Software Development Frameworks Unitcode ICT365

Student Name	Saw Gwe Aw
Student Number	33577055
Assignment 2 Title	Hospital Management System

Contents

Introduction

The Hospital Management System is integrated system, which specialises in hospital management and related affairs, appointments, and doctors' diagnosis. Developed with XML and LINQ this one provides creating value, modification, reading value, and deletion operations.

This system serves three user roles: administrators, physicians, and users of the services. Among all the actions administrators control such aspects as creating user, modifying and deleting the account. Upon registration, simple biographic information including name, email address, phone numbers as well as date of birth are taken, plus features related to the account type, for example, a patient's medical history or a doctor's area of specialty.

The system allows patients to schedule an appointment, check up an appointment or even cancel an appointment that was made by the patient. During booking, they choose date, time, doctor from the available list of the respective parameters. To manage appointments doctors deploy an attendance form where patients get to view details of the diagnosis in a convenient manner.

Currently, it provides administrators, patients, and doctors with different interfaces and functionalities as it is adjusted to the user role. The chosen platform is highly informational and functional, which helps to increase operational performance in contemporary facilities.

Assumptions

To ensure smooth operation, the Hospital Management System follows these key assumptions:

Admin Privileges: Admins are capable of generating new admin accounts though they cannot modify or delete any admin account including their own. This protects systems security and eliminates accidental interferences.

Patient Bookings: Patients can only make booking and also cancel them but cannot amend any confirmed booking either at any time. This enables the right doctor to be scheduled for a particular patient and FOA's at any given time.

Fixed Time Slots: Appointments take an hour and are also fixed and in one hourly intervals to avoid paradoxical prescheduling and make a doctor's schedule more manageable. Time selection is on the dropdown menu for better order and structuring of the utility.

These actions improve security, simplify accesses and accommodations, and increase scheduling efficiency.

Assumption for Admin Data Representation

To simplified the data structure and querying , maintenance, Admin and Doctor roles sharing the same XML elements in this hospital management system.

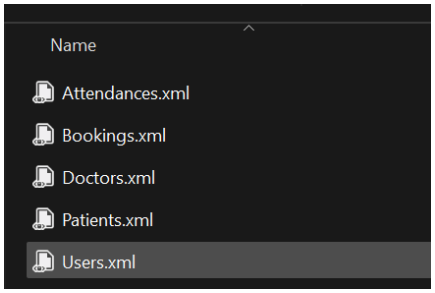
The AccountType field (stored as a sub-element) is used to distinguish between the roles:

- 0: Admin
- 1: Doctor
- 2: Patient

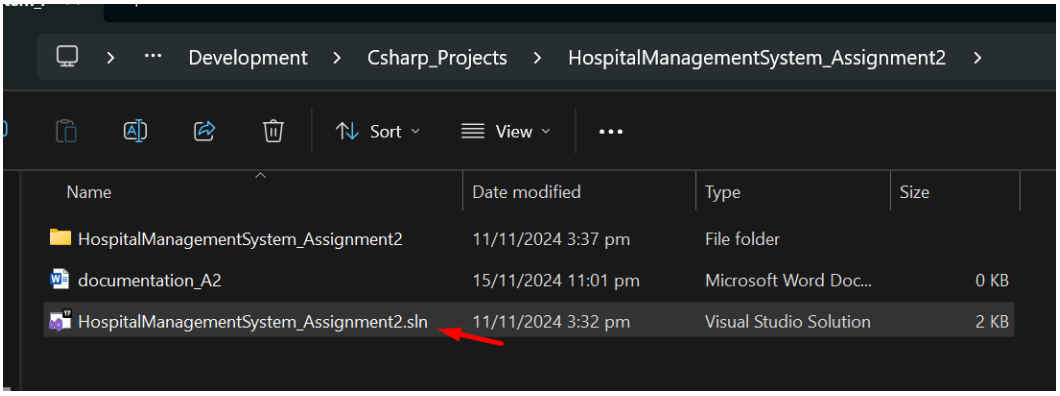
User Guide

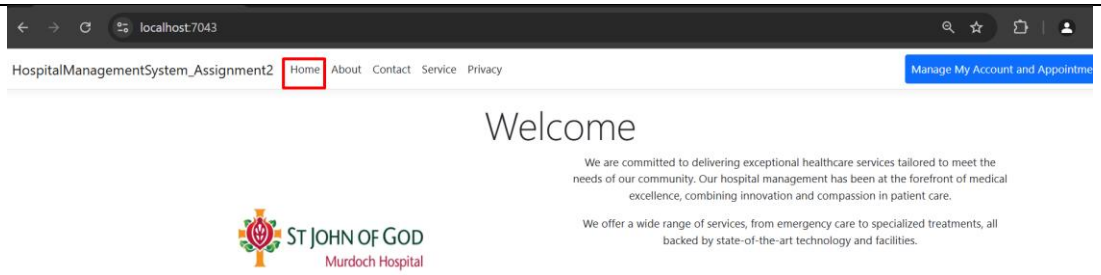
XML file path:

HospitalManagementSystem_Assignment2\HospitalManagementSystem_Assignment2\App_Data



Program Execution:



1. Home page	

HospitalManagementSystem_Assignment2 Home About Contact Service Privacy Manage My Account and App

About Us

We are one of Western Australia's leading private health campuses, providing excellence in health care for our community..

St John of God Murdoch Hospital is a private hospital and a division of St John of God Health Care, one of Australia's largest health care providers. We are one of Western Australia's leading private health campuses, providing compassionate and high quality health care for our community. Stage 2 of the hospital's redevelopment project is now complete - the last component being the renovation of the original six 1994 wards.

Established in 1994, St John of God Murdoch Hospital provides:

- a 24 hour Emergency Department
- medical and surgical services
- paediatrics
- maternity
- palliative care
- critical and coronary care.

There has been significant growth in research at the Hospital, in the areas of clinical oncology trials, orthopaedics, emergency medicine, intensive care, cardiology and nursing-led research. Many studies are delivering impressive results, with the focus on patient-centred management and innovative approaches to evaluating patient outcomes.

ınment2 Home About Contact Service Privacy 50% Reset

Contact Us

We value your feedback and inquiries. If you have any questions, concerns, or suggestions, please feel free to reach out to us using the information below.

Information	Details
Hospital Name:	St John of God Murdoch Hospital
Head Office Address:	St John of God Health Care Level 1, 556 Wellington Street Perth WA 6000
Phone:	08 6116 0000
Melbourne Office Address:	St John of God Health Care Level 4, 360 Collins Street Melbourne VIC 3000
Phone:	03 9205 6500
Email:	stjohn@email.com
Website:	stjohn.com
Office Hours:	Monday to Friday: 8:00 AM - 8:00 PM Saturday: 9:00 AM - 5:00 PM Sunday: Closed
Emergency Contact:	+61 1988 8888

We are here to help you and provide the care you need.

alManagementSystem_Assignment2 Home About Contact Service Privacy Manage My Account and Appointm

Our Service

Providing compassionate and high-quality healthcare to the community

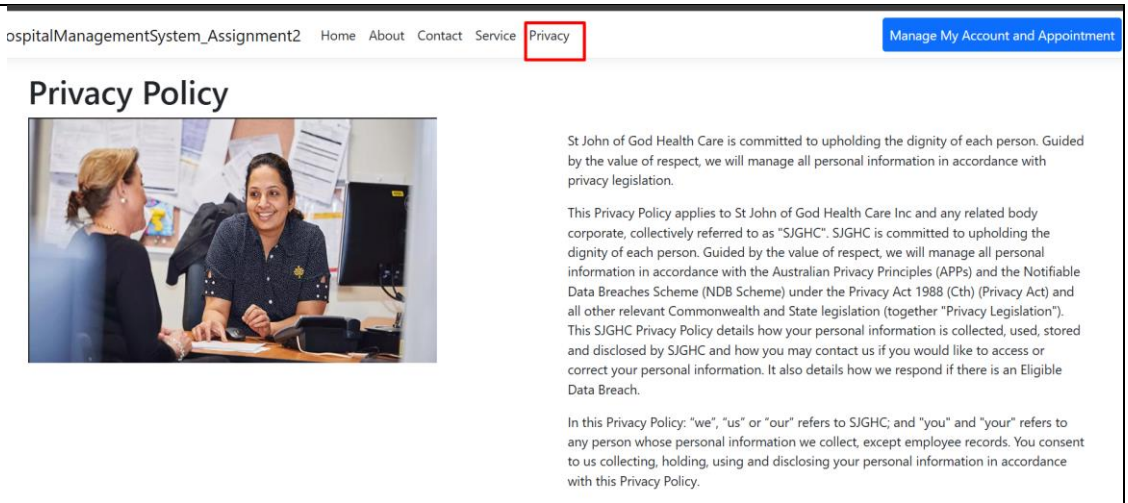
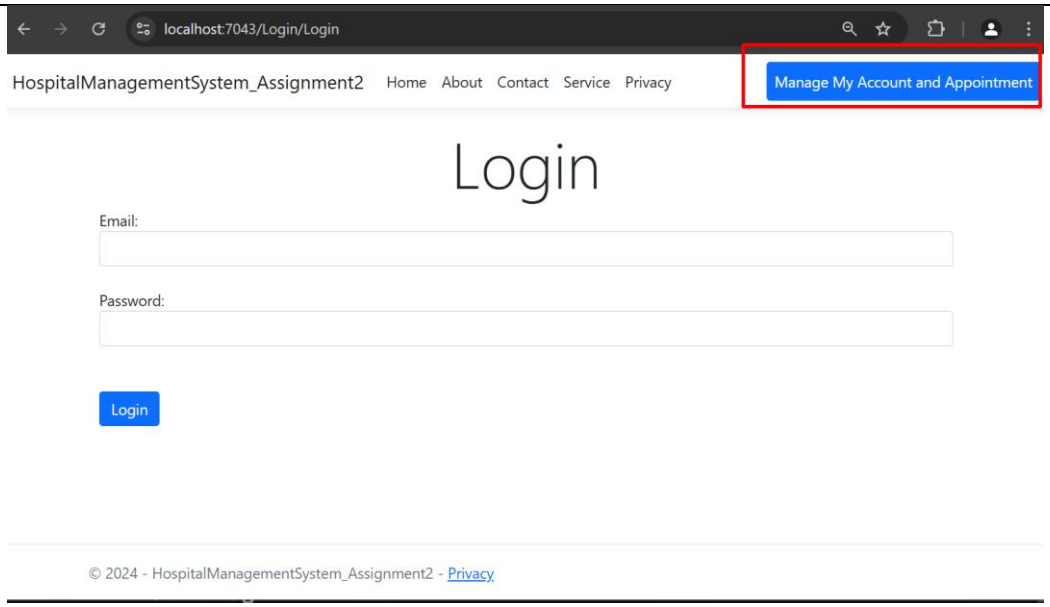
Emergency Care

Our 24-hour Emergency Department is equipped to handle a wide range of urgent medical conditions. With highly trained doctors, nurses, and support staff, we ensure prompt and efficient care in emergencies.

