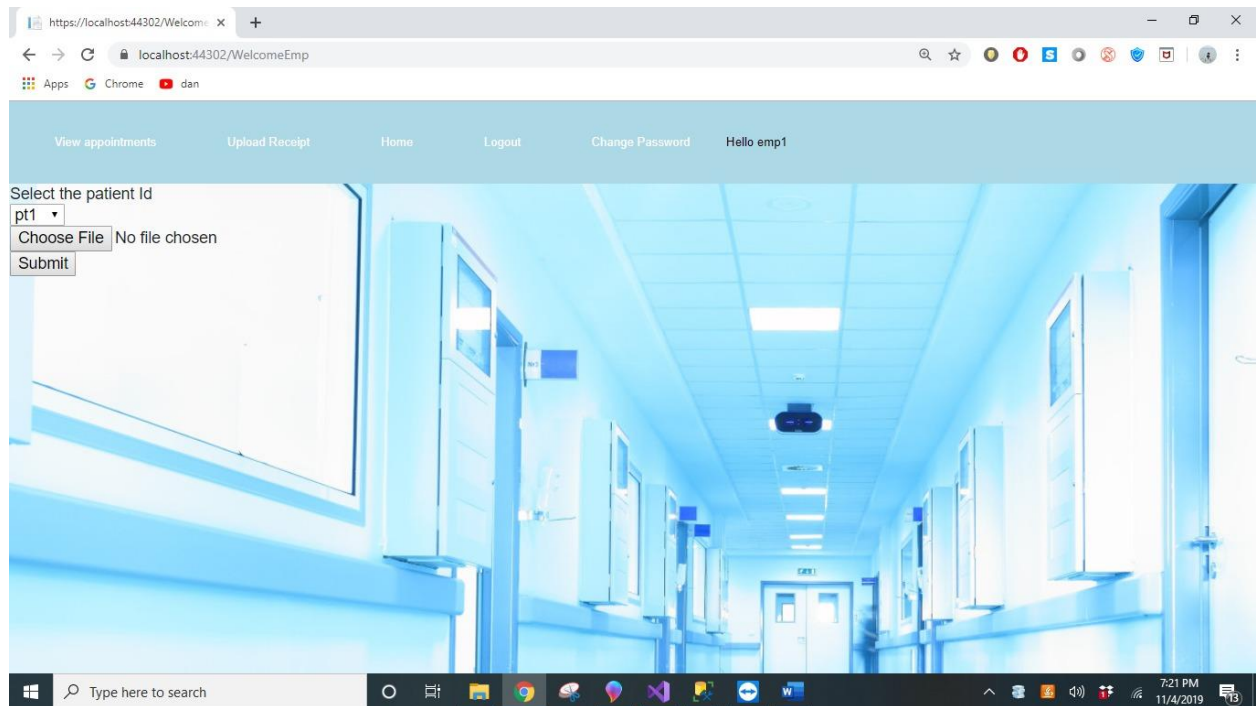
- Once the doctor logs into the account the doctor's page looks as follows:



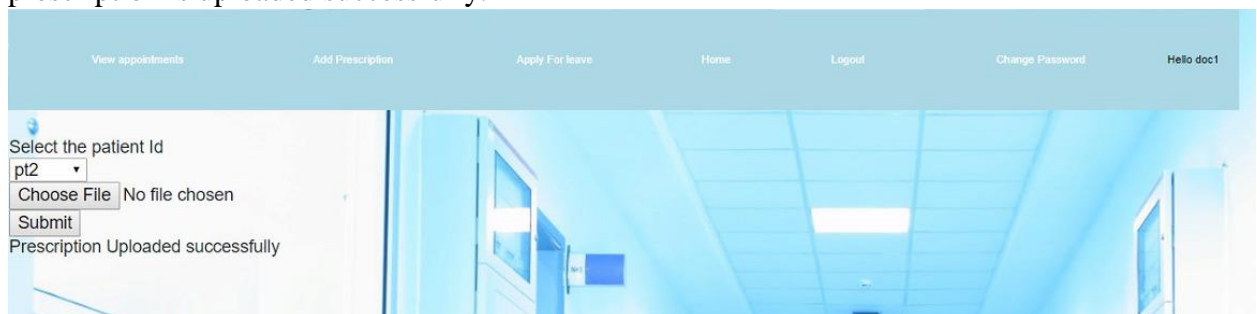
- Doctor checks the patient and updates the appointment status to completed if the treatment is done.



