MURDOCH UNIVERSITY
PERTH, WESTERN AUSTRALIA

School of Engineering and Information Technology

ASSIGNMENT COVER SHEET

- Please complete and insert this form as the first page of EACH electronic assignment.
- Submit the assignment with attached coversheet electronically as per the instructions in the Assignment Question sheet.
- Please make sure you keep a copy of the assignment.

Student Details

Surname	Tan	Given name	Hong Rui Freeman
Student Number	34177608	Email	Mikkiz1996@gmail.com

Assignment details

Unit name	Software Development Frameworks	Unit Code	ICT 365
Unit Coordinator	Mohammed Kaosar	Tutor/Tutorial time	5:34 PM
Due date/time	3rd December 2022	Submission date	3rd December 2022
Assignment title	Assignment 02		
Other information			

All forms of plagiarism, cheating and unauthorized collusion are regarded seriously by the University and could result in penalties including failure in the unit and possible exclusion from the University. If in doubt, please contact the Unit Coordinator.

Student's Declaration

Please double click on all the check boxes.

☒ Except where I have indicated, the work I am submitting in this assignment is my own work and has not been submitted for assessment in another unit.

☒ This submission complies with Murdoch University policies regarding plagiarism, cheating and collusion.

☒ I have retained a copy of this assignment for my own records.

Database Design and Database SQL Codes

- For my database design/create on Microsoft SQL Server Management Studio 18. I did 2 Database one is for the Accounts and another one is for the Bookings (Appointment Bookings). AccountIDs will be the primary key, but it will link to the Bookings database as a foreign key so that we will be able to check the existing of the account and the account roles base on their account IDs and Name.
- For my first database which is the accounts it will have
  Account: AccountID, FullName, Email, AccountPassword, PersonAddress, PersonPhoneNo, Gender BloodType, AccountRole, DOB
- For my second database which is the booking it will have
  Booking: BookingID, BookingDate, TimeSlot, Doctor Name(For Admin/Doctor), Doctor Email,
  Department/Reason, AccountID (Foreign Key), Status, Patient Name, Patient Email
- For the Account Role system Anyone can register and login to the account but only the admin can access the “Account” Database from the Admin Control Panel to help the user to update their account details or change the password.
- For the “Booking” Database Patient will be able to Create, Edit, Delete and View details of their booking (Without the Doctor Details and Status). For the Admin and Doctor, they will be also able to Create, Edit, Delete and View the booking page database but they will be able to add in the Doctor Details and Status for the patient to confirm their booking or pending for it.

Algorithm/Design Plan/Explanation

- For my Design Plan I use ASP.NET MVC for this webpage design plan. Which it has Models, View, and Controllers for each of this website page. MVC is good because it is efficient in term of creating a webpage with database because it can auto generate the website page for you when you have a database using the ADO.net Entity Data Model to create the Controllers and the Views for your website. For testing purpose ASP.NET MVC has more response time and loading time compared to webform design. Which make the developer/designer can test their website/webpage faster and debug it quicker. ASP.NET MVC has more control in designing the website/webpage using HTML, CSS (Styling) and JavaScript function for specific website/webpage if needed. Also, the good thing for ASP.NET MVC it is more object-oriented where the files is more organize such as it separates the program function implementation from the website design user interface instead if putting them all together in the same files/folder which will be very messy and hard to find the specific file/page that we wish to debug or work on. Also, when modifying or editing the webpage ASP.NET MVC does not affect the entire model when modification and changes is on-going on the development phase. Also, you can create more views using a single model on ASP.NET MVC which allow the developer/designer to finish developing the website in a fast process. Basically ASP.NET MVC is good because most of it I have full controls over the development phase and implementation phase. But ASP.NET MVC does require you to refresh your HTML, JavaScript, and CSS

(Styling) for developing a website/webpage on the Views session where it is almost like an old school developing a website/webpage using notepad++ during the past. ASP.NET MVC also use HTTP request and response to get the user input form as an action result where it will fulfill the user request without much trouble.

Limitation

< Program/Webpage >

1. The Program/Webpage is not that advanced, and it have limitation.
Which it only has basic function system like Create, Edit, View and Delete items from the webpage which will be update
on the database automatically too.

2. The patients can view other patients' appointment/booking data and details.
I not sure how to restrict the patient to view their own data and details only.

3. The AccountID for Doctor and Admin Control Panel when they wish to create a appointment for the user it show the AccountID (First Name) Instead of the AccountID. Which I not sure how to fix that too.

4. The Webpage/Program is quite simple without much function and input.

5. This webpage/Program I would say is a skeleton or mock-up page given the amount of time I have and learn MVC from the beginning. I would improve more in the future and add more for function for it.

6. I try to implement the Navigation Bar for the Admin, Doctor, and Patient so that they can click on the Navigation bar to return to their own control panel but i fail to do that. Because of the role restriction the other user role also able to view the other Navigation Bar Link/Button.

Example: The Admin can see Patient/Doctor Control Panel Navigation Bar Button same for the other way.

Therefore, I never implement this future/function for the Navigation bar.

7. This webpage only have like 2 databases, but it should have more and properly linked

User Guide

< User Guide for the Assignment Program/Website >

Murdoch Hospital -> Will also just bring the user back to the homepage content.
Home button/link -> Will just bring the user back to the homepage content.

About button/link -> Will just bring the user to the About Us page content of the hospital. Where the user can learn about the hospital history and story.
Contact button/link -> Will just bring the user to the Contact Us Page content of the hospital where the user can find the contact details for the hospital.
Service button/link -> Will just bring the user to the Services Page Content of the hospital where the user can find out what kind of services or treatment that the hospital is providing.

Registration button/link -> Will bring the user to the registration page. Were the user can register their account so that they will be able to login and use the hospital webpage system. User will have to enter their own information when register/create an account if not they won't be able to go through it.
For testing purpose, I set the Role on the registration page. The user also, can return to back to the homepage by clicking on the "Back to Homepage" button.

Login button/link -> Will bring the user to the login page. Where they can login with the account that they created or register from the registration page. They will have to enter their Email, Password and Account Role when login. The user also, can return to back to the homepage by clicking on the "Back to Homepage" button.

Admin Control Panel -> User will have to login with an Account Role "Admin" in order to go to the Admin Control Panel Webpage. Where the Admin User can Create, Edit, Delete
and view all the user account details.

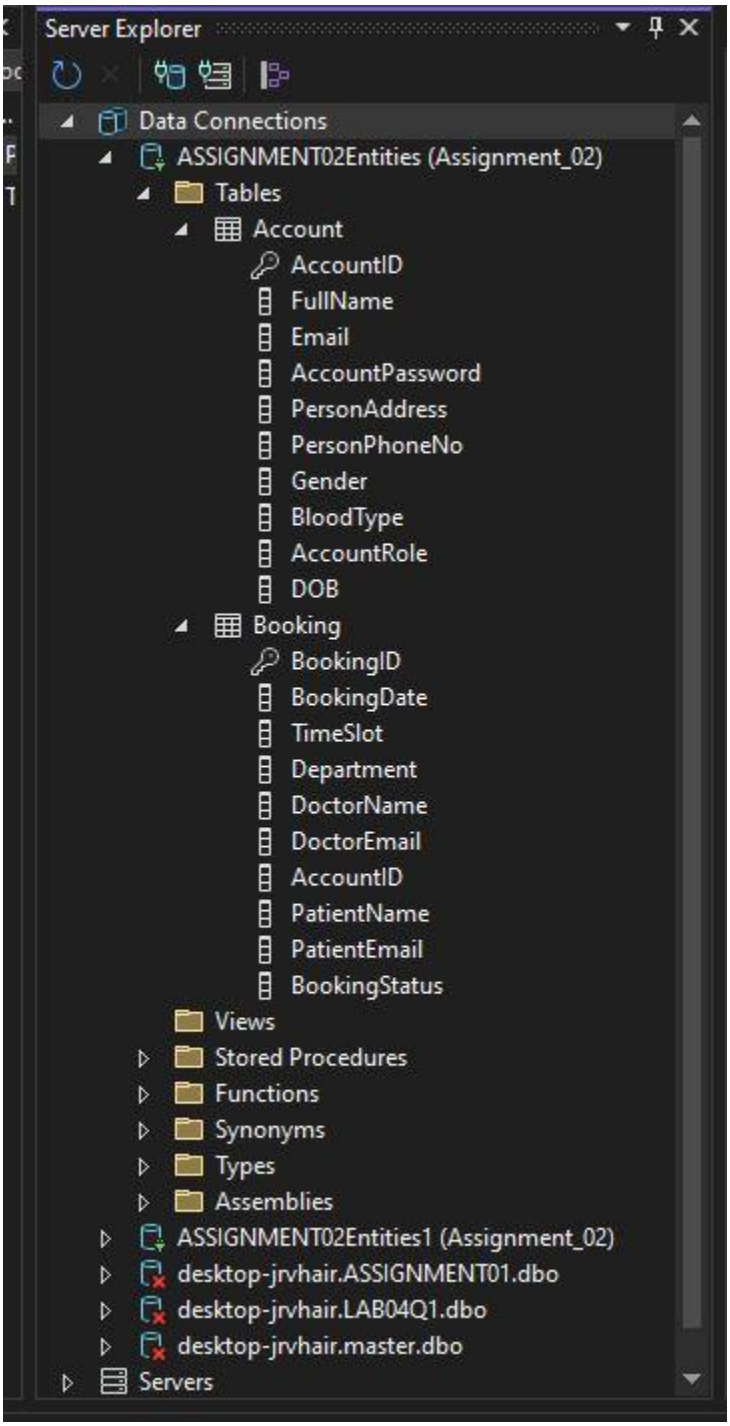
Admin Control Panel (Appointment Data) Button/Link -> The Admin User also can view the
Appointments/Booking data. They are also allowed to Create, Edit, Delete and View all the
Appointments/Booking details. The Admin User also can click on the blue button on the left below "Admin Control Panel" if they wish to return to the Admin Control Panel (Account Data) webpage.

Patient Webpage/Panel -> User will have to login with an Account Role "Patient" in order to visit the Patient webpage. Where the patients can Create, Edit, Delete and view their appointments or bookings with the hospital.

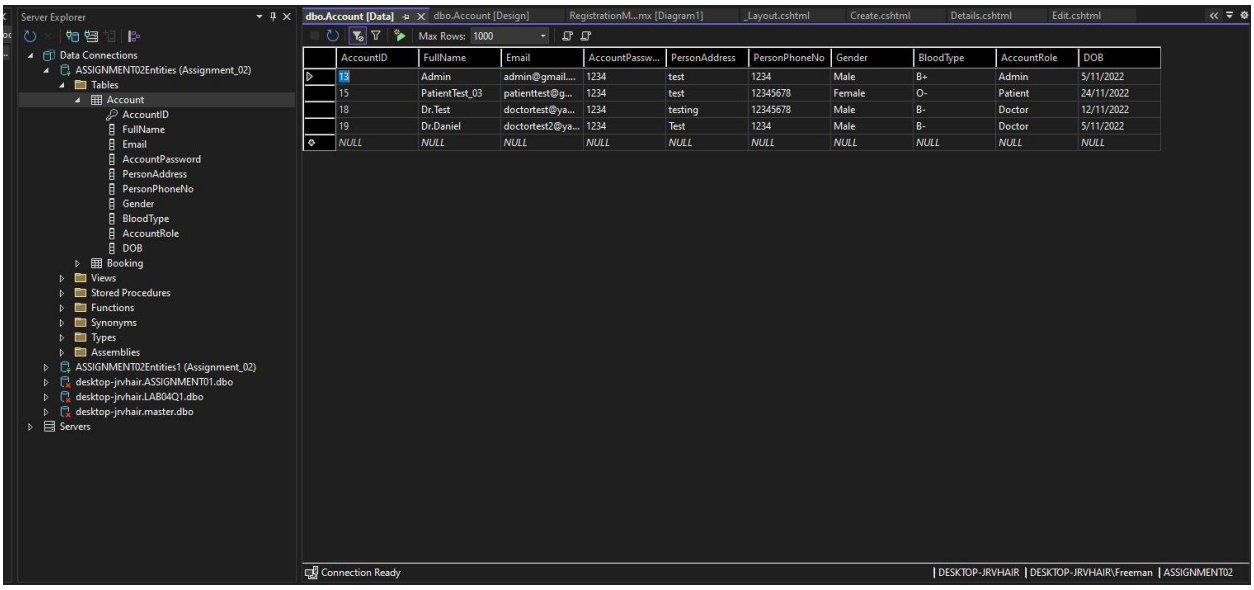
Doctor Webpage/Panel -> User will have to login with an Account Role "Doctor" to visit the doctor webpage. Where the doctor can Create, Edit, Delete and also view the patient’s appointments or booking from the hospital database.
The Doctor Account Name/ID is their First name when they register their account on the registration webpage.

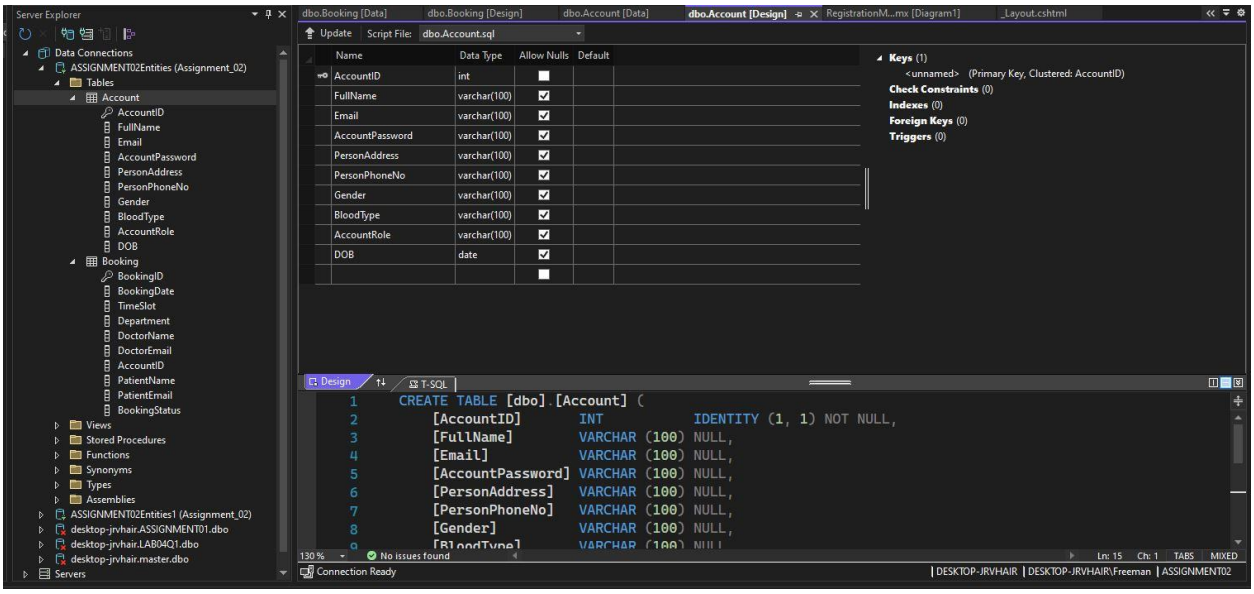
Log Out Button/Link -> User also can log out their account session by clicking on the "Log Out" Button/Link.

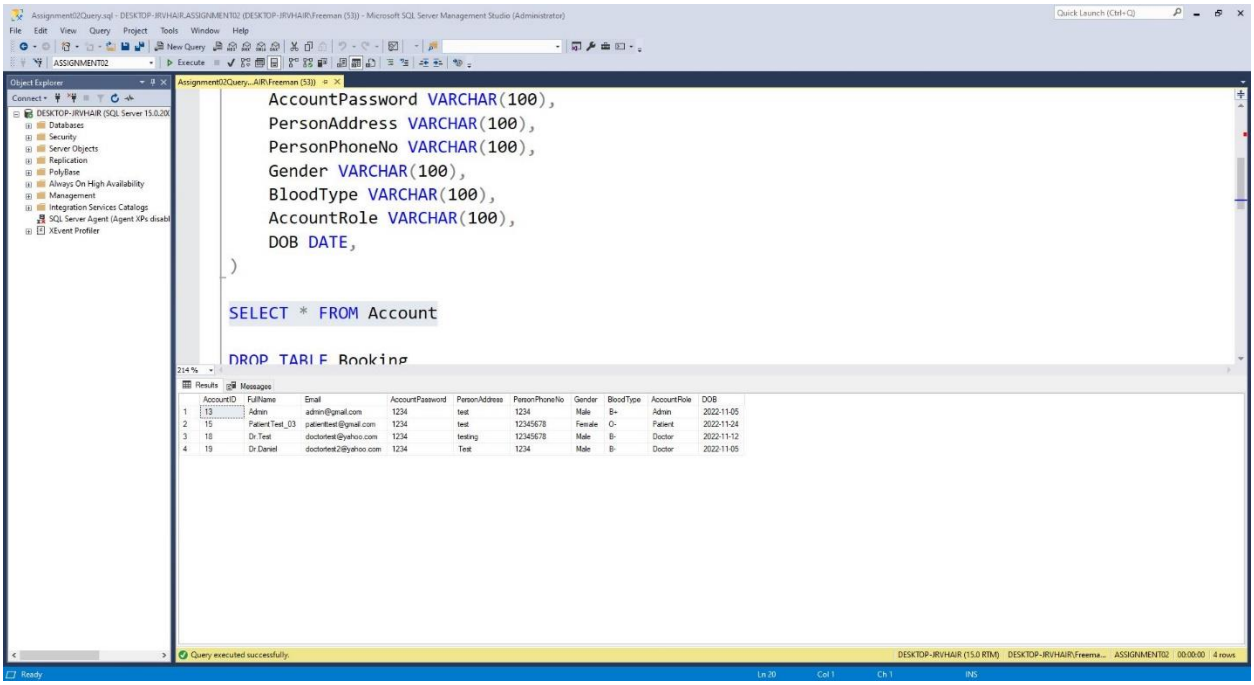
Database (Screenshot)



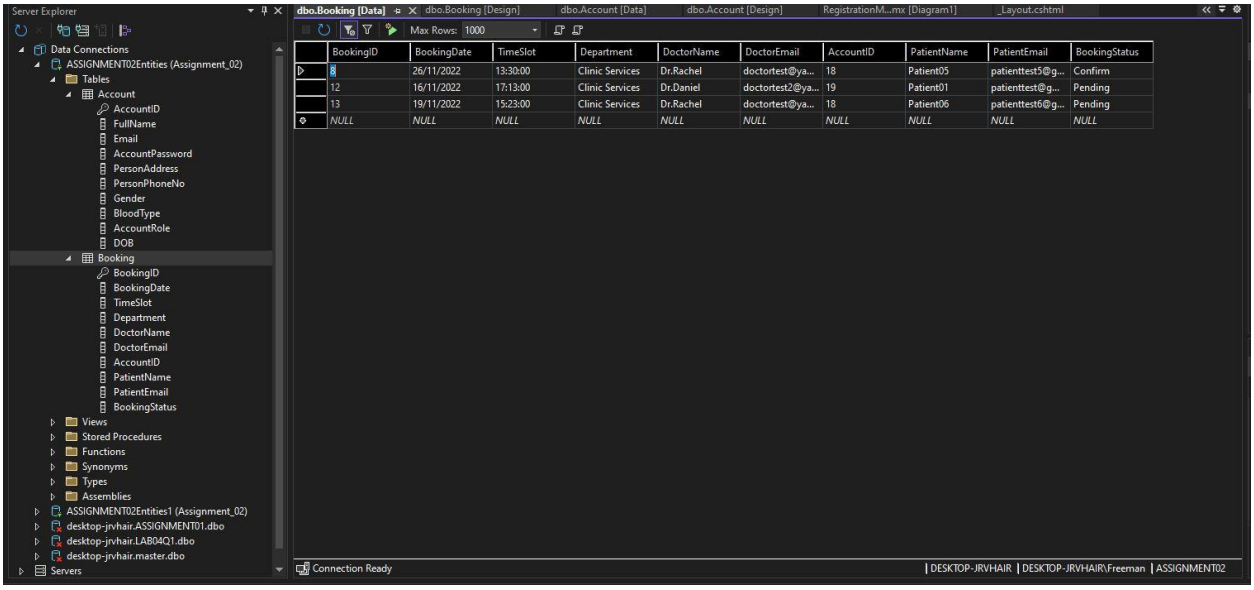
Account Database (Screenshot)

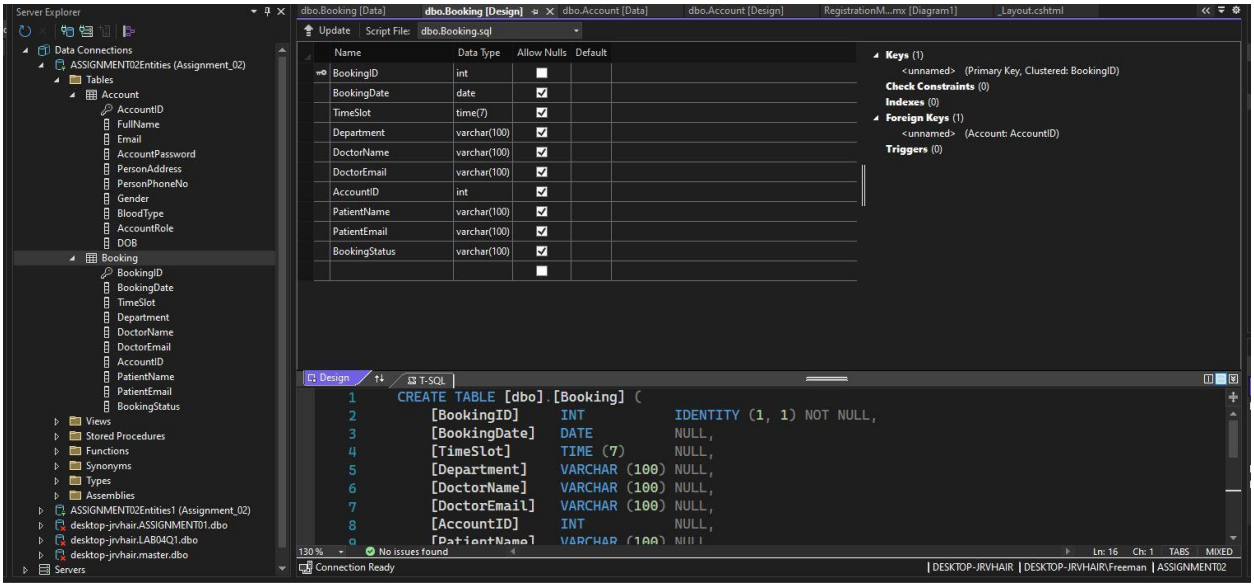


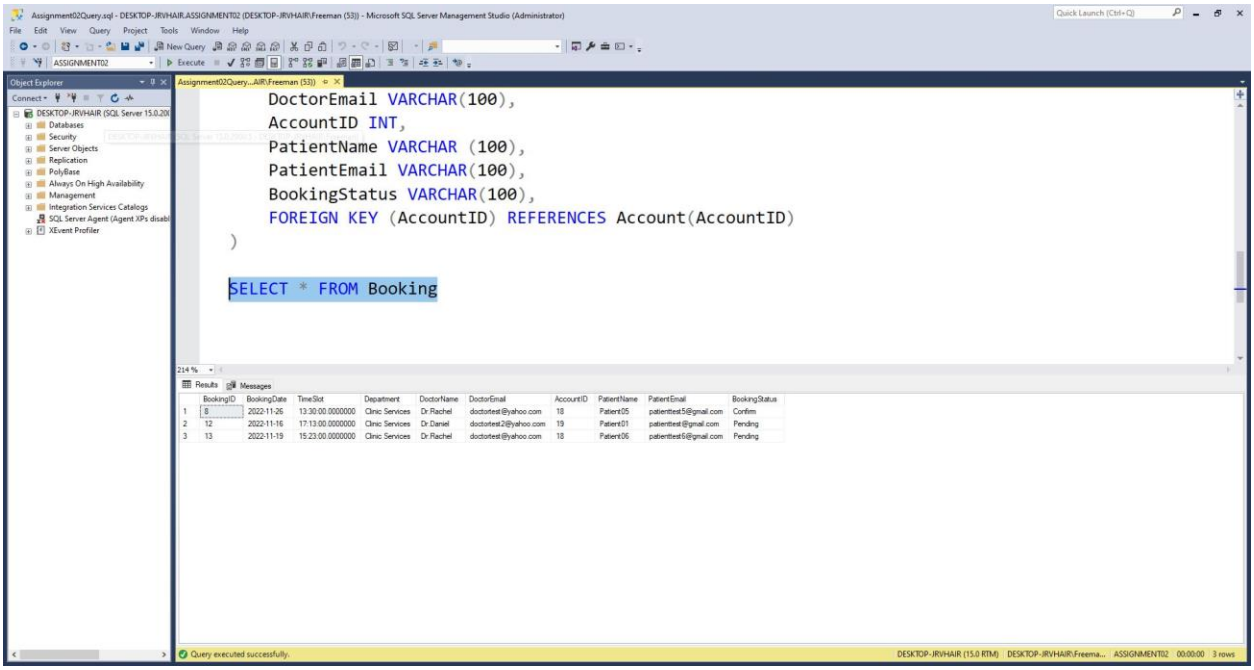




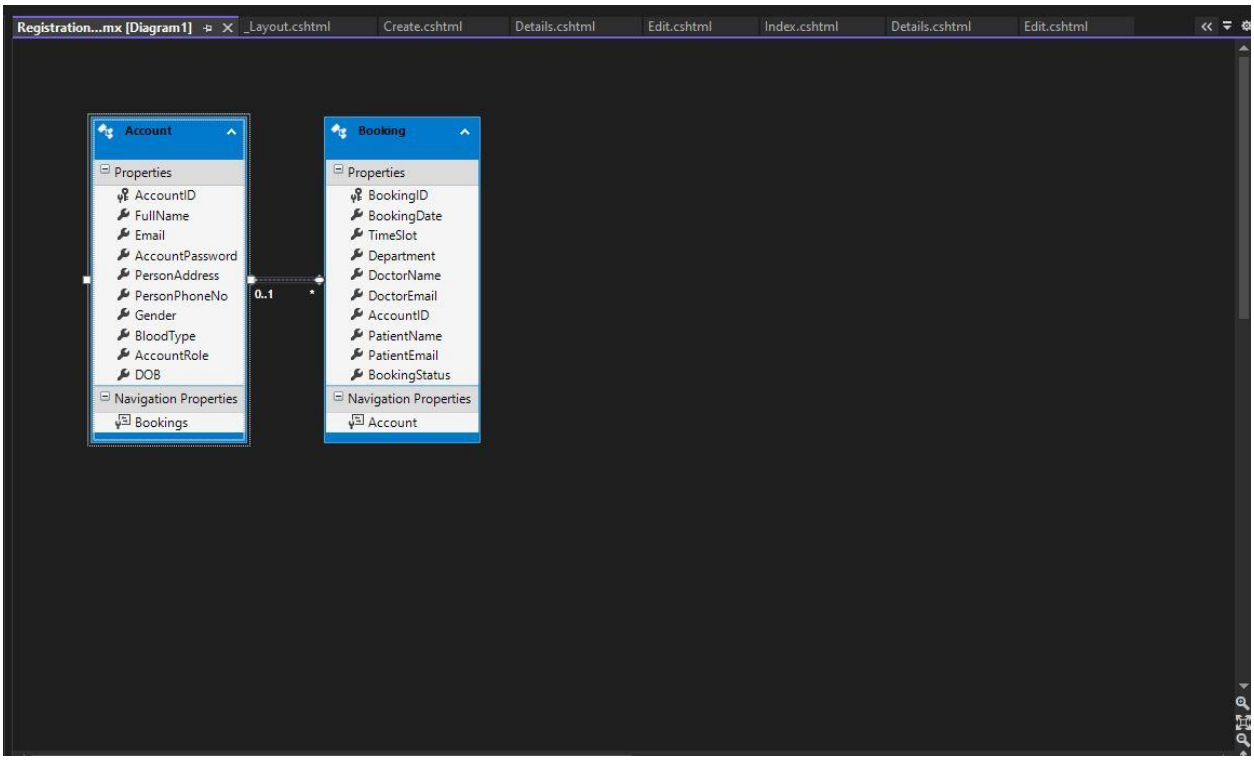
Bookings Database (Screenshot)




Server Explorer
Data Connections
ASSIGNMENT02Entities (Assignment_02)
Tables
Account
AccountID
FullName
Email
AccountPassword
PersonAddress
PersonPhoneNo
Gender
BloodType
AccountRole
DOB
Booking
BookingID
BookingDate
TimeSlot
Department
DoctorName
DoctorEmail
AccountID
PatientName
PatientEmail
BookingStatus
Views
Stored Procedures
Functions
Synonyms
Types
Assemblies
ASSIGNMENT02Entities1 (Assignment_02)
desktop-jrvhair.ASSIGNMENT01.dbo
desktop-jrvhair.LAB04Q1.dbo
desktop-jrvhair.master.dbo
Servers
dbo.Booking [Data]
dbo.Booking [Design]
dbo.Account [Data]
dbo.Account [Design]
RegistrationM...mx [Diagram1]
_Layout.cshtml
Update
Script File: dbo.Booking.sql
Name
Data Type
Allow Nulls
Default
BookingID int
BookingDate date
TimeSlot time(7)
Department varchar(100)
DoctorName varchar(100)
DoctorEmail varchar(100)
AccountID int
PatientName varchar(100)
PatientEmail varchar(100)
BookingStatus varchar(100)
Keys (1)
<unnamed> (Primary Key, Clustered: BookingID)
Check Constraints (0)
Indexes (0)
Foreign Keys (1)
<unnamed> (Account: AccountID)
Triggers (0)
Design
T-SQL
CREATE TABLE [dbo].[Booking] (
[BookingID] INT IDENTITY (1, 1) NOT NULL,
[BookingDate] DATE NULL,
[TimeSlot] TIME (7) NULL,
[Department] VARCHAR (100) NULL,
[DoctorName] VARCHAR (100) NULL,
[DoctorEmail] VARCHAR (100) NULL,
[AccountID] INT NULL,
[PatientName] VARCHAR (100) NULL
130 %
No issues found
Ln: 16 Ch: 1 TABS MIXED
Connection Ready
DESKTOP-JRVHAIR | DESKTOP-JRVHAIR\Freeman | ASSIGNMENT02


Assignment02Query.sql - DESKTOP-JRVHAIR.ASSIGNMENT02 (DESKTOP-JRVHAIR\Freeman (53)) - Microsoft SQL Server Management Studio (Administrator)
File Edit View Query Project Tools Window Help
New Query
Execute
ASSIGNMENT02
Object Explorer
Connect
DESKTOP-JRVHAIR (SQL Server 15.0.20
Databases
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabl
XEvent Profiler
Assignment02Query...AIR\Freeman (53))
DoctorEmail VARCHAR(100),
AccountID INT,
PatientName VARCHAR (100),
PatientEmail VARCHAR(100),
BookingStatus VARCHAR(100),
FOREIGN KEY (AccountID) REFERENCES Account(AccountID)
)
SELECT * FROM Booking
214 %
Results
Messages
BookingID BookingDate TimeSlot Department DoctorName DoctorEmail AccountID PatientName PatientEmail BookingStatus
1 8 2022-11-26 13:30:00.0000000 Clinic Services Dr Rachel doctortest@yahoo.com 18 Patient05 patienttest5@gmail.com Confirm
2 12 2022-11-16 17:13:00.0000000 Clinic Services Dr Daniel doctortest2@yahoo.com 19 Patient01 patienttest@gmail.com Pending
3 13 2022-11-19 15:23:00.0000000 Clinic Services Dr Rachel doctortest@yahoo.com 18 Patient06 patienttest6@gmail.com Pending
Query executed successfully.
DESKTOP-JRVHAIR (15.0 RTM) DESKTOP-JRVHAIR\Freema... ASSIGNMENT02 00:00:00 3 rows

ADO.net Entity Data Model (Screenshot) (Database)



Database SQL Source Code (Query)

```
CREATE DATABASE ASSIGNMENT02
GO
USE ASSIGNMENT02
GO

CREATE TABLE Account
(
	AccountID INT IDENTITY PRIMARY KEY,
	FullName VARCHAR (100),
	Email VARCHAR(100),
	AccountPassword VARCHAR(100),
	PersonAddress VARCHAR(100),
	PersonPhoneNo VARCHAR(100),
	Gender VARCHAR(100),
	BloodType VARCHAR(100),
	AccountRole VARCHAR(100),
	DOB DATE,
)

SELECT * FROM Account

DROP TABLE Booking

CREATE TABLE Booking
(
```

```
        BookingID INT IDENTITY PRIMARY KEY,
        BookingDate DATE,
        TimeSlot TIME,
        Department VARCHAR(100),
        DoctorName VARCHAR(100),
        DoctorEmail VARCHAR(100),
        AccountID INT,
        PatientName VARCHAR (100),
        PatientEmail VARCHAR(100),
        BookingStatus VARCHAR(100),
        FOREIGN KEY (AccountID) REFERENCES Account(AccountID)
)

SELECT * FROM Booking
```

Account Model Source Code (For the Account Database) (Model/Account.cs)

- The Account Model is created using the ADO.net Entity Data Model. That is used to getter and setter for the user input from the webpage to the database itself. Where this Account Model also generated a new List for Bookings since the AccountID is a foreign for the Bookings database. Meaning both of this Account and Model will be linked to each other using the foreign key at the Entity data model ADO.net. For me I use `using System.ComponentModel.DataAnnotations;` System is to do the “Display”, “DataType” and Required for webpage user interface form that will be display base on the things I set for the classes and it will be display in the UI and also Set the Requirement so that the user will not leave the webpage form in empty/null input.