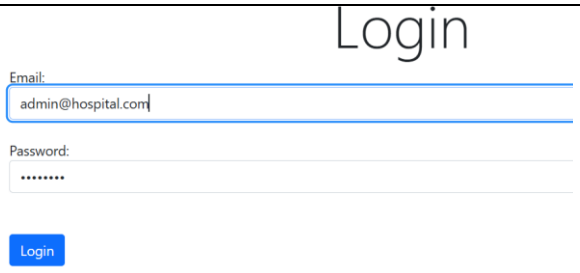
Maternity Services

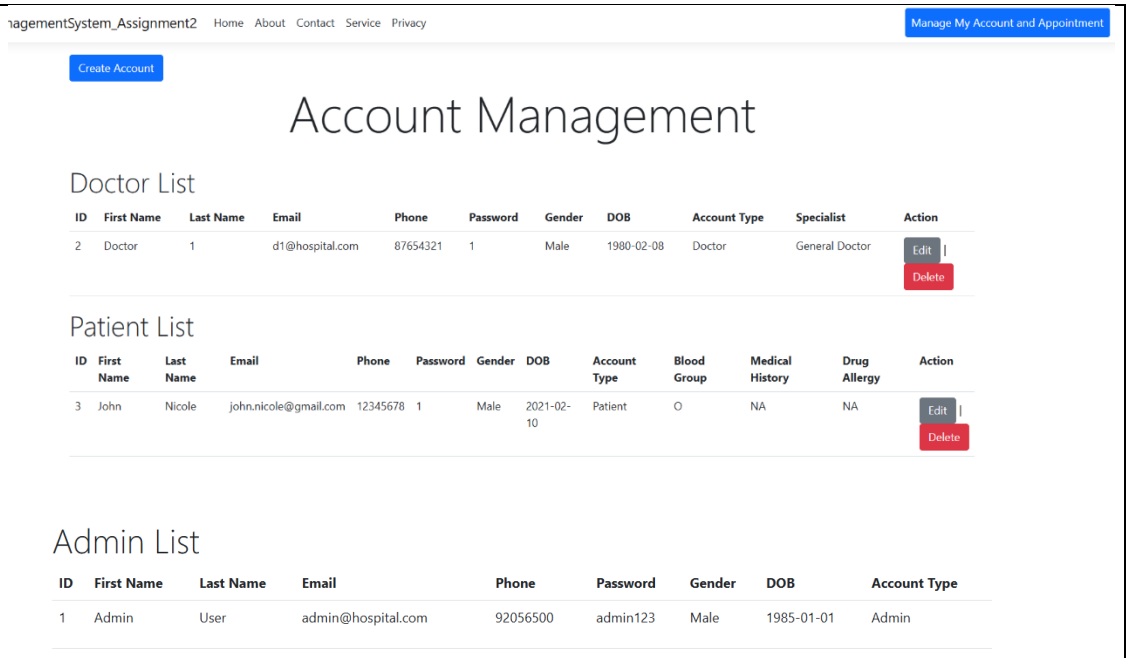
St. John of God Murdoch Hospital provides exceptional maternity care, including antenatal and postnatal support, labor and delivery services, and specialized care for both mothers and babies. Our maternity team ensures that every new parent experiences a smooth and safe journey.

Cancer Care

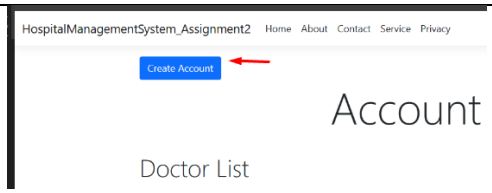
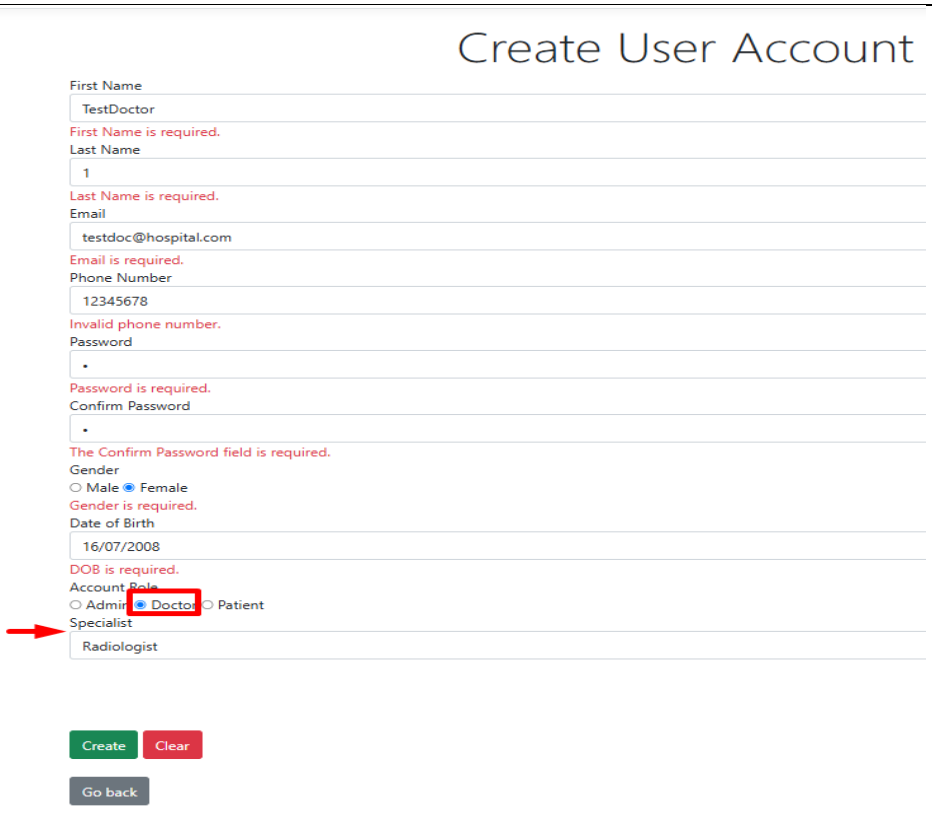
	
	

Admin role guide

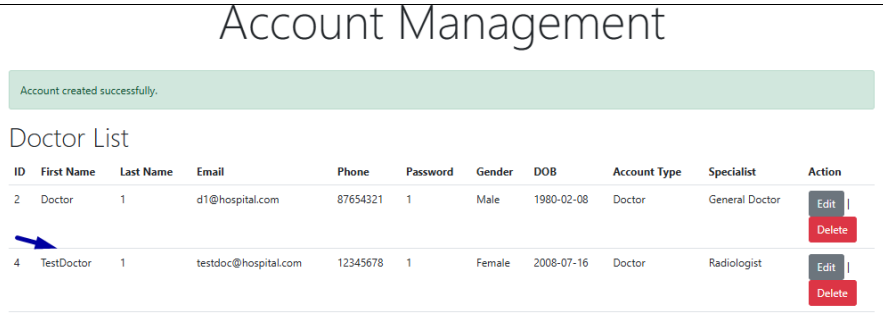
Admin account	Email: admin@hospital.com Password: admin123
Login admin account	

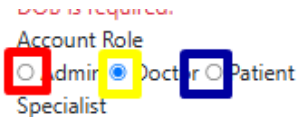
Create Admin / Doctor / Patient Account

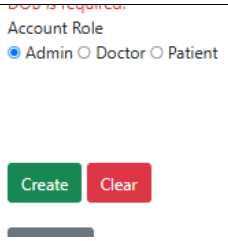
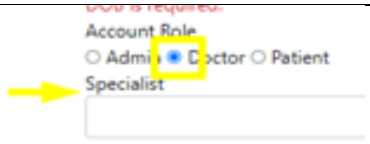
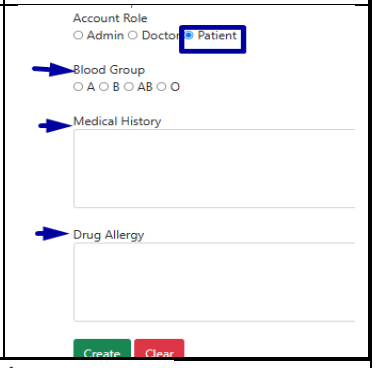
Click [Create Account]	
Fill all the required field Radio button	

New Doctor list is created.



This [Radio button] will show addition relevant field upon selection.



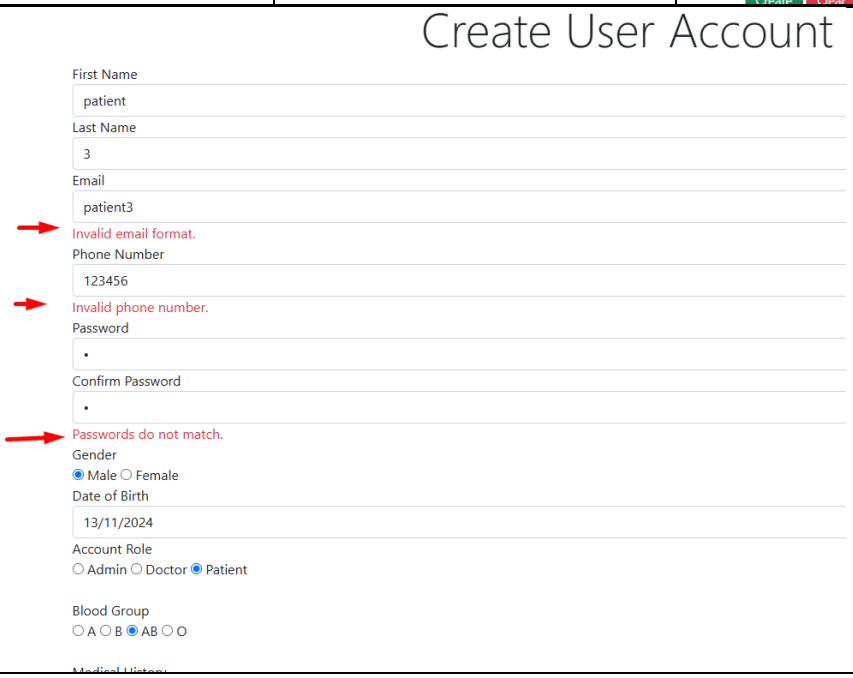
Admin: Doesn't have addition field	Doctor: has additionl field	Patient:has additional field
		

Input validation:

On Email: should have email format "@".

Phone number: should have 8 digits

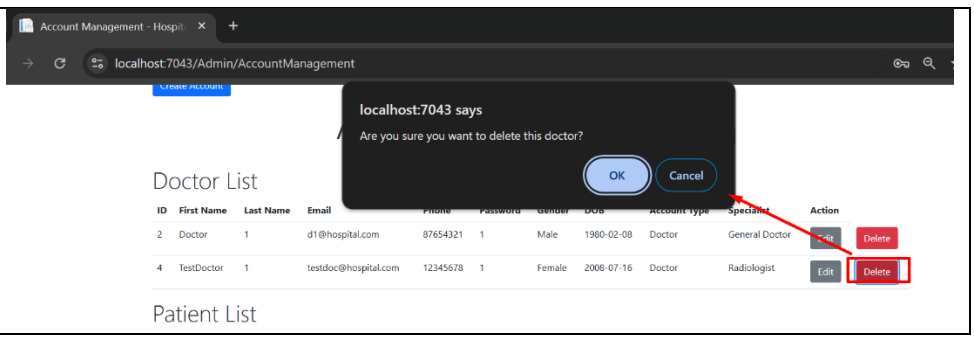
Confirm password: if the 1st password and confirm password didn't match. Show error