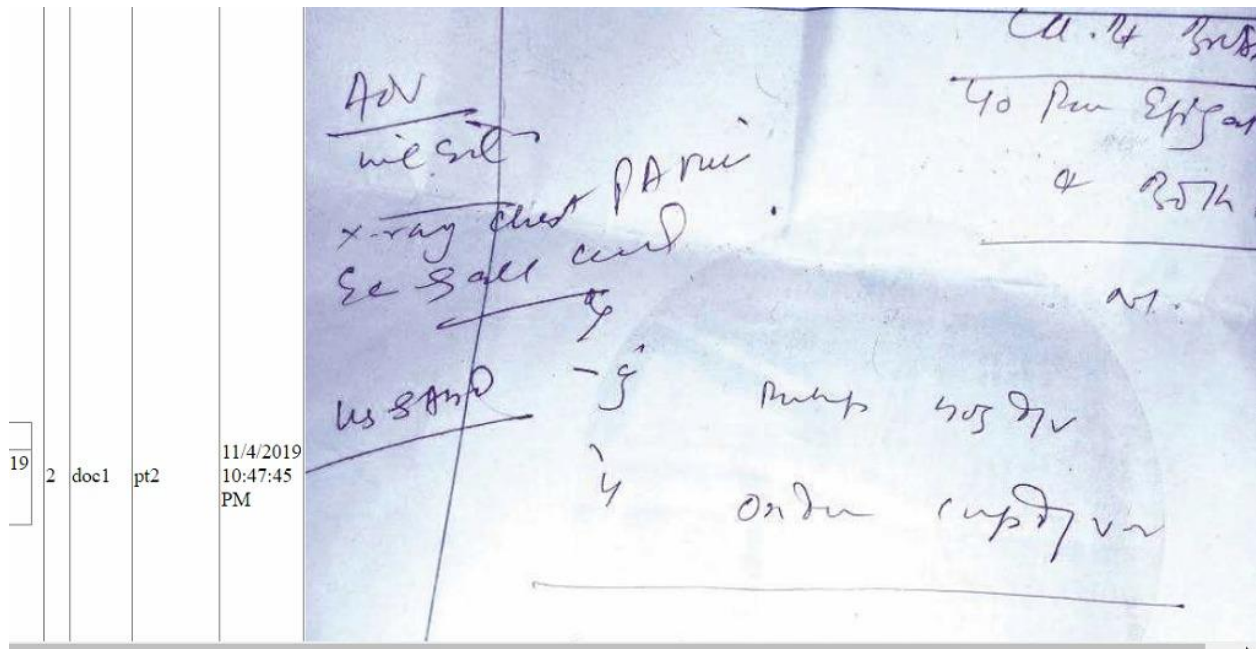
- The doctor will select the patient to whom he/she needs to upload the prescription.