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated from a template.
//
//     Manual changes to this file may cause unexpected behavior in your
application.
//     Manual changes to this file will be overwritten if the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace Assignment_02.Models
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;

    public partial class Account
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public Account()
        {
```

```
            this.Bookings = new HashSet<Booking>();
        }

        [Key]
        public int AccountID { get; set; }

        [Required(ErrorMessage = "This Field Is Required To Be Filled.")]
        [Display(Name = "Full Name")]
        public string FullName { get; set; }

        [Required(ErrorMessage = "This Field Is Required To Be Filled.")]
        [Display(Name = "Email")]
        public string Email { get; set; }

        [Required(ErrorMessage = "This Field Is Required To Be Filled.")]
        [Display(Name = "Password")]
        [DataType(DataType.Password)]
        public string AccountPassword { get; set; }

        [Required(ErrorMessage = "This Field Is Required To Be Filled.")]
        [Display(Name = "Address")]
        public string PersonAddress { get; set; }

        [Required(ErrorMessage = "This Field Is Required To Be Filled.")]
        [Display(Name = "Phone Number")]
        public string PersonPhoneNo { get; set; }

        [Required(ErrorMessage = "This Field Is Required To Be Filled.")]
        [Display(Name = "Gender")]
        public string Gender { get; set; }

        [Required(ErrorMessage = "This Field Is Required To Be Filled.")]
        [Display(Name = "Blood Type")]
        public string BloodType { get; set; }

        [Required(ErrorMessage = "This Field Is Required To Be Filled.")]
        [Display(Name = "Role")]
        public string AccountRole { get; set; }

        [Required(ErrorMessage = "This Field Is Required To Be Filled.")]
        [Display(Name = "Date Of Birth")]
        [DataType(DataType.Date)]
        public Nullable<System.DateTime> DOB { get; set; }

        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<Booking> Bookings { get; set; }
    }
}
```

Booking Model Source Code (For the Booking Database) (Model/Booking.cs)

The Booking Model is created using the ADO.net Entity Data Model. That is used to getter and setter for the user input from the webpage to the database itself. Since AccountID is a foreign key for the Booking database it will have the `public virtual Account Account { get; set; }` Getter and Setter to get the data from the Account.cs class also so that they will be able to get the data from the database base on the getter, setter and the foreign key. For me I use `using System.ComponentModel.DataAnnotations;` System is to do the “Display”, “DataType” and Required for webpage user interface form that will be display base on the things I set for the classes and it will be display in the UI. and also Set the Requirement so that the user will not leave the webpage form in empty/null input.

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated from a template.
//
//     Manual changes to this file may cause unexpected behavior in your
application.
//     Manual changes to this file will be overwritten if the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace Assignment_02.Models
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;

    public partial class Booking
    {
        [Key]
        public int BookingID { get; set; }

        [Required(ErrorMessage = "This Field Is Required To Be Filled.")]
        [Display(Name = "Appointment Date")]
        [DataType(DataType.Date)]
        public Nullable<System.DateTime> BookingDate { get; set; }

        [Required(ErrorMessage = "This Field Is Required To Be Filled.")]
        [Display(Name = "Appointment Time")]
        [DataType(DataType.Time)]
        public Nullable<System.TimeSpan> TimeSlot { get; set; }

        //[Required(ErrorMessage = "This Field Is Required To Be Filled.")]
        [Display(Name = "Hospital Department")]
        public string Department { get; set; }

        [Display(Name = "Doctor Name")]
        public string DoctorName { get; set; }

        [Display(Name = "Doctor Email")]
        public string DoctorEmail { get; set; }

        [Display(Name = "Doctor Account Name/ID")]
        public Nullable<int> AccountID { get; set; }
```

```
        [Display(Name = "Patient Name")]
        public string PatientName { get; set; }

        [Display(Name = "Patient Email")]
        public string PatientEmail { get; set; }

        [Display(Name = "Status")]
        public string BookingStatus { get; set; }

        public virtual Account Account { get; set; }
    }
}
```

HomeController Source Code (Controller/HomeController.cs)

- Highlight in yellow: Basically, the function of the ActionResult to return to the homepage (Home/Index.cshtml)
- Highlight in light blue: Basically, the function of the ActionResult to return to the About Page (Home/About.cshtml)
- Highlight in green: Basically, the function of the ActionResult to return to the Contact Us Page (Home/Contact.cshtml)the error message to the user.
- Highlight in pink: Is Basically, the function of the ActionResult to return to the Services Page (Home/Services.cshtml)the error message to the user.
- Highlight in Red: Is Basically, the function of the ActionResult to return to the Registration Page (Home/Registration.cshtml)the error message to the user.
- Highlight in Light Gray: Is the Registration Function of the ActionResult to store the account registration input to the database. It also does check if Any of the account have the same email. If it has the same email, it will show the error message to the user. Else it will just add the input to the database and return the user back to the home/login page.
- Highlight in Dark Gray: Is Basically, the function of the ActionResult for the Log Out of the account from the user. It will redirect the user to the homepage once they are log out from their account.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Principal;
using System.Web;
using System.Web.Mvc;
using System.Web.Security;
using Assignment_02.Models;

namespace Assignment_02.Controllers
{
    public class HomeController : Controller
    {
        ASSIGNMENT02Entities db = new ASSIGNMENT02Entities();

        [AllowAnonymous]
```

```
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult About()
        {
            ViewBag.Message = "About us";

            return View();
        }

        public ActionResult Contact()
        {
            ViewBag.Message = "contact us";

            return View();
        }

        public ActionResult Services()
        {
            ViewBag.Message = "Services/Departments/Facility";

            return View();
        }

        public ActionResult Registration()
        {
            //ViewBag.Message = "Registration Page";

            return View();
        }


        /*[HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Registration([Bind(Include =
"AccountID,FullName,Email,AccountPassword,reAccountPassword,PersonAddress,PersonPh
oneNo,Gender,BloodType,AccountRole,DOB")] Account account)
        {
            if (ModelState.IsValid)
            {
                db.Accounts.Add(account);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            return View(account);
        }*/

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Registration(Account accountDB)
        {
            if (ModelState.IsValid)
            {
                if (db.Accounts.Any(x => x.Email == accountDB.Email))
                {
```

```
                    ViewBag.SameEmailMsg = ("<script>alert('Email has been use
before!.')</script>");
                    return View();
                }
                else
                {
                    db.Accounts.Add(accountDB);
                    db.SaveChanges();

                    Session["AccountIDSS"] = accountDB.AccountID.ToString();
                    Session["EmailSS"] = accountDB.Email.ToString();

                    ViewBag.CreateAccountMsg = ("<script>alert('Account Create
Successfully!.')</script>");

                    return RedirectToAction("Index", "Home");
                }
            }
            return View(accountDB);
        }

        public ActionResult Logout()
        {
            Session.Clear();
            return RedirectToAction("Index", "Home");
        }
    }
 }
```

LoginController (Controller/LoginController.cs)

- Highlight in yellow: Basically, the function of the ActionResult to return to the Login Page (Login/Login.cshtml), Using [HttpGet] to basically retrieve the data from the server database. (Account Database)

- Highlight in light blue: Basically, the function of the ActionResult that using [HttpPost] is to Post/Re-submit the Account that is login by the user from there we will do a checking to check if the Account exist on the database. If it is existed on the database, it will successfully login the user and redirect back the user to their webpage. But if the account does not exist it will show an alert/error message to the user. For this function we also do check the user AccountRole by using the switch case method If the AccountRole is admin, doctor, patient (It will return the user to the specific webpage base on their role). If not, it will just return back the user to the Home page.

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

```
using Assignment_02.Models;

namespace Assignment_02.Controllers
{
    public class LoginController : Controller
    {
        ASSIGNMENT02Entities db = new ASSIGNMENT02Entities();

        // GET: Login

        [HttpGet]
        public ActionResult Login()
        {
            return View();
        }

        /*
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Login(Account accountDB)
        {
            var checklogin = db.Accounts.Where(i =>
i.Email.Equals(accountDB.Email) &&
i.AccountPassword.Equals(accountDB.AccountPassword) &&
i.AccountRole.Equals(accountDB.AccountRole)).FirstOrDefault();

            if (checklogin != null)
            {
                Session["AccountIDSS"] = accountDB.AccountID.ToString();
                Session["EmailSS"] = accountDB.Email.ToString();
                Session["AccountSS"] = accountDB.AccountRole.ToString();

                if (accountDB.AccountRole == "Admin")
                {
                    return RedirectToAction("Index", "Admin");
                }
                else if(accountDB.AccountRole == "Doctor")
                {
                    return RedirectToAction("Index", "Home");
                }
                else if(accountDB.AccountRole == "Patient")
                {
                    return RedirectToAction("Index", "Patients");
                }
                else
                {
                    ViewBag.LoginMsg = ("<script>alert('Invalid Role, Please try
again!.')</script>");
                }
                //return RedirectToAction("Index", "Home");
            }
            else
            {
                ViewBag.LoginMsg = ("<script>alert('Wrong Email or Password or
Account Role!.')</script>");
                //ViewBag.Notification = "Wrong Email or Password or Account Role,
Please Try Again!";
            }
```

```
            return View();
        }*/

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Login(Account accountDB)
        {
            var checklogin = db.Accounts.Where(i =>
i.Email.Equals(accountDB.Email) &&
i.AccountPassword.Equals(accountDB.AccountPassword) &&
i.AccountRole.Equals(accountDB.AccountRole)).FirstOrDefault();

            if (checklogin != null)
            {
                Session["Login"] = accountDB;
                Session["EmailSS"] = accountDB.Email;
                Session["AccountSS"] = accountDB.AccountRole;

                switch (accountDB.AccountRole)
                {
                    case "Admin":
                        return RedirectToAction("Index", "Admin");

                    case "Patient":
                        return RedirectToAction("Index", "Patient");

                    case "Doctor":
                        return RedirectToAction("Index", "Doctor");

                    default:
                        return RedirectToAction("Index", "Home");
                }
            }
            else
            {
                ViewBag.LoginMsg = ("<script>alert('Wrong Email or Password or
Account Role!.')</script>");
                //ViewBag.Notification = "Wrong Email or Password or Account Role,
Please Try Again!";
            }
            return View();
        }
    }
}
```

AdminController (Controller/AdminController.cs) (Admin Control Panel For Account)

- Highlight in Brown: Basically, the function of the ActionResult to get the list of the user data and return it back to the user.
- Highlight in Yellow: Basically, the function of the ActionResult to return the user to the Admin Control Panel Page which is (Admin/Index.cshtml). Where the

admin can Create, Edit, View and Delete the User Account from the Control Panel.But it will do a validation check to see if the Account Session Login is Admin role. If it is not an Admin role account, it will return the user to the Login page.

- Highlight in Teal Color: Basically, the function of the ActionResult for the admin to view the Account Data Details on the Details Page (Admin/Details.cshtml). I also did a validation check to see if the Account Session Login is Admin role. If it is not an Admin role account, it will return the user to the login page.

- Highlight in Light Blue: Basically, the function of the ActionResult for the admin to create the account for the user if needed. The account that admin created will store at the Account Database same as the registration page. It also did a validation check to see if the Account Session Login is Admin role. If it is not an Admin role account, it will return the user to the login page. And it also check if the account email it used by other user so that the user won't be able to create the account with the same email.

- Highlight in Green: Basically, the function of the ActionResult for the admin to Edit the user account by updating their account details using the Edit webpage (Admin/Edit.cshtml) if needed. Once is update on the webpage it will also update it on the "Account" database automatically. It also did a validation check to see if the Account Session Login is Admin role. If it is not an Admin role account, it will return the user to the login page. And it also checks if the account email it used by other user so that the admin won't be able to update/edit the account with the same email.

- Highlight in Pink: Basically, the function of the ActionResult for the admin to Delete the user account base on their AccountID at the Delete Webpage (Admin/Delete.cshtml) It will also do a confirmation check to see if the admin wishes to delete this user account. Lastly It also did a validation check to see if the Account Session Login is Admin role. If it is not an Admin role account, it will return the user to the login page.

- Highlight in Red: It basically, the Dispose method that will release the resources. If it is true, it will release both managed and unmanaged resources; if it is false, it will only release unmanaged resources.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Security.Principal;
```

```
using System.Web;
using System.Web.Mvc;
using Assignment_02.Models;

namespace Assignment_02.Controllers
{
    public class AdminController : Controller
    {
        ASSIGNMENT02Entities db = new ASSIGNMENT02Entities();

        public ActionResult getUserList()
        {
            var users = db.Accounts.ToList();
            return View(users);
        }

        // GET: Admin
        public ActionResult Index()
        {
            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];

                if (obj.AccountRole != "Admin")
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }

        return View(db.Accounts.ToList());
        }

        /*public ActionResult Index(Account accountDB)
        {
            var checklogin = db.Accounts.Where(i => i.Email.Equals(accountDB.Email)
&& i.AccountPassword.Equals(accountDB.AccountPassword) &&
i.AccountRole.Equals(accountDB.AccountRole)).FirstOrDefault();

            if (checklogin != null)
            {
                Session["AccountIDSS"] = accountDB.AccountID.ToString();
                Session["EmailSS"] = accountDB.Email.ToString();
```

```
            Session["AccountSS"] = accountDB.AccountRole.ToString();

            if (accountDB.AccountRole == "Admin")
            {
                return RedirectToAction("Index", "Admin");
            }
            else if (accountDB.AccountRole == "Doctor")
            {
                return RedirectToAction("Index", "Home");
            }
            else if (accountDB.AccountRole == "Patient")
            {
                return RedirectToAction("Index", "Patients");
            }
            else
            {
                ViewBag.LoginMsg = ("<script>alert('Invalid Role, Please try
again!.')</script>");
            }
            //return RedirectToAction("Index", "Home");
        }
        else
        {
            ViewBag.LoginMsg = ("<script>alert('Wrong Email or Password or Account
Role!.')</script>");
            //ViewBag.Notification = "Wrong Email or Password or Account Role,
Please Try Again!";
        }
        return View(db.Accounts.ToList());
    }*/

    // GET: Admin/Details/5
    public ActionResult Details(int? id)
    {

        if (Session["Login"] != null)
        {
            Models.Account obj = (Models.Account)Session["Login"];

            if (obj.AccountRole != "Admin")
            {
                return RedirectToAction("Login", "Login");
            }
        }
        else
        {
```

```
            return RedirectToAction("Login", "Login");
        }

        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        Account account = db.Accounts.Find(id);
        if (account == null)
        {
            return HttpNotFound();
        }
        return View(account);
    }


    // GET: Admin/Create
    public ActionResult Create()
    {
        if (Session["Login"] != null)
        {
            Models.Account obj = (Models.Account)Session["Login"];

            if (obj.AccountRole != "Admin")
            {
                return RedirectToAction("Login", "Login");
            }
        }
        else
        {
            return RedirectToAction("Login", "Login");
        }

        return View();
    }

    // POST: Admin/Create
    // To protect from overposting attacks, enable the specific properties you want to
bind to, for
    // more details see https://go.microsoft.com/fwlink/?LinkId=317598.

    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Create(Account accountDB)
    {
        if (Session["Login"] != null)
```

```
            {
                Models.Account obj = (Models.Account)Session["Login"];

                if (obj.AccountRole != "Admin")
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }

            if (ModelState.IsValid)
            {
                if (db.Accounts.Any(x => x.Email == accountDB.Email))
                {
                    ViewBag.SameEmailMsg = ("<script>alert('Email has been use
before!.')</script>");
                    return View();
                }
                else
                {
                    db.Accounts.Add(accountDB);
                    db.SaveChanges();

                    Session["AccountIDSS"] = accountDB.AccountID.ToString();
                    Session["EmailSS"] = accountDB.Email.ToString();

                    ViewBag.CreateAccountMsg = ("<script>alert('Account Create
Successfully!.')</script>");

                    return RedirectToAction("Index", "Admin");
                }
            }
            return View(accountDB);
        }


        // GET: Admin/Edit/5
        public ActionResult Edit(int id)
        {
            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];
```

```
            if (obj.AccountRole != "Admin")
            {
                return RedirectToAction("Login", "Login");
            }
        }
        else
        {
            return RedirectToAction("Login", "Login");
        }

        var edit = db.Accounts.Where(model => model.AccountID ==
id).FirstOrDefault();

        return View(edit);
    }

    /*
    // POST: Admin/Edit/5
    // To protect from overposting attacks, enable the specific properties you want to
bind to, for
    // more details see https://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Edit([Bind(Include =
"AccountID,FullName,Email,AccountPassword,PersonAddress,PersonPhoneNo,Gend
er,BloodType,AccountRole,DOB")] Account account)
    {
        if (ModelState.IsValid)
        {
            db.Entry(account).State = EntityState.Modified;
            db.SaveChanges();
            return RedirectToAction("Index");
        }
        return View(account);
    }*/


    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Edit(Account accountDB)
    {
        if (Session["Login"] != null)
        {
            Models.Account obj = (Models.Account)Session["Login"];

            if (obj.AccountRole != "Admin")
```

```
            {
                return RedirectToAction("Login", "Login");
            }
        }
        else
        {
            return RedirectToAction("Login", "Login");
        }


        if (ModelState.IsValid)
        {
            if (db.Accounts.Any(x => x.Email == accountDB.Email))
            {
                ViewBag.SameEmailMsg = ("<script>alert('Email has been use
before!.')</script>");
                return View();
            }
            else
            {
                db.Entry(accountDB).State = EntityState.Modified;
                int edit = db.SaveChanges();

                Session["AccountIDSS"] = accountDB.AccountID.ToString();
                Session["EmailSS"] = accountDB.Email.ToString();

                if (edit > 0)
                {
                    ViewBag.EditMsg2 = ("<script>alert('Edit Account
Successfully!.')</script>");

                    return RedirectToAction("Index", "Admin");
                }
                else
                {
                    ViewBag.EditMsg = ("<script>alert('Something is wrong with the
input...')</script>");
                }
            }
        }
        return View(accountDB);
    }


    // GET: Admin/Delete/5
    public ActionResult Delete(int? id)
```

```
        {
            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];

                if (obj.AccountRole != "Admin")
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }

            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Account account = db.Accounts.Find(id);
            if (account == null)
            {
                return HttpNotFound();
            }
            return View(account);
        }


        // POST: Admin/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];

                if (obj.AccountRole != "Admin")
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }
```

```
            Account account = db.Accounts.Find(id);
            db.Accounts.Remove(account);
            db.SaveChanges();
            return RedirectToAction("Index");
        }


        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

AdminTwoController (Controller/AdminTwoController.cs) (Admin Control Panel for Bookings/Appointment)

- Highlight in Yellow: Basically, the function of the ActionResult to return the user to the Admin Control Panel Page which is (AdminTwo/Index.cshtml). Where the admin can Create, Edit, View and Delete the Bookings/Appointment from the Control Panel. But it will do a validation check to see if the Account Session Login is Admin role. If it is not an Admin role account, it will return the user to the Login page.

- Highlight in Teal Color: Basically, the function of the ActionResult for the admin to view the Bookings/Appointment Details on the Details Page (AdminTwo/Details.cshtml) from the Bookings database itself. I also did a validation check to see if the Account Session Login is Admin role. If it is not an Admin role account, it will return the user to the login page.

- Highlight in Light Blue: Basically, the function of the ActionResult for the admin to create an Appointment/Booking for the user if needed from the Admin Control Panel Page (AdminTwo/Create.cshtml). The Appointment/Booking that admin created will store at the Bookings Database. It also did a validation check to see if the Account Session Login is Admin role. If it is not an Admin role account, it will return the user to the login page.

- Highlight in Green: Basically, the function of the ActionResult for the admin to Edit the user appointment/booking by updating their appointment/booking details using the Edit webpage (AdminTwo/Edit.cshtml) if needed. Once is update on the webpage it will also update it on the “Bookings” database automatically. It also

did a validation check to see if the Account Session Login is Admin role. If it is not an Admin role account, it will return the user to the login page.

- Highlight in Pink: Basically, the function of the ActionResult for the admin to Delete the user Appointment/Booking base on their BookingID at the Delete Webpage (AdminTwo/Delete.cshtml) It will also do a confirmation check to see if the admin wishes to delete this user appointment/booking. Lastly It also did a validation check to see if the Account Session Login is Admin role. If it is not an Admin role account, it will return the user to the login page.

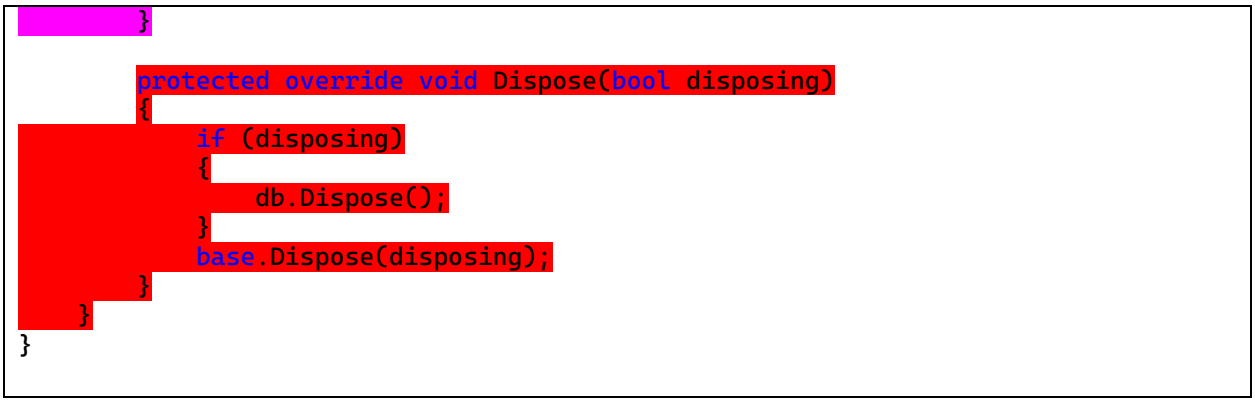
- Highlight in Red: It basically, the Dispose method that will release the resources. If it is true, it will release both managed and unmanaged resources; if it is false, it will only release unmanaged resources.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using Assignment_02.Models;

namespace Assignment_02.Controllers
{
    public class AdminTwoController : Controller
    {
        private ASSIGNMENT02Entities db = new ASSIGNMENT02Entities();

        // GET: AdminTwo
        public ActionResult Index()
        {
            var bookings = db.Bookings.Include(b => b.Account);

            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];

                if (obj.AccountRole != "Admin")
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }

            return View(bookings.ToList());
        }

        // GET: AdminTwo/Details/5
        public ActionResult Details(int? id)
        {
```

```
            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];

                if (obj.AccountRole != "Admin")
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }

            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Booking booking = db.Bookings.Find(id);
            if (booking == null)
            {
                return HttpNotFound();
            }
            return View(booking);
        }

        // GET: AdminTwo/Create
        public ActionResult Create()
        {
            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];

                if (obj.AccountRole != "Admin")
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }

            ViewBag.AccountID = new SelectList(db.Accounts, "AccountID",
"FullName");
            return View();
        }

        // POST: AdminTwo/Create
        // To protect from overposting attacks, enable the specific properties you
want to bind to, for
        // more details see https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include =
"BookingID,BookingDate,TimeSlot,Department,DoctorName,DoctorEmail,AccountID,Patien
tName,PatientEmail,BookingStatus")] Booking booking)
        {
```

```
            if (ModelState.IsValid)
            {
                db.Bookings.Add(booking);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.AccountID = new SelectList(db.Accounts, "AccountID",
"FullName", booking.AccountID);
            return View(booking);
        }

        // GET: AdminTwo/Edit/5
        public ActionResult Edit(int? id)
        {
            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];

                if (obj.AccountRole != "Admin")
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }

            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Booking booking = db.Bookings.Find(id);
            if (booking == null)
            {
                return HttpNotFound();
            }
            ViewBag.AccountID = new SelectList(db.Accounts, "AccountID",
"FullName", booking.AccountID);
            return View(booking);
        }

        // POST: AdminTwo/Edit/5
        // To protect from overposting attacks, enable the specific properties you
want to bind to, for
        // more details see https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include =
"BookingID,BookingDate,TimeSlot,Department,DoctorName,DoctorEmail,AccountID,Patien
tName,PatientEmail,BookingStatus")] Booking booking)
        {
            if (ModelState.IsValid)
            {
                db.Entry(booking).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
```

```
            }
            ViewBag.AccountID = new SelectList(db.Accounts, "AccountID",
"FullName", booking.AccountID);
            return View(booking);
        }

        // GET: AdminTwo/Delete/5
        public ActionResult Delete(int? id)
        {
            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];

                if (obj.AccountRole != "Admin")
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }

            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Booking booking = db.Bookings.Find(id);
            if (booking == null)
            {
                return HttpNotFound();
            }
            return View(booking);
        }

        // POST: AdminTwo/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];

                if (obj.AccountRole != "Admin")
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }

            Booking booking = db.Bookings.Find(id);
            db.Bookings.Remove(booking);
            db.SaveChanges();
            return RedirectToAction("Index");
```



```
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

PatientController (Controller/PatientController.cs) Action Function for Patient Booking the Appointments

- Highlight in Yellow: Basically, the function of the ActionResult to return the user to the Patient Booking/Appointment Page which is (Patient/Index.cshtml). Where the Patients can Create, Edit, View and Delete the Bookings/Appointment from the Appointment/Booking webpage. But it will also do a validation check to see if the Account Session Login is Admin or Patient role. If it is not an Admin or Patient role account, it will redirect the user to the Login page.

- Highlight in Teal Color: Basically, the function of the ActionResult for the Patient to view the Bookings/Appointment Details on the Details Page (Patient/Details.cshtml) from the Bookings database itself. I also did a validation check to see if the Account Session Login is Admin or Patient role. If it is not an Admin or Patient role account, it will return the user to the login page.

- Highlight in Light Blue: Basically, the function of the ActionResult for the Patients to create or add an Appointment/Booking for themselves at the Appointments/Booking Webpage (Patient/Create.cshtml). The Appointment/Booking that the patient created will be store at the Bookings Database automatically. It also did a validation check to see if the Account Session Login is Admin or Patient role. If it is not an Admin or Patient role account, it will redirect the user to the login page

- Highlight in Green: Basically, the function of the ActionResult for the Patients to Edit the own appointment/booking by updating their appointment/booking details using the Edit webpage (Patient/Edit.cshtml). Once is update on the webpage “Edit” it will also update it on the “Bookings” database automatically. It also did a validation check to see if the Account Session Login is Admin or Patient role. If it is not an Admin or Patient role account, it will redirect the user to the login page

- Highlight in Pink: Basically, the function of the ActionResult for the Patients to Delete their Appointment/Booking base on their BookingID at the Delete Webpage (Patient/Delete.cshtml). It will also do a confirmation check to see if the

patient wishes to delete this appointment/booking. It will also automatically delete it from the database once the user wishes to delete it. Lastly It also did a validation check to see if the Account Session Login is Admin or Patient role. If it is not an Admin or Patient role account, it will return the user to the login page.

- Highlight in Red: It basically, the Dispose method that will release the resources. If it is true, it will release both managed and unmanaged resources; if it is false, it will only release unmanaged resources.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using Assignment_02.Models;

namespace Assignment_02.Controllers
{
    public class PatientController : Controller
    {
        ASSIGNMENT02Entities db = new ASSIGNMENT02Entities();

        // GET: Patient
        public ActionResult Index()
        {
                var bookings = db.Bookings.Include(b => b.Account);

                if (Session["Login"] != null)
                {
                    Models.Account obj = (Models.Account)Session["Login"];

                    if (obj.AccountRole != "Patient" && obj.AccountRole != 
"Admin")
                    {
                        return RedirectToAction("Login", "Login");
                    }
                }
                else
                {
                    return RedirectToAction("Login", "Login");
                }

                return View(bookings.ToList());
        }

        // GET: Patient/Details/5
        public ActionResult Details(int? id)
        {
            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];

                if (obj.AccountRole != "Patient" && obj.AccountRole != "Admin")
```

```
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }


            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Booking booking = db.Bookings.Find(id);
            if (booking == null)
            {
                return HttpNotFound();
            }
            return View(booking);
        }

        // GET: Patient/Create
        public ActionResult Create()
        {
            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];

                if (obj.AccountRole != "Patient" && obj.AccountRole != "Admin")
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }


            ViewBag.AccountID = new SelectList(db.Accounts, "AccountID",
"FullName");
            return View();
        }

        // POST: Patient/Create
        // To protect from overposting attacks, enable the specific properties you
want to bind to, for
        // more details see https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Booking booking)
        {
            if (ModelState.IsValid)
            {
                if (db.Bookings.Any(x => x.PatientEmail == booking.PatientEmail))
                {
```

```
                    ViewBag.SameEmailBookingMsg = ("<script>alert('Email has been
use before!.')</script>");
                    return View();
                }
                else
                {
                    db.Bookings.Add(booking);
                    db.SaveChanges();

                    Session["AccountIDSS"] = booking.AccountID.ToString();
                    Session["EmailSS"] = booking.PatientEmail.ToString();

                    ViewBag.CreateAppointmentMsg = ("<script>alert('Appointment
Create Successfully!.')</script>");

                    return RedirectToAction("Index", "Patient");
                }
            }
            ViewBag.AccountID = new SelectList(db.Accounts, "AccountID",
"FullName", booking.AccountID);
            return View(booking);
        }

        // GET: Patient/Edit/5
        public ActionResult Edit(int? id)
        {
            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];

                if (obj.AccountRole != "Patient" && obj.AccountRole != "Admin")
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }


            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Booking booking = db.Bookings.Find(id);
            if (booking == null)
            {
                return HttpNotFound();
            }
            ViewBag.AccountID = new SelectList(db.Accounts, "AccountID",
"FullName", booking.AccountID);
            return View(booking);
        }

        // POST: Patient/Edit/5
        // To protect from overposting attacks, enable the specific properties you
want to bind to, for
```

```
        // more details see https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include =
"BookingID,BookingDate,TimeSlot,Department,DoctorName,DoctorEmail,AccountID,Patien
tName,PatientEmail,BookingStatus")] Booking booking)
        {
            if (ModelState.IsValid)
            {
                if (db.Bookings.Any(x => x.PatientEmail == booking.PatientEmail))
                {
                    ViewBag.SameEmailMsg3 = ("<script>alert('Email has been use
before!.')</script>");
                    return View();
                }
                else
                {
                    db.Entry(booking).State = EntityState.Modified;
                    int edit = db.SaveChanges();
                    Session["AccountIDSS"] = booking.AccountID.ToString();
                    Session["EmailSS"] = booking.PatientEmail.ToString();

                    if (edit > 0)
                    {
                        ViewBag.PatientEditMsg2 = ("<script>alert('Edit
Appointment Successfully!.')</script>");

                        return RedirectToAction("Index", "Patient");
                    }
                    else
                    {
                        ViewBag.PatientEditMsg = ("<script>alert('Something is
wrong with the input...')</script>");
                    }
                }
            }
            ViewBag.AccountID = new SelectList(db.Accounts, "AccountID",
"FullName", booking.AccountID);
            return View(booking);
        }

        // GET: Patient/Delete/5
        public ActionResult Delete(int? id)
        {
            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];

                if (obj.AccountRole != "Patient" && obj.AccountRole != "Admin")
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }
```

```
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Booking booking = db.Bookings.Find(id);
            if (booking == null)
            {
                return HttpNotFound();
            }
            return View(booking);
        }

        // POST: Patient/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];

                if (obj.AccountRole != "Patient" && obj.AccountRole != "Admin")
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }


            Booking booking = db.Bookings.Find(id);
            db.Bookings.Remove(booking);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

DoctorController (Controller/DoctorController.cs) Action Function for Doctor to Create, Edit, Delete, View the Patients Bookings or Appointments

- Highlight in Yellow: Basically, the function of the ActionResult to return the user to the Doctor Control Panel Page which is (Doctor/Index.cshtml). Where the Doctors can Create, Edit, View and Delete the Patients Bookings/Appointment from the Appointment/Booking webpage/Database. But it will also do a validation check to see if the Account Session Login is Admin or Doctor role. If it is not an Admin or Doctor role account, it will redirect the user to the Login page.

- Highlight in Teal Color: Basically, the function of the ActionResult for the Doctors to view the Patients Bookings/Appointment Details on the Doctor control panel Details Page (Doctor/Details.cshtml) from the Bookings database itself. I also did a validation check to see if the Account Session Login is Admin or Doctor role. If it is not an Admin or Doctor role account, it will return the user to the login page.

- Highlight in Light Blue: Basically, the function of the ActionResult for the Doctors to create or add an Appointment/Booking for the Patients at the Doctor Control Panel Appointments/Booking Webpage (Doctor/Create.cshtml). The Appointment/Booking that the Doctor created for the Patient will be store at the Bookings Database automatically. It also did a validation check to see if the Account Session Login is Admin or Doctor role. If it is not an Admin or Doctor role account, it will redirect the user to the login page

- Highlight in Green: Basically, the function of the ActionResult for the Doctors to Edit or Update the Patients appointment/booking by updating their appointment/booking details using the Doctor Control Panel Edit webpage (Doctor/Edit.cshtml). Once is update on the webpage “Edit” it will also update it on the “Bookings” database automatically. It also did a validation check to see if the Account Session Login is Admin or Doctor role. If it is not an Admin or Doctor role account, it will redirect the user to the login page

- Highlight in Pink: Basically, the function of the ActionResult for the Doctor to Delete the Patients Appointment/Booking base on their BookingID at the Doctor Control Panel Delete Webpage (Doctor/Delete.cshtml). It will also do a confirmation check to see if the Doctor wishes to delete this Patient appointment/booking. It will also automatically delete it from the database once the user wishes to delete it. Lastly It also did a validation check to see if the Account Session Login is Admin or Doctor role. If it is not an Admin or Doctor role account, it will return the user to the login page.

- Highlight in Red: It basically, the Dispose method that will release the resources. If it is true, it will release both managed and unmanaged resources; if it is false, it will only release unmanaged resources.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
```

```
using System.Net;
using System.Web;
using System.Web.Mvc;
using Assignment_02.Models;

namespace Assignment_02.Controllers
{
    public class DoctorController : Controller
    {
        private ASSIGNMENT02Entities db = new ASSIGNMENT02Entities();

        // GET: Doctor
        public ActionResult Index()
        {
            var bookings = db.Bookings.Include(b => b.Account);

            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];

                if (obj.AccountRole != "Doctor" && obj.AccountRole != "Admin")
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }

            return View(bookings.ToList());
        }

        // GET: Doctor/Details/5
        public ActionResult Details(int? id)
        {
            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];

                if (obj.AccountRole != "Doctor" && obj.AccountRole != "Admin")
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }

            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Booking booking = db.Bookings.Find(id);
            if (booking == null)
            {
                return HttpNotFound();
```

```
            }
            return View(booking);
        }

        // GET: Doctor/Create
        public ActionResult Create()
        {
            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];

                if (obj.AccountRole != "Doctor" && obj.AccountRole != "Admin")
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }

            ViewBag.AccountID = new SelectList(db.Accounts, "AccountID",
"FullName");
            return View();
        }

        // POST: Doctor/Create
        // To protect from overposting attacks, enable the specific properties you
want to bind to, for
        // more details see https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Booking booking)
        {
            if (ModelState.IsValid)
            {
                if (db.Bookings.Any(x => x.PatientEmail == booking.PatientEmail))
                {
                    ViewBag.DoctorCreateEmailMsg = ("<script>alert('Email has been
use before!.')</script>");
                    return View();
                }
                else
                {
                    db.Bookings.Add(booking);
                    db.SaveChanges();

                    Session["AccountIDSS"] = booking.AccountID.ToString();
                    Session["EmailSS"] = booking.PatientEmail.ToString();

                    ViewBag.CreateAccountMsg = ("<script>alert('Account Create
Successfully!.')</script>");

                    return RedirectToAction("Index", "Doctor");
                }
            }
            ViewBag.AccountID = new SelectList(db.Accounts, "AccountID",
"FullName", booking.AccountID);
```

```
            return View(booking);
        }

        // GET: Doctor/Edit/5
        public ActionResult Edit(int? id)
        {
            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];

                if (obj.AccountRole != "Doctor" && obj.AccountRole != "Admin")
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }

            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Booking booking = db.Bookings.Find(id);
            if (booking == null)
            {
                return HttpNotFound();
            }
            ViewBag.AccountID = new SelectList(db.Accounts, "AccountID",
"FullName", booking.AccountID);
            return View(booking);
        }

        // POST: Doctor/Edit/5
        // To protect from overposting attacks, enable the specific properties you
want to bind to, for
        // more details see https://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include =
"BookingID,BookingDate,TimeSlot,Department,DoctorName,DoctorEmail,AccountID,Patien
tName,PatientEmail,BookingStatus")] Booking booking)
        {
           if (ModelState.IsValid)
            {
                if (db.Bookings.Any(x => x.PatientEmail == booking.PatientEmail))
                {
                    ViewBag.SameEmailMsg2 = ("<script>alert('Email has been use
before!.')</script>");
                    return View();
                }
                else
                {
                    db.Entry(booking).State = EntityState.Modified;
                    int edit = db.SaveChanges();
                    Session["AccountIDSS"] = booking.AccountID.ToString();
                    Session["EmailSS"] = booking.PatientEmail.ToString();
```

```
                    if (edit > 0)
                    {
                        ViewBag.DoctorEditMsg2 = ("<script>alert('Edit Appointment
Successfully!.')</script>");

                        return RedirectToAction("Index", "Doctor");
                    }
                    else
                    {
                        ViewBag.DoctorEditMsg = ("<script>alert('Something is
wrong with the input...')</script>");
                    }
                }
            }
            ViewBag.AccountID = new SelectList(db.Accounts, "AccountID",
"FullName", booking.AccountID);
            return View(booking);
        }

        // GET: Doctor/Delete/5
        public ActionResult Delete(int? id)
        {
            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];

                if (obj.AccountRole != "Doctor" && obj.AccountRole != "Admin")
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }

            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Booking booking = db.Bookings.Find(id);
            if (booking == null)
            {
                return HttpNotFound();
            }
            return View(booking);
        }

        // POST: Doctor/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            if (Session["Login"] != null)
            {
                Models.Account obj = (Models.Account)Session["Login"];
```

```
                if (obj.AccountRole != "Doctor" && obj.AccountRole != "Admin")
                {
                    return RedirectToAction("Login", "Login");
                }
            }
            else
            {
                return RedirectToAction("Login", "Login");
            }

            Booking booking = db.Bookings.Find(id);
            db.Bookings.Remove(booking);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

Hospital Homepage Design (Views/Home/Index.cshtml) (Source Code)

- It basically the Hospital Home webpage coding that is to design and style the webpage content outcome for the user to view the Homepage Content.

```
@{
    ViewBag.Title = "Home Page";
}

<style>
    .Home-Content {
        background-color: lightgrey;
        padding: 50px 100px 50px 100px;
        font-family: arial;
    }

    .Home-Content h1
    {
        color: indianred;

    }
</style>

<div class="Home-Content" >
    <center><h1>Welcome to Murdoch Hospital</h1>
    <br />
            <h3>
```

```
                Murdoch Hospital is one of the leading private health campuses in
Western Australia and a major health care hub serving the southern region.
                <br /><br />
                Our reputation for excellence in clinical care and first class
tertiary level services makes us the hospital of choice for both medical
professionals and patients.
            </h3>
            <br />
            <iframe width="960" height="540"
src="https://www.youtube.com/embed/sIFupQUA9bg" title="YouTube video player"
frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media;
gyroscope; picture-in-picture" allowfullscreen></iframe>
    </center>
</div>

<br /><br />
<center>
    <div class="row">
        <div class="col-md-4">
            <img src="~/Image/AboutUsIcon.png" alt="About_Us_Icon_Image"
width="150" height="120" />
            <h2>About Us</h2>
            <p>
                Give you information and history of Murdoch Hospital of the
history and stories about it.
            </p>
            <p><a class="btn btn-default"
href="https://localhost:44399/Home/About">Learn more &raquo;</a></p>
        </div>
        <div class="col-md-4">
            <img src="~/Image/ContactUsIcon.png" alt="Contact_Us_Icon_Image"
width="150" height="120" />
            <h2>Contact</h2>
            <p>Our Murdoch Hospital Contact Page which have our contact details
and visiting hour details.</p>
            <p><a class="btn btn-default"
href="https://localhost:44399/Home/Contact">Learn more &raquo;</a></p>
        </div>
        <div class="col-md-4">
            <img src="~/Image/ServiceIcon.png" alt="Service_Icon_Image"
width="150" height="120" />
            <h2>Services</h2>
            <p>Information of what kind of treatment and services we have in
Murdoch Hospital</p>
            <p><a class="btn btn-default"
href="https://localhost:44399/Home/Services">Learn more &raquo;</a></p>
        </div>
    </div>
</center>
```

Hospital About Page Design (Views/Home/About.cshtml) (Source Code)

- It basically the Hospital About Us webpage coding that is to design and style the webpage content outcome for the user to view the About Page Content where the user can learn about the hospital history and their story.

```
@{
    ViewBag.Title = "About";
}
<style>
    .About-Page {
        font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
        font-size: large;
    }
</style>

<center>
    <h2>@ViewBag.Title.</h2>
    <h3>@ViewBag.Message</h3>
</center>
<br />
<center>
    <div class="About-Photo">
        <img src="~/Image/Murdoch_Hospital.jpg" alt="Murdoch_Hospital_IMG" />
    </div>
</center>
<br />
<div class="About-Page">
    <div class="About-Content">
        <h2 style="text-decoration: underline;">History</h2>
        <p>
            In 1989, St John of God Health Care, in conjunction with the Sisters
of St Joseph of the
            Apparition and the Sisters of Mercy in Perth, lodged a proposal
            for new private hospital facilities in the south metropolitan area of
Perth.
        </p>
        <br />
        <p>The success of this proposal led to St John of God Health Care
undertaking the construction of what is now St John of God Murdoch Hospital.</p>
        <br />
        <p>Staff from St Joseph's Hospital Bicton and St John of God Rivervale
transferred to St John of God Murdoch Hospital in preparation for the opening of
the new hospital.</p>
        <br />
        <p>Other key staff for the development of the new hospital were recruited
and appointed, starting with the founding Chief Executive Officer Bill Shields in
1991.</p>
        <br />
        <p>The site of the new hospital was blessed on 21 September 1991 and the
first patients were admitted on 14 February 1994.</p>
        <br />
        <p>The vision of a unique hospital south of the river became a reality and
enabled us to continue the work of St John of God and the founding sisters.</p>
        <br />
    </div>
    <center>
        <div class=row>
            <div class="About-Content-02">
```

```
                <h3 style="text-decoration: underline;">More Information About
Murdoch Hospital</h3>
                <p>
                    For more information and history of Murdoch Hospital of the
history and stories about it
                    <br />do click the link below.
                </p>
                <br />
                <p><a class="btn btn-default" href="https://www.sjog.org.au/our-
locations/st-john-of-god-murdoch-hospital/about">About Murdoch Hospital
&raquo;</a></p>
            </div>
        </div>
    </center>
</div>
```

Hospital Contact Us Page Design (Views/Home/Contact.cshtml) (Source Code)

- It basically the Hospital Contact Us webpage coding that is to design and style the webpage content outcome for the user to view the Contact Us Webpage content where the user can see the hospital details and contact them if needed.