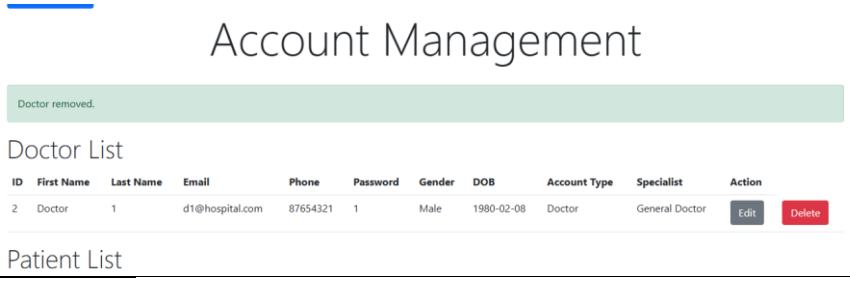
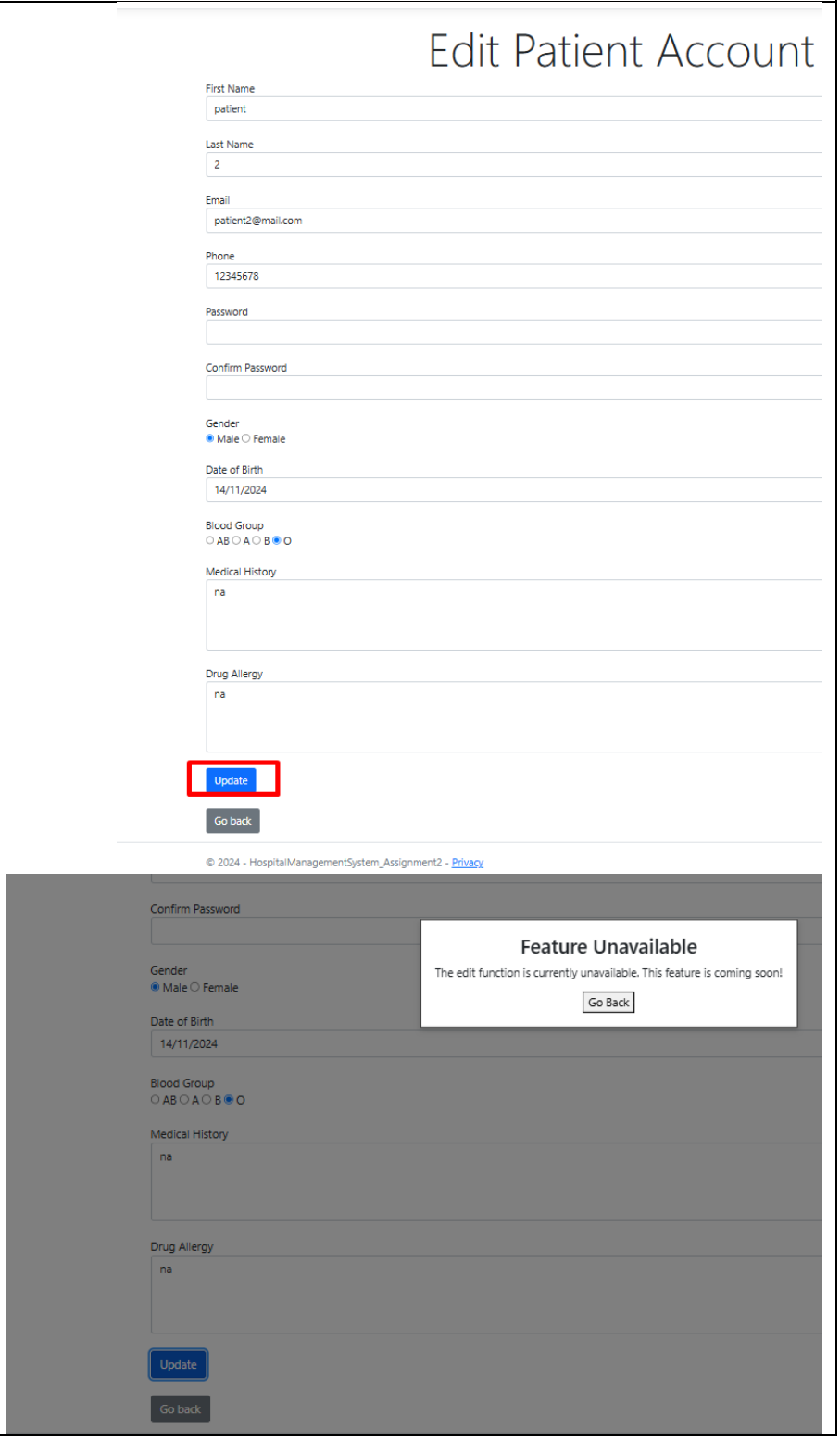


Admin Role user guide Edit/ Delete

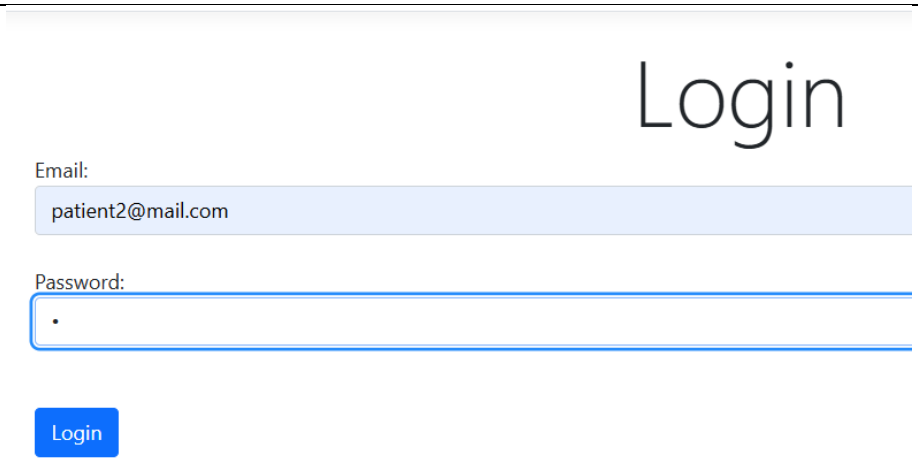
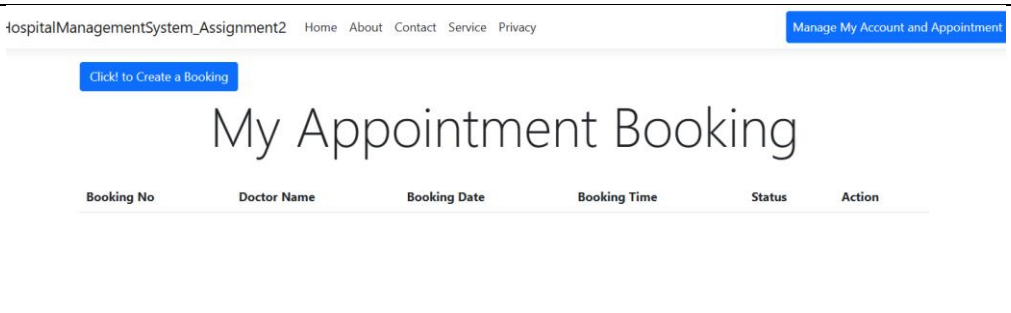
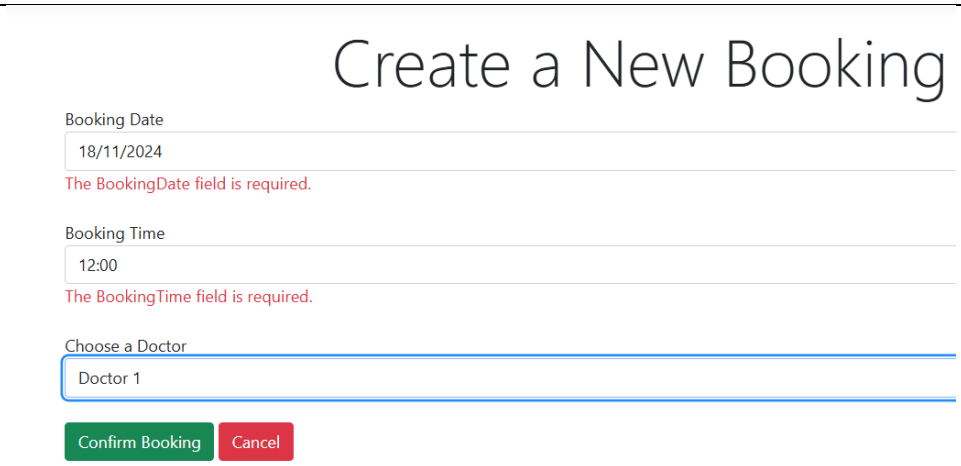
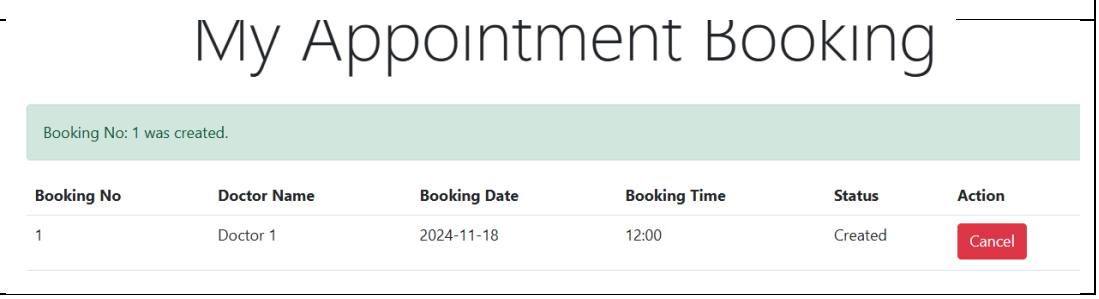
[delete] button on Doctor List

Pop up for the confirmation.

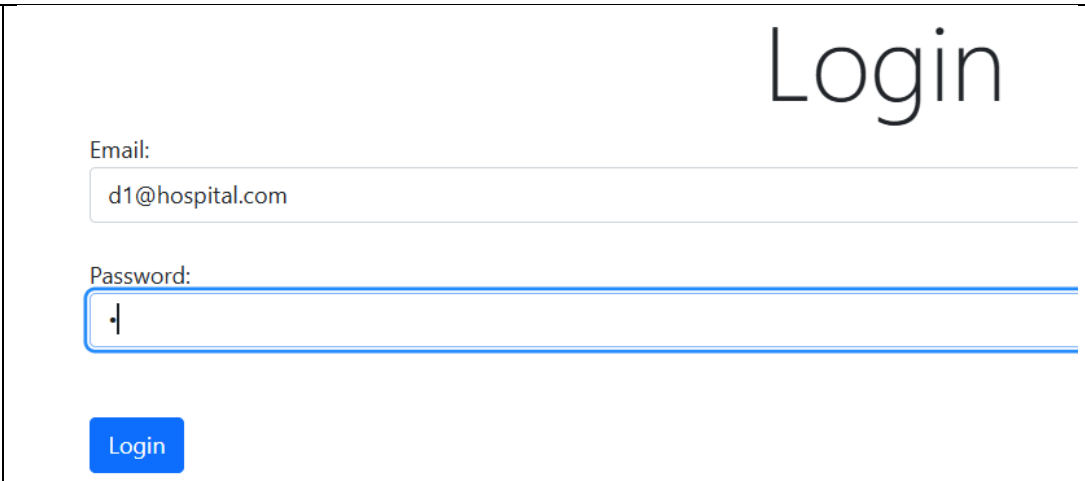
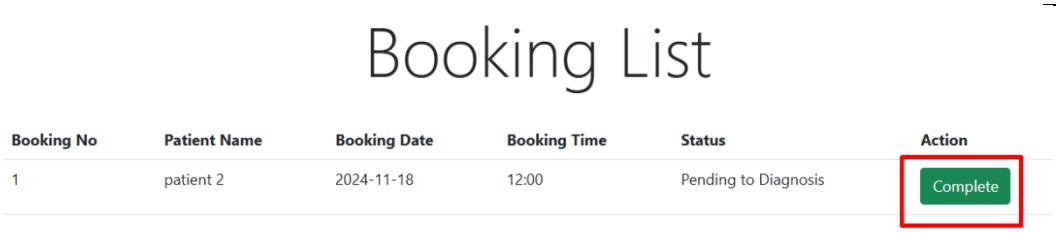
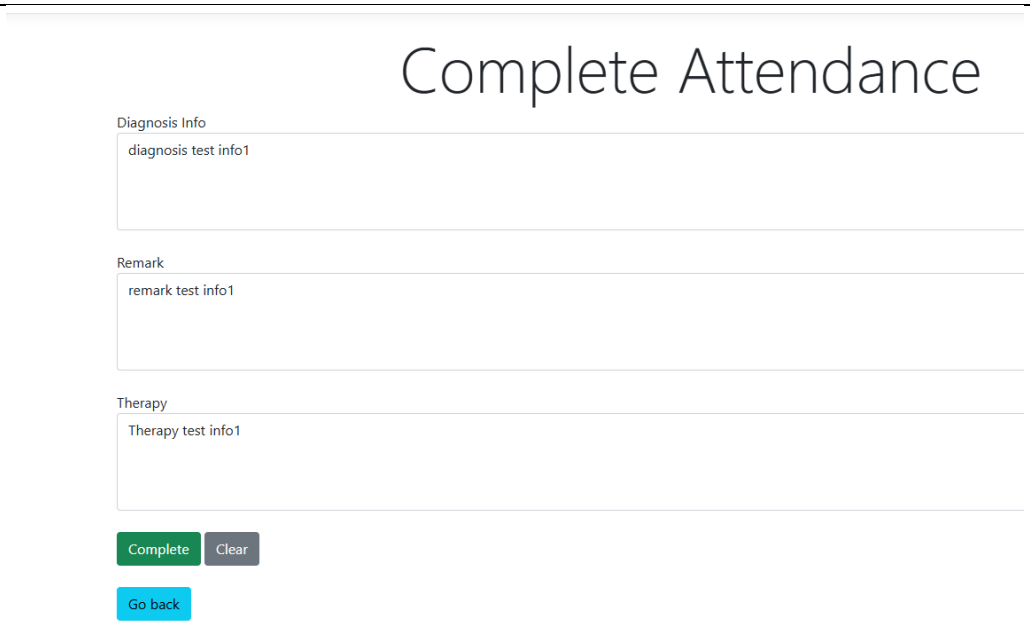
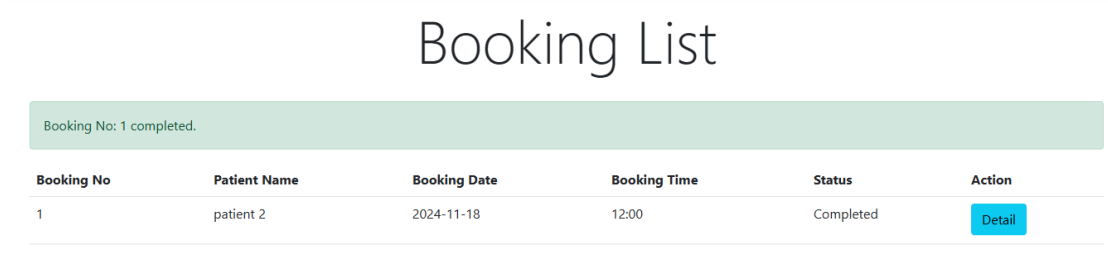