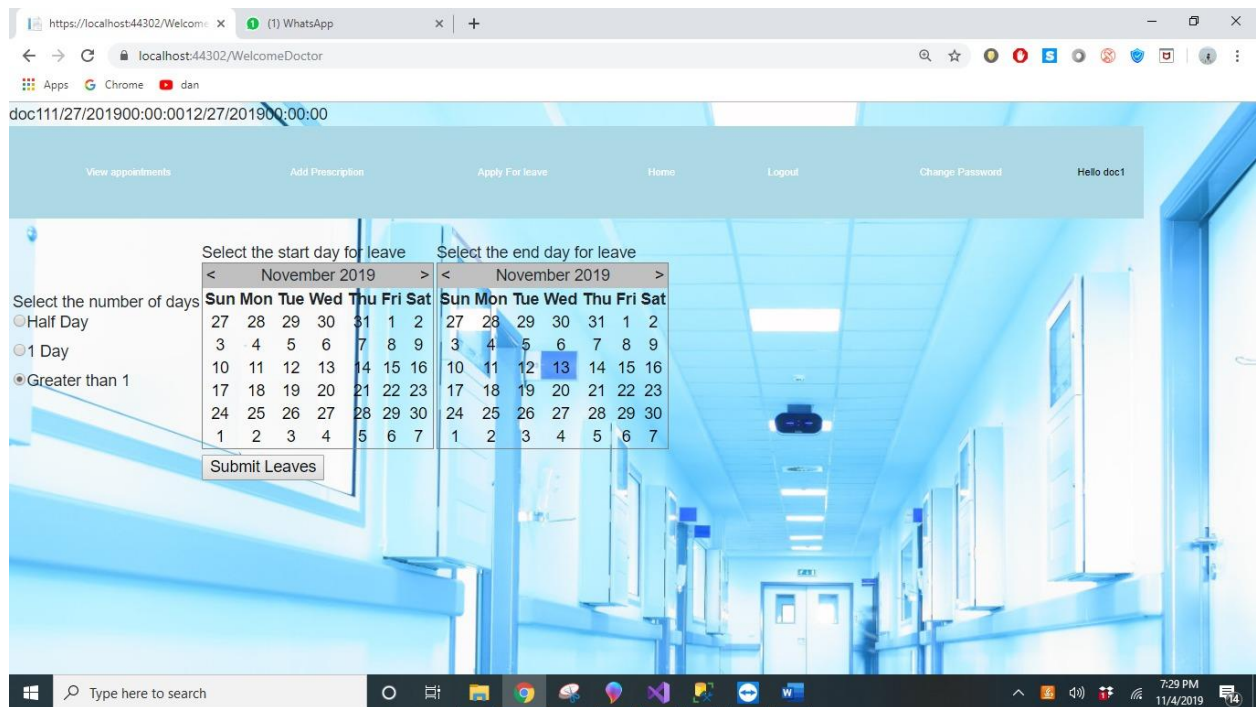
- After selecting the patient to whom the prescription needs to be sent, the doctor will upload the prescription to the respective patients account.
- Once the prescription is uploaded then the notification will be displayed that the prescription is uploaded successfully.



- The patients can view the prescription in their account.



- 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.



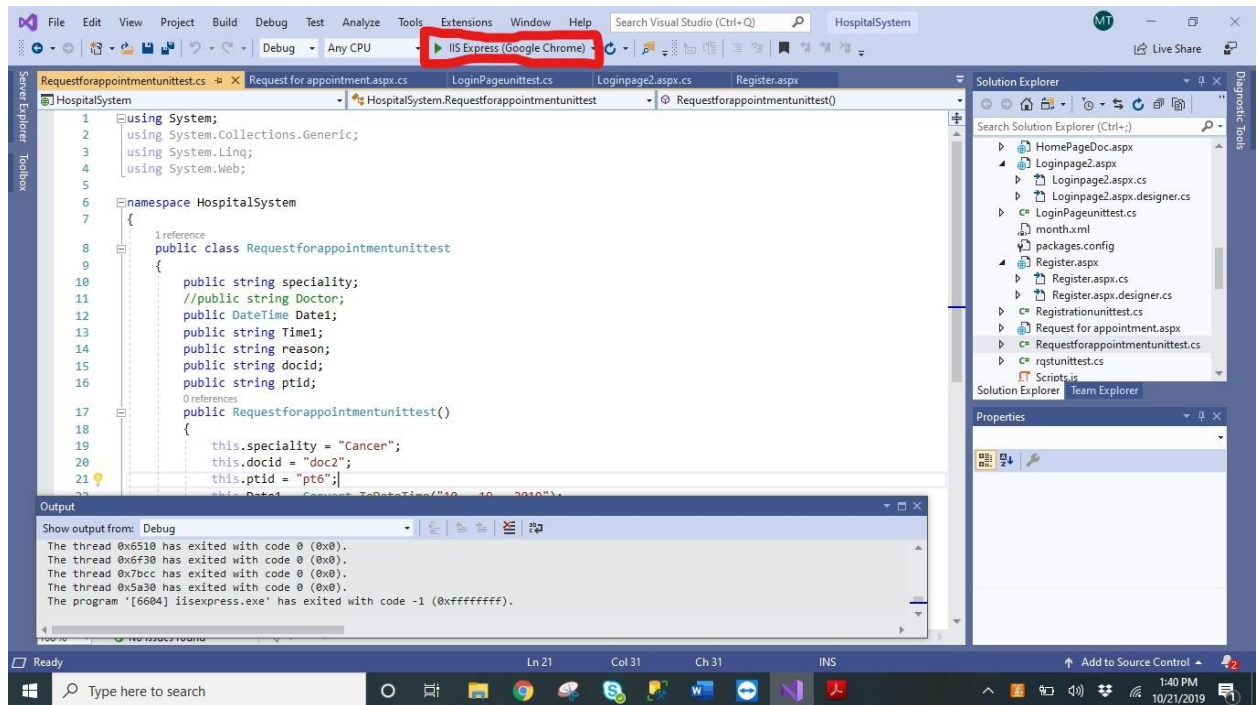
6. To compile/run the program and test cases