```
@{
    ViewBag.Title = "Contact Us";
}


<style>
    .Contact-Page
    {
        font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
        font-size: large;
    }

    .Contact-List
    {
        padding-right: 45px;
    }
</style>
<div class="Contact-Page">
    <center>
        <h2>@ViewBag.Title.</h2>
        <h3 style="text-decoration: underline;">Contact us</h3>
        <br />

        <div class="Contact-Content">
            <address>
                Tel: (08) 9438 9000
                <br />
                Freecall: 1800 640 300 (country only)
                <br /> <br />
                Our main reception phone number is (08) 9438 9000.
```

```
                <br /> <br />
                Patients can phone our new patient helpline on (08) 9428 8838 to
find out more about gaps and out-of-pocket expenses that relate to your health
fund and which may be associated with your hospital treatment.
                <br /> <br />
                Open between 8.00am and 4.30pm weekdays.
            </address>
        </div>

        <br />
        <div class="Contact-Content-02">
            <h3 style="text-decoration: underline;">Find us</h3>
            <address>
                <br />
                <div class="Contact-List">
                    <ul style="list-style-type: none">
                        <li>Street addresses:</li>
                        <br />
                        <li>Main Hospital, Emergency and Murdoch Medical
Clinic</li>
                        <li>Gate 1, Barry Marshall Parade</li>
                        <li> Murdoch WA 6150</li>
                        <br />
                        <li>Wexford Medical Centre</li>
                        <li> Gate 2, Barry Marshall Parade</li>
                        <li>Murdoch WA 6150</li>
                        <br />
                        <li> Hospice and Deliveries</li>
                        <li>Gate 3, Fiona Wood Road</li>
                        <li>Murdoch WA 6150</li>
                        <br />
                        <li>Epic Pharmacy</li>
                        <li>Gate 3, Fiona Wood Road</li>
                        <li>Murdoch WA 6150</li>
                        <br />
                        <li>MURTEC and Function Rooms</li>
                        <li>Gate 3, Fiona Wood Road</li>
                        <li>Murdoch WA 6150</li>
                    </ul>
                </div>
            </address>
        </div>

        <br />
        <h3 style="text-decoration: underline;">Our links</h3>
        <br />
        <div class="Contact-Content-03">
            <address>
                <strong>Work With Us:</strong> <a
href="https://www.sjog.org.au/our-locations/st-john-of-god-murdoch-hospital/work-
with-us">recruitment.hr@sjog.org.au</a><br />
                <strong>Our Facebook:</strong> <a
href="https://www.facebook.com/stjohnofgodmurdoch/">Facebook</a><br />
                <strong>Murdoch Helpline:</strong> <a
href="https://www.sjog.org.au/murdochhelpline">Murdoch Hospital Helpline</a>
            </address>
        </div>
    </center>
```

```
</div>
```

Hospital Services Page Design (Views/Home/Services.cshtml) (Source Code)

- It basically the Hospital Service webpage coding that is to design and style the webpage content outcome for the user to view Service Webpage content to see what the hospital provides what kind of services.

```
@{
    ViewBag.Title = "Services";
}

<style>
    .Services-Page {
        font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
        font-size: large;
    }
</style>

<center>
    <h1>@ViewBag.Title.</h1>
    <h2>@ViewBag.Message</h2>
</center>
<br />
<center>
    <div class="Services-Page">
        <h1 style="text-decoration: underline;">Emergency Department</h1>
        <h2>Our Emergency Department is open 24 hours a day, 7 days a week and is
easily accessible via Barry Marshall Parade.</h2>
        <h3>If you have a life-threatening medical condition, call 000 and ask for
an ambulance. You can request paramedics to bring you to our hospital.</h3><br />
        <strong>Emergency Services Link:</strong> <a
href="https://www.sjog.org.au/our-locations/st-john-of-god-murdoch-hospital/our-
services/emergency-department">Emegerency Services</a>
        <br /><br />
        <h1 style="text-decoration: underline;">maternity-services</h1>
        <h3>When you choose St John of God Murdoch Hospital, you are choosing the
option to have your partner stay with you overnight, a longer hospital stay, your
own obstetrician, and access to breastfeeding and mental health support with us.
Our maternity ward, called St Mary's Ward, gives you access to the support and
care of our midwives, nurses and experienced caregivers in our modern hospital,
right in the heart of Perth's southern suburbs</h3><br />
        <strong>Maternity Services Link:</strong> <a
href="https://www.sjog.org.au/our-locations/st-john-of-god-murdoch-hospital/our-
services/maternity-services">Maternity Services</a>
        <br /><br />
        <h1 style="text-decoration: underline;">Medical and surgical</h1>
        <h3>We offer a wide range of medical and surgical services. To find out
more about each service, By browsing the link below.</h3><br />
        <strong>Medical and surgical Link:</strong> <a
href="https://www.sjog.org.au/our-locations/st-john-of-god-murdoch-hospital/our-
services/medical-and-surgical">Medical and surgical</a>
        <br /><br />
```

```
        <h1 style="text-decoration: underline;">Mental Health Services</h1>
        <h3>St John of God Murdoch Hospital is working on providing outpatient
(day) services for adult patients while development of the standalone mental
health facility continues.</h3><br />
        <strong>Mental Health Services Link:</strong> <a
href="https://www.sjog.org.au/our-locations/st-john-of-god-murdoch-hospital/our-
services/mental-health">Mental Health Services</a>
        <br /><br />
        <h1 style="text-decoration: underline;">Rehabilitation services</h1>
        <h3>Our rehabilitation services help you return to health and independence
as soon as possible after injury, illness or surgery. To find out more, browse our
services by clicking the link below.</h3><br />
        <strong>Rehabilitation services Link:</strong> <a
href="https://www.sjog.org.au/our-locations/st-john-of-god-murdoch-hospital/our-
services/rehabilitation-services">Rehabilitation services</a>
        <br /><br />
        <h1 style="text-decoration: underline;">Community and youth Services</h1>
        <h3>We provide community and youth services to people who are experiencing
disadvantage to improve their health and wellbeing. Programs are available at no
or low cost.</h3><br />
        <strong>Community and Youth Link:</strong> <a
href="https://www.sjog.org.au/our-locations/st-john-of-god-murdoch-hospital/our-
services/community-and-youth">Community and Youth</a>
        <br /><br />
        <h1 style="text-decoration: underline;">Home Healthcare Services</h1>
        <h3>St John of God Healthcare at Home gives you the choice to receive
compassionate nursing care in the comfort of your own home, when you need
it.</h3><br />
        <strong>Home Healthcare Services Link:</strong> <a
href="https://www.sjog.org.au/our-locations/st-john-of-god-murdoch-hospital/our-
services/home-nursing-and-care">Home Healthcare Services</a>
        <br /><br />
    </div>
</center>
```

Hospital Registration Page Design (Views/Home/Registration.cshtml) (Source Code)

- It basically the Hospital Registration webpage coding that is to design, style and implement the function of HomeController the webpage content outcome for the user to register their account for the hospital to use the hospital webpage system. The account that has been register on this webpage will be store inside the Account Model Entity ADO.net Database. Which is why it have a @model implementation on the top.

```
@model Assignment_02.Models.Account

@{
    ViewBag.Title = "Registration";
}
```

```
<script>
    function submit_alert() {
        alert("Account has been created!.");
    }

    function reset_alert() {
        alert("Form has been clear!");
    }
</script>

<style>
    .BackButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: 15px;
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    button {
        outline: none;
        border: none;
    }

    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }

    a:visited {
        color: white;
        text-decoration: none;
    }
</style>

<h2>Registration</h2>

@Html.Raw(ViewBag.SameEmailMsg)
@Html.Raw(ViewBag.CreateAccountMsg)

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
```

```
        <div class="form-group">
            @Html.LabelFor(model => model.FullName, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.FullName, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.FullName, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Email, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Email, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Email, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.AccountPassword, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.AccountPassword, new
{ htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.AccountPassword, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.PersonAddress, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.PersonAddress, new { htmlAttributes
= new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.PersonAddress, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.PersonPhoneNo, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.PersonPhoneNo, new { htmlAttributes
= new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.PersonPhoneNo, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Gender, htmlAttributes: new { @class =
"control-label col-md-2" })
```

```
            <div class="col-md-10">
                @Html.DropDownListFor(m => m.Gender, new List<SelectListItem>
              { new SelectListItem{Text="Male", Value="Male"},
                new SelectListItem{Text="Female", Value="Female"}}, "Please select
Your Gender", new { style = "width:250px; height: 50px;" })
                @Html.ValidationMessageFor(model => model.Gender)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.BloodType, htmlAttributes: new { @class
= "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownListFor(m => m.BloodType, new List<SelectListItem>
               { new SelectListItem{Text="A+", Value="A+"},
                 new SelectListItem{Text="A-", Value="A-"},
                 new SelectListItem{Text="B+", Value="B+"},
                 new SelectListItem{Text="B-", Value="B-"},
                 new SelectListItem{Text="O+", Value="O+"},
                 new SelectListItem{Text="O-", Value="O-"},
                 new SelectListItem{Text="AB+", Value="AB+"},
                 new SelectListItem{Text="AB-", Value="AB-"}

                 ,
            }, "Please select Blood Type", new { style = "width:250px; height:
50px;" })
                @Html.ValidationMessageFor(model => model.BloodType)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.AccountRole, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownListFor(m => m.AccountRole, new List<SelectListItem>
              { new SelectListItem{Text="Admin", Value="Admin"},
                new SelectListItem{Text="Patient", Value="Patient"},
                new SelectListItem{Text="Doctor", Value="Doctor"}},
                "Please select Your Account Role", new { style = "width:250px;
height: 50px;" })
                @Html.ValidationMessageFor(model => model.AccountRole)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.DOB, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.DOB, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.DOB, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
```

```
                <input style="background-color:deepskyblue;color:white"
type="submit" value="Create" class="btn btn-default" onclick="submit_alert()" />
            </div>
        </div>


        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <label style="color:red;">@ViewBag.Notification</label>
            </div>
        </div>

    </div>
}
<div>
    <button class="BackButton">@Html.ActionLink("Back to Homepage",
"Index")</button>
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Hospital Login Page Design (Views/Login/Login.cshtml) (Source Code)

- It basically the Hospital Login webpage coding that is to design, style and implement the function of LoginController the webpage content where the user can login their account using this webpage to use the Hospital System. The user have to enter and select their Email, Password and Account Role in order to login successfully to the Hospital System Webpage. In order to check if the Account is exist in the database this is why it have a @models Account implementation on the top to use the Account Ado Entity Database Models to check for the account existence.

```
@model Assignment_02.Models.Account

@{
    ViewBag.Title = "Login";
}

<script>
    function submit_alert() {
        alert("Account has successfully login!.");
    }

    function reset_alert() {
        alert("Form has been clear!");
    }
</script>

<style>
    .BackButton {
        background-color: dodgerblue;
```

```
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: 15px;
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    button {
        outline: none;
        border: none;
    }

    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }

    a:visited {
        color: white;
        text-decoration: none;
    }
</style>

<h2>Login</h2>

@Html.Raw(ViewBag.LoginMsg)

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })


        <div class="form-group">
            @Html.LabelFor(model => model.Email, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Email, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Email, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
```

```
            @Html.LabelFor(model => model.AccountPassword, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.AccountPassword, new
{ htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.AccountPassword, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.AccountRole, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownListFor(m => m.AccountRole, new List<SelectListItem>
              { new SelectListItem{Text="Admin", Value="Admin"},
                new SelectListItem{Text="Patient", Value="Patient"},
                new SelectListItem{Text="Doctor", Value="Doctor"}},
                "Please select Your Account Role", new { style = "width:250px;
height: 50px;" })
                @Html.ValidationMessageFor(model => model.AccountRole)
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input style="background-color:deepskyblue;color:white"
type="submit" value="Login" class="btn btn-default" onclick="submit_alert()" />
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <label style="color:red">@ViewBag.Notification</label>
            </div>
        </div>
    </div>
}

<div>
    <button class="BackButton">@Html.ActionLink("Back to Homepage", "Index",
"Home")</button>
</div>


@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Hospital Admin Control Panel Homepage Design (Views/Admin/Index.cshtml) (Source Code)

- It basically the Hospital Admin Control Panel webpage coding that is to design, style and implement the function of AdminController to the Admin Control Panel webpage content for the admin to View, Edit, Create, Delete the user account from the Account Database. Instead of updating on the database the admin can just update it on this webpage. Therefore, I use @model for this webpage so is linked to the Account Database Model.

```
@model IEnumerable<Assignment_02.Models.Account>

@{
    ViewBag.Title = "Admin Panel";
}


<style>
    .CreateButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 20px;
        text-decoration: none;
        height: 65px;
        width: 150px;
        font-family: Arial;
        font-size: medium;
        border-radius: 12px;
        border: none;
    }

    .AdminButton {
        background-color: red;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 75px;
        width: 200px;
        font-family: Arial;
        font-size: medium;
        border-radius: 12px;
        border: none;
    }

    .EditButton {
        background-color: dodgerblue;
        padding: 10px;
        text-decoration-color: white;
        text-decoration: none;
        height: 65px;
        width: 80px;
        font-family: Arial;
        font-size: medium;
        border-radius: 12px;
        text-align: center;
        outline: none;
        outline: none;
    }
```

```
    .DetailButton {
        background-color: forestgreen;
        padding: 10px;
        text-decoration-color: white;
        text-decoration: none;
        height: 65px;
        width: 80px;
        font-family: Arial;
        font-size: Medium;
        border-radius: 12px;
        text-align: center;
    }

    .DeleteButton {
        background-color: red;
        padding: 10px;
        text-decoration-color: white;
        text-decoration: none;
        height: 65px;
        width: 80px;
        font-family: Arial;
        font-size: Medium;
        border-radius: 12px;
        text-align: center;
    }

    button {
        outline: none;
        border: none;
    }

    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }

    a:visited {
        color: white;
        text-decoration: none;
    }
</style>

<h2>Admin Control Panel</h2>

<center>
    <div class="CreateButton">
        <a>@Html.ActionLink("Create New Account", "Create")</a>
    </div>
    <br />
    <div class="AdminButton">
        <a>@Html.ActionLink("Admin Control Panel (Appointment Data)", "Index",
"AdminTwo")</a>
    </div>
```

```
</center>
<br />
<h2>Account Data</h2>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.FullName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Email)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.AccountPassword)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.PersonAddress)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.PersonPhoneNo)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Gender)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.BloodType)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.AccountRole)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DOB)
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.FullName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Email)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.AccountPassword)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.PersonAddress)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.PersonPhoneNo)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Gender)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.BloodType)
```

```
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.AccountRole)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.DOB)
            </td>
            <td>
                <button class="EditButton">@Html.ActionLink("Edit", "Edit", new
{ id = item.AccountID }) </button>
                <button class="DetailButton">@Html.ActionLink("Details",
"Details", new { id = item.AccountID }) </button>
                <button class="DeleteButton">@Html.ActionLink("Delete", "Delete",
new { id = item.AccountID })</button>
            </td>
        </tr>
    }

</table>
```

- |

Hospital Admin Control Panel Create Page Design (Views/Admin/Create.cshtml) (Source Code)

- It basically the Hospital Admin Control Panel webpage coding that is to design, style and implement the function of AdminController to the Admin Control Panel webpage content for the admin to Create the user account if needed and store it to the Account Database itself. Instead of Creating on the database the admin can just Create it on this webpage. Therefore, I use @model for this webpage so is linked to the Account Database Model.

```
@model Assignment_02.Models.Account

@{
    ViewBag.Title = "Create";
}


<style>
    .BackButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: 15px;
        border-radius: 12px;
        border: none;
        text-align: center;
```

```
    }

    button {
        outline: none;
        border: none;
    }

    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }

    a:visited {
        color: white;
        text-decoration: none;
    }
</style>

<script>
    function create_alert() {
        alert("Account has been created!.");
    }
</script>

<h2>Create</h2>

@Html.Raw(ViewBag.SameEmailMsg)
@Html.Raw(ViewBag.CreateAccountMsg)

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Account</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.FullName, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.FullName, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.FullName, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Email, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
```

```
                @Html.EditorFor(model => model.Email, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Email, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.AccountPassword, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.AccountPassword, new
{ htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.AccountPassword, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.PersonAddress, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.PersonAddress, new { htmlAttributes
= new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.PersonAddress, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.PersonPhoneNo, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.PersonPhoneNo, new { htmlAttributes
= new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.PersonPhoneNo, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Gender, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownListFor(model => model.Gender, new
List<SelectListItem>
               { new SelectListItem{Text="Male", Value="Male"},
                 new SelectListItem{Text="Female", Value="Female"}},
                 "Please select Your Gender", new { style = "width:250px; height:
50px;" })
                @Html.ValidationMessageFor(model => model.Gender)
                @Html.ValidationMessageFor(model => model.Gender)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.BloodType, htmlAttributes: new { @class
= "control-label col-md-2" })
```

```
            <div class="col-md-10">
                @Html.DropDownListFor(model => model.BloodType, new
List<SelectListItem>
                { new SelectListItem{Text="A+", Value="A+"},
                  new SelectListItem{Text="A-", Value="A-"},
                  new SelectListItem{Text="B+", Value="B+"},
                  new SelectListItem{Text="B-", Value="B-"},
                  new SelectListItem{Text="O+", Value="O+"},
                  new SelectListItem{Text="O-", Value="O-"},
                  new SelectListItem{Text="AB+", Value="AB+"},
                  new SelectListItem{Text="AB-", Value="AB-"}

                  ,
                }, "Please select Blood Type", new { style = "width:250px; height:
50px;" })
                @Html.ValidationMessageFor(model => model.BloodType)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.AccountRole, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownListFor(model => model.AccountRole, new
List<SelectListItem>
                { new SelectListItem{Text="Admin", Value="Admin"},
                  new SelectListItem{Text="Patient", Value="Patient"},
                  new SelectListItem{Text="Doctor", Value="Doctor"}},
                  "Please select Your Account Role", new { style = "width:250px;
height: 50px;" })
                @Html.ValidationMessageFor(model => model.AccountRole)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.DOB, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.DOB, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.DOB, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input style="background-color:deepskyblue;color:white"
onclick="create_alert()" type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}
<div>
    <button class="BackButton">@Html.ActionLink("Back to List", "Index")</button>
</div>
@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
```

```
}
```

Hospital Admin Control Panel Delete Page Design (Views/Admin/Delete.cshtml) (Source Code)

- It basically the Hospital Admin Control Panel webpage coding that is to design, style and implement the function of AdminController to the Admin Control Panel webpage content for the admin to Delete the user account if needed from the Account Database itself. Instead of Deleting on the database the admin can just Delete it on this webpage. Therefore, I use @model for this webpage so is linked to the Account Database Model.

```
@model Assignment_02.Models.Account

@{
    ViewBag.Title = "Delete";
}


<style>
    .BackButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: 15px;
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    button {
        outline: none;
        border: none;
    }

    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }

    a:visited {
        color: white;
        text-decoration: none;
    }
```

```
</style>

<script>
    function delete_alert()
    {
        alert("You have delete the Account!.");
    }
</script>


<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Admin Delete User Account</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.FullName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.FullName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Email)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Email)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.AccountPassword)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.AccountPassword)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.PersonAddress)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.PersonAddress)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.PersonPhoneNo)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.PersonPhoneNo)
        </dd>

        <dt>
```

```
            @Html.DisplayNameFor(model => model.Gender)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Gender)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.BloodType)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.BloodType)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.AccountRole)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.AccountRole)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DOB)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DOB)
        </dd>

    </dl>


    @using (Html.BeginForm())
    {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input style="background-color:red;color:white"
onclick="delete_alert()" type="submit" value="Delete" class="btn btn-default" />
        </div>
    }
        <br />
        <div>
            <button class="BackButton">@Html.ActionLink("Back to List",
"Index")</button>
        </div>
    </div>
```

Hospital Admin Control Panel Edit Page Design (Views/Admin/Edit.cshtml) (Source Code)

- It basically the Hospital Admin Control Panel webpage coding that is to design, style and implement the function of AdminController to the Admin Control Panel webpage content for the admin to Edit the user account details if needed from the Account Database itself. Instead of Editing it on the database the admin can just edit it from this webpage. Therefore, I use @model for this webpage so is linked to the Account Database Model.

```
@model Assignment_02.Models.Account

@{
    ViewBag.Title = "Edit";
}

<style>
    .BackButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: 15px;
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    button {
        outline: none;
        border: none;
    }

    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }

    a:visited {
        color: white;
        text-decoration: none;
    }
</style>

<script>
    function edit_alert() {
        alert("Account details/information has been updated!.");
```

```
    }
</script>

<h2>Admin Edit User Account</h2>

@Html.Raw(ViewBag.EditMsg)
@Html.Raw(ViewBag.EditMsg2)
@Html.Raw(ViewBag.SameEmailMsg)

@using (Html.BeginForm())
{

    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Account</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.AccountID)

        <div class="form-group">
            @Html.LabelFor(model => model.FullName, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.FullName, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.FullName, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Email, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Email, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Email, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.AccountPassword, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.AccountPassword, new
{ htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.AccountPassword, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.PersonAddress, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
```

```
                @Html.EditorFor(model => model.PersonAddress, new { htmlAttributes
= new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.PersonAddress, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.PersonPhoneNo, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.PersonPhoneNo, new { htmlAttributes
= new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.PersonPhoneNo, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Gender, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownListFor(model => model.Gender, new
List<SelectListItem>
                    { new SelectListItem{Text="Male", Value="Male"},
                      new SelectListItem{Text="Female", Value="Female"}},
                      "Please select Your Gender", new { style = "width:250px;
height: 50px;" })
                @Html.ValidationMessageFor(model => model.Gender)
                @Html.ValidationMessageFor(model => model.Gender)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.BloodType, htmlAttributes: new { @class
= "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownListFor(model => model.BloodType, new
List<SelectListItem>
                    { new SelectListItem{Text="A+", Value="A+"},
                      new SelectListItem{Text="A-", Value="A-"},
                      new SelectListItem{Text="B+", Value="B+"},
                      new SelectListItem{Text="B-", Value="B-"},
                      new SelectListItem{Text="O+", Value="O+"},
                      new SelectListItem{Text="O-", Value="O-"},
                      new SelectListItem{Text="AB+", Value="AB+"},
                      new SelectListItem{Text="AB-", Value="AB-"}

                    }, "Please select Blood Type", new { style = "width:250px;
height: 50px;" })
                @Html.ValidationMessageFor(model => model.BloodType)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.AccountRole, htmlAttributes: new
{ @class = "control-label col-md-2" })
```

```
            <div class="col-md-10">
                @Html.DropDownListFor(m => m.AccountRole, new List<SelectListItem>
               { new SelectListItem{Text="Admin", Value="Admin"},
               new SelectListItem{Text="Patient", Value="Patient"},
               new SelectListItem{Text="Doctor", Value="Doctor"}},
               new { style = "width:250px; height: 50px;" })
                @Html.ValidationMessageFor(model => model.AccountRole)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.DOB, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.DOB, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.DOB, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input style="background-color:green;color:white"
onclick="edit_alert()" type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    <button class="BackButton">@Html.ActionLink("Back to List", "Index")</button>
</div>



@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Hospital Admin Control Panel Details Page Design (Views/Admin/Details.cshtml) (Source Code)

- It basically the Hospital Admin Control Panel webpage coding that is to design, style and implement the function of AdminController to the Admin Control Panel webpage content for the admin to view the specific user account details if needed from the Account Database itself. Instead of viewing it on the database. The admin can just view the details from this webpage itself. Therefore, I use @model for this webpage so is linked to the Account Database Model.

```
@model Assignment_02.Models.Account

@{
```

```
    ViewBag.Title = "Details";
}

<style>
    .BackButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: 15px;
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    .EditButton {
        background-color: green;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: medium;
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    .AccountDetails {
        font-size: 24px;
    }

    button {
        outline: none;
        border: none;
    }

    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }

    a:visited {
        color: white;
        text-decoration: none;
    }
</style>

<h2>Details</h2>
```

```
<div class="AccountDetails">
    <h4>Admin Check User Account Details</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.FullName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.FullName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Email)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Email)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.AccountPassword)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.AccountPassword)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.PersonAddress)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.PersonAddress)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.PersonPhoneNo)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.PersonPhoneNo)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Gender)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Gender)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.BloodType)
        </dt>

        <dd>
```

```
            @Html.DisplayFor(model => model.BloodType)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.AccountRole)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.AccountRole)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DOB)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DOB)
        </dd>

    </dl>
</div>
<p>
    <button class="EditButton">@Html.ActionLink("Edit", "Edit", new { id =
Model.AccountID })</button>
    <button class="BackButton">@Html.ActionLink("Back to List", "Index")</button>
</p>
```

Hospital Admin Control Panel Homepage Design for Bookings/Appointments (Views/AdminTwo/Index.cshtml) (Source Code)

- It basically the Hospital Admin Control Panel webpage coding that is to design, style and implement the function of AdminTwoController to the Admin Control Panel webpage content for the admin to View, Edit, Create, Delete the user appointments/booking from the Booking Database. Instead of updating on the database the admin can just update it on this webpage. Therefore, I use @model for this webpage so is linked to the Booking Database Entity Model.

```
@model IEnumerable<Assignment_02.Models.Booking>

@{
    ViewBag.Title = "Index";
}

<style>
    .CreateButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 20px;
        text-decoration: none;
        height: 65px;
        width: 130px;
```

```
        font-family: Arial;
        font-size: medium;
        border-radius: 12px;
        border: none;
    }

    .AdminButton {
        background-color: red;
        text-decoration-color: white;
        padding: 20px;
        text-decoration: none;
        height: 75px;
        width: 200px;
        font-family: Arial;
        font-size: medium;
        border-radius: 12px;
        border: none;
    }
    .EditButton {
        background-color: dodgerblue;
        padding: 10px;
        text-decoration-color: white;
        text-decoration: none;
        height: 65px;
        width: 80px;
        font-family: Arial;
        font-size: medium;
        border-radius: 12px;
        text-align: center;
        outline: none;
        outline: none;
    }

    .DetailButton {
        background-color: forestgreen;
        padding: 10px;
        text-decoration-color: white;
        text-decoration: none;
        height: 65px;
        width: 80px;
        font-family: Arial;
        font-size: Medium;
        border-radius: 12px;
        text-align: center;
    }

    .DeleteButton {
        background-color: red;
        padding: 10px;
        text-decoration-color: white;
        text-decoration: none;
        height: 65px;
        width: 80px;
        font-family: Arial;
        font-size: Medium;
        border-radius: 12px;
        text-align: center;
    }
```

```
    button {
        outline: none;
        border: none;
    }

    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }

    a:visited {
        color: white;
        text-decoration: none;
    }

    .BackButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 15px;
        text-decoration: none;
        height: 50px;
        width: 200px;
        font-family: Arial;
        font-size: 15px;
        border-radius: 12px;
        border: none;
        text-align: center;
    }
</style>

<h2>Admin Control Panel (Appointment Data)</h2>

<div class="CreateButton">
    <a>@Html.ActionLink("Create New", "Create")</a>
</div>

<h2>Booking/Appointment Data</h2>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.BookingDate)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TimeSlot)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Department)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DoctorName)
        </th>
        <th>
```

```
            @Html.DisplayNameFor(model => model.DoctorEmail)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.PatientName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.PatientEmail)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.BookingStatus)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Account.FullName)
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.BookingDate)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.TimeSlot)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Department)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.DoctorName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.DoctorEmail)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.PatientName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.PatientEmail)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.BookingStatus)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Account.FullName)
            </td>
            <td>
                <button class="EditButton">@Html.ActionLink("Edit", "Edit", new
{ id = item.BookingID })</button>
                <button class="DetailButton">@Html.ActionLink("Details",
"Details", new { id = item.BookingID })</button>
                <button class="DeleteButton">@Html.ActionLink("Delete", "Delete",
new { id = item.BookingID })</button>
            </td>
        </tr>
    }
</table>
```

```
<div>
    <button class="BackButton">@Html.ActionLink("Admin Control Panel", "Index",
"Admin")</button>
</div>
```

Hospital Admin Control Panel Create Page Design for Bookings/Appointments (Views/AdminTwo/Create.cshtml) (Source Code)

- It basically the Hospital Admin Control Panel webpage coding that is to design, style and implement the function of AdminTwoController to the Admin Control Panel webpage content for the admin to Create the user appointments/booking and store it to the Booking Database if needed. Instead of Creating on the database the admin can just create it on this webpage. Therefore, I use @model for this webpage so is linked to the Booking Database Entity Model.

```
@model Assignment_02.Models.Booking

@{
    ViewBag.Title = "Create";
}

<style>
    .BackButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: 15px;
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    button {
        outline: none;
        border: none;
    }

    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }
```

```
    a:visited {
        color: white;
        text-decoration: none;
    }
</style>

<script>
    function create_alert()
    {
        alert("Product has been created!.");
    }
</script>

<h2>Admin Control Panel (Create Appointment Data)</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Booking</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.BookingDate, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.BookingDate, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.BookingDate, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.TimeSlot, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.TimeSlot, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.TimeSlot, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Department, htmlAttributes: new { @class
= "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Department, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Department, "", new
{ @class = "text-danger" })
            </div>
        </div>
```

```
        <div class="form-group">
            @Html.LabelFor(model => model.DoctorName, htmlAttributes: new { @class
= "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.DoctorName, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.DoctorName, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.DoctorEmail, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.DoctorEmail, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.DoctorEmail, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.AccountID, "Doctor Account Name/ID",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("AccountID", null, htmlAttributes: new { @class
= "form-control" })
                @Html.ValidationMessageFor(model => model.AccountID, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.PatientName, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.PatientName, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.PatientName, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.PatientEmail, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.PatientEmail, new { htmlAttributes
= new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.PatientEmail, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.BookingStatus, htmlAttributes: new
{ @class = "control-label col-md-2" })
```

```
            <div class="col-md-10">
                @Html.DropDownListFor(model => model.BookingStatus, new
List<SelectListItem>
                 { new SelectListItem{Text="Pending", Value="Pending"},
                 new SelectListItem{Text="Confirm", Value="Confirm"}}, "Please
select the status", new { style = "width:250px; height: 50px;" })
                @Html.ValidationMessageFor(model => model.BookingStatus)
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input style="background-color:deepskyblue;color:white"
onclick="create_alert()" type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    <button class="BackButton">@Html.ActionLink("Back to List", "Index")</button>
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Hospital Admin Control Panel Delete Page Design for Bookings/Appointments (Views/AdminTwo/Delete.cshtml) (Source Code)

- It basically the Hospital Admin Control Panel webpage coding that is to design, style and implement the function of AdminTwoController to the Admin Control Panel webpage content for the admin to Delete the user appointments/booking from the Booking Database if needed. Instead of Deleting on the database the admin can just delete it on this webpage. Therefore, I use @model for this webpage so is linked to the Booking Database Entity Model.

```
@model Assignment_02.Models.Booking

@{
    ViewBag.Title = "Delete";
}

<style>
    .BackButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: 15px;
```

```
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    button {
        outline: none;
        border: none;
    }

    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }

    a:visited {
        color: white;
        text-decoration: none;
    }
</style>

<script>
    function delete_alert()
    {
        alert("You have delete the Account!.");
    }
</script>

<h2>Admin Control Panel (Delete Appointment Data)</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Booking</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.BookingDate)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.BookingDate)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.TimeSlot)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.TimeSlot)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Department)
```

```
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Department)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DoctorName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DoctorName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DoctorEmail)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DoctorEmail)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.PatientName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.PatientName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.PatientEmail)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.PatientEmail)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.BookingStatus)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.BookingStatus)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Account.FullName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Account.FullName)
        </dd>

    </dl>

    @using (Html.BeginForm())
    {
```

```
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input style="background-color:red;color:white"
onclick="delete_alert()" type="submit" value="Delete" class="btn btn-default" />
        </div>
    }

    <br />
    <div>
        <button class="BackButton">@Html.ActionLink("Back to List",
"Index")</button>
    </div>
</div>
```

Hospital Admin Control Panel Edit Page Design for Bookings/Appointments (Views/AdminTwo/Edit.cshtml) (Source Code)

- It basically the Hospital Admin Control Panel webpage coding that is to design, style and implement the function of AdminTwoController to the Admin Control Panel webpage content for the admin to Edit the user appointments/booking details from the Booking Database if needed. Instead of Editing on the database the admin can just edit it on this webpage. Therefore, I use @model for this webpage so is linked to the Booking Database Entity Model.

```
@model Assignment_02.Models.Booking

@{
    ViewBag.Title = "Edit";
}

<style>
    .BackButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: 15px;
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    button {
        outline: none;
        border: none;
    }
```

```
    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }

    a:visited {
        color: white;
        text-decoration: none;
    }
</style>

<script>
    function edit_alert() {
        alert("Product details/information has been updated!.");
    }
</script>

<h2>Admin Control Panel (Edit Appointment Data)</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Booking</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.BookingID)

        <div class="form-group">
            @Html.LabelFor(model => model.BookingDate, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.BookingDate, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.BookingDate, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.TimeSlot, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.TimeSlot, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.TimeSlot, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
```

```
            @Html.LabelFor(model => model.Department, htmlAttributes: new { @class
= "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Department, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Department, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.DoctorName, htmlAttributes: new { @class
= "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.DoctorName, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.DoctorName, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.DoctorEmail, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.DoctorEmail, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.DoctorEmail, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.AccountID, "Doctor Account Name/ID",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("AccountID", null, htmlAttributes: new { @class
= "form-control" })
                @Html.ValidationMessageFor(model => model.AccountID, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.PatientName, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.PatientName, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.PatientName, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.PatientEmail, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
```

```
                @Html.EditorFor(model => model.PatientEmail, new { htmlAttributes
= new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.PatientEmail, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.BookingStatus, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownListFor(model => model.BookingStatus, new
List<SelectListItem>
                { new SelectListItem{Text="Pending", Value="Pending"},
                 new SelectListItem{Text="Confirm", Value="Confirm"}}, "Please
select the status", new { style = "width:250px; height: 50px;" })
                @Html.ValidationMessageFor(model => model.BookingStatus)
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input style="background-color:green;color:white"
onclick="edit_alert()" type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    <button class="BackButton">@Html.ActionLink("Back to List", "Index")</button>
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Hospital Admin Control Panel Details Page Design for Bookings/Appointments (Views/AdminTwo/Details.cshtml) (Source Code)

- It basically the Hospital Admin Control Panel webpage coding that is to design, style and implement the function of AdminTwoController to the Admin Control Panel webpage content for the admin to view the specific user appointments/booking details from the Booking Database if needed. Instead of viewing on the database the admin can just view it on this webpage. Therefore, I use @model for this webpage so is linked to the Booking Database Entity Model.

```
@model Assignment_02.Models.Booking

@{
    ViewBag.Title = "Details";
```

```
}

<style>
    .BackButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: 15px;
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    .EditButton {
        background-color: green;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: medium;
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    .AppointmentDetails {
        font-size: 24px;
    }

    button {
        outline: none;
        border: none;
    }

    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }

    a:visited {
        color: white;
        text-decoration: none;
    }
</style>

<h2>Admin Control Panel (View Appointment Data)</h2>
```

```
<div class="AppointmentDetails">
    <h4>Booking</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.BookingDate)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.BookingDate)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.TimeSlot)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.TimeSlot)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Department)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Department)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DoctorName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DoctorName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DoctorEmail)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DoctorEmail)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.PatientName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.PatientName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.PatientEmail)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.PatientEmail)
```

```
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.BookingStatus)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.BookingStatus)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Account.FullName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Account.FullName)
        </dd>

    </dl>
</div>
<p>
    <button class="EditButton">@Html.ActionLink("Edit", "Edit", new { id =
Model.BookingID })</button>
    <button class="BackButton">@Html.ActionLink("Back to List", "Index")</button>
</p>
```

Hospital Doctor Control Panel Homepage Design (Views/Doctor/Index.cshtml) (Source Code)

- It basically the Hospital Doctor Control Panel webpage coding that is to design, style and implement the function of DoctorController to the Doctor Control Panel webpage content for the Doctors to View, Edit, Create, Delete the Patients Booking/Appointment from the Booking Database. Instead of updating on the database the doctor can just update it on this webpage. Therefore, I use @model for this webpage so is linked to the Booking Database Model.

```
@model IEnumerable<Assignment_02.Models.Booking>

@{
    ViewBag.Title = "Index";
}

<style>
    .CreateButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 15px;
        text-decoration: none;
        height: 65px;
        width: 140px;
        font-family: Arial;
        font-size: medium;
        border-radius: 12px;
        border: none;
```

```
    }

    .AdminButton {
        background-color: red;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 75px;
        width: 200px;
        font-family: Arial;
        font-size: medium;
        border-radius: 12px;
        border: none;
    }

    .EditButton {
        background-color: dodgerblue;
        padding: 5px;
        text-decoration-color: white;
        text-decoration: none;
        height: 65px;
        width: 80px;
        font-family: Arial;
        font-size: medium;
        border-radius: 12px;
        text-align: center;
        outline: none;
        outline: none;
    }

    .DetailButton {
        background-color: forestgreen;
        padding: 5px;
        text-decoration-color: white;
        text-decoration: none;
        height: 65px;
        width: 80px;
        font-family: Arial;
        font-size: Medium;
        border-radius: 12px;
        text-align: center;
    }

    .DeleteButton {
        background-color: red;
        padding: 5px;
        text-decoration-color: white;
        text-decoration: none;
        height: 65px;
        width: 80px;
        font-family: Arial;
        font-size: Medium;
        border-radius: 12px;
        text-align: center;
    }

    button {
        outline: none;
```

```
        border: none;
    }

    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }

    a:visited {
        color: white;
        text-decoration: none;
    }
</style>

<h2>Doctor System</h2>

<center>
    <div class="CreateButton">
        <a>@Html.ActionLink("Create New Appointment", "Create")</a>
    </div>
</center>
<br />

<h2>Appointments Data</h2>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.BookingID)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.BookingDate)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TimeSlot)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Department)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.AccountID)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DoctorName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DoctorEmail)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.PatientName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.PatientEmail)
        </th>
```

```
        <th>
            @Html.DisplayNameFor(model => model.BookingStatus)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Account.FullName)
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.BookingID)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.BookingDate)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.TimeSlot)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Department)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.AccountID)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.DoctorName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.DoctorEmail)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.PatientName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.PatientEmail)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.BookingStatus)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Account.FullName)
            </td>
            <td>
                <button class="EditButton">@Html.ActionLink("Edit", "Edit", new
{ id = item.BookingID })</button>
                <button class="DetailButton">@Html.ActionLink("Details",
"Details", new { id = item.BookingID })</button>
                <button class="DeleteButton">@Html.ActionLink("Delete", "Delete",
new { id = item.BookingID })</button>
            </td>
        </tr>
    }

</table>
```

Hospital Doctor Control Panel Homepage Design (Views/Doctor/Create.cshtml) (Source Code)

- It basically the Hospital Doctor Control Panel webpage coding that is to design, style and implement the function of DoctorController to the Doctor Control Panel webpage content for the Doctors Create the Patients Booking/Appointment and store to the Booking Database. Instead of Creating on the database itself the doctor can just create it on this webpage. Therefore, I use @model for this webpage so is linked to the Booking Database Model.

```
@model Assignment_02.Models.Booking

@{
    ViewBag.Title = "Create";
}

<style>
    .BackButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: 15px;
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    button {
        outline: none;
        border: none;
    }

    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }

    a:visited {
        color: white;
        text-decoration: none;
    }
</style>

<script>
```

```
    function create_alert() {
        alert("Appointment has been created!.");
    }
</script>

<h2>Doctor (Create Patient Appointment)</h2>

@Html.Raw(ViewBag.PatientAppointmentEmailMsg)
@Html.Raw(ViewBag.DoctorCreateAppointmentMsg)

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Booking</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.BookingDate, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.BookingDate, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.BookingDate, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.TimeSlot, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.TimeSlot, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.TimeSlot, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Department, htmlAttributes: new { @class
= "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Department, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Department, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.DoctorName, htmlAttributes: new { @class
= "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.DoctorName, new { htmlAttributes =
new { @class = "form-control" } })
```

```
                @Html.ValidationMessageFor(model => model.DoctorName, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.DoctorEmail, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.DoctorEmail, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.DoctorEmail, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.AccountID, "Doctor Account Name/ID",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("AccountID", null, htmlAttributes: new { @class
= "form-control" })
                @Html.ValidationMessageFor(model => model.AccountID, "", new
{ @class = "text-danger" });
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.PatientName, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.PatientName, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.PatientName, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.PatientEmail, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.PatientEmail, new { htmlAttributes
= new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.PatientEmail, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.BookingStatus, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownListFor(model => model.BookingStatus, new
List<SelectListItem>
         { new SelectListItem{Text="Pending", Value="Pending"},
           new SelectListItem{Text="Confirm", Value="Confirm"}}, "Please select
the status", new { style = "width:250px; height: 50px;" })
```

```
                @Html.ValidationMessageFor(model => model.BookingStatus)
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input style="background-color:deepskyblue;color:white"
onclick="create_alert()" type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    <button class="BackButton">@Html.ActionLink("Back to List", "Index")</button>
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Hospital Doctor Control Panel Delete Page Design (Views/Doctor/Delete.cshtml) (Source Code)

- It basically the Hospital Doctor Control Panel webpage coding that is to design, style and implement the function of DoctorController to the Doctor Control Panel webpage content for the Doctors to Delete the Patients Booking/Appointment from the Booking Database if needed. Instead of deleting on the database the doctor can just delete it from this webpage. Therefore, I use @model for this webpage so is linked to the Booking Database Model.