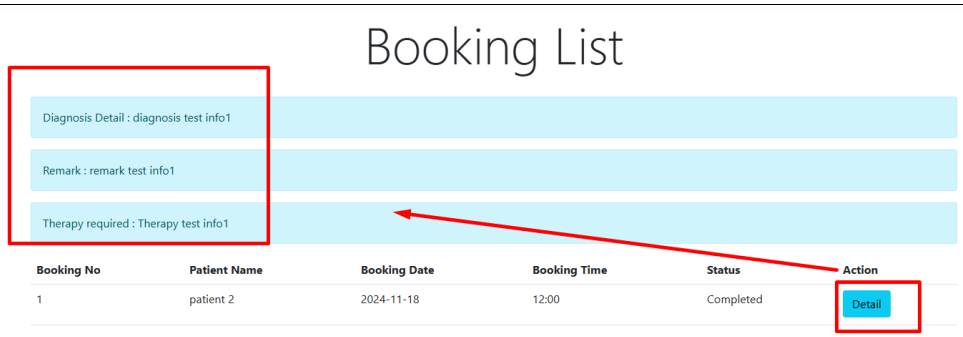
	
[update] Existing data is currently unavailable	

Patient Role user guide

Login Patient account	
Currently this user doesn't has any booking information. [click to create a booking]	
Fill in all the required field [Confirm Booking]	
	

Doctor Role user guide

Login Doctor Account	
Doctor see the booking made by the patient. With status Pending to diagnosis [Complete] to next action to diagnose	
Fill in the all the field. And click [complete]	
Redirect back to the booking list with Status complete.	

Click [Detail] To view the diagnosis detail	

Limitation

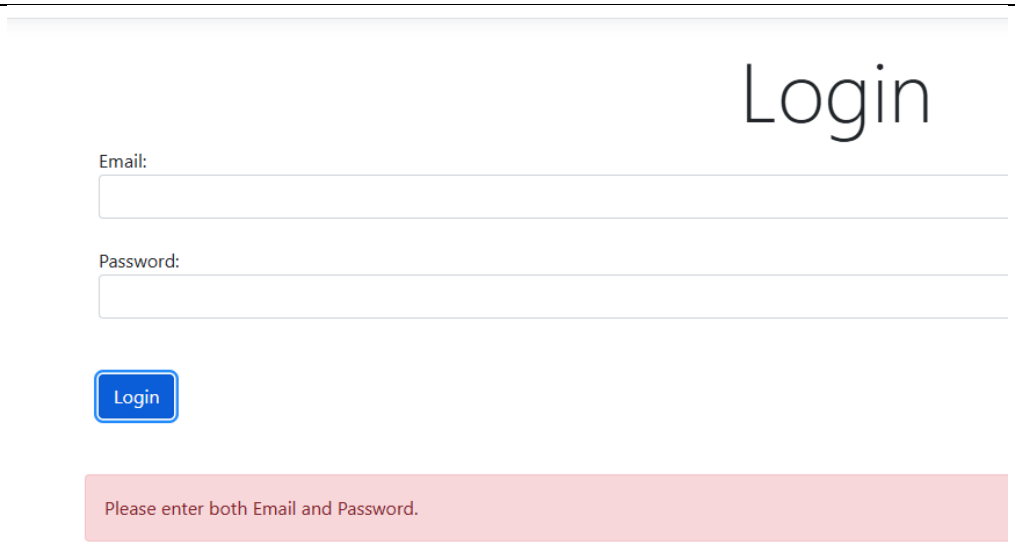
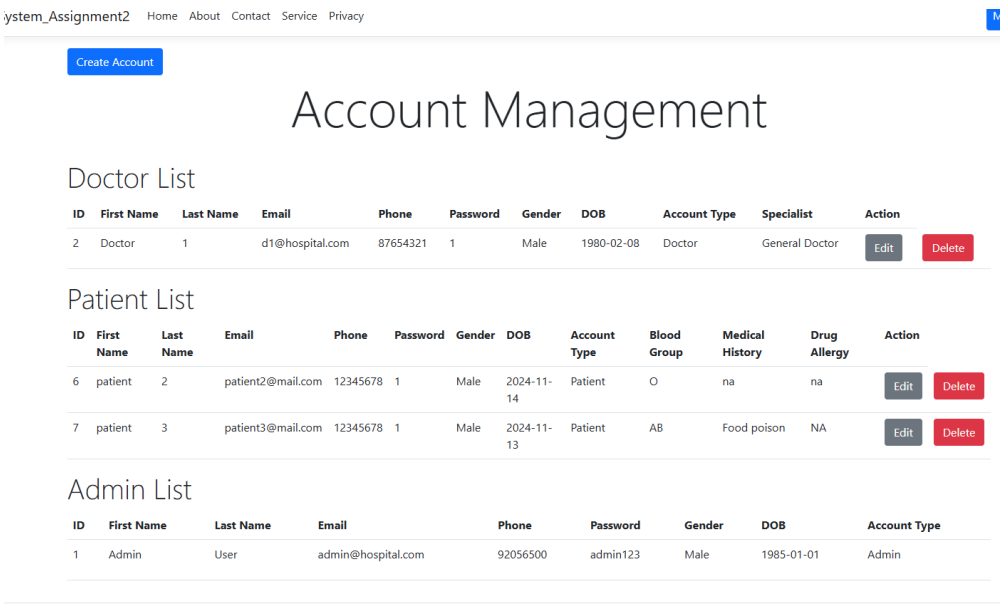
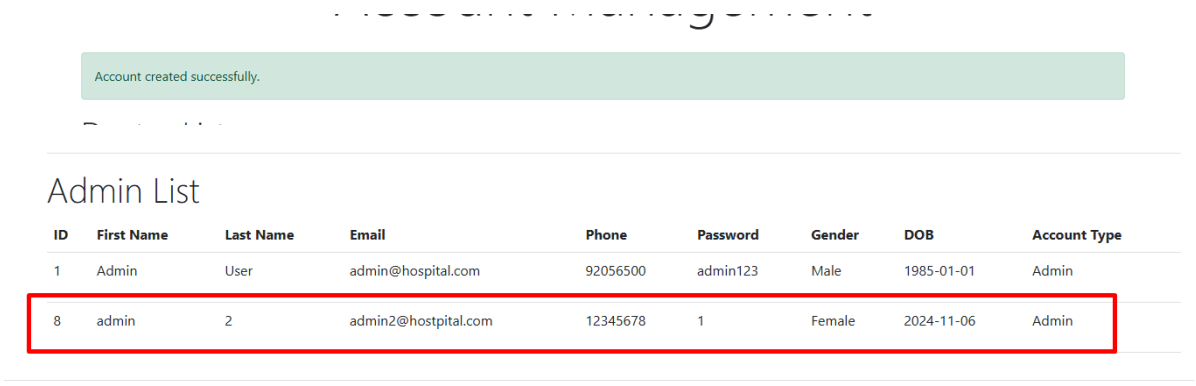
The followings are the limitation of the current Hospital Management system.

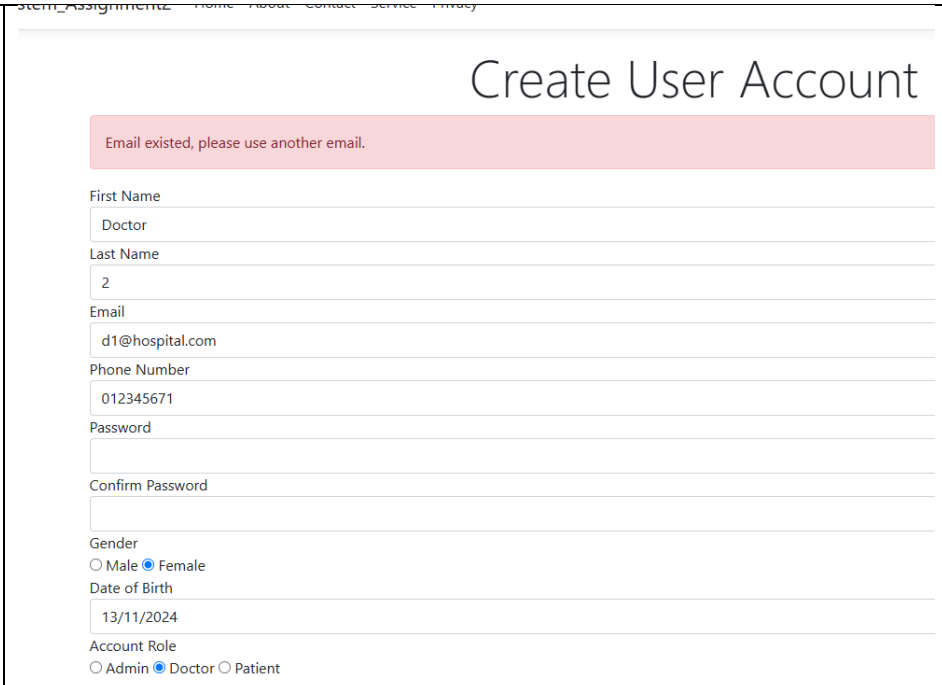
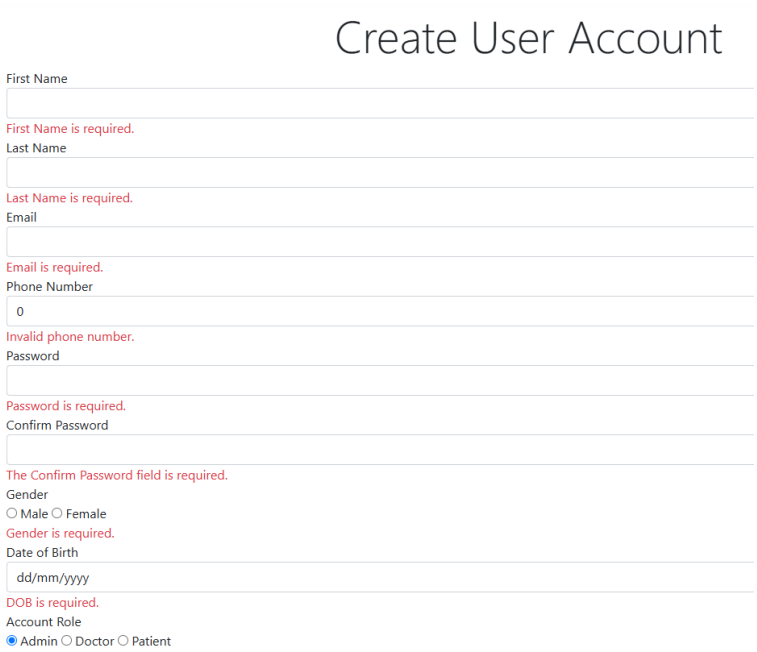
1. Inability to Edit the existing Records: Edit account of Doctors and Patient functions are not well developed.
2. Field Validation Limitations:
    a. Password validation: there is no checks for minimum length, presence of the letters/numbers in uppercase or lowercase and special characters.
    b. Username/ Email validation: can input any letter name on this email field with any domain name.
    c. Date of Birth (DOB): there is no mechanisms to guarantee that the users is of reasonable age (for example, some of the DOB can be set into the future or any unrealistic number).
3. Passwords are stored in plain text in XML file, and it is not encrypted.
4. Doctor's specialization: not enforced for users assigned the Doctors role, allowing them without a Specialisation field element.
5. Admin List: existing Admin user List can't be modified