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.



- 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

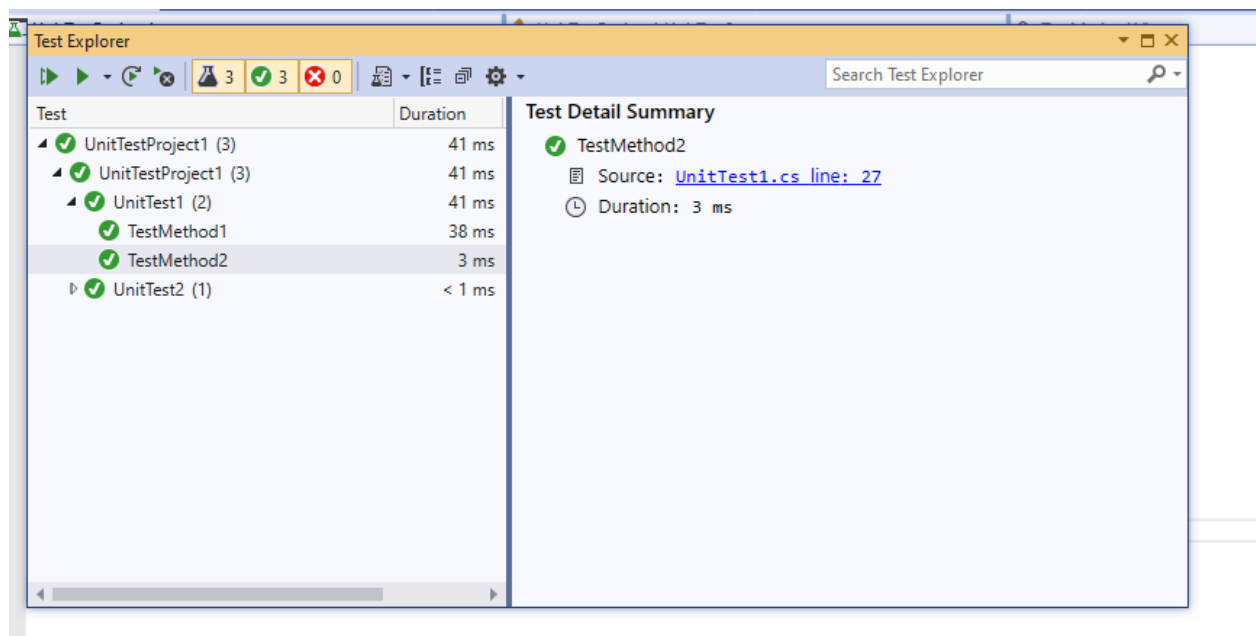
Doctor:

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 of change password we create a class unittest3.cs
- We initialize the variables used in that web form and hard code the values in the constructor.
- Next, we check these 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:



7. Feedback received during peer review Session

- User can create multiple accounts which should be validated. So, we need to make sure that every user can create only one account.
- The date of birth and age should match exactly which hasn't been validated.

7.1. Actions need to be taken based on the feedback

- We will make sure that each user can create a maximum of only one account.
- We will also make sure that age and date of birth matches exactly.