```
@model Assignment_02.Models.Booking

@{
    ViewBag.Title = "Delete";
}

<style>
    .BackButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: 15px;
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    button {
```

```
        outline: none;
        border: none;
    }

    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }

    a:visited {
        color: white;
        text-decoration: none;
    }
</style>

<script>
    function delete_alert()
    {
        alert("You have delete the Appointment!.");
    }
</script>

<h2>Doctor (Delete Patient Appointment)</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Booking</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.BookingDate)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.BookingDate)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.TimeSlot)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.TimeSlot)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Department)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Department)
        </dd>
```

```
        <dt>
            @Html.DisplayNameFor(model => model.DoctorName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DoctorName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DoctorEmail)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DoctorEmail)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.PatientName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.PatientName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.PatientEmail)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.PatientEmail)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.BookingStatus)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.BookingStatus)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Account.FullName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Account.FullName)
        </dd>

    </dl>

    @using (Html.BeginForm())
    {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input  style="background-color:red;color:white"
onclick="delete_alert()" type="submit" value="Delete" class="btn btn-default" />
        </div>
```

```
    }

    <br />
    <div>
        <button class="BackButton">@Html.ActionLink("Back to List",
"Index")</button>
    </div>
</div>
```

Hospital Doctor Control Panel Edit Page Design (Views/Doctor/Edit.cshtml) (Source Code)

- It basically the Hospital Doctor Control Panel webpage coding that is to design, style and implement the function of DoctorController to the Doctor Control Panel webpage content for the Doctors to Edit the Patients Booking/Appointment from the Booking Database. Instead of editing it on the database the doctor can just edit it from this webpage. Therefore, I use @model for this webpage so is linked to the Booking Database Model.

```
@model Assignment_02.Models.Booking

@{
    ViewBag.Title = "Edit";
}

<style>
    .BackButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: 15px;
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    button {
        outline: none;
        border: none;
    }

    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
```

```
    }

    a:visited {
        color: white;
        text-decoration: none;
    }
</style>

<script>
    function edit_alert() {
        alert("Patient Appointment details/information has been updated!.");
    }
</script>

<h2>Doctor Patient Schedule (Edit)</h2>

@Html.Raw(ViewBag.DoctorEditMsg)
@Html.Raw(ViewBag.DoctorEditMsg2)
@Html.Raw(ViewBag.SameEmailMsg2)

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Booking</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.BookingID)

        <div class="form-group">
            @Html.LabelFor(model => model.BookingDate, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.BookingDate, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.BookingDate, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.TimeSlot, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.TimeSlot, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.TimeSlot, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Department, htmlAttributes: new { @class
= "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Department, new { htmlAttributes =
new { @class = "form-control" } })
```

```
                @Html.ValidationMessageFor(model => model.Department, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.DoctorName, htmlAttributes: new { @class
= "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.DoctorName, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.DoctorName, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.DoctorEmail, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.DoctorEmail, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.DoctorEmail, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.AccountID, "Doctor Account Name/ID",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("AccountID", null, htmlAttributes: new { @class
= "form-control" })
                @Html.ValidationMessageFor(model => model.AccountID, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.PatientName, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.PatientName, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.PatientName, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.PatientEmail, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.PatientEmail, new { htmlAttributes
= new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.PatientEmail, "", new
{ @class = "text-danger" })
            </div>
```

```
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.BookingStatus, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownListFor(model => model.BookingStatus, new
List<SelectListItem>
           { new SelectListItem{Text="Pending", Value="Pending"},
             new SelectListItem{Text="Confirm", Value="Confirm"}}, "Please select
the status", new { style = "width:250px; height: 50px;" })
                @Html.ValidationMessageFor(model => model.BookingStatus)
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input style="background-color:green;color:white"
onclick="edit_alert()" type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}


<div>
    <button class="BackButton">@Html.ActionLink("Back to List", "Index")</button>
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Hospital Doctor Control Panel Details Page Design (Views/Doctor/Details.cshtml) (Source Code)

- It basically the Hospital Doctor Control Panel webpage coding that is to design, style and implement the function of DoctorController to the Doctor Control Panel webpage content for the Doctors to View the specific Patient Booking/Appointment details from the Booking Database. Instead of viewing it on the database the doctor can just view it from this webpage as a list of data. Therefore, I use @model for this webpage so is linked to the Booking Database Model.

```
@model Assignment_02.Models.Booking

@{
    ViewBag.Title = "Details";
}

<style>
    .BackButton {
```

```
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: 15px;
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    .EditButton {
        background-color: green;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: medium;
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    .AppointmentDetails {
        font-size: 24px;
    }

    button {
        outline: none;
        border: none;
    }

    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }

    a:visited {
        color: white;
        text-decoration: none;
    }
</style>

<h2>Doctor (Patient Appointment Details)</h2>

<div class="AppointmentDetails">
    <h4>Booking</h4>
    <hr />
    <dl class="dl-horizontal">
```

```
        <dt>
            @Html.DisplayNameFor(model => model.BookingDate)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.BookingDate)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.TimeSlot)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.TimeSlot)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Department)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Department)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DoctorName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DoctorName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DoctorEmail)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DoctorEmail)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.PatientName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.PatientName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.PatientEmail)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.PatientEmail)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.BookingStatus)
```

```
        </dt>

        <dd>
            @Html.DisplayFor(model => model.BookingStatus)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Account.FullName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Account.FullName)
        </dd>

    </dl>
</div>
<p>
    <button class="EditButton">@Html.ActionLink("Edit", "Edit", new { id =
Model.BookingID })</button>
    <button class="BackButton">@Html.ActionLink("Back to List", "Index")</button>
</p>
```

Hospital Patients Control Panel Homepage Design (Views/Patient/Index.cshtml) (Source Code)

- It basically the Hospital Patients Control Panel webpage coding that is to design, style and implement the function of PatientController to the Patient Control Panel webpage content for the Patients to View, Edit, Create, Delete their Booking/Appointment from the Booking Database. Instead of updating on the database the patients can just update it on this webpage. Therefore, I use @model for this webpage so is linked to the Booking Database Model.

```
@model IEnumerable<Assignment_02.Models.Booking>

@{
    ViewBag.Title = "Index";
}

<style>
    .CreateButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 15px;
        text-decoration: none;
        height: 65px;
        width: 140px;
        font-family: Arial;
        font-size: medium;
        border-radius: 12px;
        border: none;
    }

    .AdminButton {
```

```
        background-color: red;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 75px;
        width: 200px;
        font-family: Arial;
        font-size: medium;
        border-radius: 12px;
        border: none;
    }

    .EditButton {
        background-color: dodgerblue;
        padding: 5px;
        text-decoration-color: white;
        text-decoration: none;
        height: 65px;
        width: 80px;
        font-family: Arial;
        font-size: medium;
        border-radius: 12px;
        text-align: center;
        outline: none;
        outline: none;
    }

    .DetailButton {
        background-color: forestgreen;
        padding: 5px;
        text-decoration-color: white;
        text-decoration: none;
        height: 65px;
        width: 80px;
        font-family: Arial;
        font-size: Medium;
        border-radius: 12px;
        text-align: center;
    }

    .DeleteButton {
        background-color: red;
        padding: 5px;
        text-decoration-color: white;
        text-decoration: none;
        height: 65px;
        width: 80px;
        font-family: Arial;
        font-size: Medium;
        border-radius: 12px;
        text-align: center;
    }

    button {
        outline: none;
        border: none;
    }
```

```
    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }

    a:visited {
        color: white;
        text-decoration: none;
    }
</style>

<h2>Patients Appointment</h2>

<center>
<div class="CreateButton">
    <a>@Html.ActionLink("Create New Appointment", "Create")</a>
</div>
</center>
<br />

<h2>Appointments Data</h2>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.BookingID)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.BookingDate)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TimeSlot)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Department)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DoctorName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DoctorEmail)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.PatientName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.PatientEmail)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.BookingStatus)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Account.FullName)
        </th>
```

```
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.BookingID)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.BookingDate)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.TimeSlot)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Department)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.DoctorName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.DoctorEmail)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.PatientName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.PatientEmail)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.BookingStatus)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Account.FullName)
            </td>
            <td>
                <button class="EditButton">@Html.ActionLink("Edit", "Edit", new
{ id = item.BookingID })</button>
                <button class="DetailButton">@Html.ActionLink("Details",
"Details", new { id = item.BookingID })</button>
                <button class="DeleteButton">@Html.ActionLink("Delete", "Delete",
new { id = item.BookingID })</button>
            </td>
        </tr>
    }

</table>
```

Hospital Patients Control Panel Create Appointment/Booking page Design (Views/Patient/Create.cshtml) (Source Code)

- It basically the Hospital Patients Control Panel webpage coding that is to design, style and implement the function of PatientController to the Patient Control Panel

webpage content for the Patients to Create their Booking/Appointment for themselves and store them to the Hospital Booking Database once they are done with creating the appointments/booking. Instead of updating themselves on the database the patients can just update it on this webpage. Therefore, I use @model for this webpage so is linked to the Booking Database Model.

```
@model Assignment_02.Models.Booking

@{
    ViewBag.Title = "Create";
}

<style>
    .BackButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: 15px;
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    button {
        outline: none;
        border: none;
    }

    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }

    a:visited {
        color: white;
        text-decoration: none;
    }
</style>

<script>
    function create_alert() {
        alert("Appointment has been created!.");
    }
</script>


<h2>Create Appointment Booking</h2>
```

```
@Html.Raw(ViewBag.SameEmailBookingMsg)
@Html.Raw(ViewBag.CreateAppointmentMsg)

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Booking</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.BookingDate, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.BookingDate, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.BookingDate, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.TimeSlot, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.TimeSlot, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.TimeSlot, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Department, htmlAttributes: new { @class
= "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Department, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Department, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.PatientName, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.PatientName, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.PatientName, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
```

```
            @Html.LabelFor(model => model.PatientEmail, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.PatientEmail, new { htmlAttributes
= new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.PatientEmail, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input style="background-color:deepskyblue;color:white"
onclick="create_alert()" type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    <button class="BackButton">@Html.ActionLink("Back to List", "Index")</button>
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Hospital Patients Control Panel Delete Appointment/Booking page (Views/Patient/Delete.cshtml) (Source Code)

- It basically the Hospital Patients Control Panel webpage coding that is to design, style and implement the function of PatientController to the Patients Control Panel webpage content for the Patients to Delete their appointments/booking from the Hospital Booking/Appointment Database itself. Instead of Deleting on the database the Patients can just Delete it on this webpage. Therefore, I use @model for this webpage so is linked to the Booking Database Entity Model.

```
@model Assignment_02.Models.Booking

@{
    ViewBag.Title = "Delete";
}

<style>
    .BackButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: 15px;
```

```
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    button {
        outline: none;
        border: none;
    }

    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }

    a:visited {
        color: white;
        text-decoration: none;
    }
</style>

<script>
    function delete_alert()
    {
        alert("You have delete the Appointment Booking.");
    }
</script>


<h2>Delete Appointment Booking</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Appointment Booking</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.BookingDate)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.BookingDate)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.TimeSlot)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.TimeSlot)
        </dd>

        <dt>
```

```
            @Html.DisplayNameFor(model => model.Department)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Department)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DoctorName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DoctorName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DoctorEmail)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DoctorEmail)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.PatientName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.PatientName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.PatientEmail)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.PatientEmail)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.BookingStatus)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.BookingStatus)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Account.FullName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Account.FullName)
        </dd>

    </dl>

    @using (Html.BeginForm())
```

```
    {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input style="background-color:red;color:white"
onclick="delete_alert()" type="submit" value="Delete" class="btn btn-default" />
        </div>
    }
    <br />
    <div>
        <button class="BackButton">@Html.ActionLink("Back to List",
"Index")</button>
    </div>
</div>
```

Hospital Patients Control Panel Edit Appointment/Booking page (Views/Patient/Edit.cshtml) (Source Code)

- It basically the Hospital Patients Control Panel webpage coding that is to design, style and implement the function of PatientController to the Patients Control Panel webpage content for the Patients to Edit their appointments/booking details from the Hospital Booking/Appointment Database itself. Instead of Editing on the database the Patients can just Edit/Update their appointments/booking details it on this webpage. Therefore, I use @model for this webpage so is linked to the Booking Database Entity Model.

```
@model Assignment_02.Models.Booking

@{
    ViewBag.Title = "Edit";
}

<style>
    .BackButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: 15px;
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    button {
        outline: none;
        border: none;
    }
```

```
    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }

    a:visited {
        color: white;
        text-decoration: none;
    }
</style>

<script>
    function edit_alert() {
        alert("Appointment details/information has been updated!.");
    }
</script>

<h2>Edit Patient Appointment</h2>

@Html.Raw(ViewBag.PatientEditMsg)
@Html.Raw(ViewBag.PatientEditMsg2)
@Html.Raw(ViewBag.SameEmailMsg3)

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Booking</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.BookingID)

        <div class="form-group">
            @Html.LabelFor(model => model.BookingDate, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.BookingDate, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.BookingDate, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.TimeSlot, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.TimeSlot, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.TimeSlot, "", new
{ @class = "text-danger" })
            </div>
```

```
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Department, htmlAttributes: new { @class
= "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Department, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Department, "", new
{ @class = "text-danger" })
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.PatientName, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.PatientName, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.PatientName, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.PatientEmail, htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.PatientEmail, new { htmlAttributes
= new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.PatientEmail, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input style="background-color:green;color:white"
onclick="edit_alert()" type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    <button class="BackButton">@Html.ActionLink("Back to List", "Index")</button>
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Hospital Patients Control Panel Details Appointment/Booking page (Views/Patient/Details.cshtml) (Source Code)

- It basically the Hospital Patients Control Panel webpage coding that is to design, style and implement the function of PatientController to the Patients Control Panel webpage content for the Patients to View their appointments/booking details from the Hospital Booking/Appointment Database itself. Instead of Viewing on the database the Patients can just View their appointments/booking details it on this webpage. Therefore, I use @model for this webpage so is linked to the Booking Database Entity Model.

```
@model Assignment_02.Models.Booking

@{
    ViewBag.Title = "Details";
}

<style>
    .BackButton {
        background-color: dodgerblue;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: 15px;
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    .EditButton {
        background-color: green;
        text-decoration-color: white;
        padding: 10px;
        text-decoration: none;
        height: 60px;
        width: 120px;
        font-family: Arial;
        font-size: medium;
        border-radius: 12px;
        border: none;
        text-align: center;
    }

    .BookingDetail {
        font-size: 24px;
    }

    button {
        outline: none;
        border: none;
    }
```

```
    a:link {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: black;
        text-decoration: none;
    }

    a:visited {
        color: white;
        text-decoration: none;
    }
</style>



<h2>Appointments Details</h2>

<div class="BookingDetail">
    <h4>Booking</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.BookingDate)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.BookingDate)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.TimeSlot)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.TimeSlot)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Department)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Department)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DoctorName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DoctorName)
        </dd>

        <dt>
```

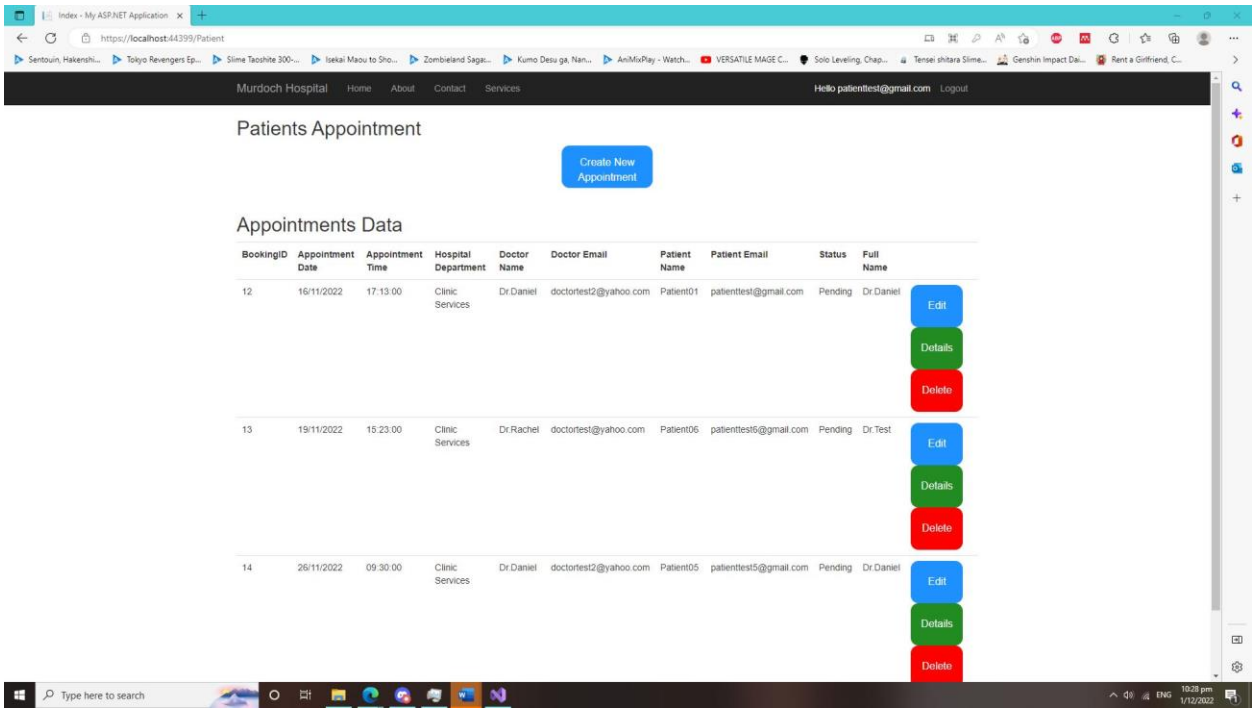
```
            @Html.DisplayNameFor(model => model.DoctorEmail)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DoctorEmail)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.PatientName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.PatientName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.PatientEmail)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.PatientEmail)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.BookingStatus)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.BookingStatus)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Account.FullName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Account.FullName)
        </dd>

    </dl>
</div>
<p>
    <button class="EditButton">@Html.ActionLink("Edit", "Edit", new { id =
Model.BookingID })</button>
    <button class="BackButton">@Html.ActionLink("Back to List", "Index")</button>
</p>
```

Unit/Program Testing Output (Screenshot) and Text

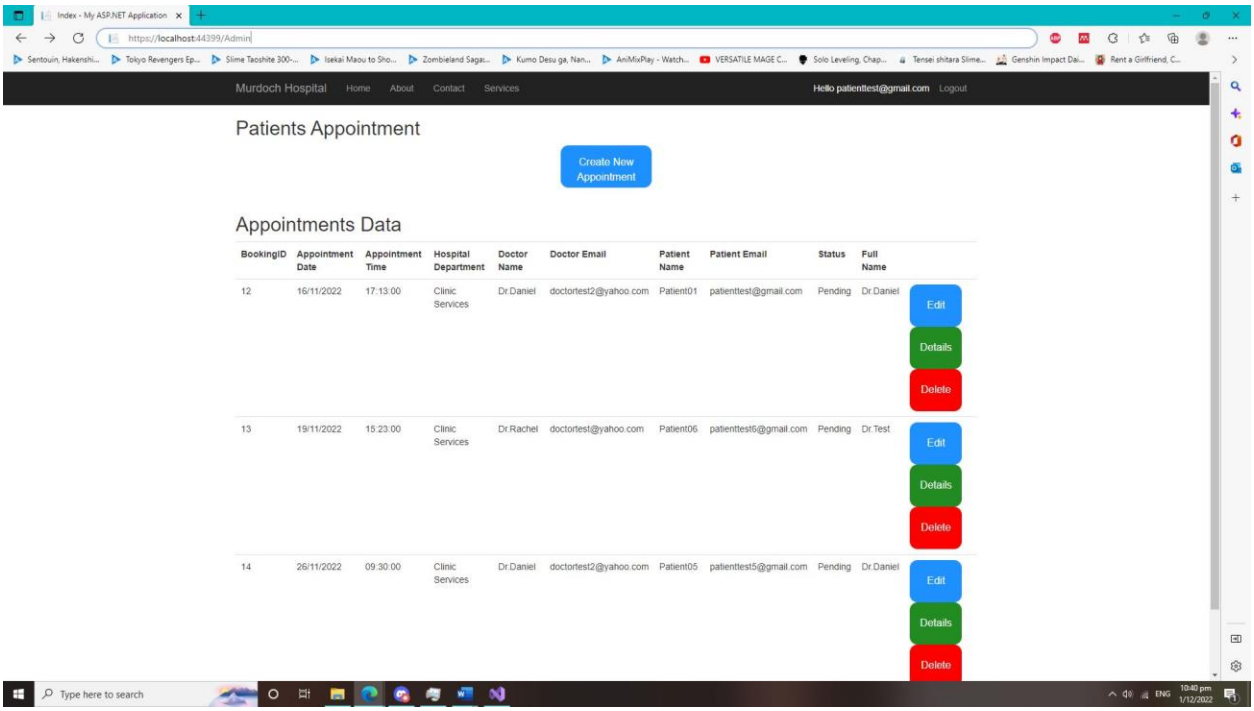
Bypass Webpage Test Using Webpage Link.

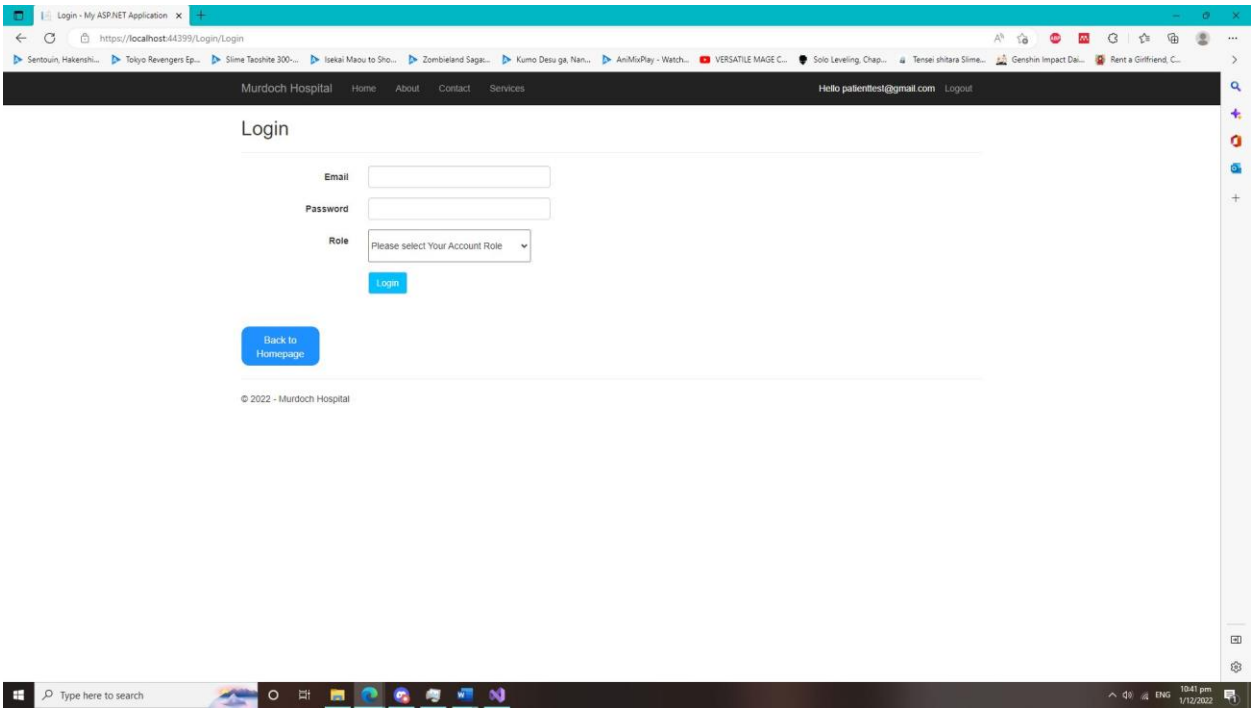
- This Test is to prevent Account Role Bypass using the webpage URL of Admin/Doctor/Patient Webpage Links. So that the non-authorize user won't be able to access those links and even if they try if they will be straight redirect them to the login page. Example Test 1: Patient Account Role Trying to Access to Admin webpage using the admin webpage URL.

Patient Webpage (Login by Patienttest@gmail.com)



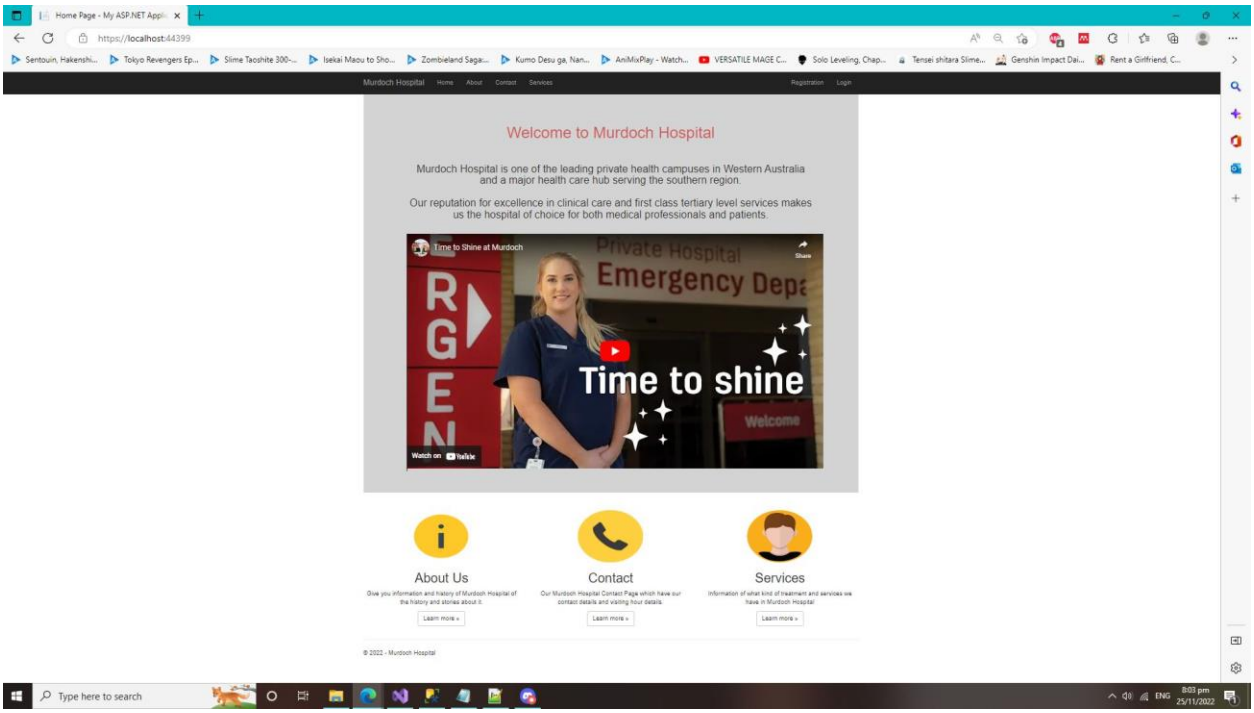
The Patient trying to access the Admin Webpage using this URL: https://localhost:44399/Admin. They will get redirect it back to the Login Page.





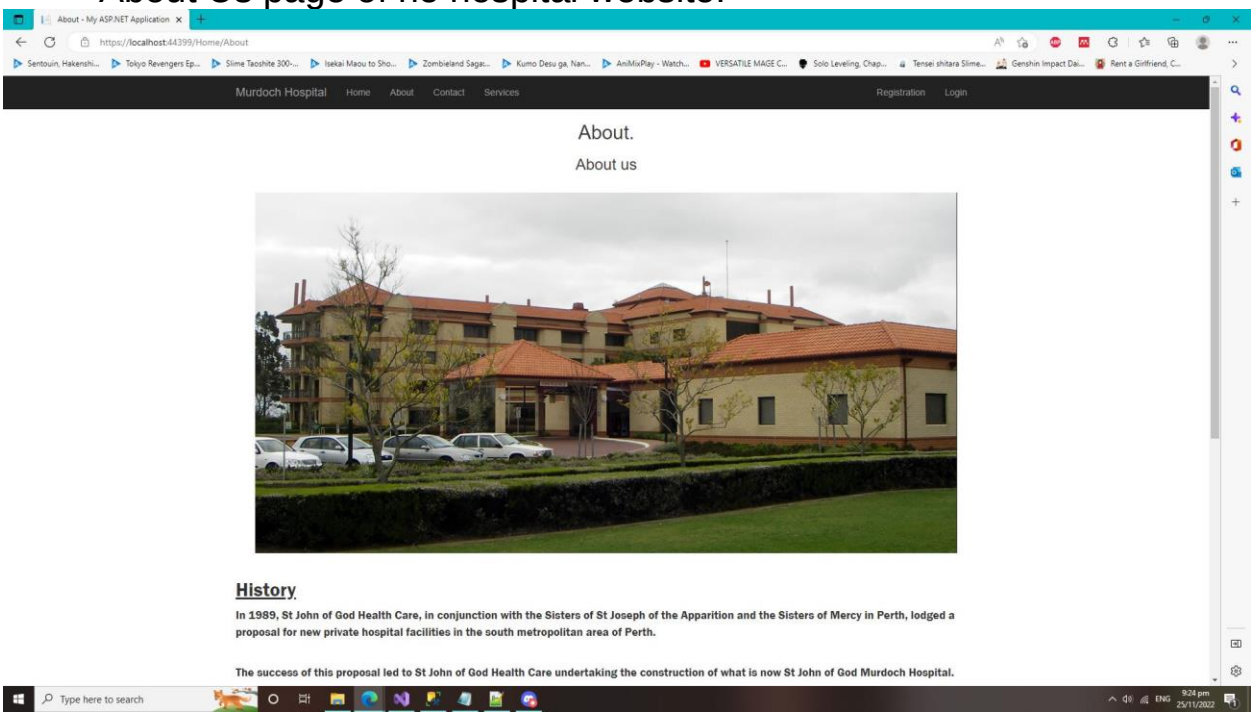
Homepage Test (Screenshot) and Text

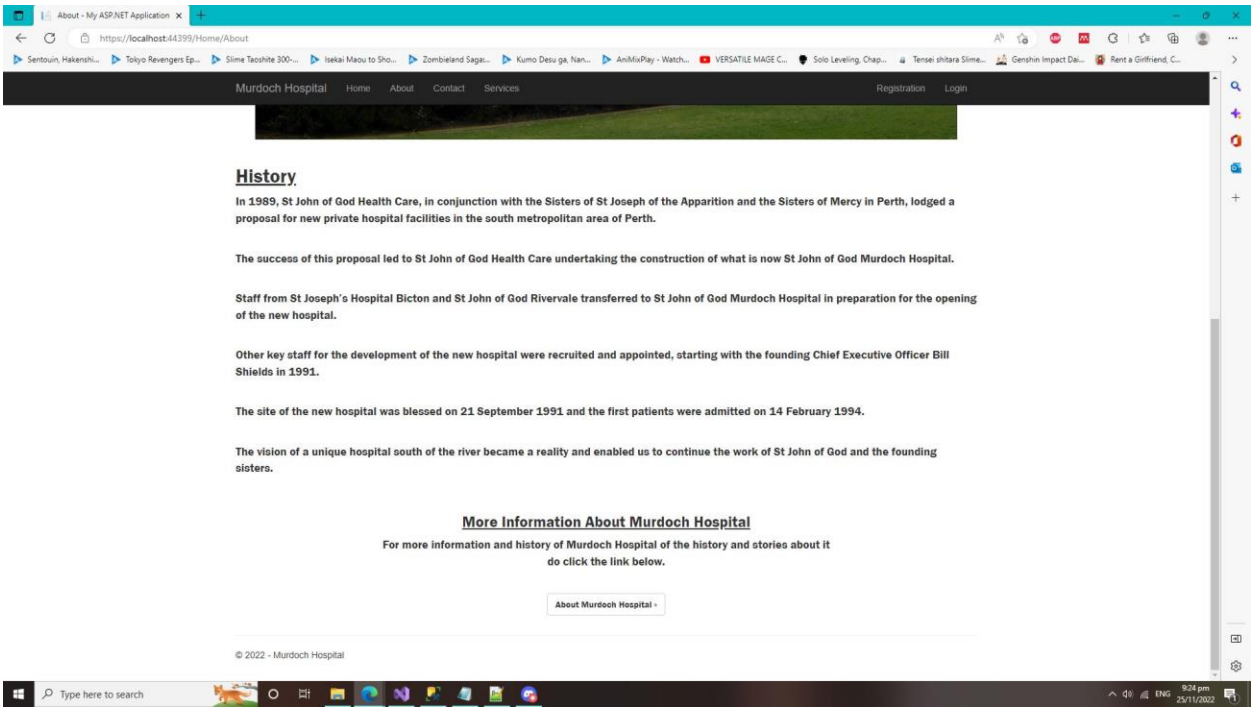
- Homepage of the Hospital Website.



About Us page Test (Screenshot) and Text

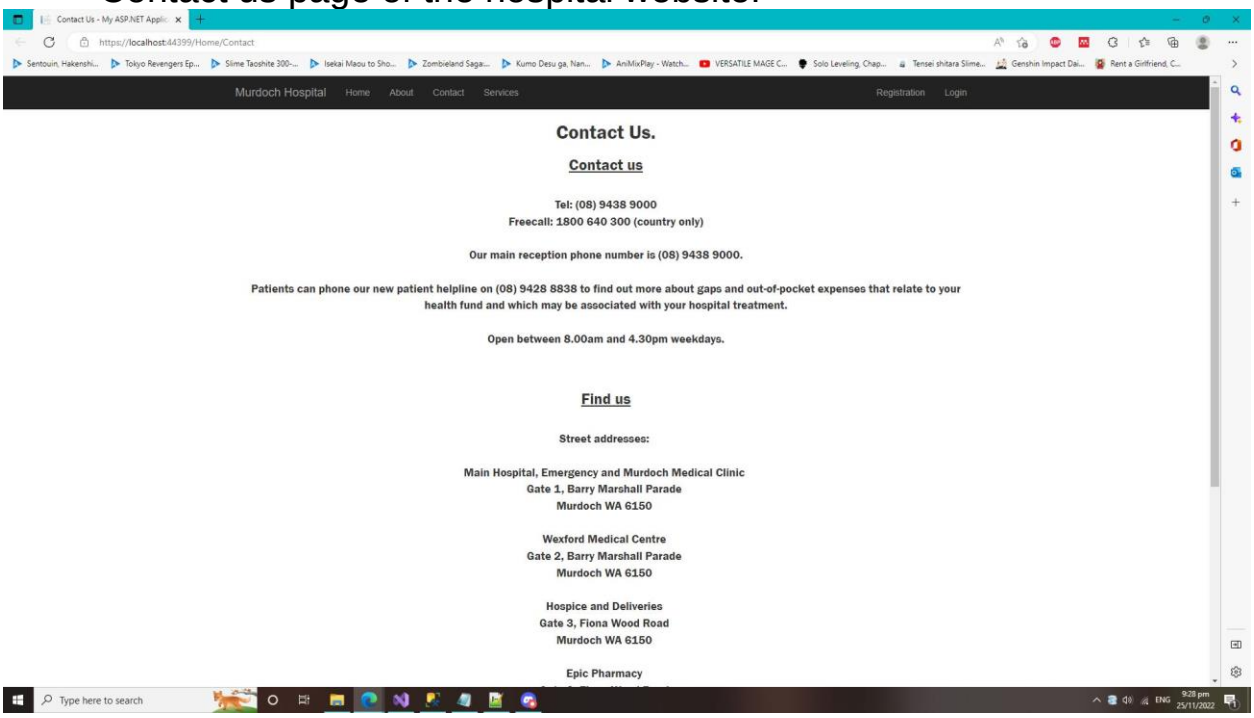
- About Us page of he hospital website.