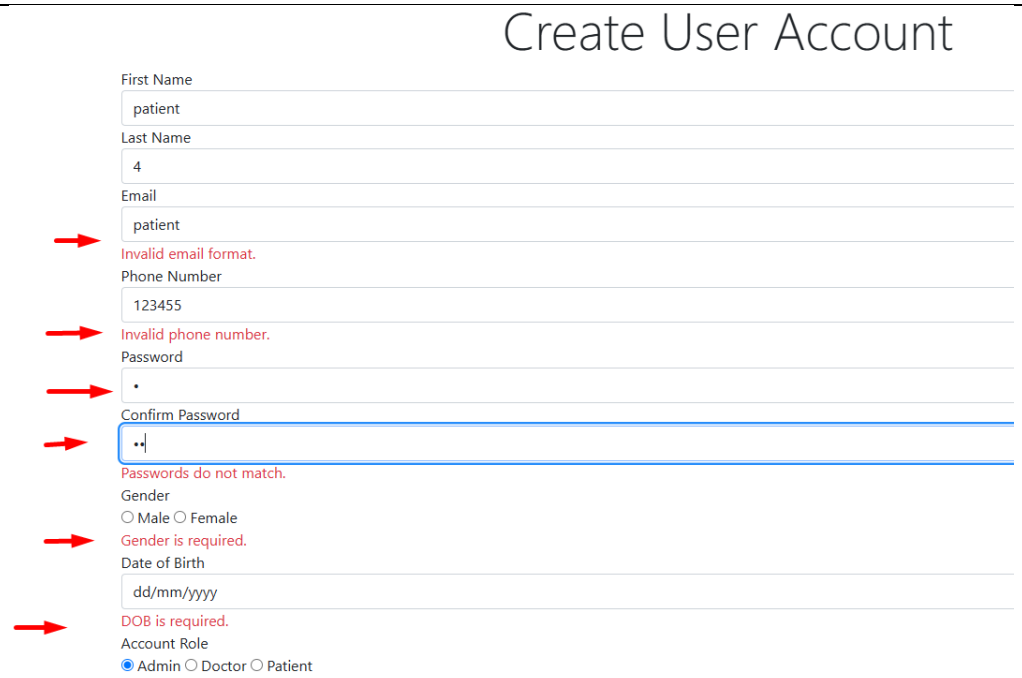
Testing

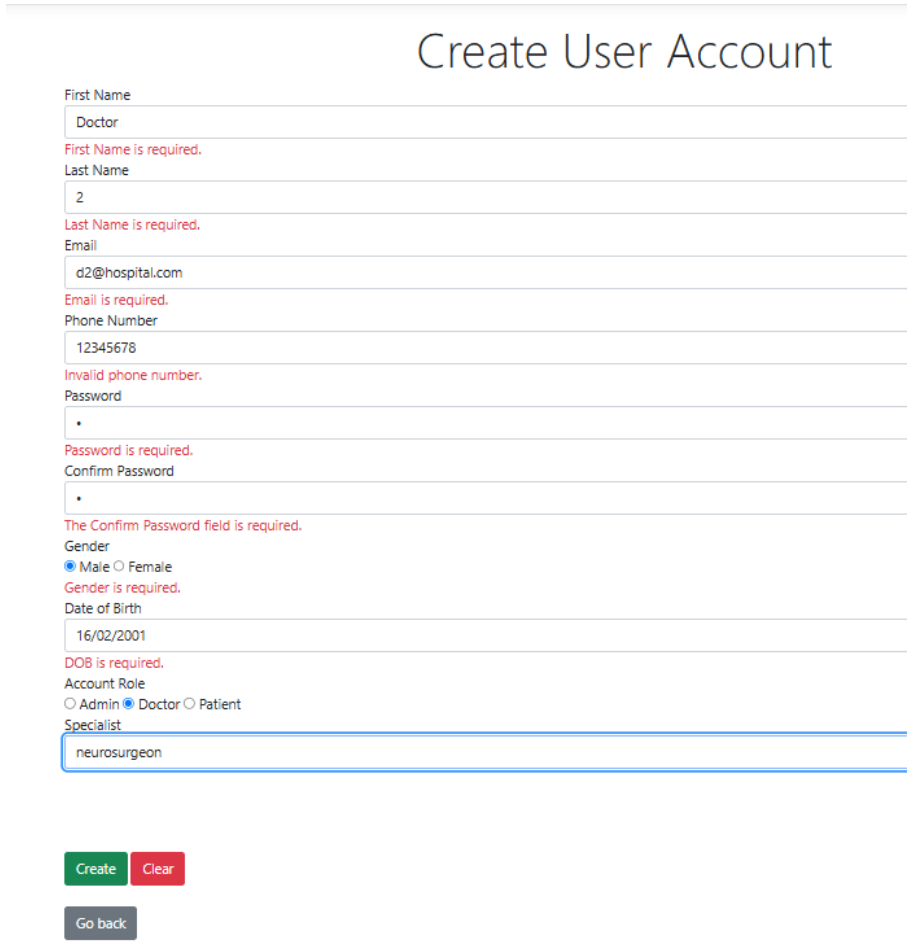
Test Plan and Test Result

No	Unit Test Scenario	Result
Login		
1	Login with empty input. System will prompt error.	Pass

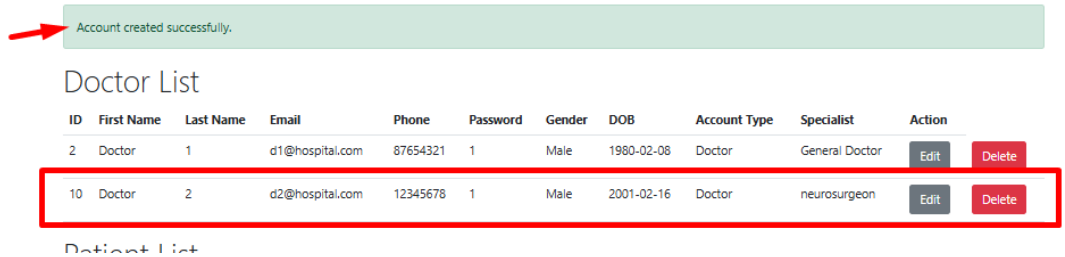
		
Admin		
2	Login as admin and display account management page. 	Pass
3	Add admin account. 	Pass
4	Using existed email to create account. System will reject to register.	Pass

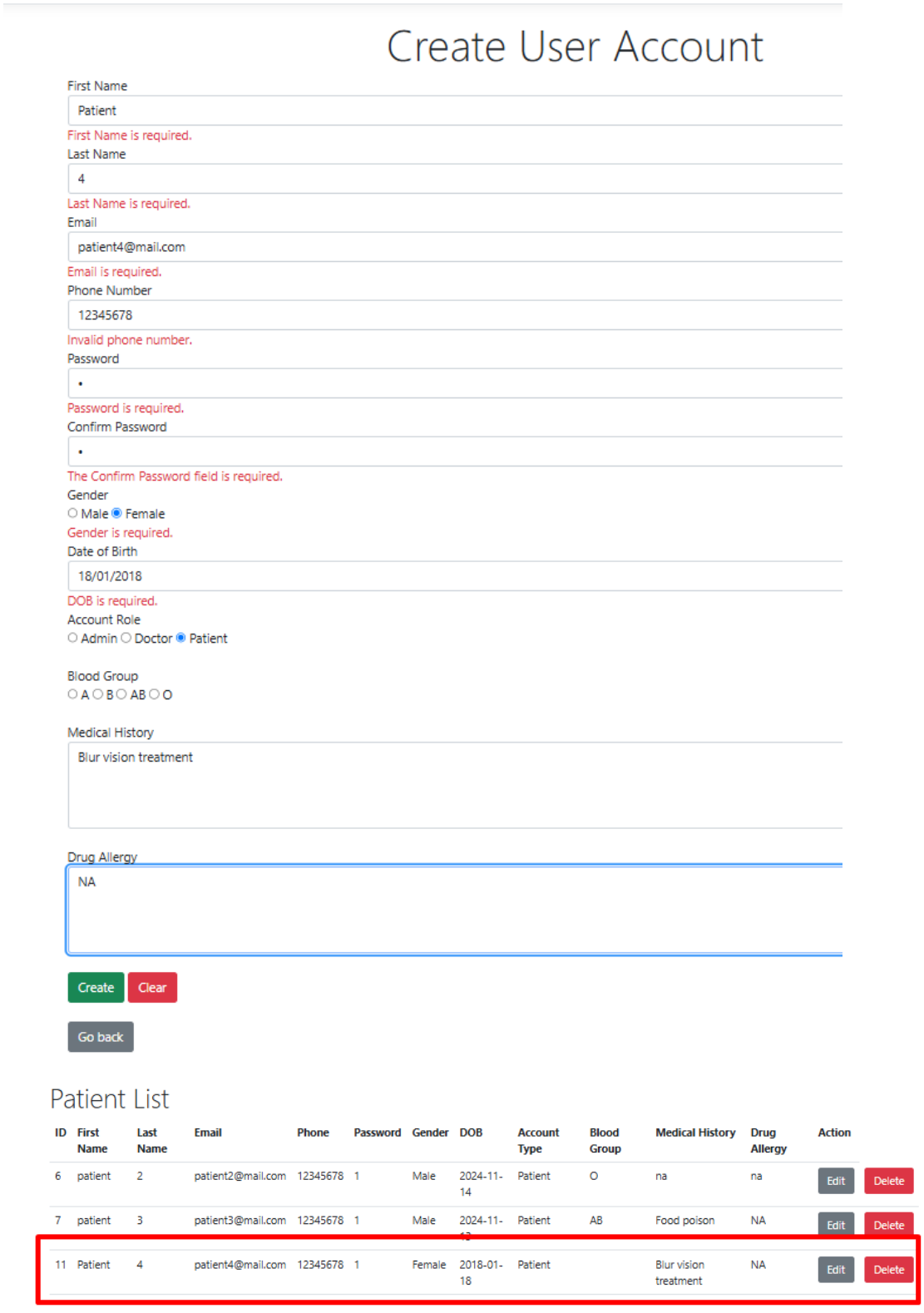
		
5	Input validation. Empty input:  Email: Phone: Password matching check:	Pass

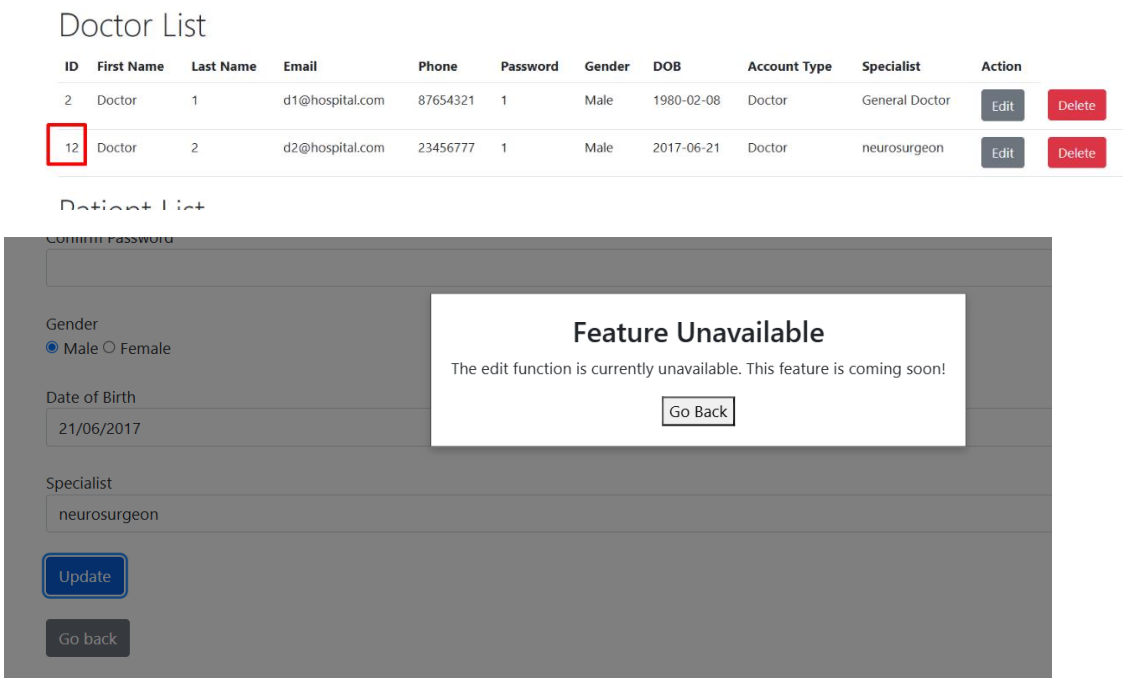
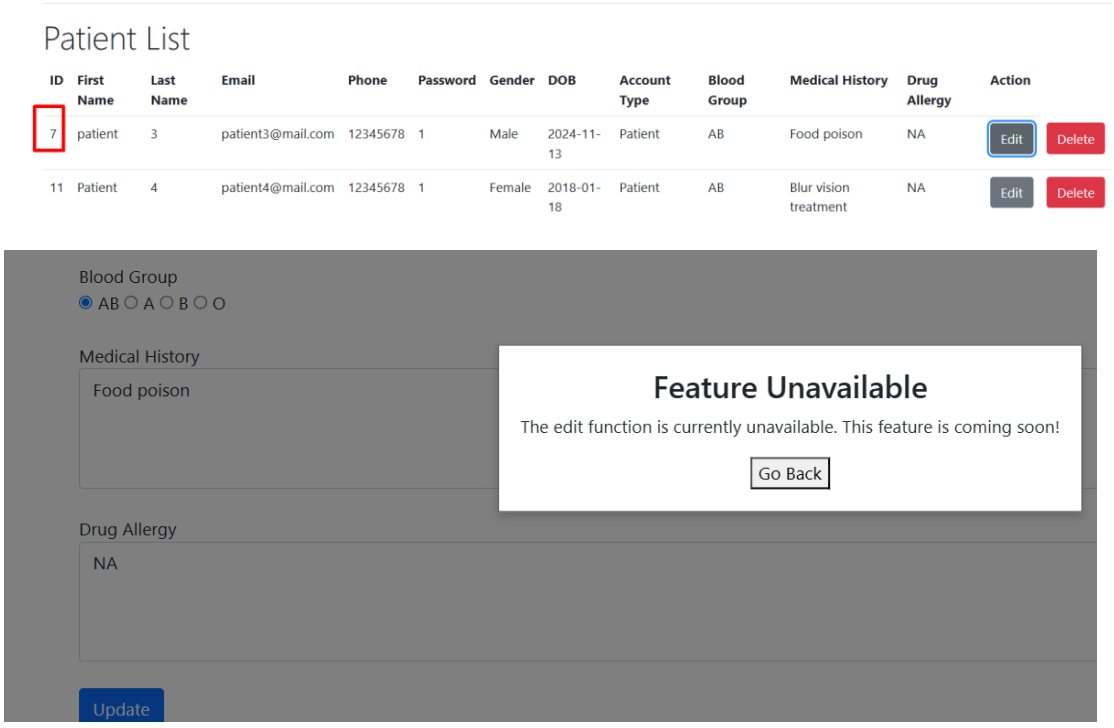
<table>
<tr>
<td></td>
<td>


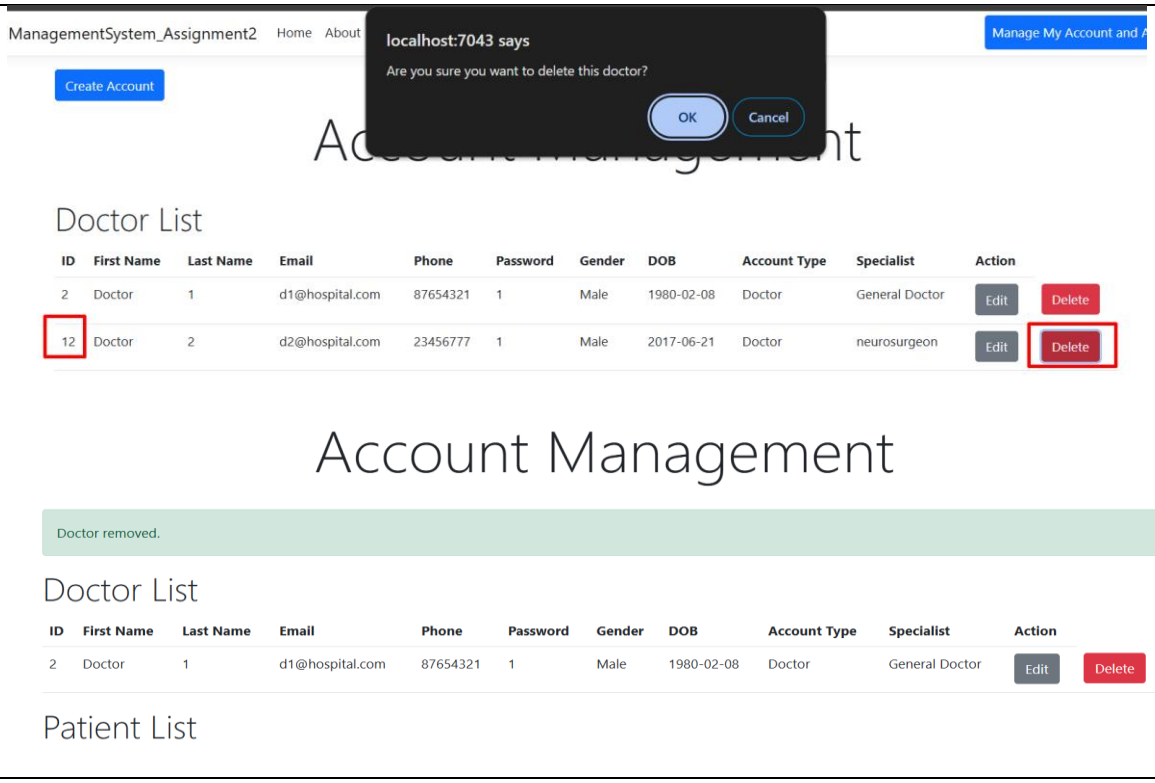
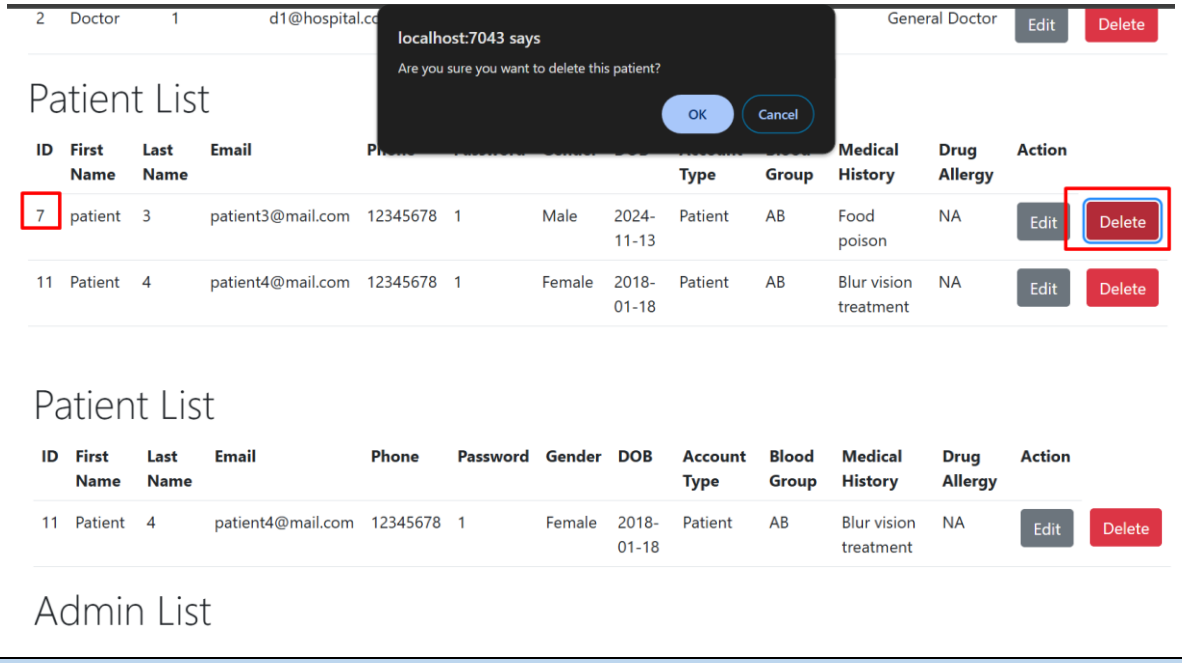
</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Add doctor account.


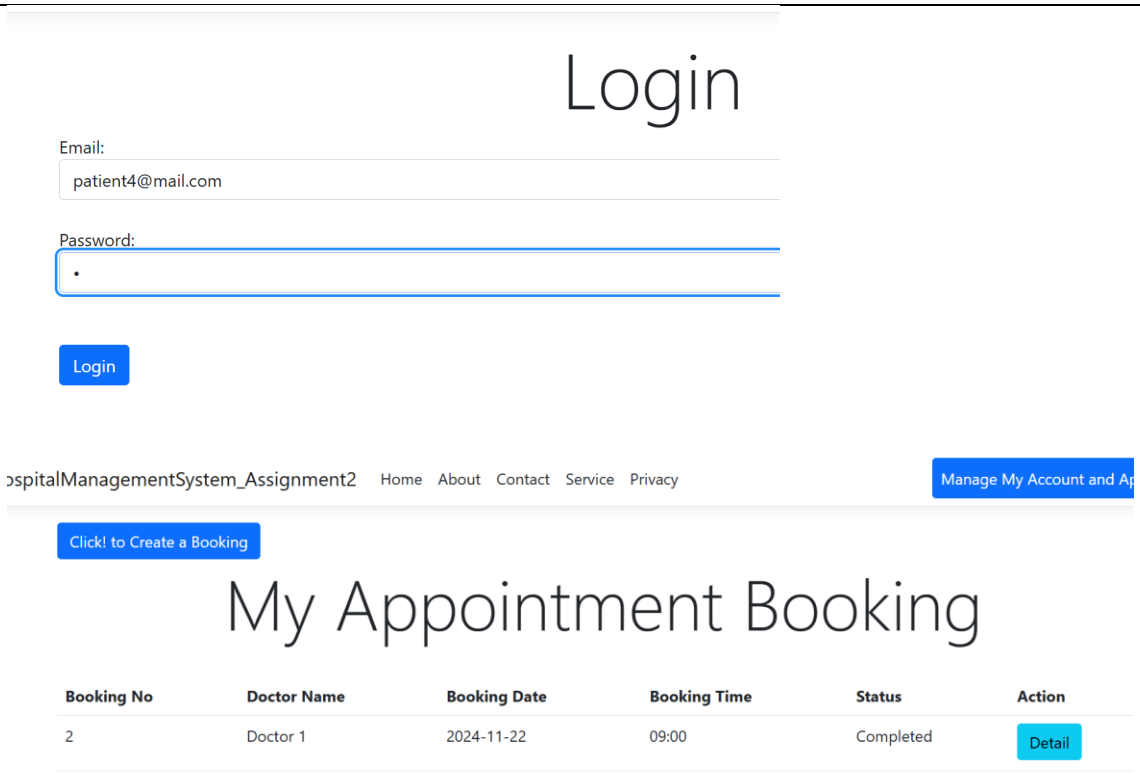
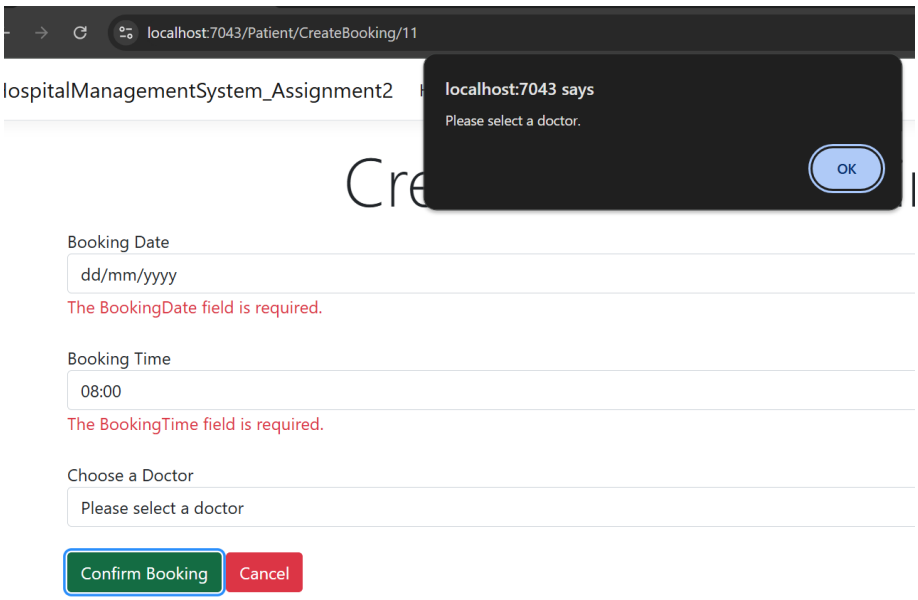
</td>
<td>Pass</td>
</tr>
</table>