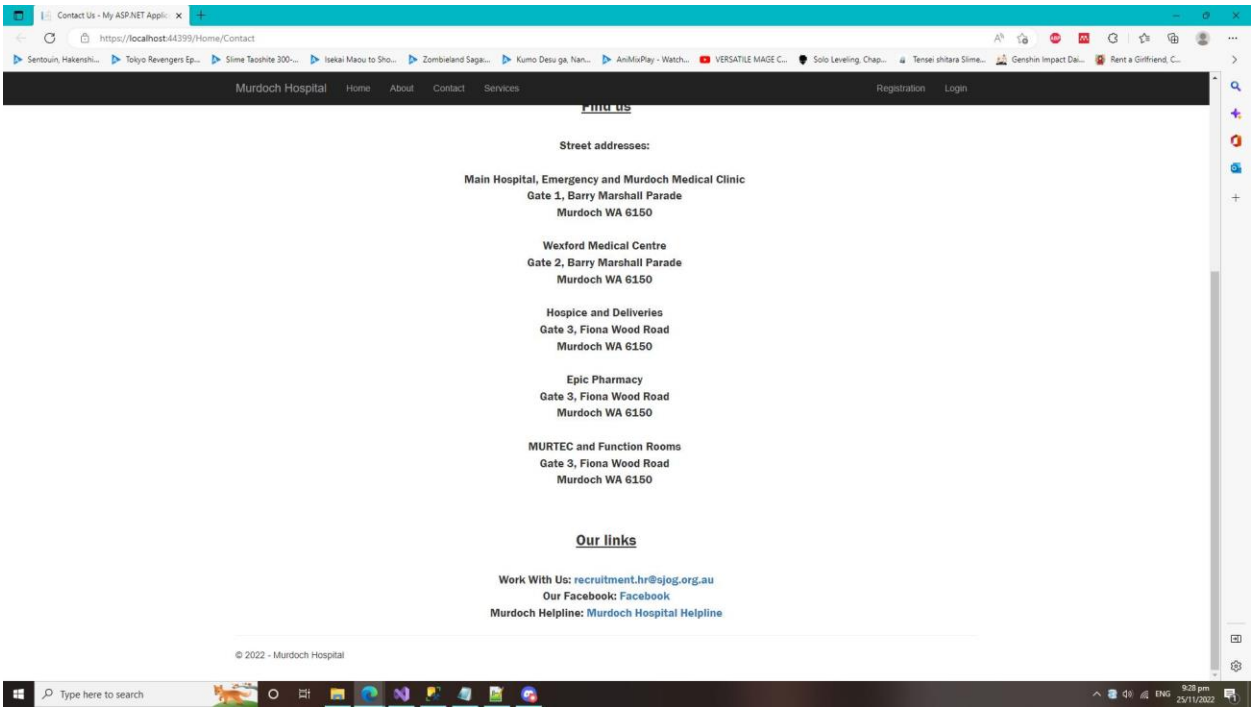




Contact Us Page Test (Screenshot) and Text

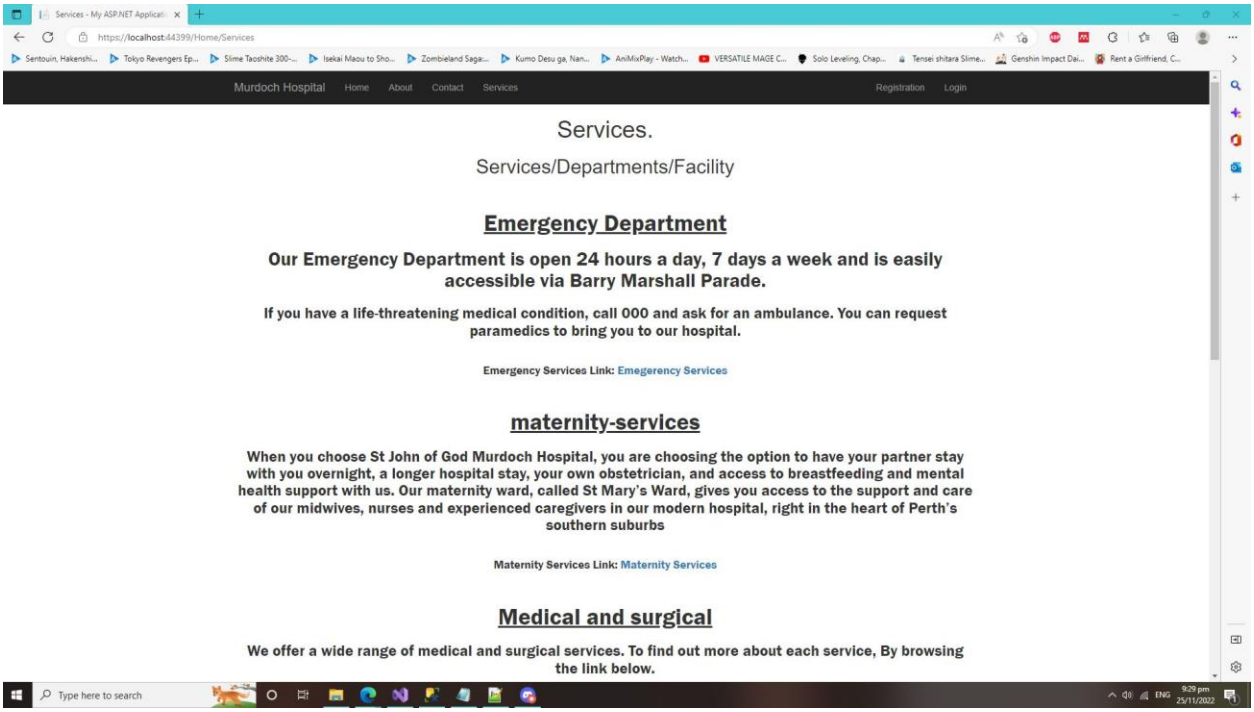
- Contact us page of the hospital website.

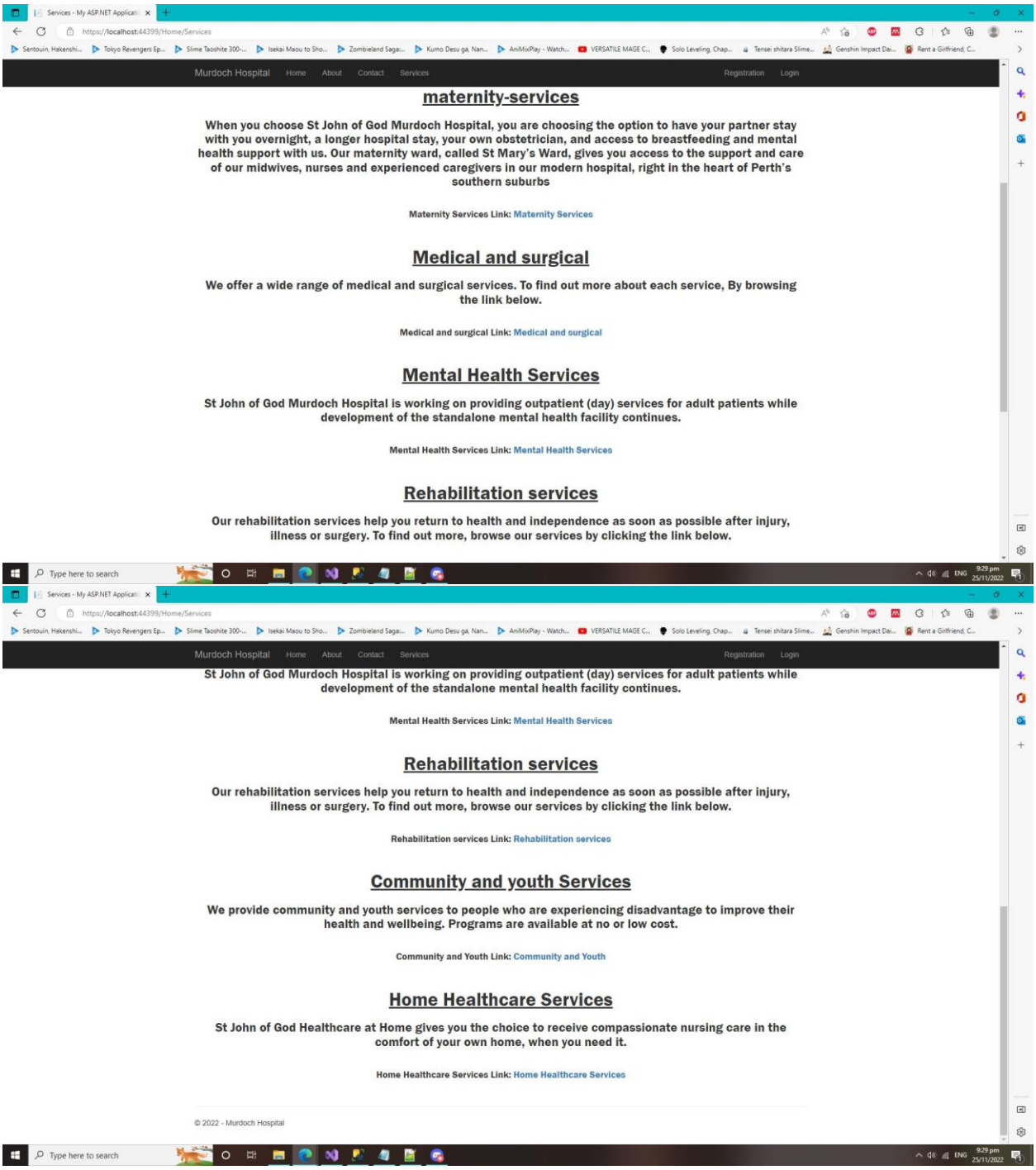




Services Webpage Test (Screenshot) and Text

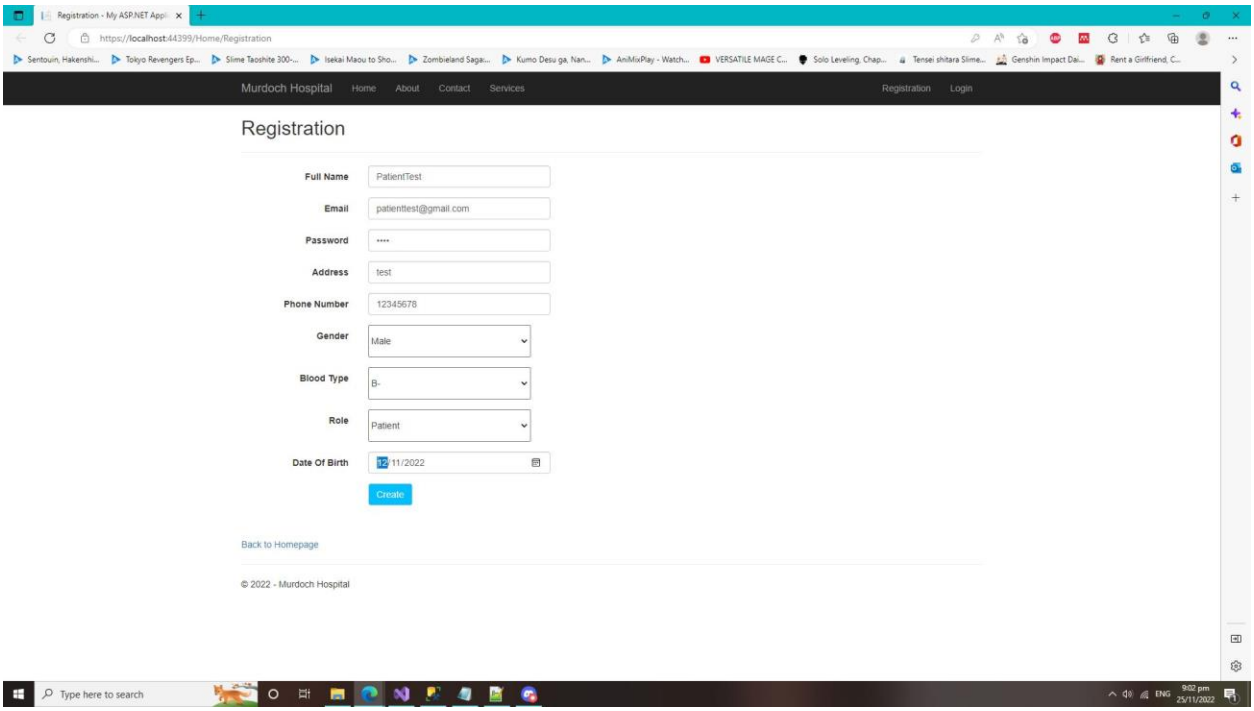
- Services Page of the Hospital Website.



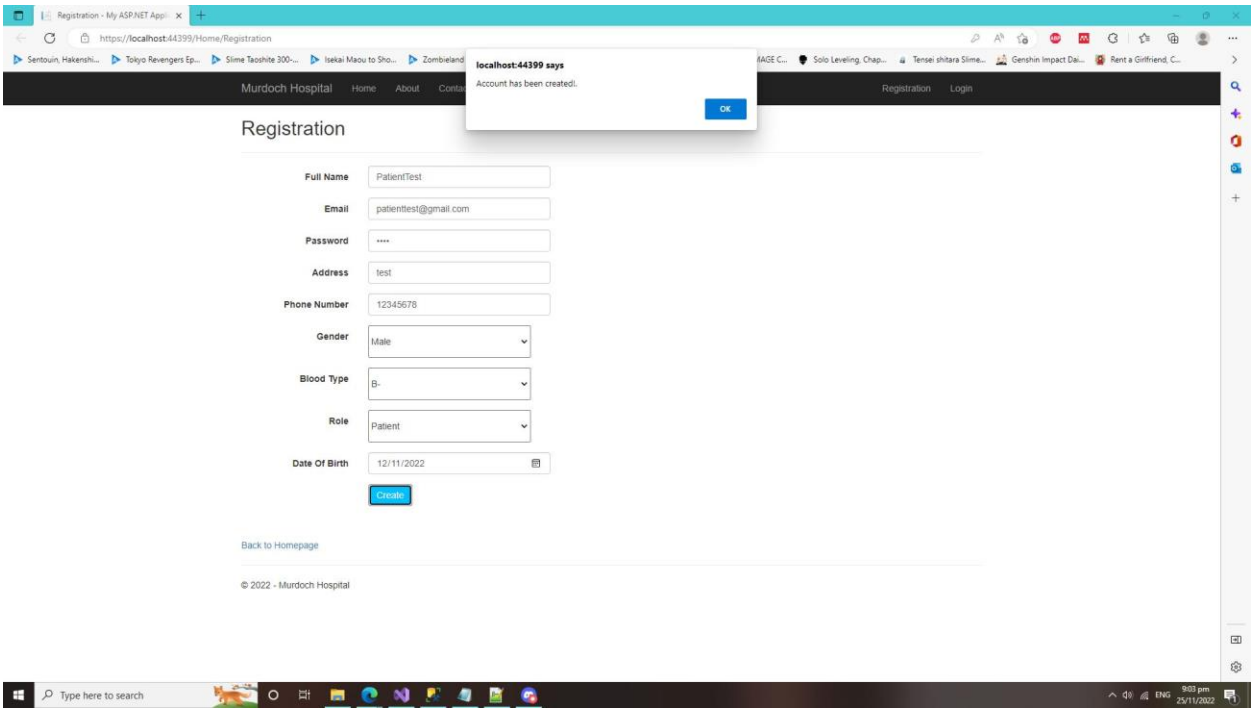


Registration Account Test (Screenshot) and Text

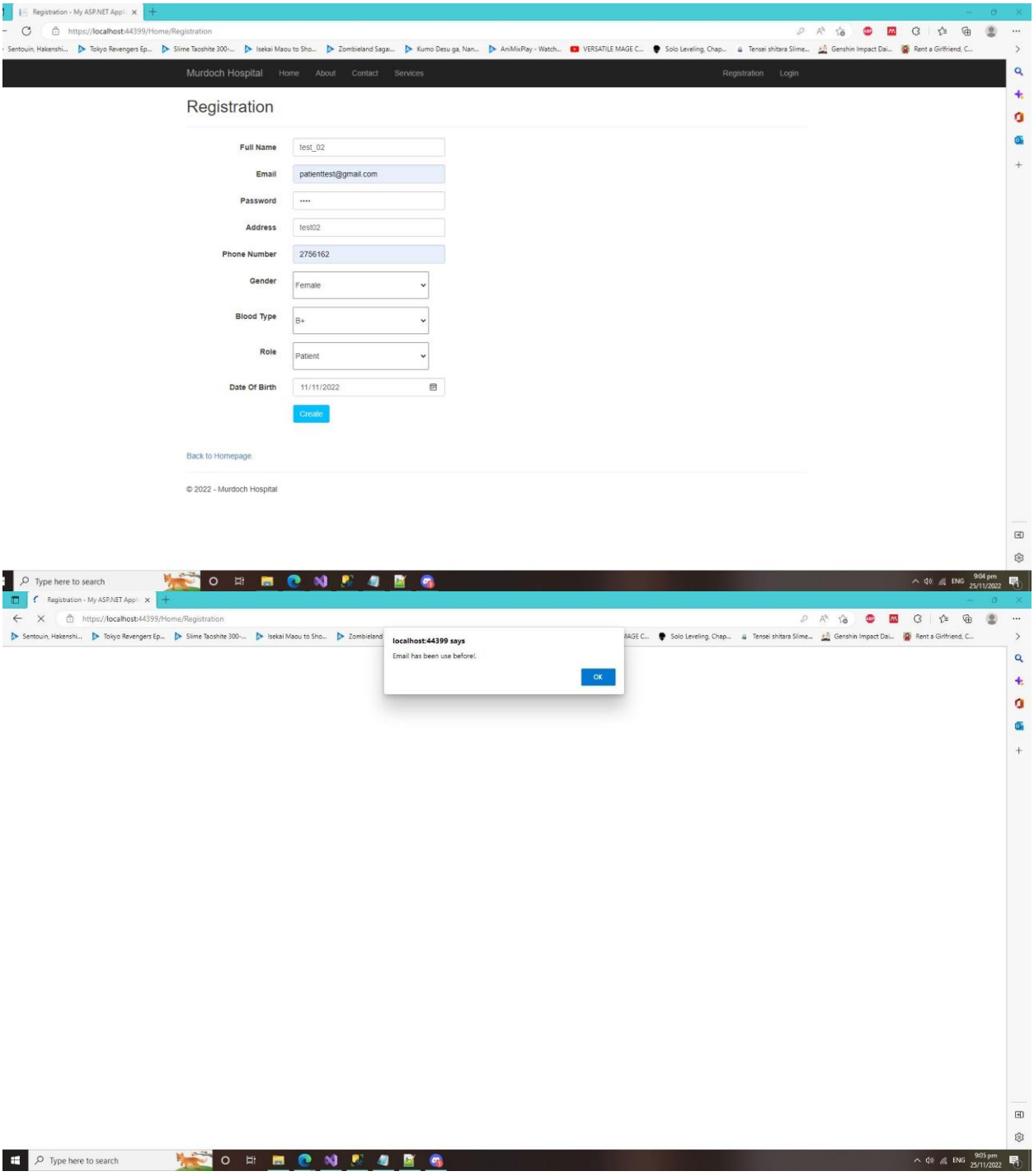
- Let the user create a new account with specific of roles, so that the user can login to use the Hospital System webpage.



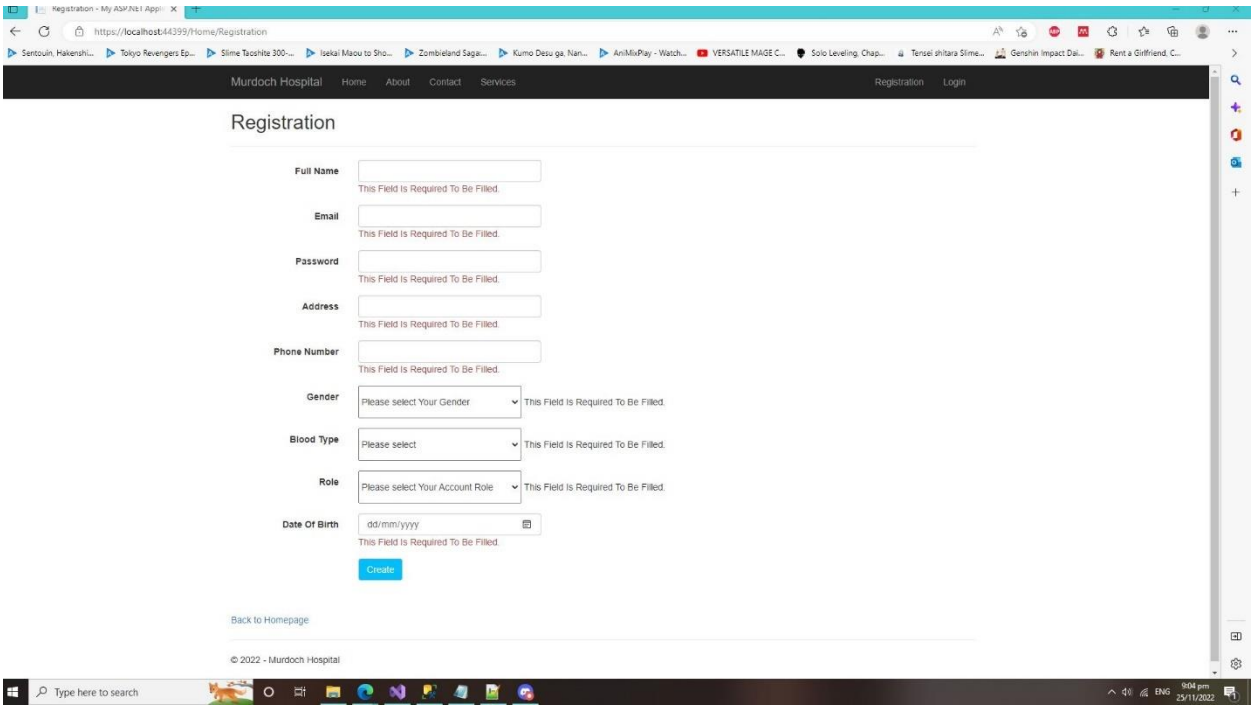
- Create New Account Successfully Check Test (Patient Account)



- Validation checks to check if the account “Email” if it is existed. If it is exist inside the hospital database. The user won’t be able to use the Same “Email” to register for the account.

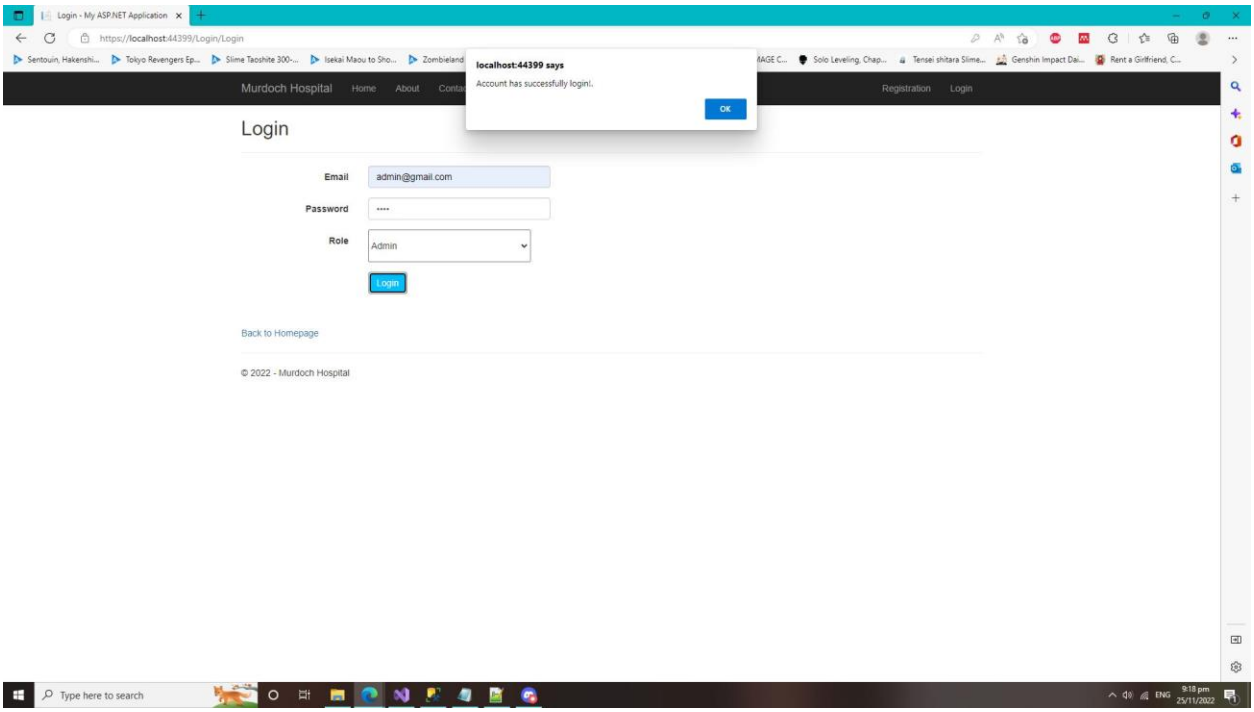


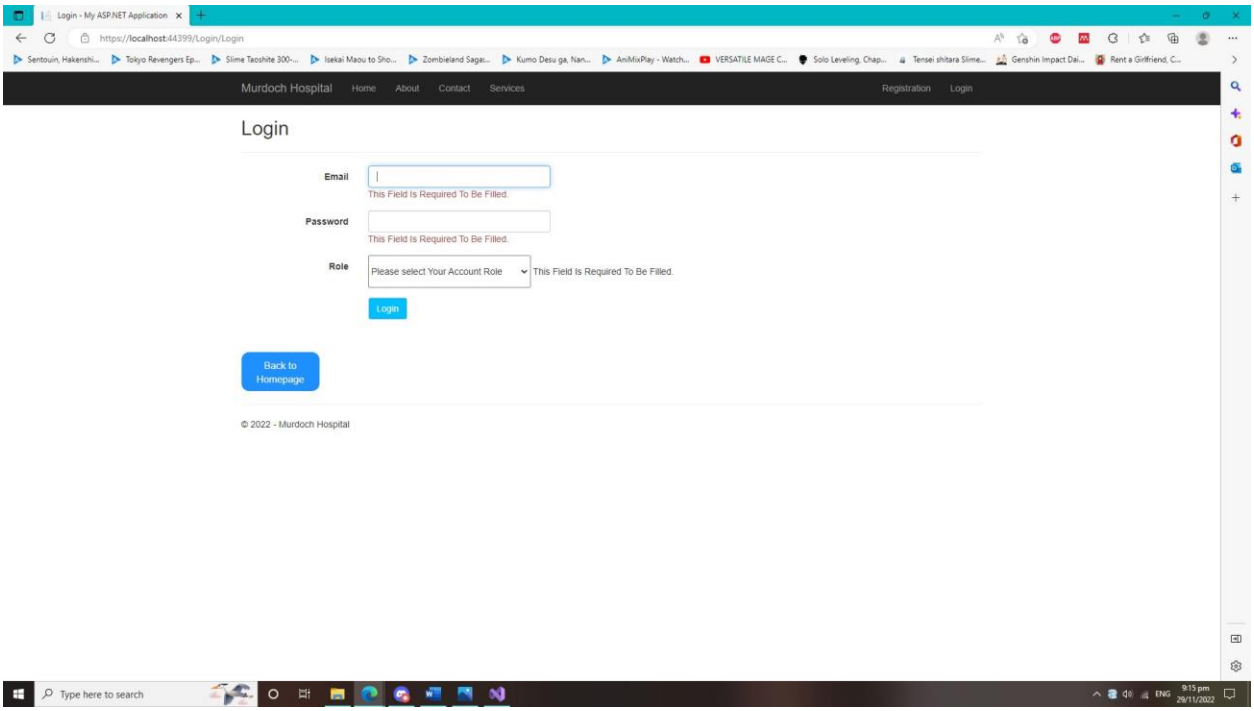
- Validation Check for all the registration form input must be fill in, if not it will show an error message to the user.



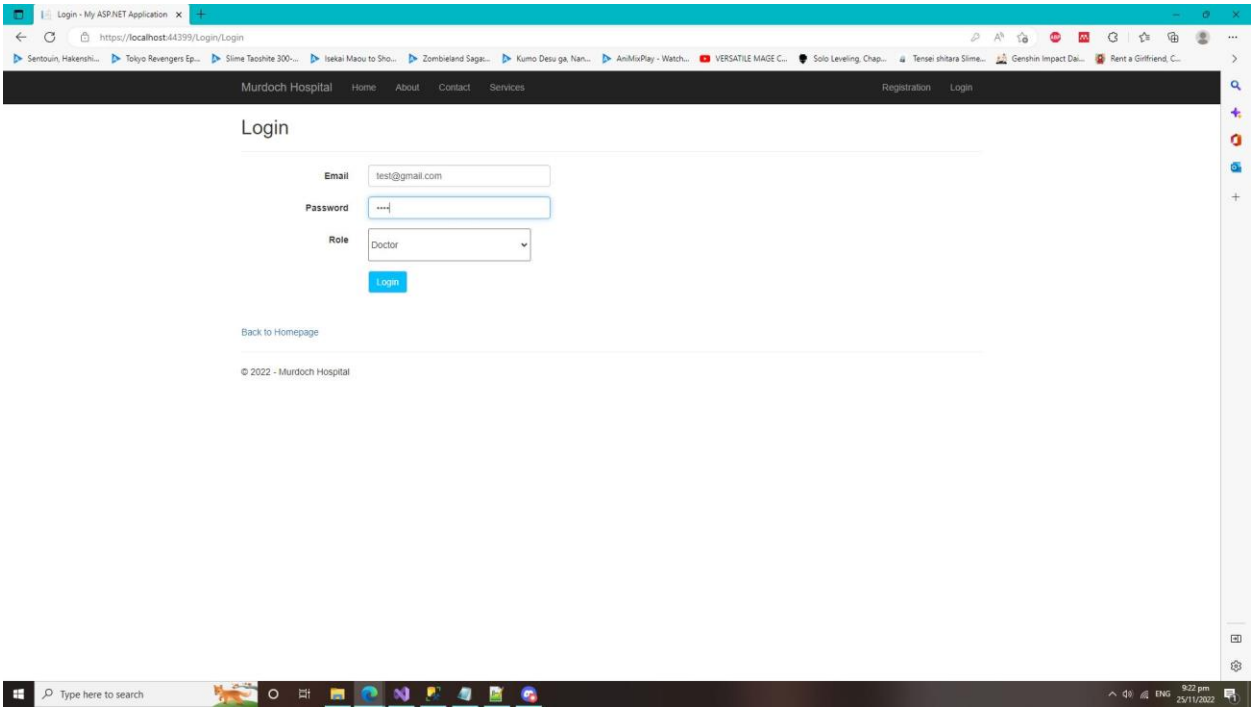
Login Test (Screenshot) and Text

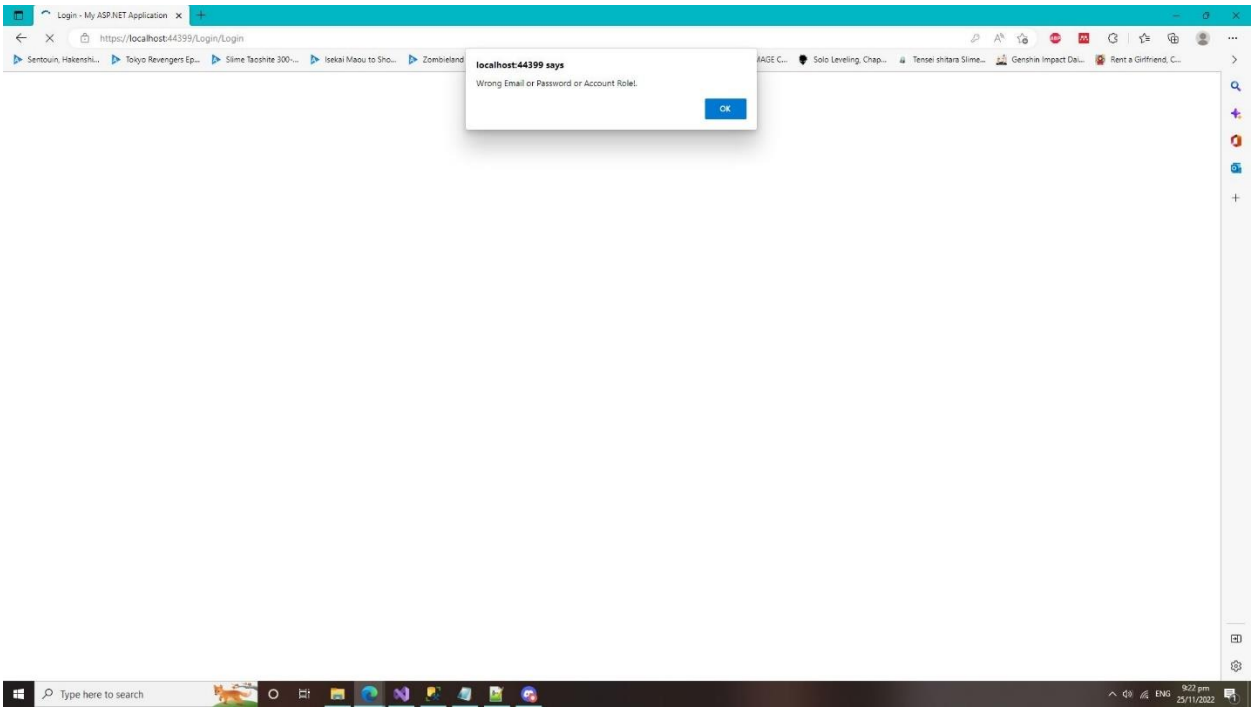
- Login Successfully Account Test
  It is to check if the login “Username” and “Password” is empty the user won’t be able to login, and it will show an error message to the user.





- If the database does not have your existing account or incorrect Email, Password or Role it will show this error message to the user if they try to login.

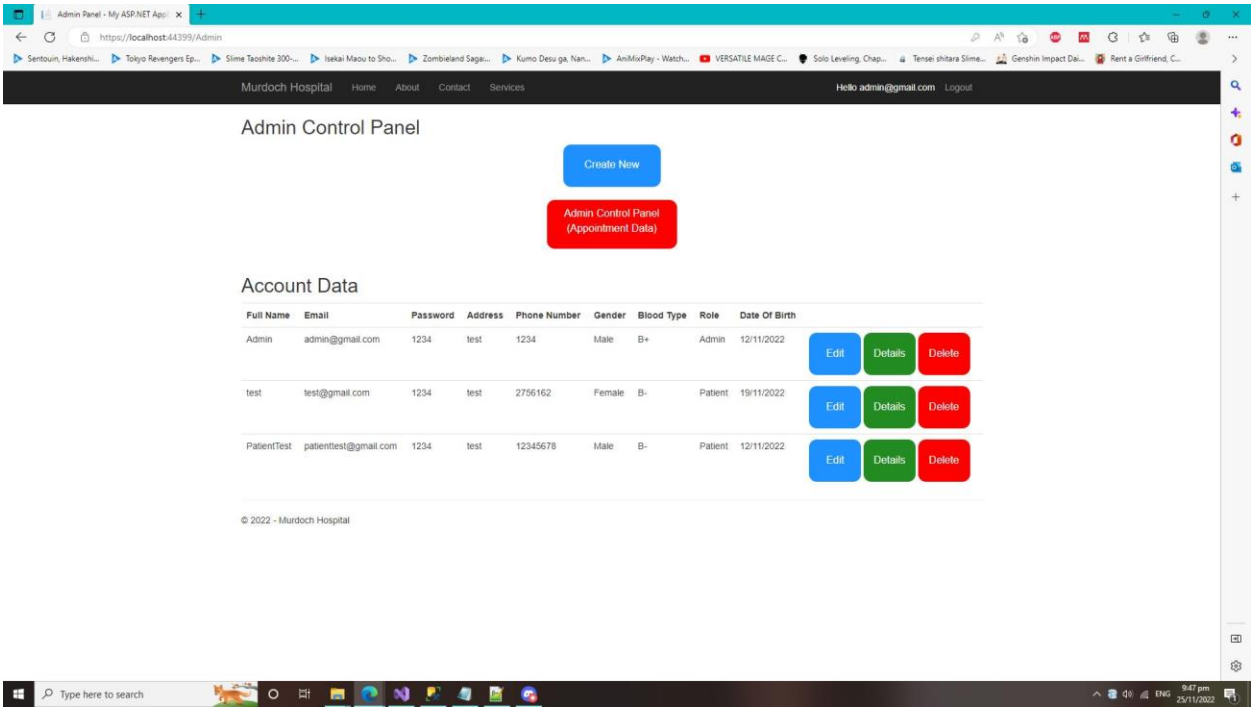




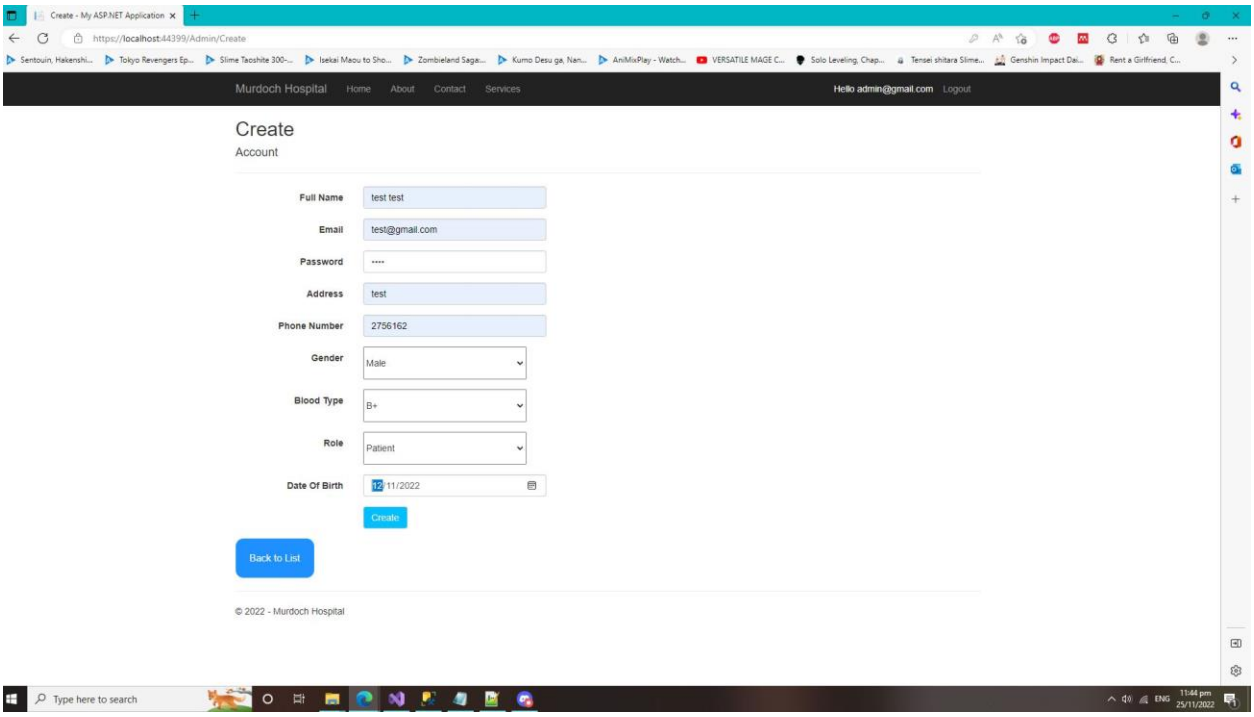
Admin Control Panel Test (Screenshot) and Text (Account Only)

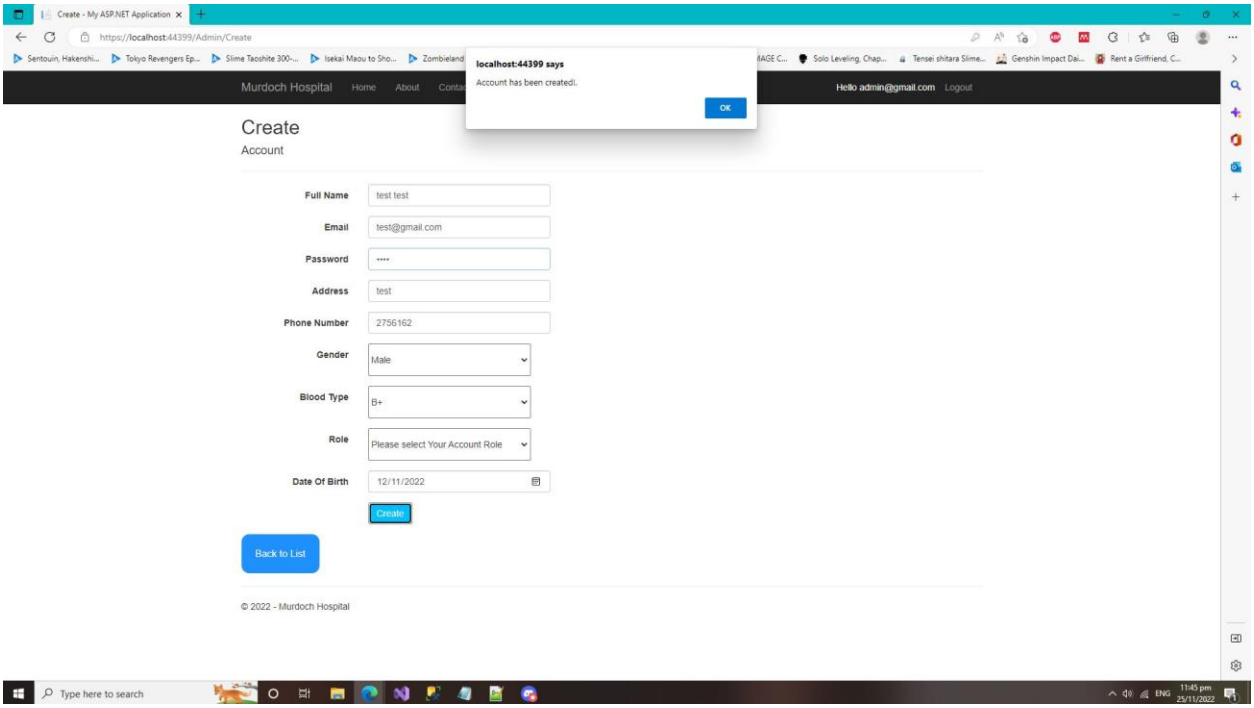
- This test is for the Admin Control Panel where the admin can Create, Edit, Delete, and view details of the user accounts.