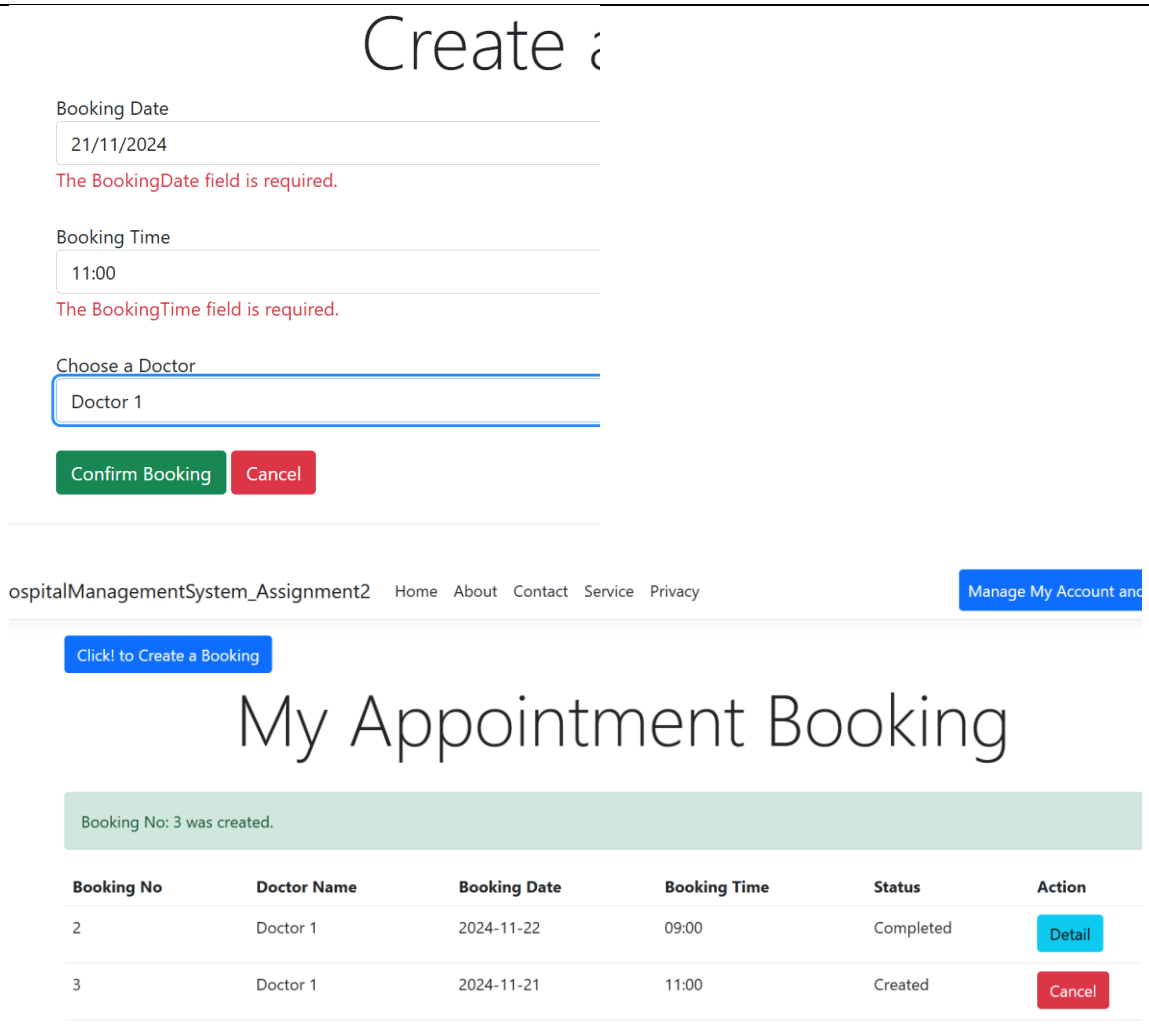
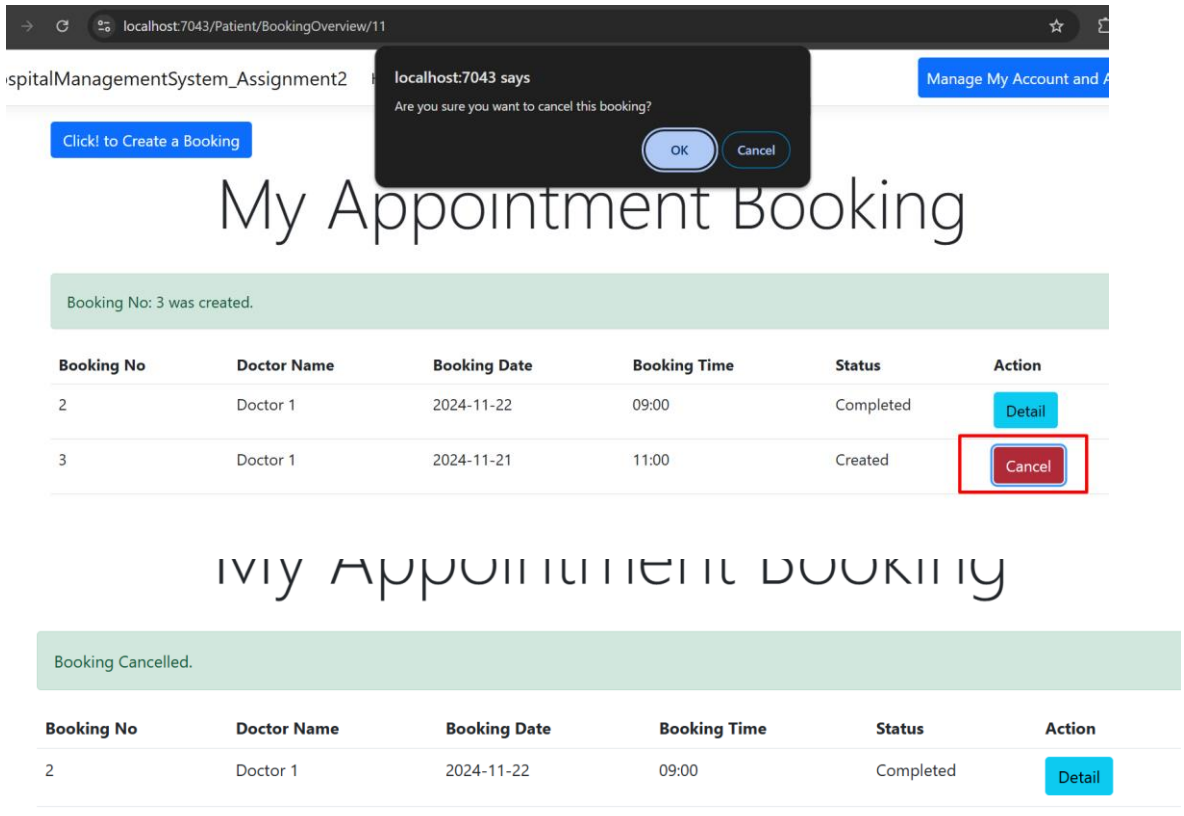


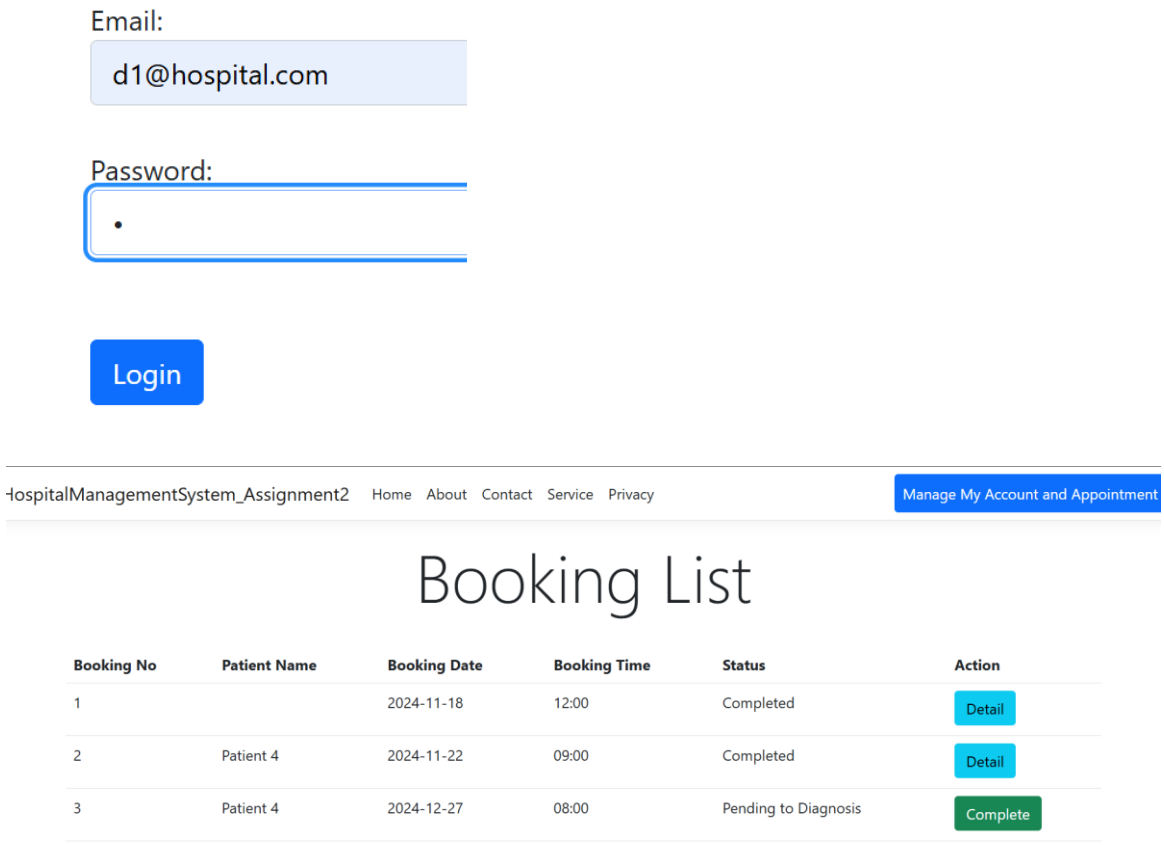
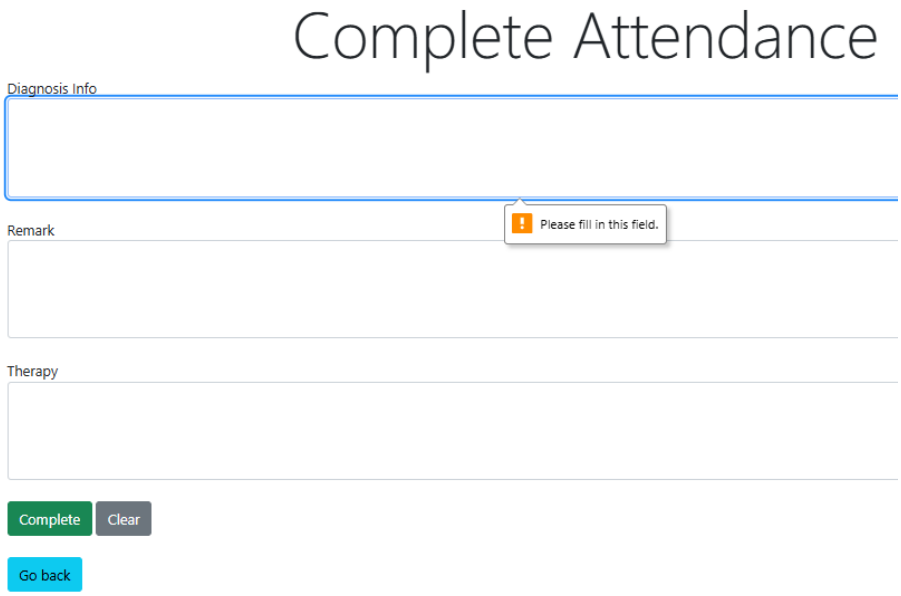
7	Add patient account.  	Pass