Admin Control Panel

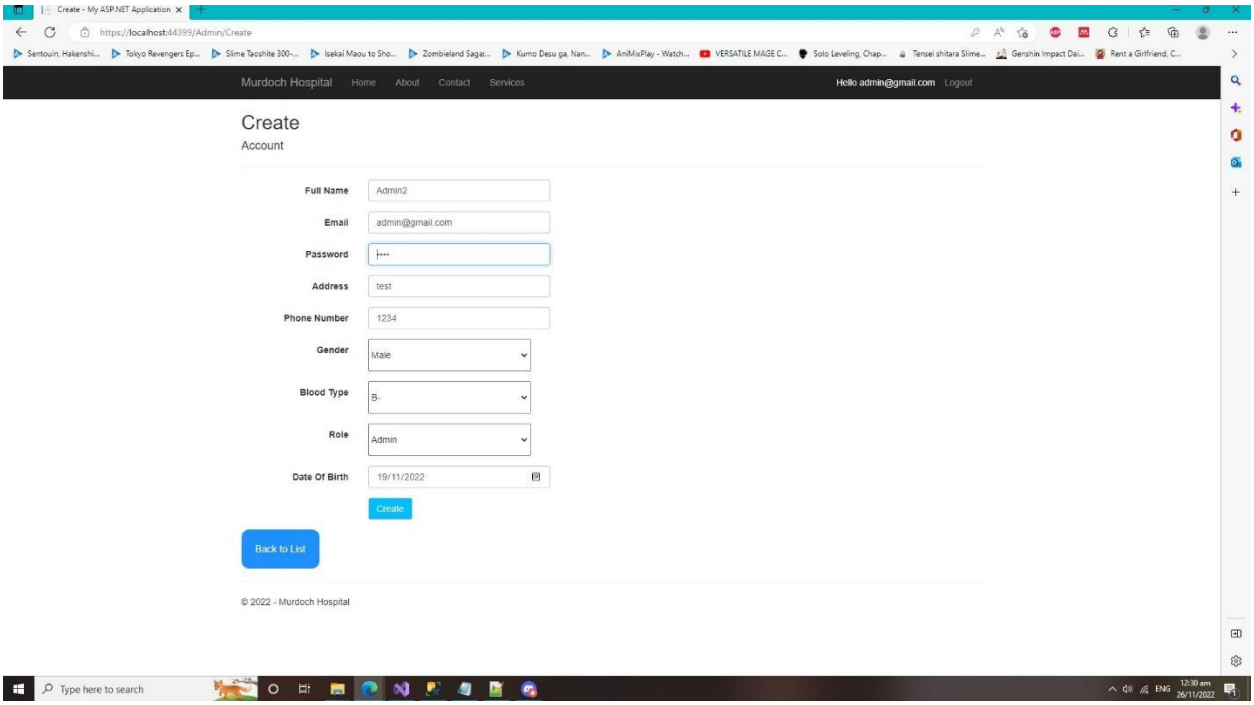


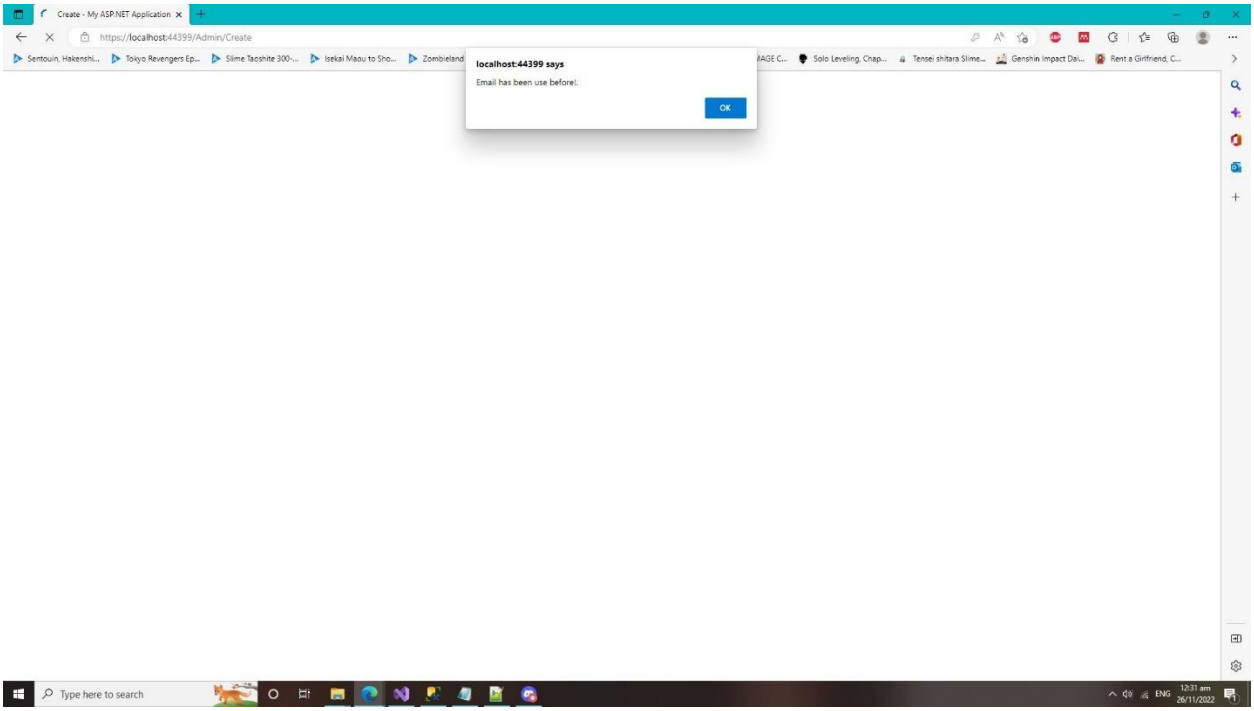
- Admin Control Panel (Create Page). Where the admin can create a new account for the user and store it inside into the Account Database.



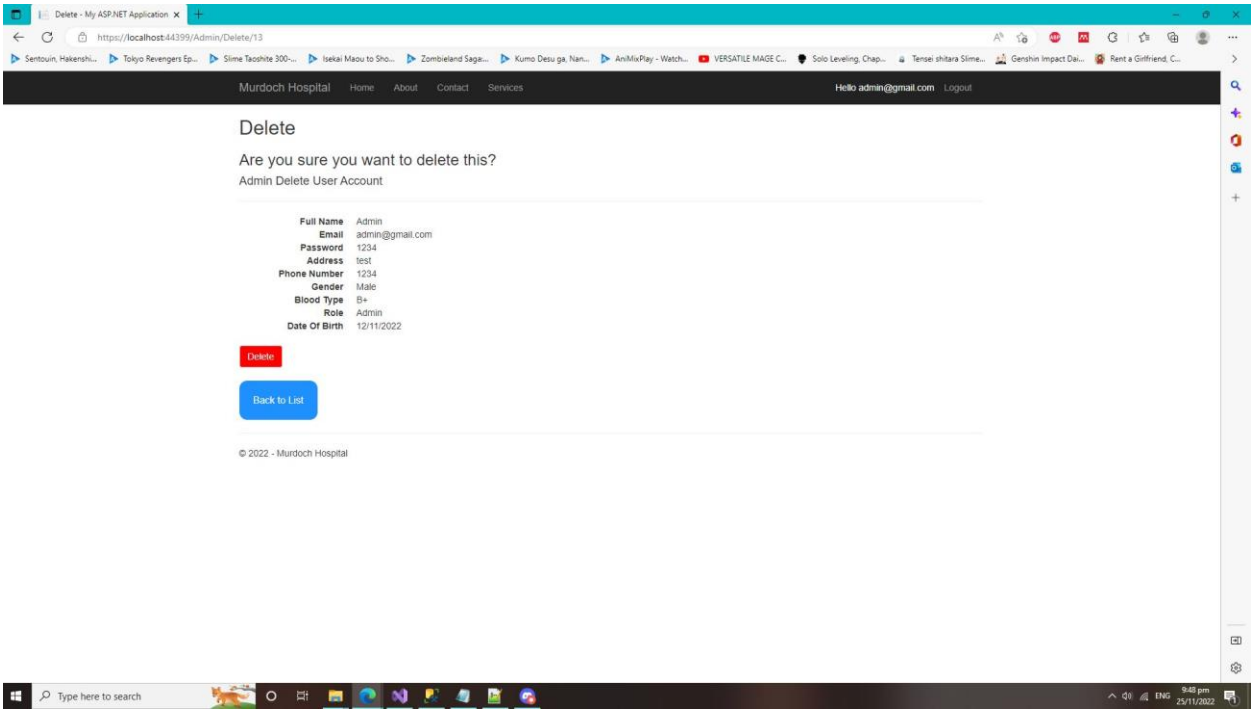


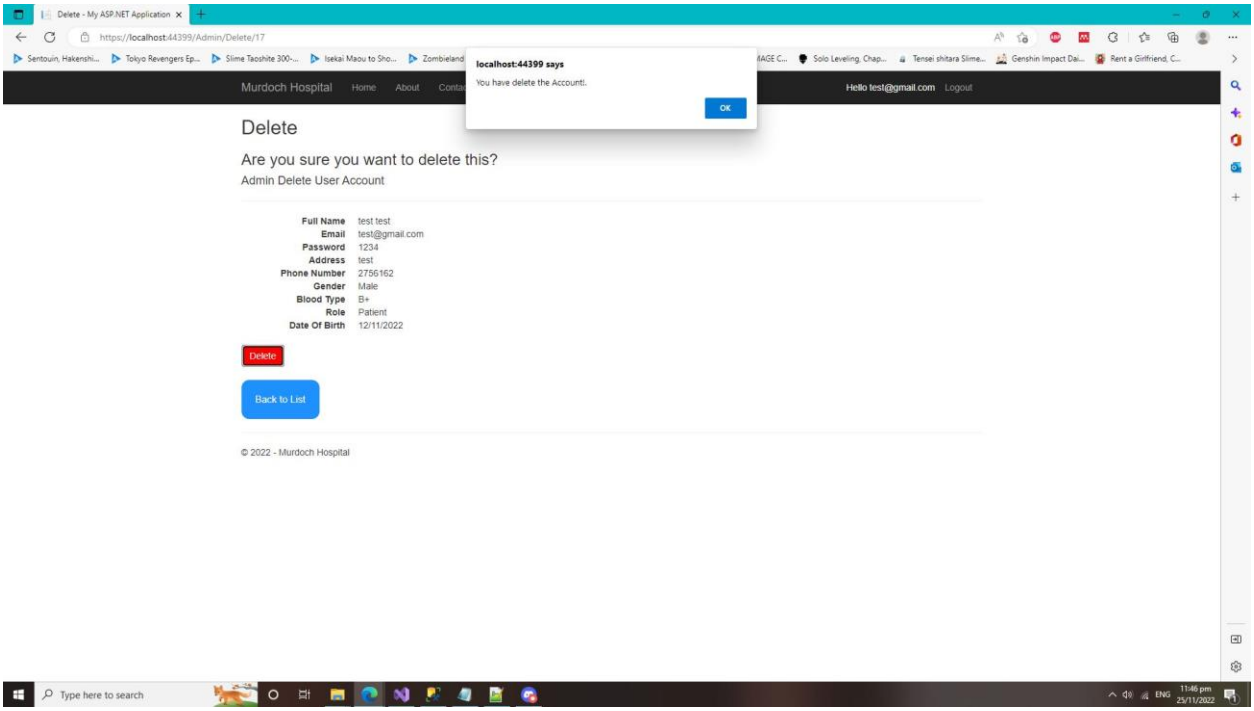
- It also does a validation check if the Email exist on the database. If the Email is existed on the database, the admin can't create the user account with the same email on the database.



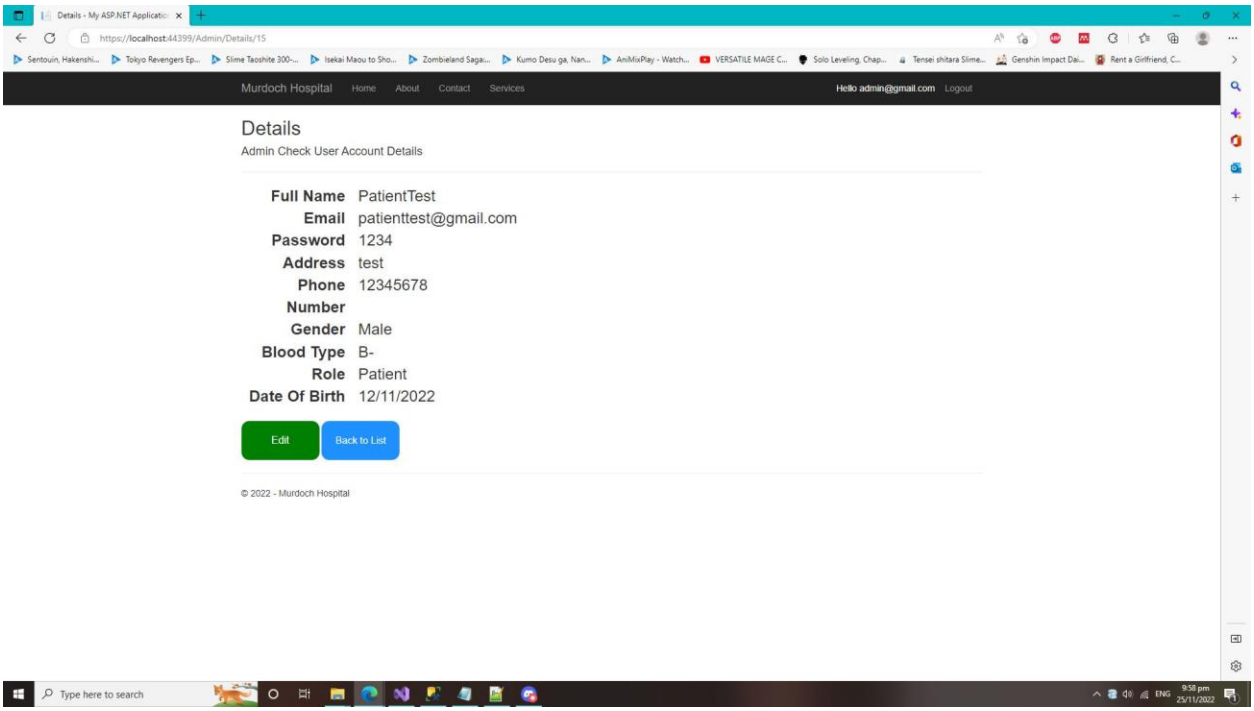


- Admin Control Panel (Delete Page). This is where the admin can delete the user account if it is needed.

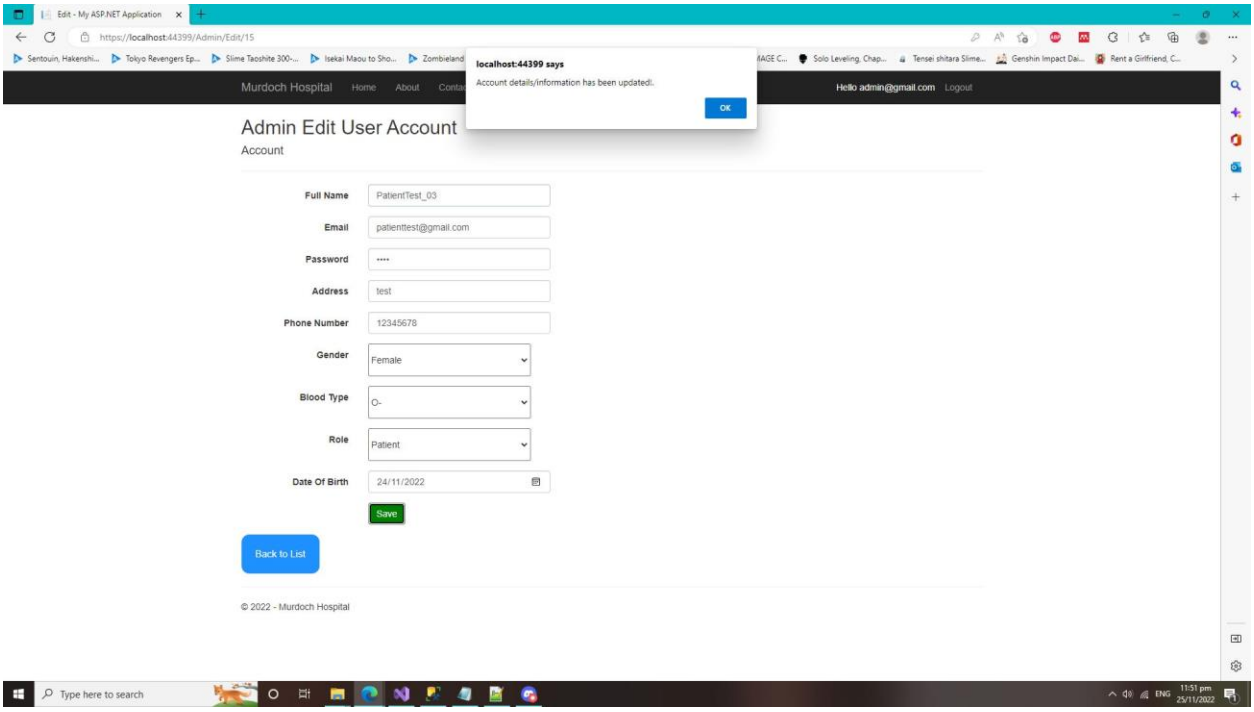




- Admin Control Panel (Details Page). This is where the admin can view the specific account details on this webpage



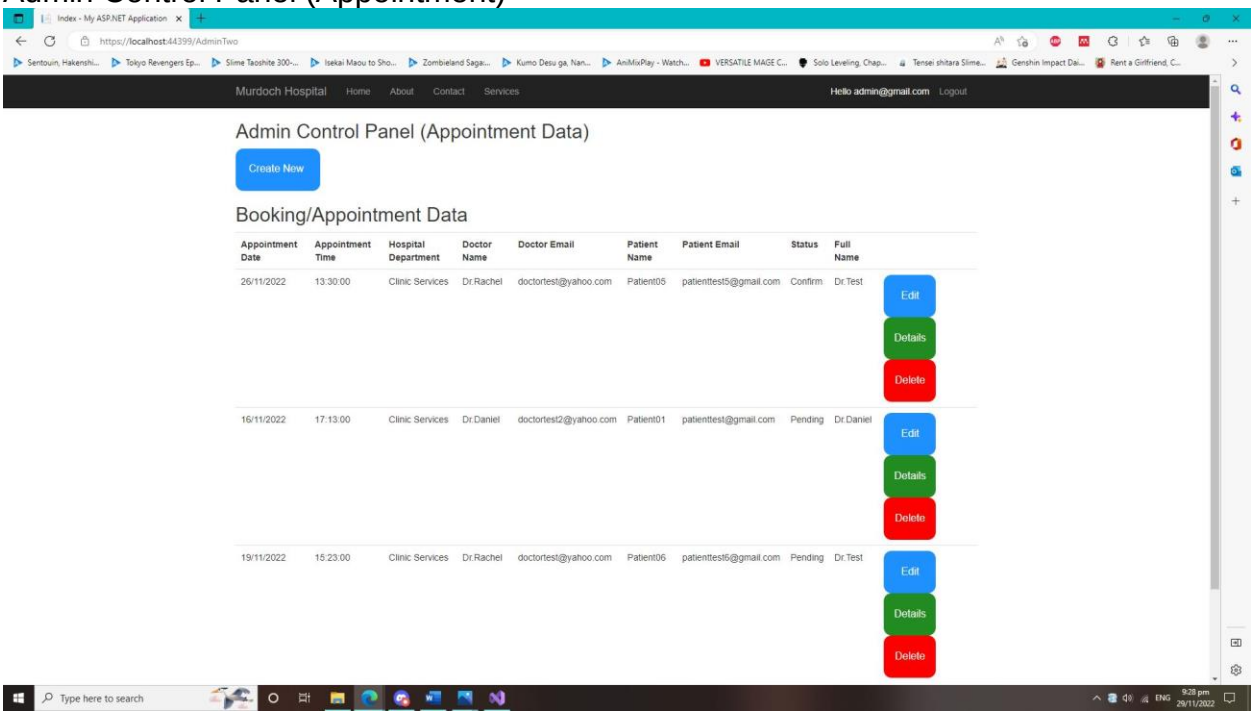
- Admin Control Panel (Edit Page). This is where the admin can edit the user account details or information if needed.



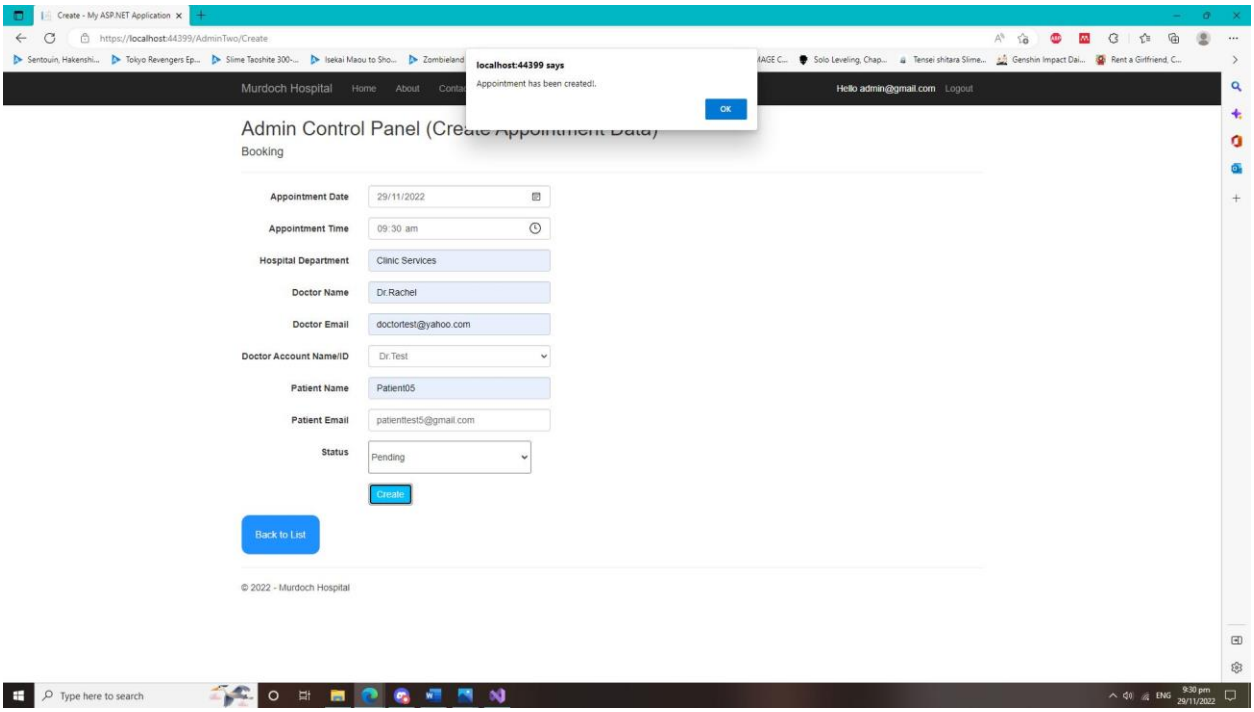
Admin Control Panel Test (Screenshot) and Text (Appointments/Booking Only)

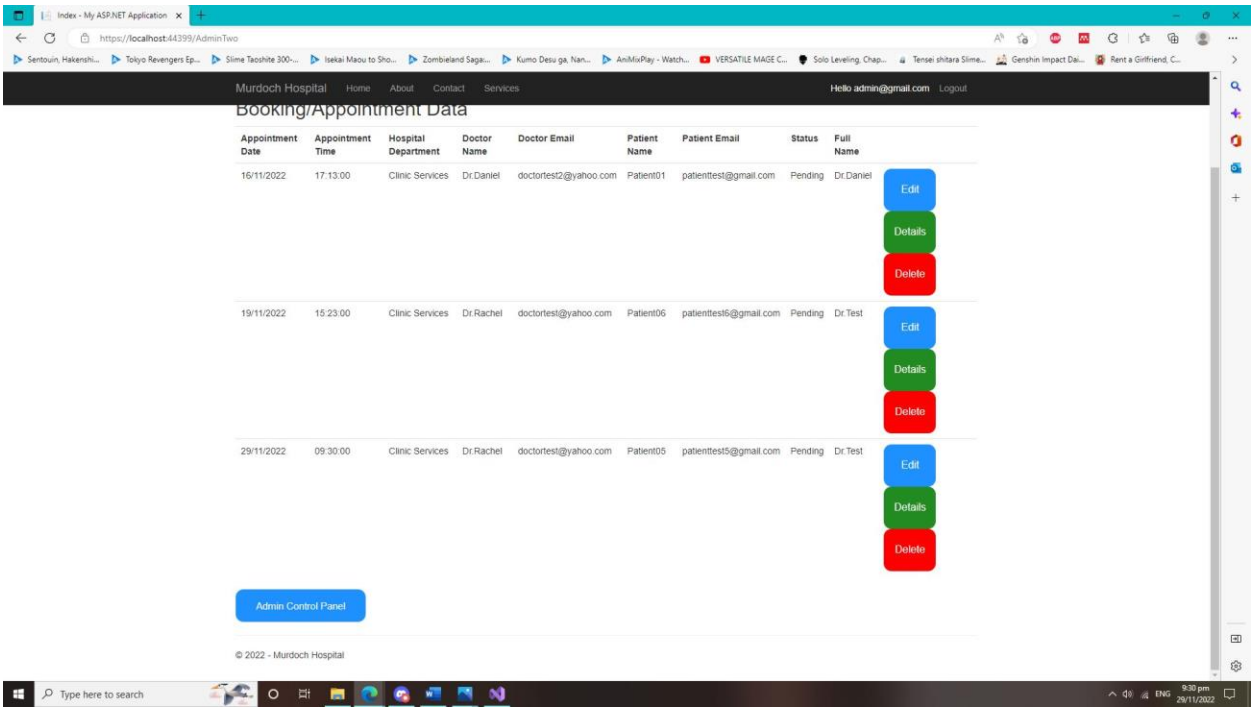
- This test is for the Admin Control Panel where the admin can Create, Edit, Delete, and view details of the user Appointments/Booking.

  Admin Control Panel (Appointment)

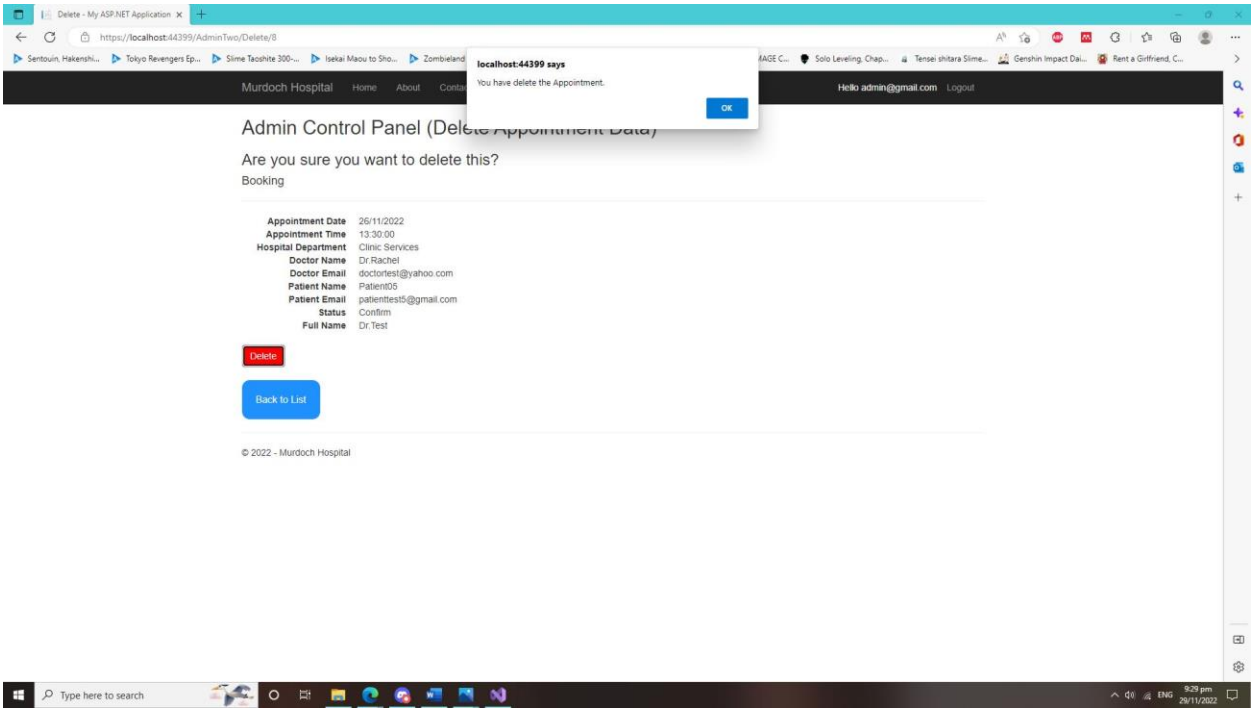


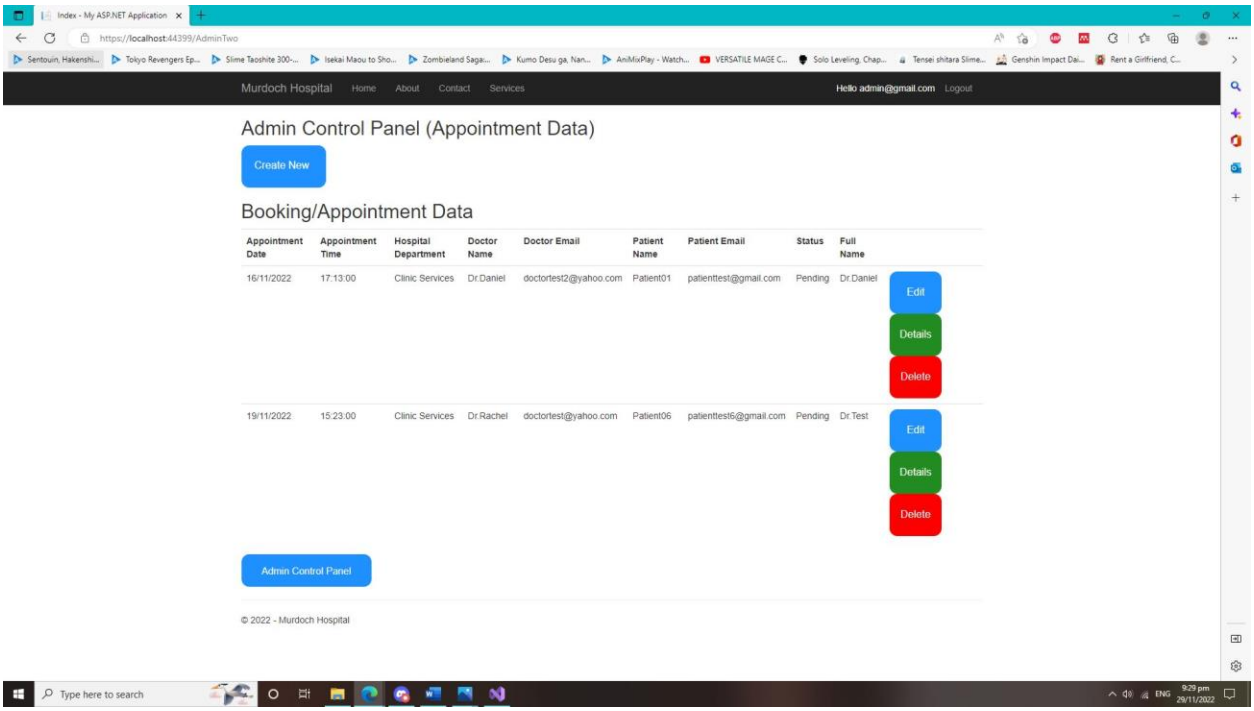
- Admin Control Panel (Create Page). Where the admin can create a new Appointment/Booking for the user and store it inside into the Booking Database.



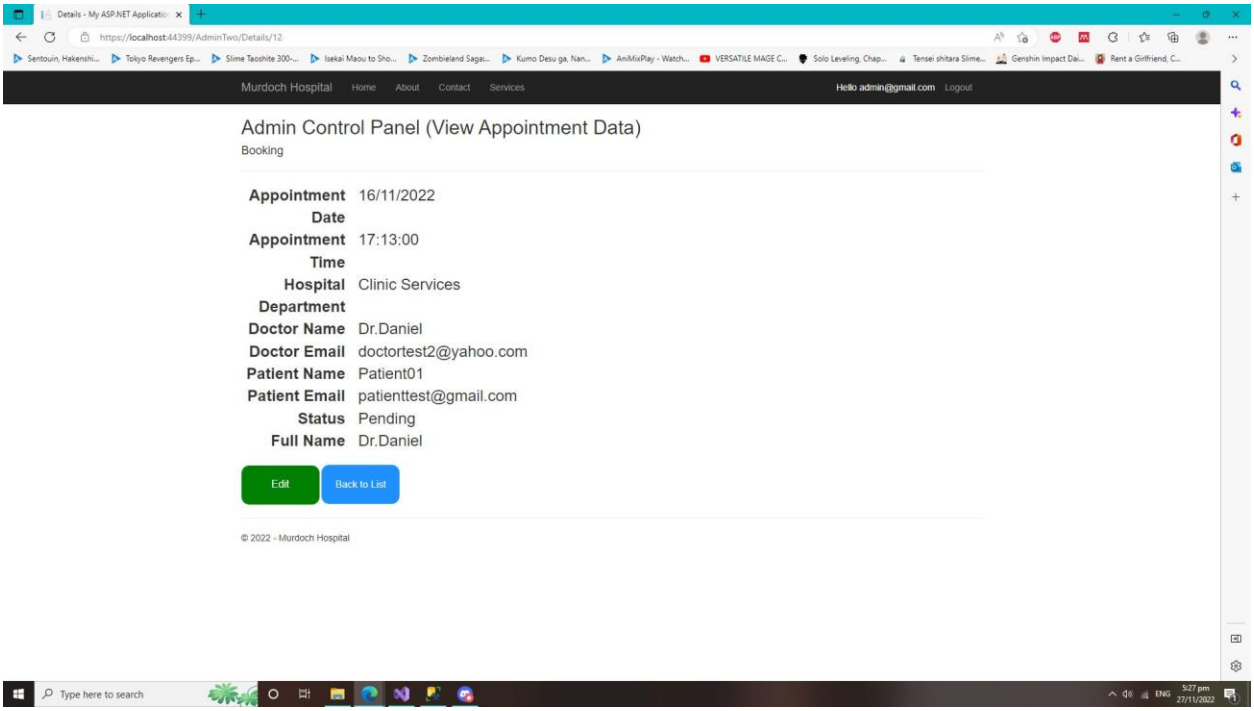


- Admin Control Panel (Delete Page). This is where the admin can delete the user appointment or booking if it is needed.

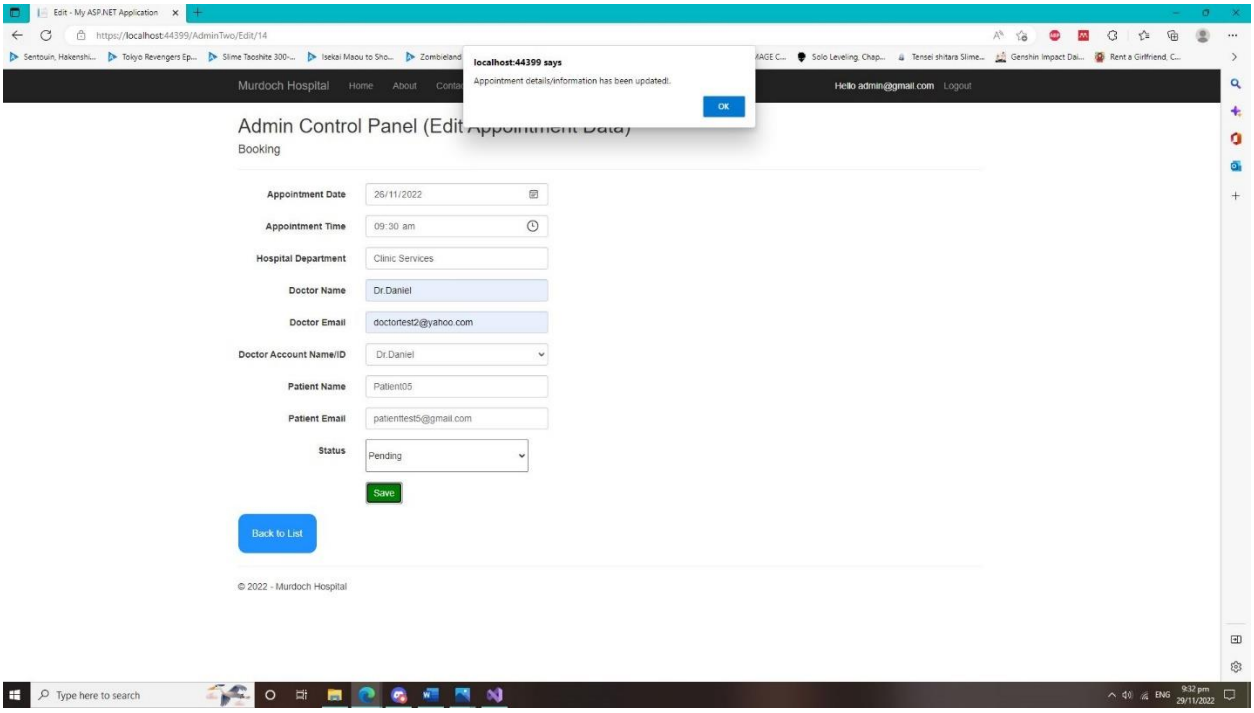


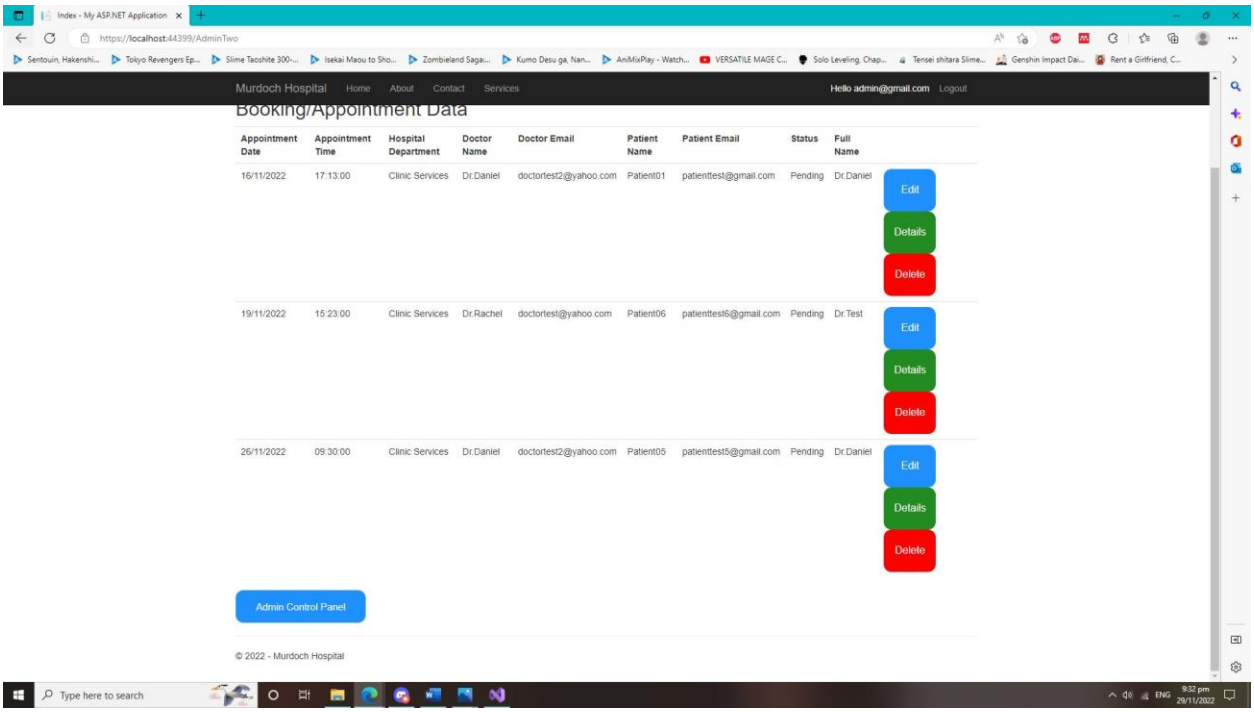


- Admin Control Panel (Details Page). This is where the admin can view the specific user appointment or details on this webpage



- Admin Control Panel (Edit Page). This is where the admin can edit the user appointment or booking details or information if needed.

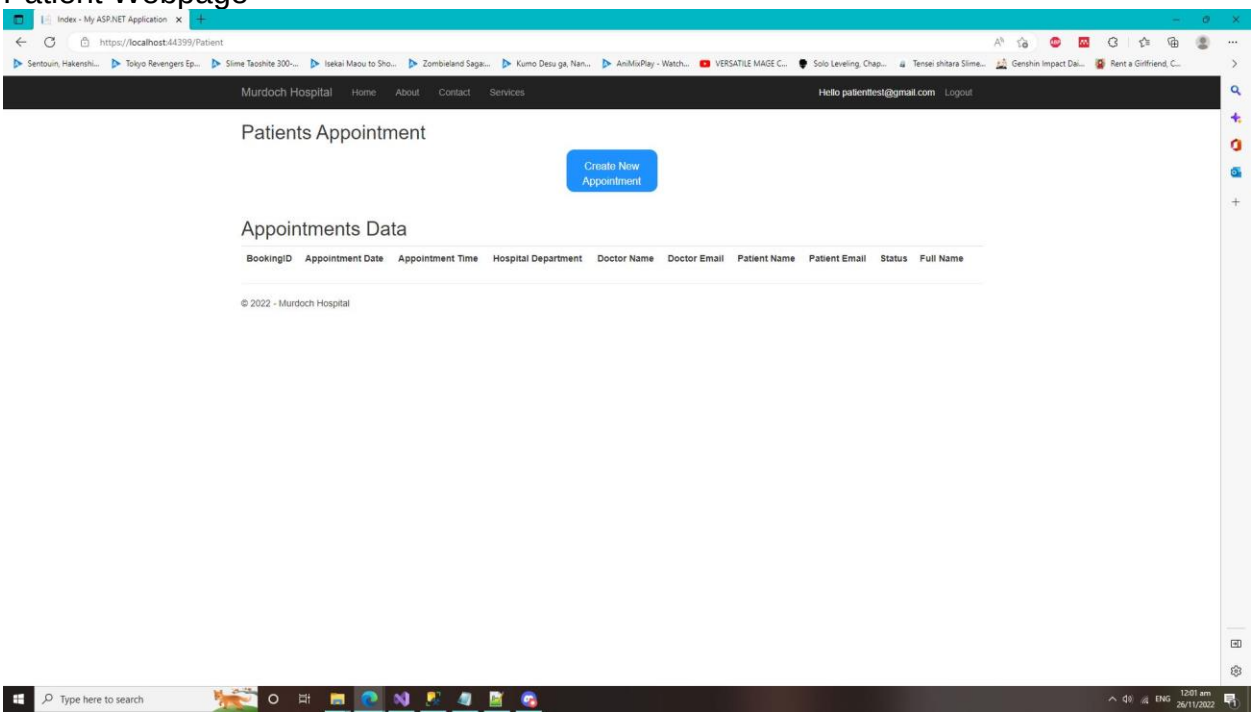




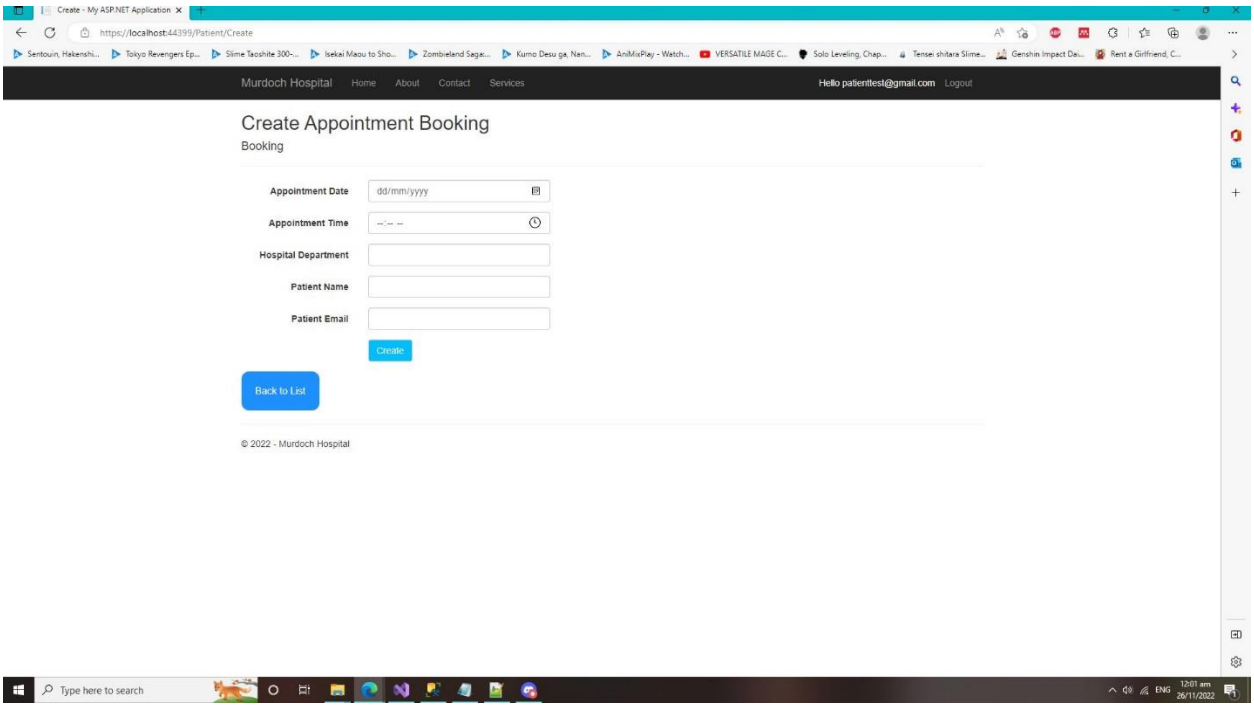
Patient Webpage Test (Screenshot) and Text (Account Only)

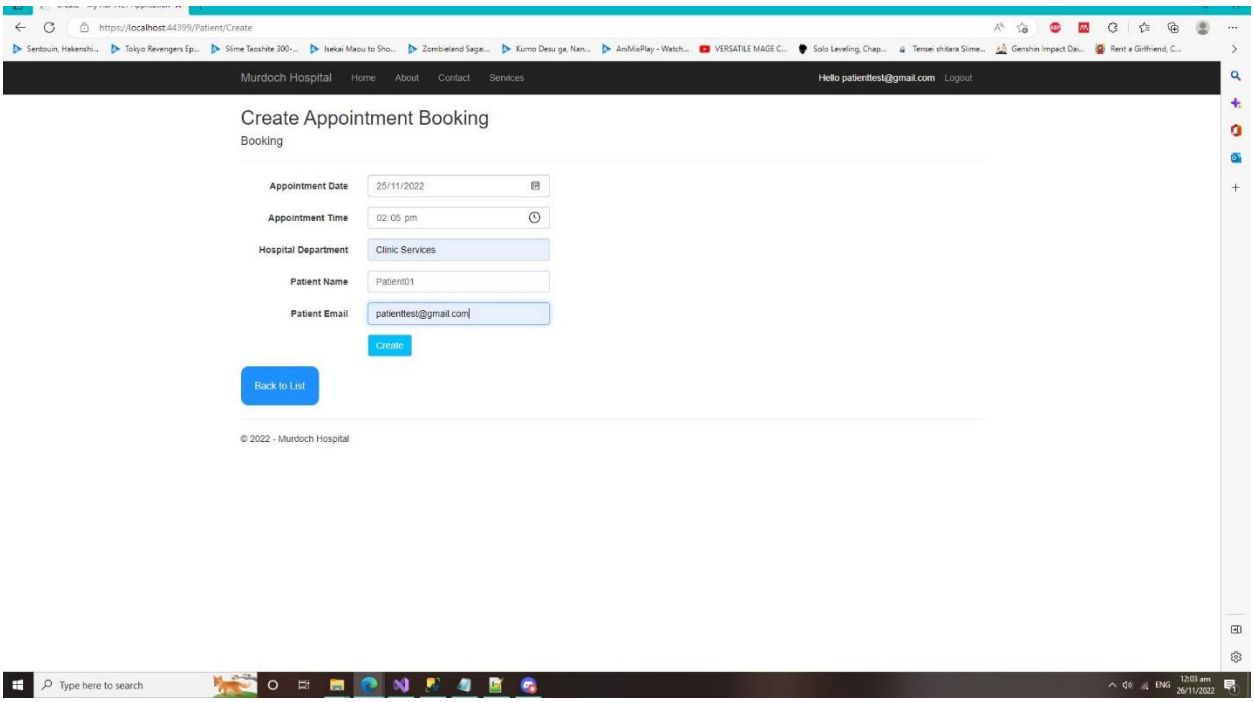
- Patient Control Panel Test. This test is for the patient where the patients can Create, Edit, Delete and View details of their booking or appointments from this control panel webpage.

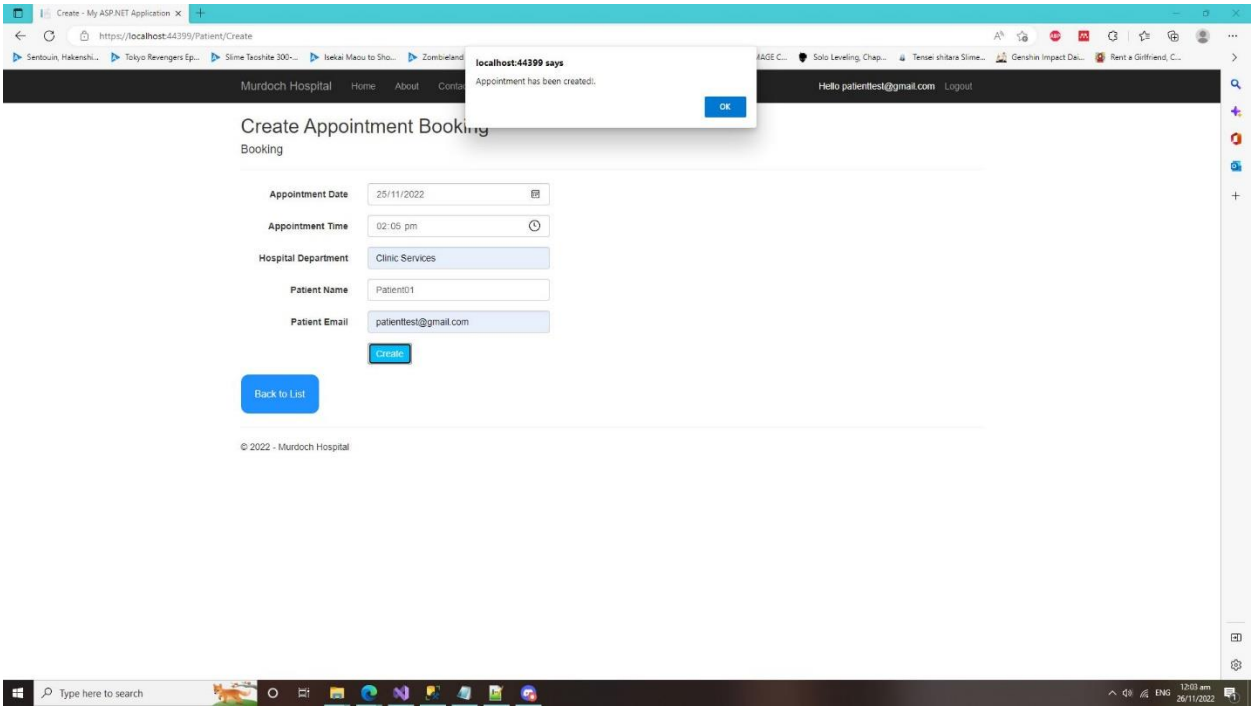
Patient Webpage

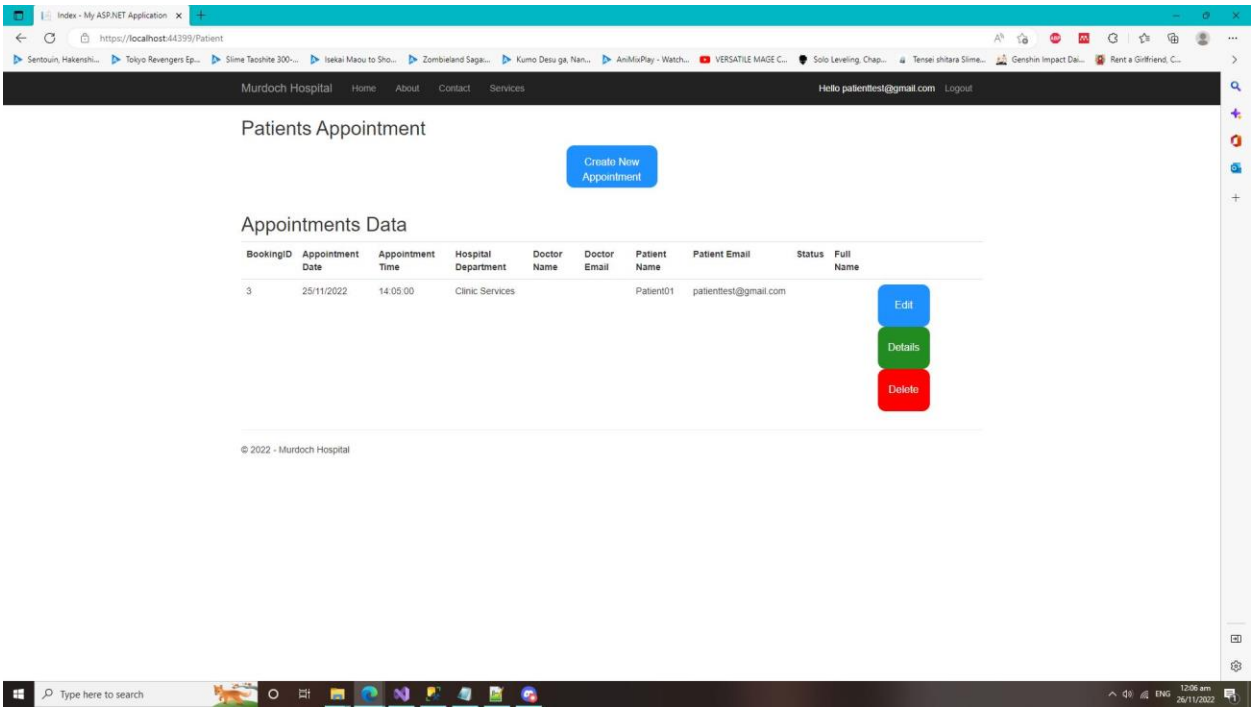


- Patient Webpage (Create Page). Where the patients can create a new Appointment/Booking for the themselves and it automatically store it inside into the Booking Database.

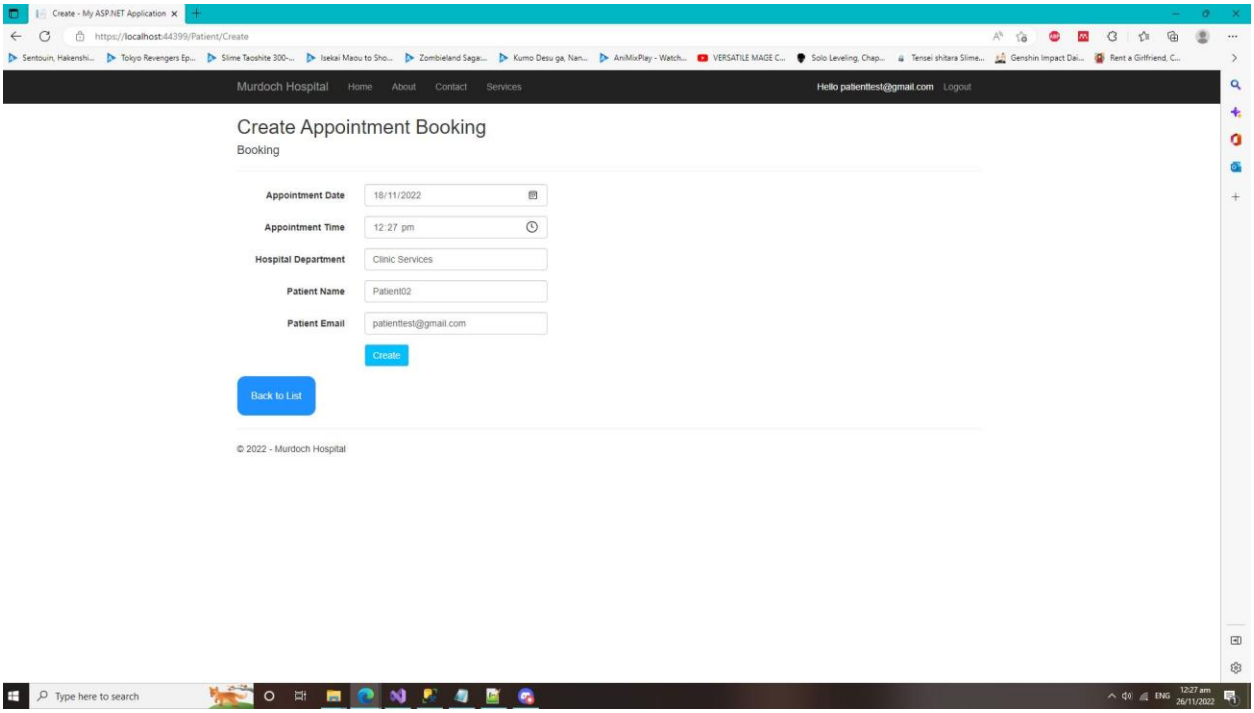


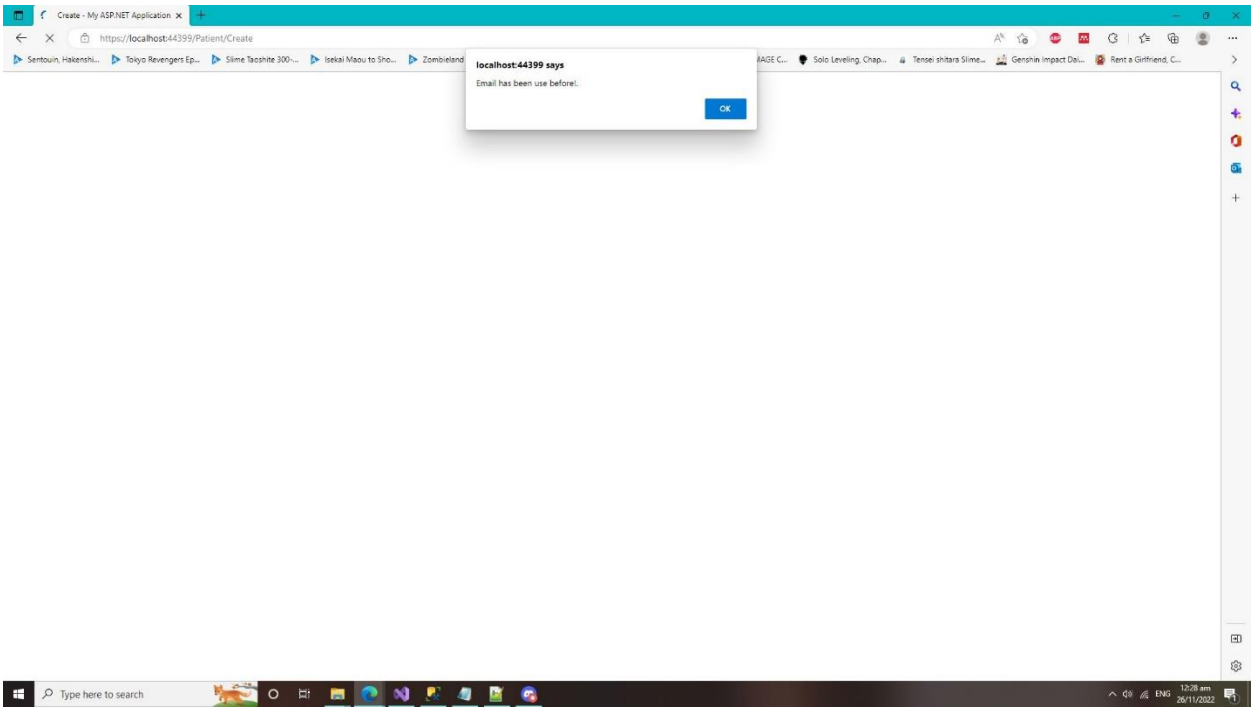

https://localhost:44399/Patient/Create
Murdoch Hospital
Home
About
Contact
Services
Hello patienttest@gmail.com
Logout
Create Appointment Booking
Booking
Appointment Date
25/11/2022
Appointment Time
02:05 pm
Hospital Department
Clinic Services
Patient Name
Patient01
Patient Email
patienttest@gmail.com
Create
Back to List
© 2022 - Murdoch Hospital
Type here to search
ENG
12:03 am
26/11/2022


Create - My ASP.NET Application
https://localhost:44399/Patient/Create
localhost:44399 says
Appointment has been created!
OK
Murdoch Hospital
Hello patienttest@gmail.com
Logout
Create Appointment Booking
Booking
Appointment Date
25/11/2022
Appointment Time
02:05 pm
Hospital Department
Clinic Services
Patient Name
Patient01
Patient Email
patienttest@gmail.com
Create
Back to List
© 2022 - Murdoch Hospital
Type here to search
ENG
12:03 am
26/11/2022

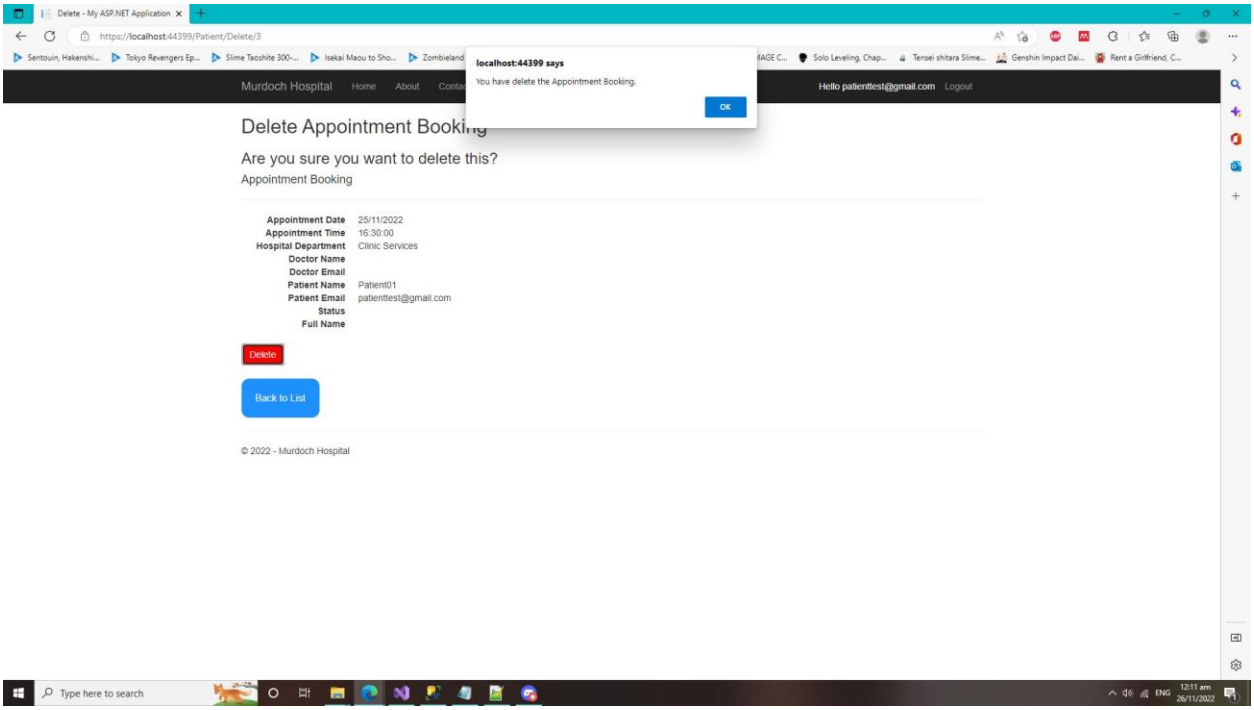


- It also does a validation check if the other patient email exists on the database. If the email does exist on the database, the patient can't create the appointment or booking with the same email that store on the database.

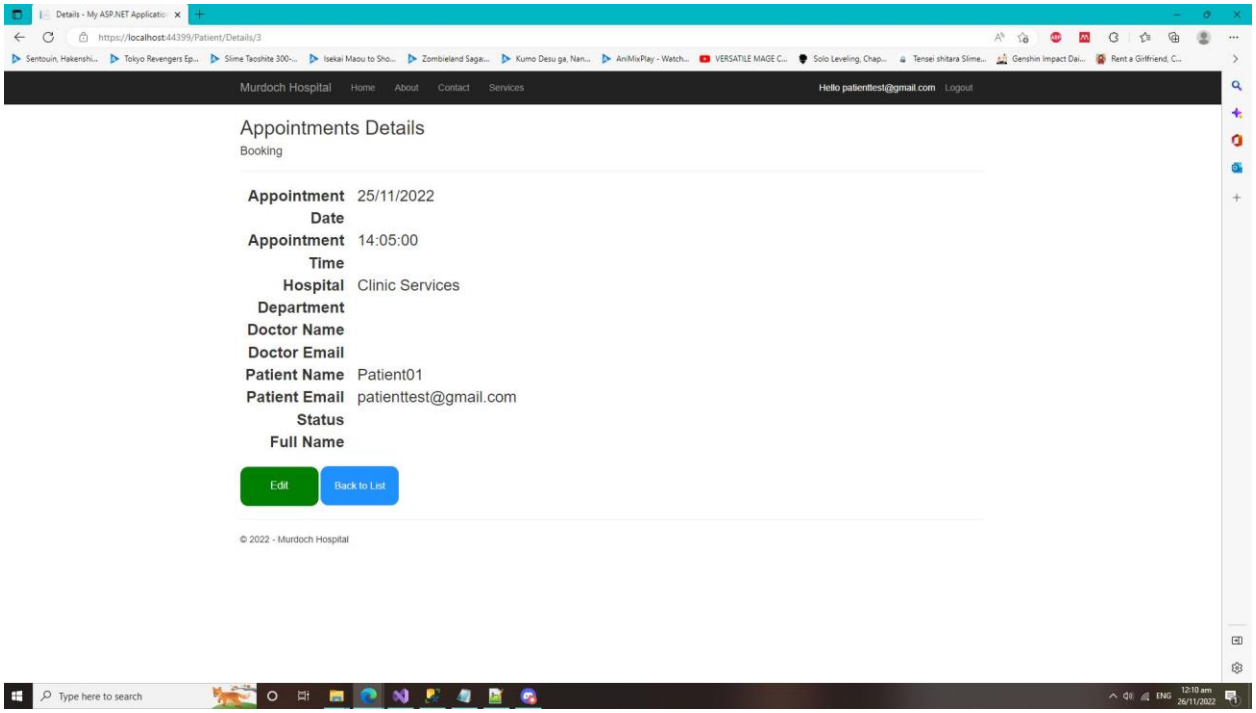




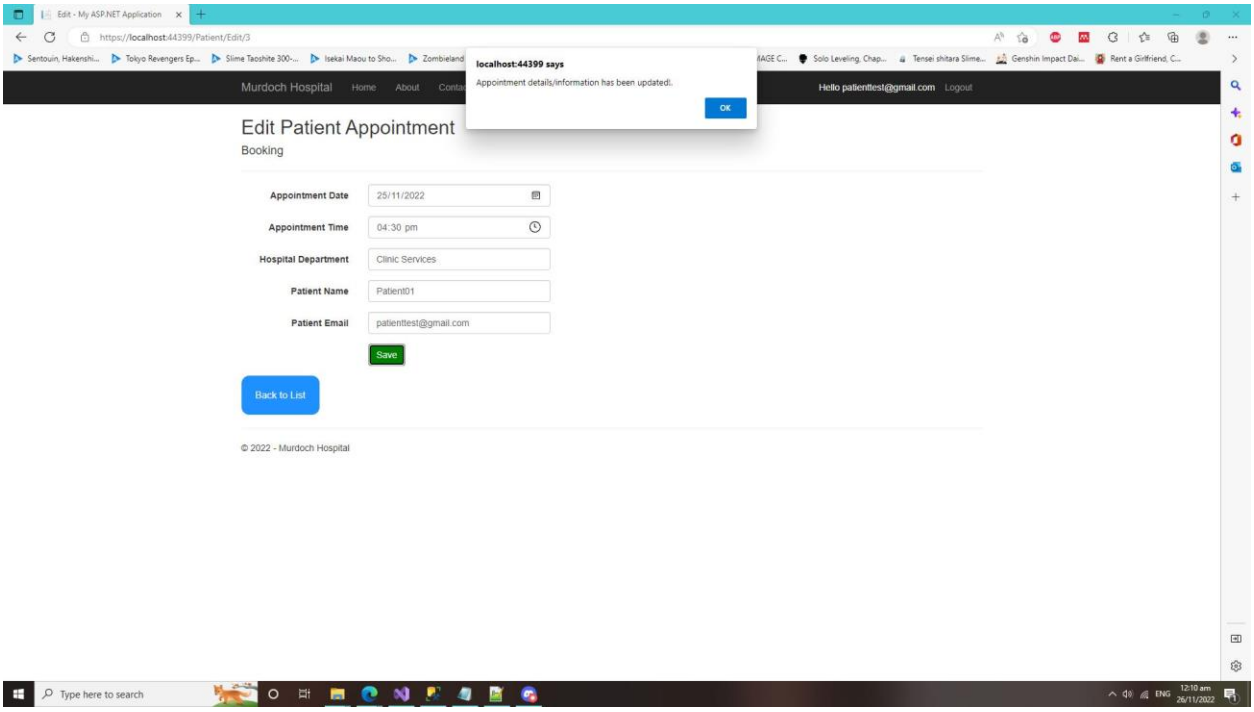
- Patient Web page (Delete Page). This webpage is where the patients can delete or cancel their appointments of booking. The appointment or booking that deleted will also be automatically deleted from the Booking database itself.

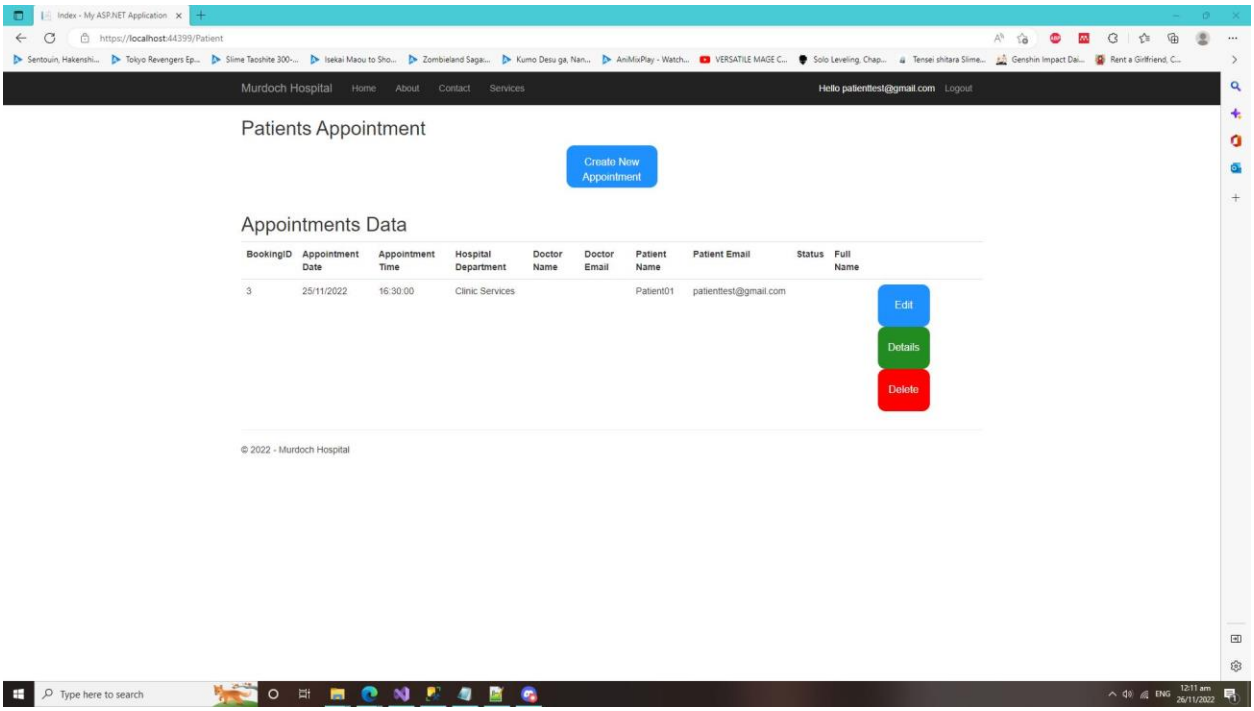


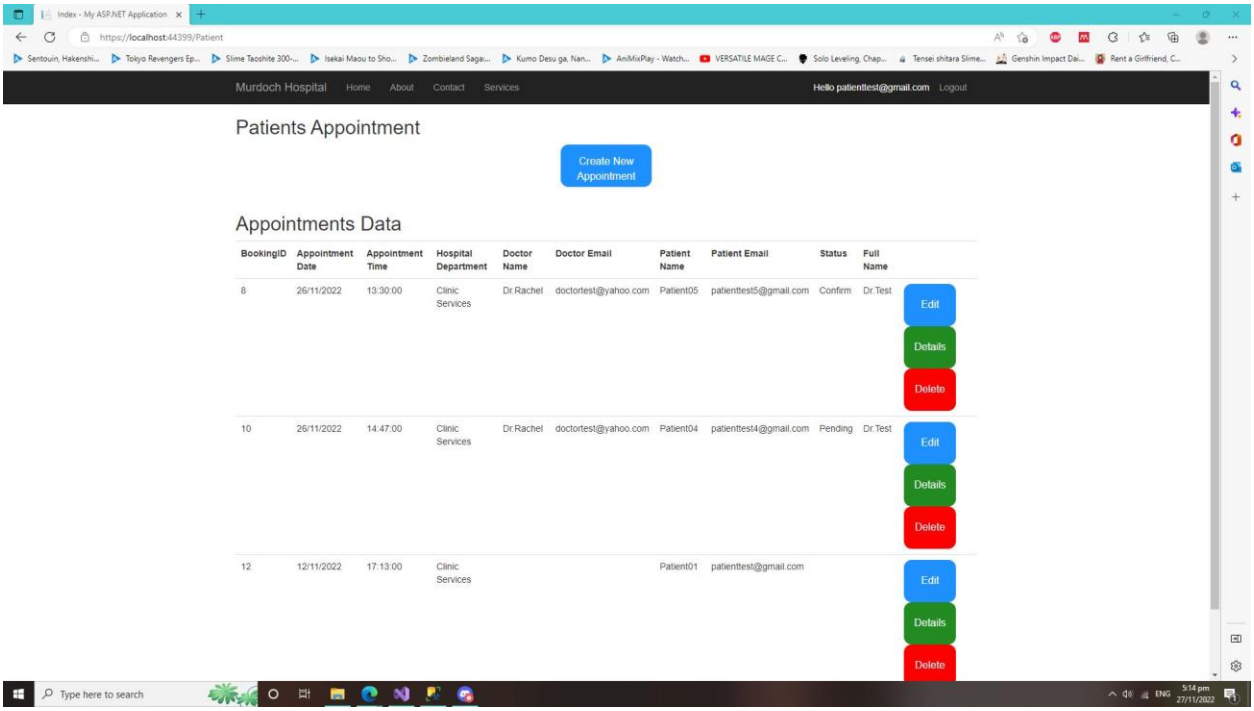
- Patient Webpage (Details Page). This is where the patient can view their own appointments/booking details from this webpage itself.



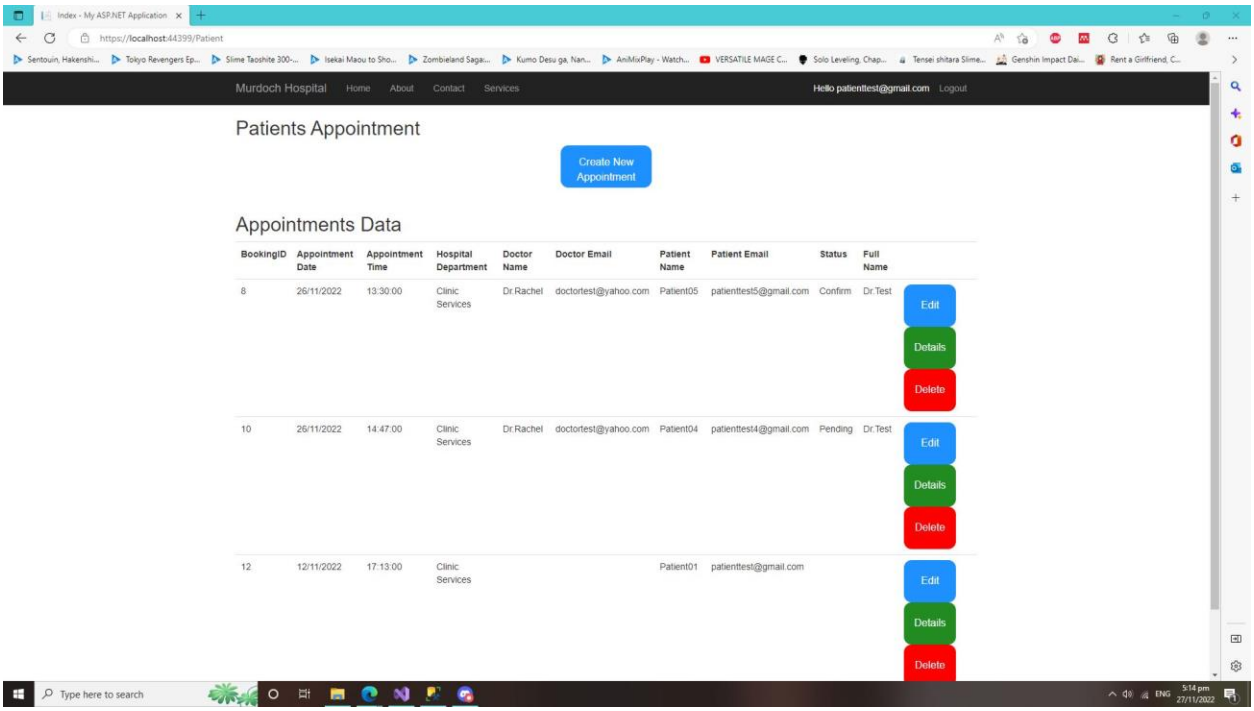
- Patient Webpage (Edit Page). This is where patient can edit their appointments and booking details if they wish to update any of it.

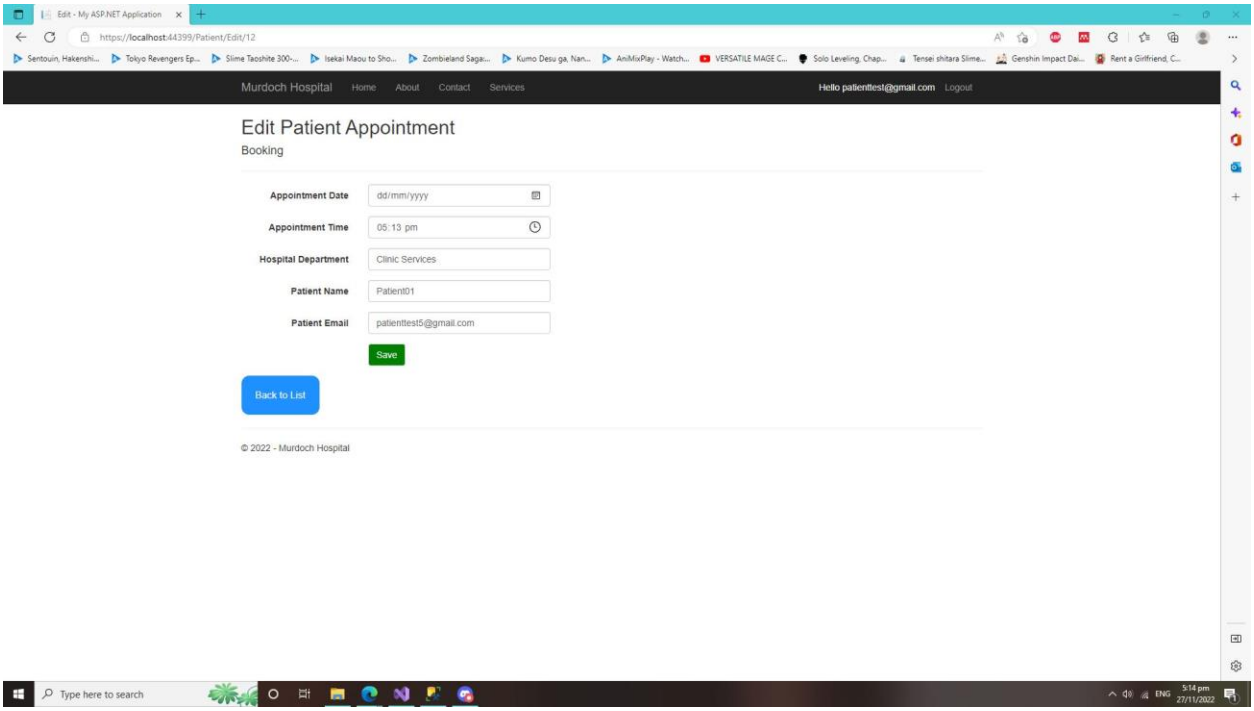


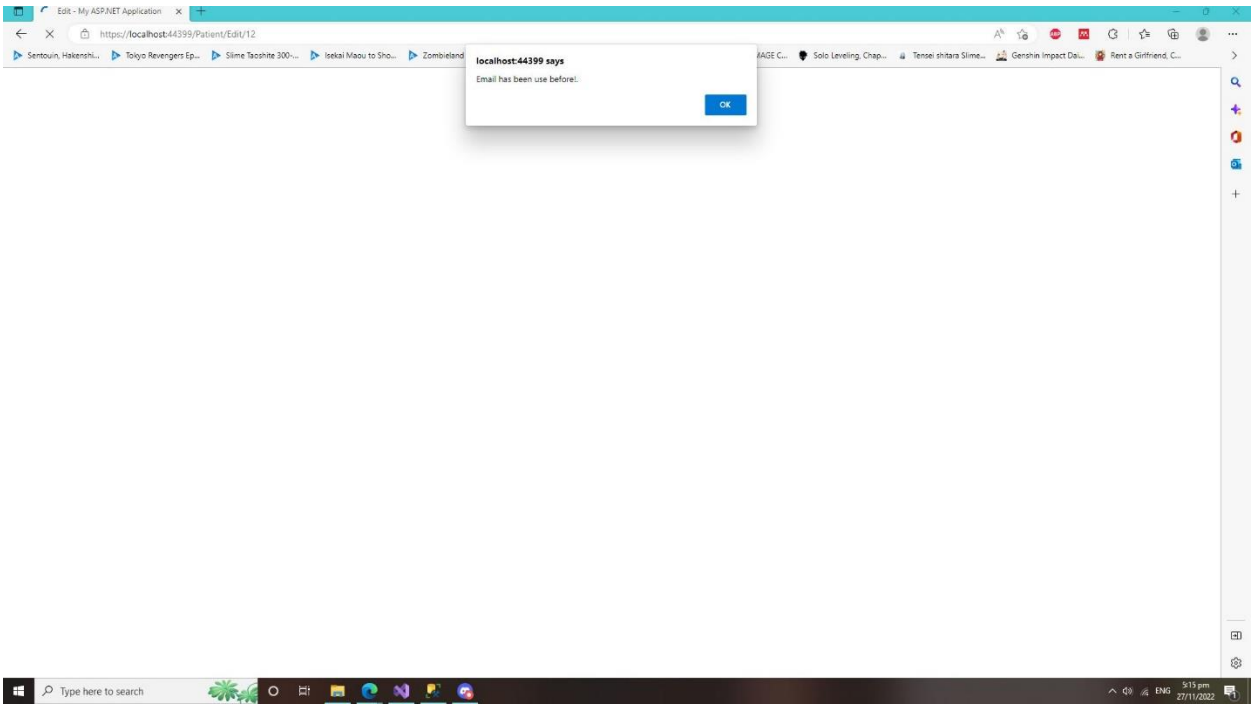




- It also does a validation check if the other patient email exists on the database. If the email does exist on the database, the patient can't edit the appointment or booking with the same email that store on the database.


Index - My ASP.NET Application
https://localhost:44399/Patient
Murdoch Hospital
Home
About
Contact
Services
Hello patienttest@gmail.com
Logout
Patients Appointment
Create New Appointment
Appointments Data
BookingID
Appointment Date
Appointment Time
Hospital Department
Doctor Name
Doctor Email
Patient Name
Patient Email
Status
Full Name
8
26/11/2022
13:30:00
Clinic Services
Dr.Rachel
doctortest@yahoo.com
Patient05
patienttest5@gmail.com
Confirm
Dr.Test
10
26/11/2022
14:47:00
Clinic Services
Dr.Rachel
doctortest@yahoo.com
Patient04
patienttest4@gmail.com
Pending
Dr.Test
12
12/11/2022
17:13:00
Clinic Services
Patient01
patienttest@gmail.com
Edit
Details
Delete
Type here to search

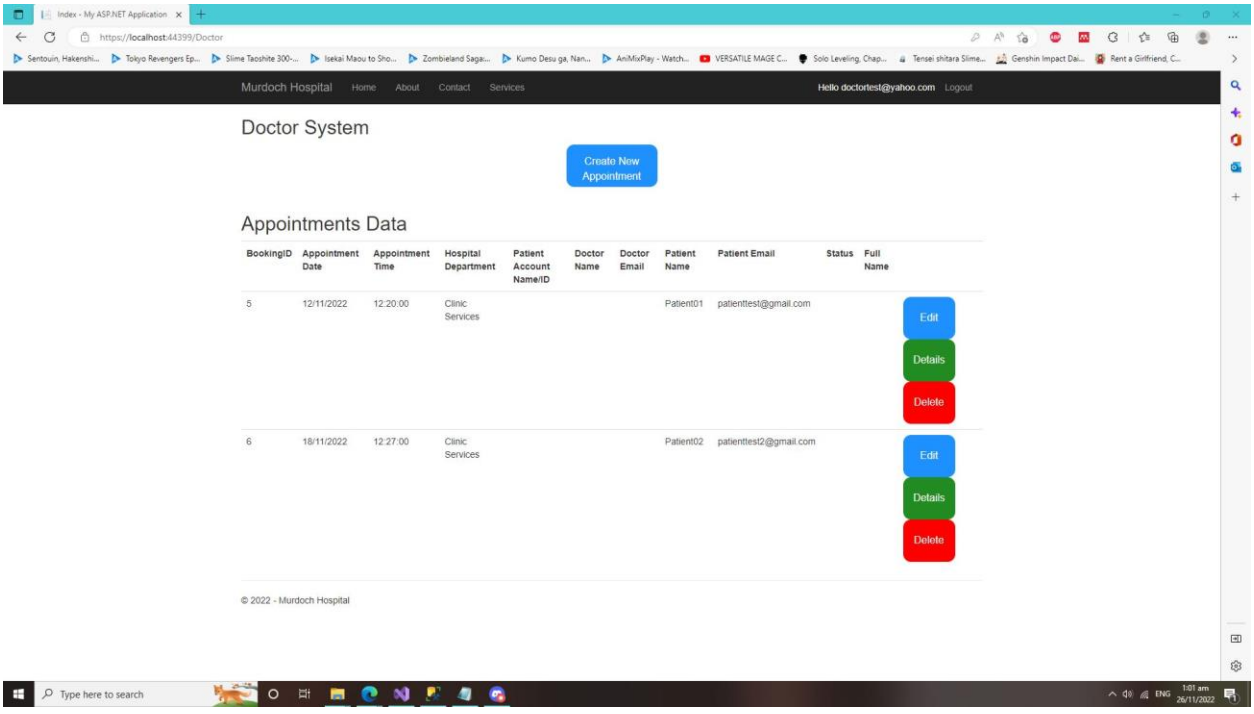

Edit - My ASP.NET Application
https://localhost:44399/Patient/Edit/12
Murdoch Hospital
Home
About
Contact
Services
Hello patienttest@gmail.com
Logout
Edit Patient Appointment
Booking
Appointment Date
dd/mm/yyyy
Appointment Time
05:13 pm
Hospital Department
Clinic Services
Patient Name
Patient01
Patient Email
patienttest5@gmail.com
Save
Back to List
© 2022 - Murdoch Hospital
Type here to search



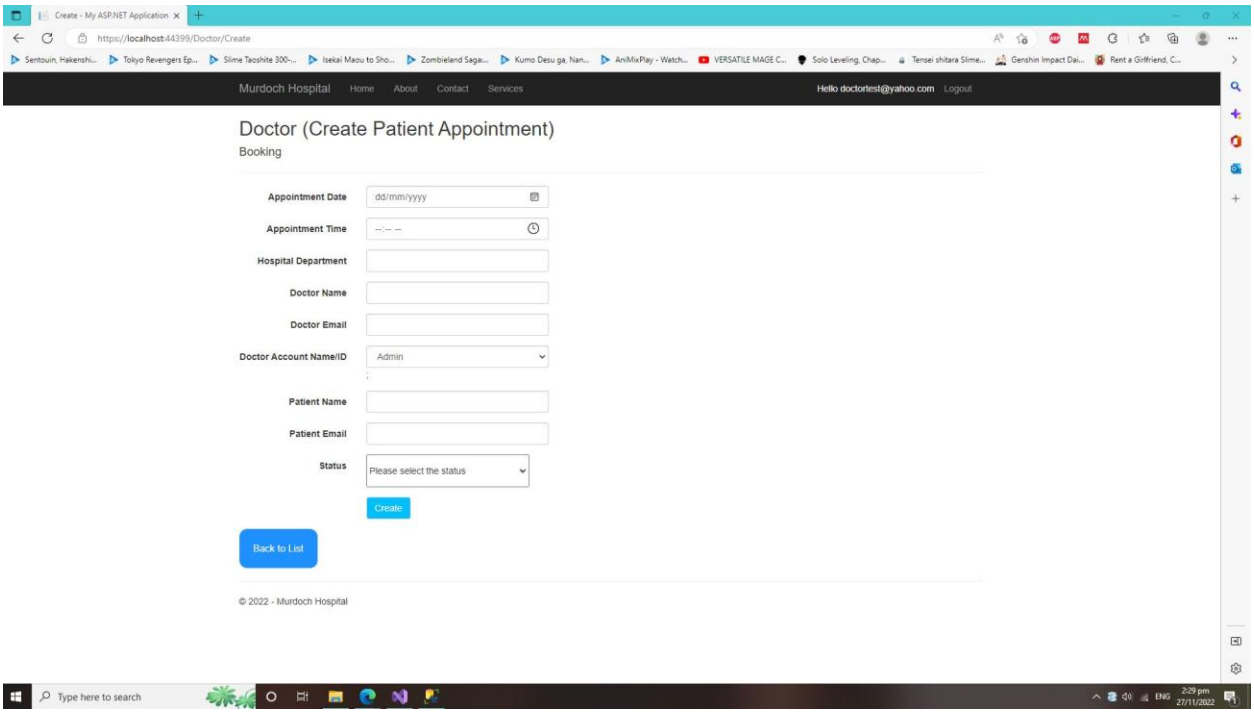
Doctor Control Panel Test (Screenshot) and Text (Account Only)

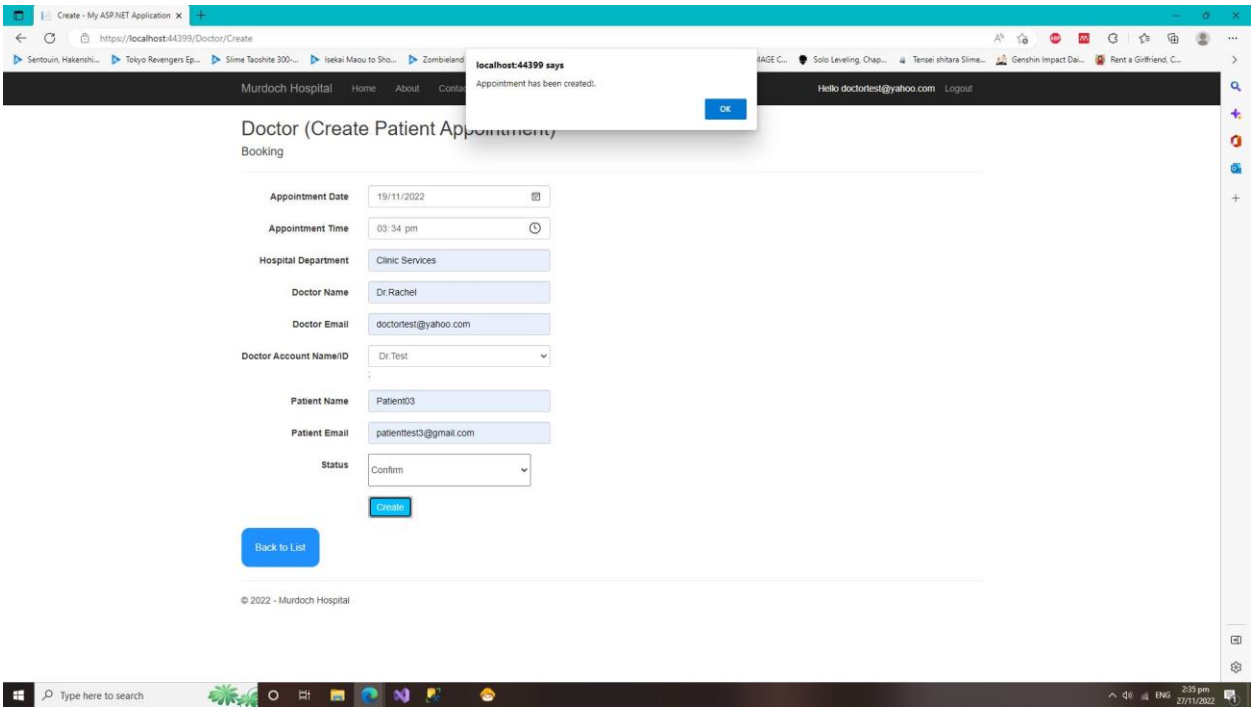
- Doctor Control Panel Test. This test is for the doctors where the doctors can Create, Edit, Delete and View details of their patients booking or appointments from this control panel webpage.

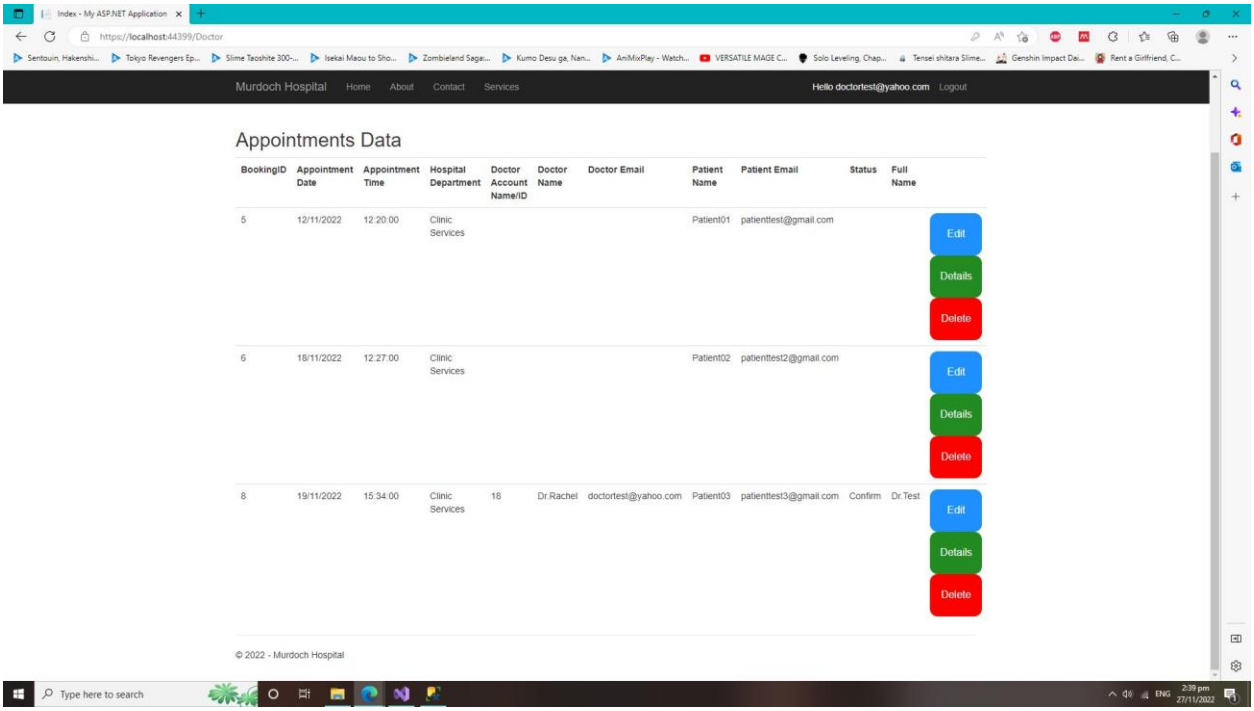
  Doctor Control Panel



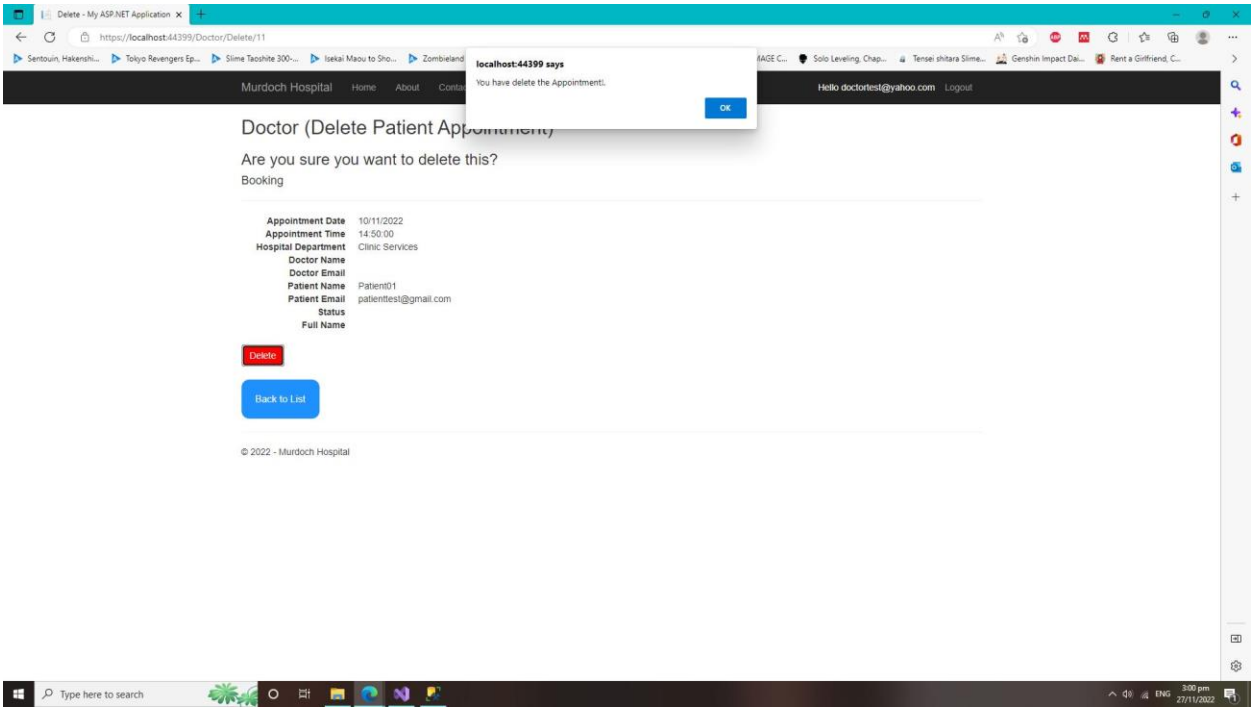
- Doctor Control Panel (Create Page). Where the doctors can create/schedule a new Appointment/Booking for their patients and it automatically store it inside into the Booking Database. (For this Doctor Control Panel. I never did any validation check on same email because one patient can have more then one appointments/bookings in any other days)

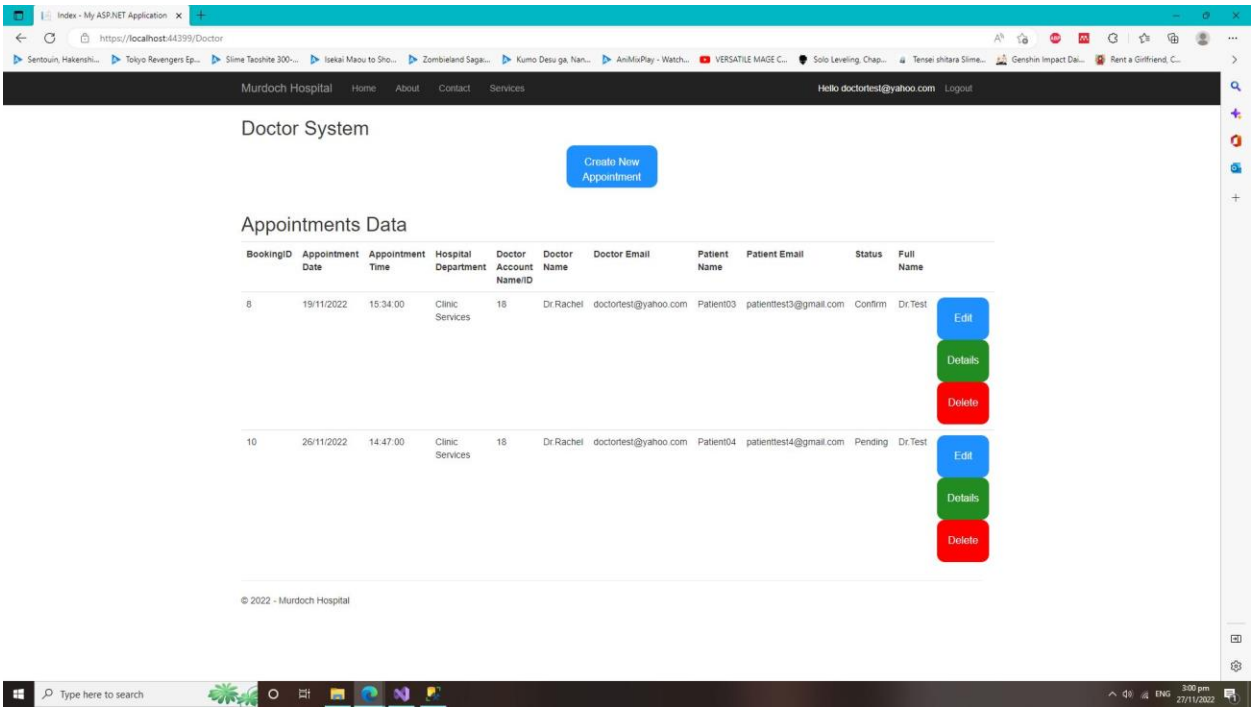




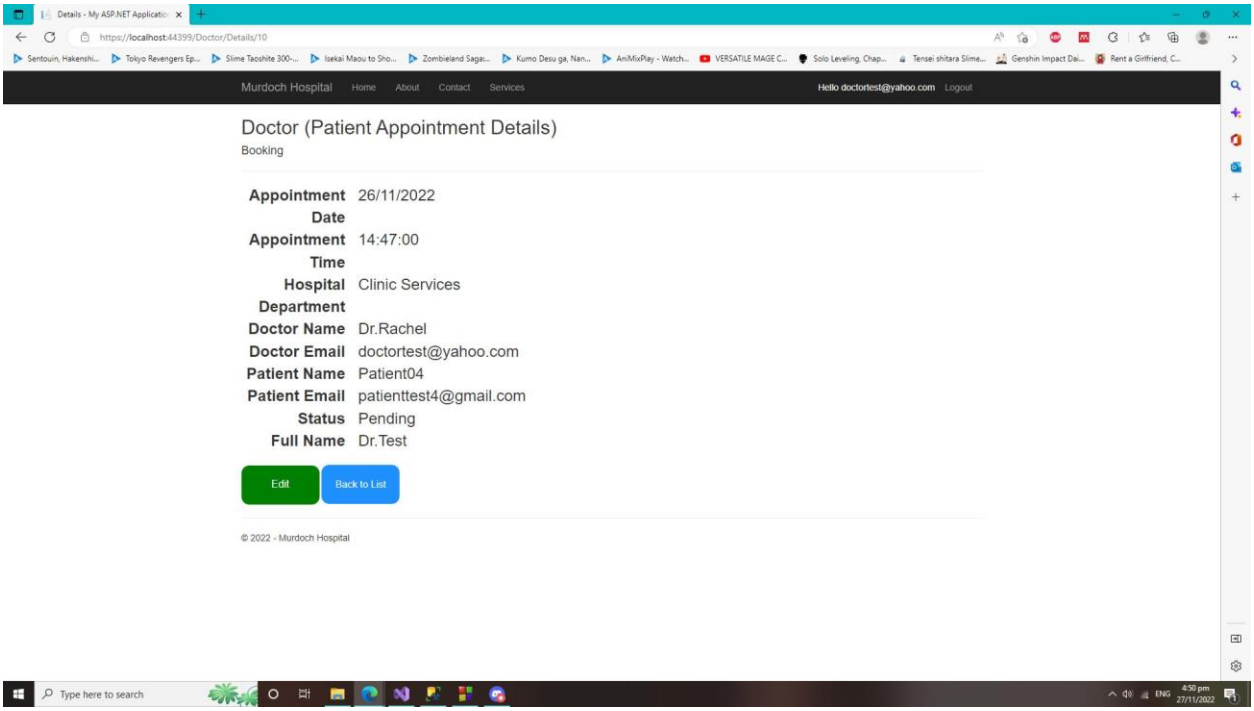


- Doctor Control Panel (Delete Page). This webpage is where the doctors can delete or cancel their patients' appointments of booking. The appointment or booking that deleted will also be automatically deleted from the Booking database itself.

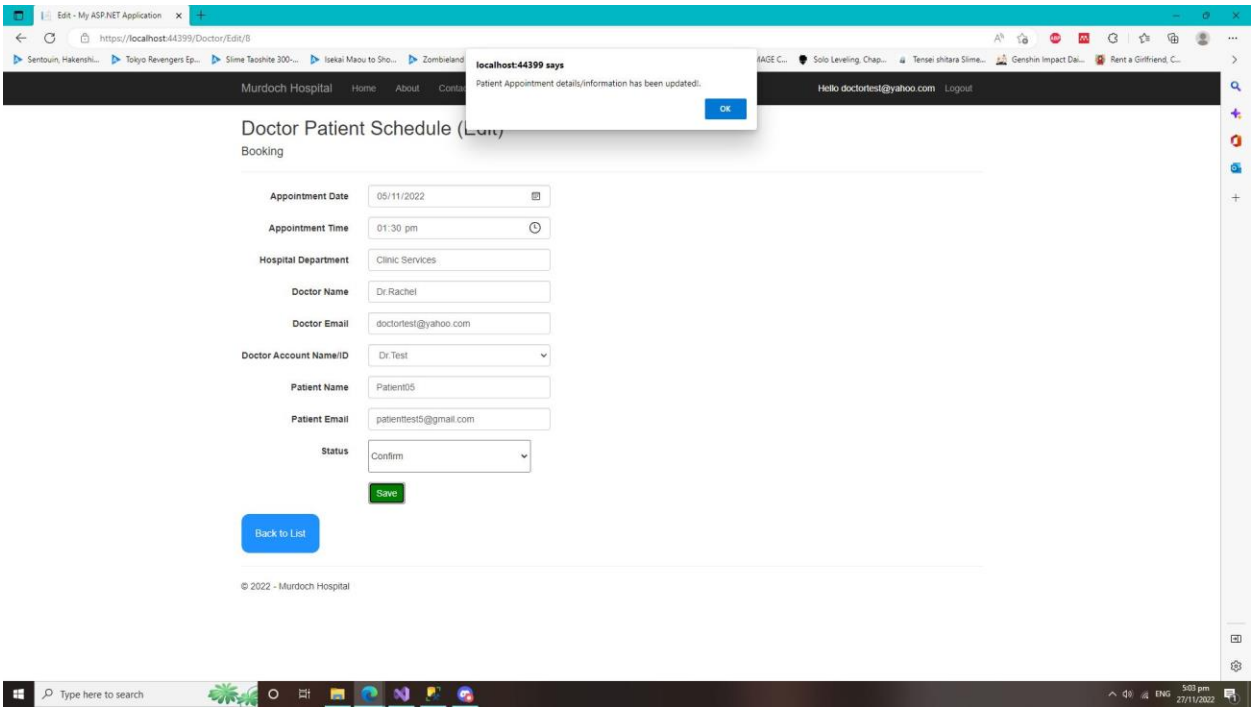


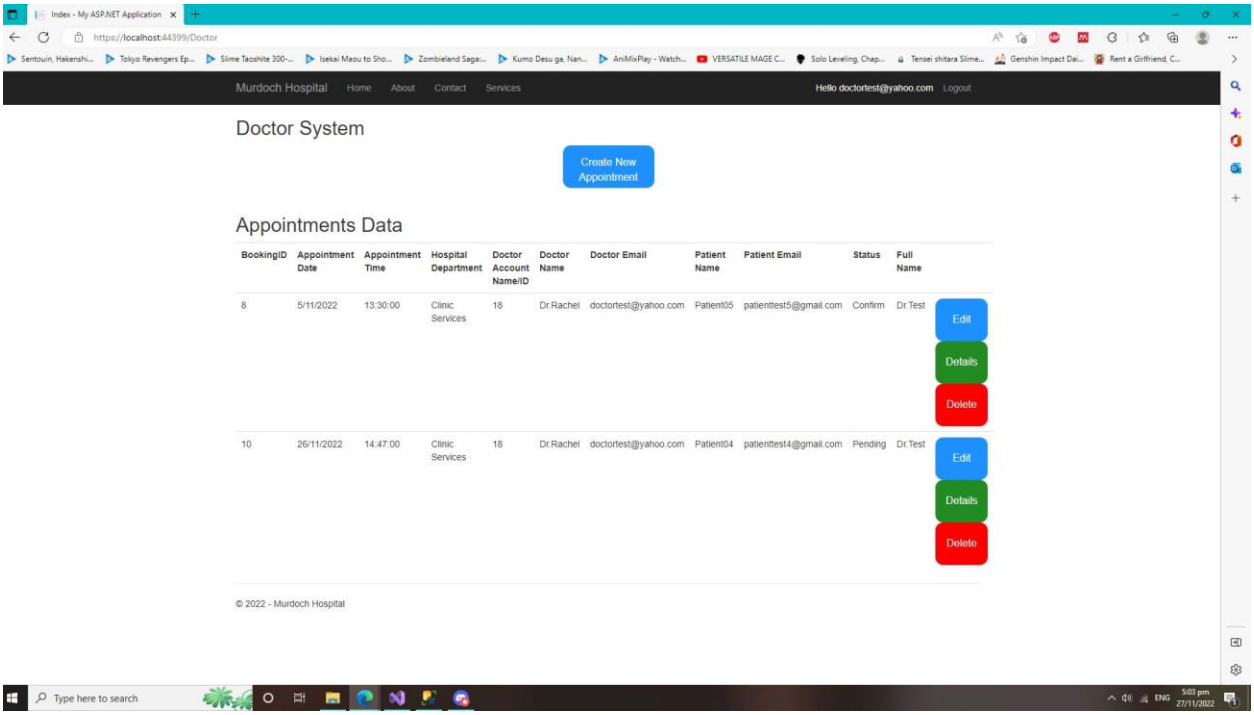


- Doctor Control Panel (Details Page). This is where the doctors can view their specific patients' appointments/booking details from this webpage itself.



- Doctor Control Panel (Edit Page). This is where doctors can edit their patients appointments and booking details if they wish to update any of it.


Edit - My ASP.NET Application
https://localhost:44399/Doctor/Edit/8
localhost:44399 says
Patient Appointment details/information has been updated!.
OK
Murdoch Hospital
Home
About
Hello doctortest@yahoo.com
Logout
Doctor Patient Schedule (Edit)
Booking
Appointment Date
05/11/2022
Appointment Time
01:30 pm
Hospital Department
Clinic Services
Doctor Name
Dr.Rachel
Doctor Email
doctortest@yahoo.com
Doctor Account Name/ID
Dr.Test
Patient Name
Patient05
Patient Email
patienttest5@gmail.com
Status
Confirm
Save
Back to List
© 2022 - Murdoch Hospital
Type here to search


Index - My ASP.NET Application
https://localhost:44399/Doctor
Murdoch Hospital
Home
About
Contact
Services
Hello doctortest@yahoo.com
Logout
Doctor System
Create New Appointment
Appointments Data
BookingID
Appointment Date
Appointment Time
Hospital Department
Doctor Account Name/ID
Doctor Name
Doctor Email
Patient Name
Patient Email
Status
Full Name
8
5/11/2022
13:30:00
Clinic Services
18
Dr.Rachel
doctortest@yahoo.com
Patient05
patienttest5@gmail.com
Confirm
Dr.Test
Edit
Details
Delete
10
26/11/2022
14:47:00
Clinic Services
18
Dr.Rachel
doctortest@yahoo.com
Patient04
patienttest4@gmail.com
Pending
Dr.Test
Edit
Details
Delete
© 2022 - Murdoch Hospital
Type here to search