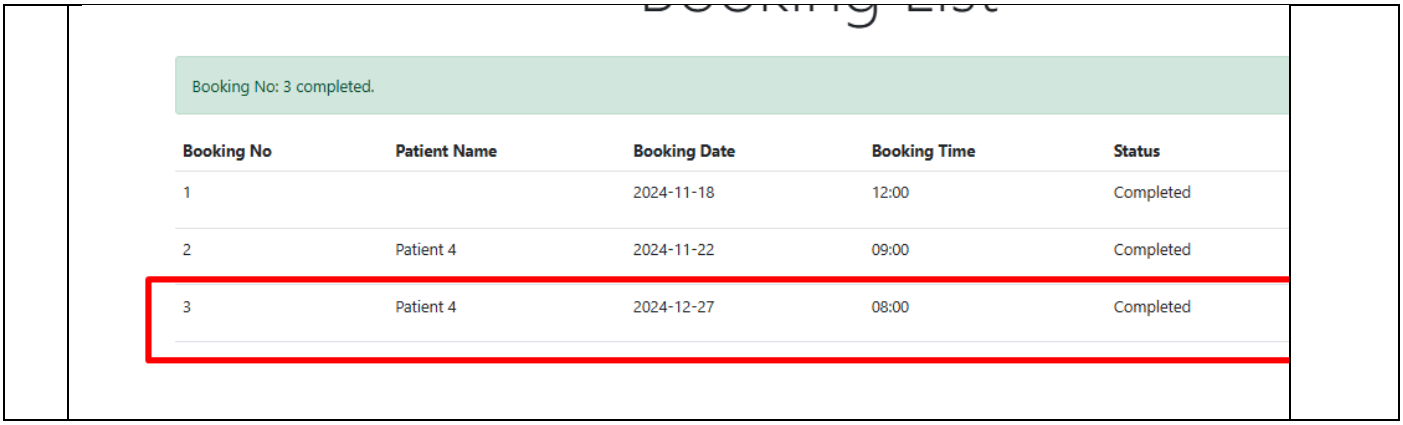
8	Update doctor account ID 12  	Fail
9	Update patient account.  	Fail
10	Delete a doctor ID 12	Pass

<table>
<tr><td></td><td> 
</td><td></td></tr>
<tr><td>11</td><td>Delete a patient ID 7
 
</td><td>Pass</td></tr>
<tr><td colspan="3">Patient</td></tr>
<tr><td>12</td><td>Login as patient and display booking overview page.</td><td>Pass</td></tr>
</table>

		
13	Null input validation. 	Pass
14	Add a new booking.	Pass

		
15	Cancel a booking. 	Pass

Doctor		
16	Login as doctor and display booking list page. 	Pass
17	Null input validation. 	Pass
18	Complete a booking attendance.	Pass


Booking No: 3 completed.
Booking No
Patient Name
Booking Date
Booking Time
Status
1
2024-11-18
12:00
Completed
2
Patient 4
2024-11-22
09:00
Completed
3
Patient 4
2024-12-27
08:00
Completed

Assertion

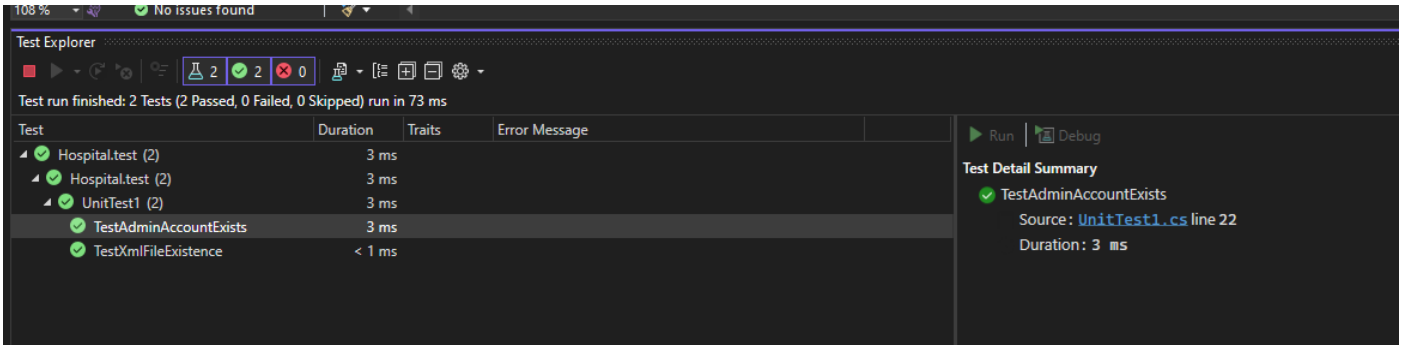
Test Admin Account Exits

Test Description: to check if an Admin account exist in the Users.xml file.

-Load the User.xml file from the specified directory.

-Search and verify user element where the AccountType is set 0 for (admin).

Test Result



Source Code:

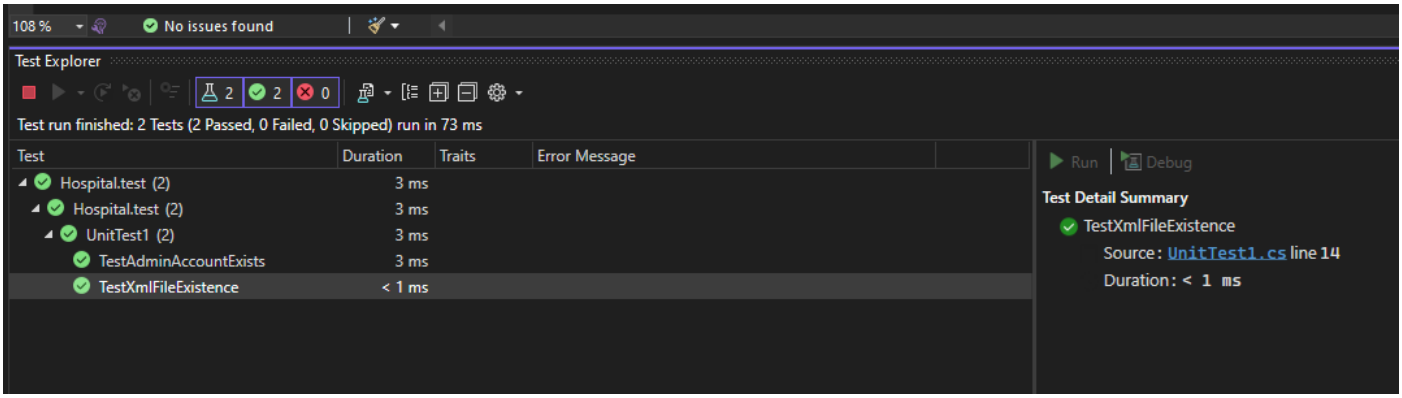
```
[TestMethod]
 public void TestAdminAccountExists()
{
   // Ensure the file exists
   Assert.IsTrue(File.Exists(xmlFilePath), "Users.xml file does not exist.");

   // Read the XML file
   XDocument xmlDoc = XDocument.Load(xmlFilePath);

   // Check if any user has an AccountType of "Admin"
   var adminAccount = xmlDoc.Descendants("User")
                .FirstOrDefault(user => user.Element("AccountType")?.Value == "0");

   // Assert that an admin account is found
   Assert.IsNotNull(adminAccount, "Admin account not found in the Users.xml file.");
 }
```

Test Xml File Exist

Test Description: check whether the User.xml file exits in the specific directory.

- To ensure the file is present , confirming that the application can read the XML data.

Test Result



Source Code:

```
[TestMethod]
public void TestXmlFileExistence()
{
   // Assert that the file exists
   Assert.IsTrue(File.Exists(xmlFilePath), $"Users.xml file does not exist at path:
{xmlFilePath}");
}
```

Source Code

Model

Account.cs	using System.ComponentModel.DataAnnotations;
	
	namespace HospitalManagementSystem_Assignment2.Models
	{
	  public class Account
	  {
	  public int ID { get; set; }
	
	  [Required(ErrorMessage = "First Name is required.")]
	  public string FirstName { get; set; }
	
	  [Required(ErrorMessage = "Last Name is required.")]
	  public string LastName { get; set; }
	
	  [Required(ErrorMessage = "Email is required.")]
	  [EmailAddress(ErrorMessage = "Invalid email format.")]
	  public string Email { get; set; }
	
	  [Required(ErrorMessage = "Password is required.")]
	  [DataType(DataType.Password)]
	  public string Password { get; set; }
	
	  [Compare("Password", ErrorMessage = "Passwords do not match.")]
	  [Display(Name = "Confirm Password")]

	[DataType(DataType.Password)] public string ConfirmPassword { get; set; } [Required(ErrorMessage = "DOB is required.")] public string DOB { get; set; } [Required(ErrorMessage = "Phone is required.")] [RegularExpression(@"^[0-9]{8,}$", ErrorMessage = "Invalid phone number.")] public int Phone { get; set; } [Required(ErrorMessage = "Gender is required.")] public string Gender { get; set; } public int AccountType { get; set; } // 0: Admin | 1: Doctor | 2: Patient public string? Specialist { get; set; } public string? MedicalHistory { get; set; } public string? BloodGroup { get; set; } public string? DrugAllergy { get; set; } } }

Admin.cs	namespace HospitalManagementSystem_Assignment2.Models
	{
	public class Admin : User
	{
	public int ID { get; set; }
	
	}
	}

AdminList.cs	namespace HospitalManagementSystem_Assignment2.Models
	{
	public class AdminList
	{
	public int ID { get; set; }
	public string FirstName { get; set; }
	public string LastName { get; set; }
	public string Email { get; set; }
	public string Password { get; set; }
	public string DOB { get; set; }
	public int Phone { get; set; }
	public string Gender { get; set; }
	public int AccountType { get; set; } // 0: Admin | 1: Doctor | 2: Patient
	
	}
	}

Attendance.cs	namespace HospitalManagementSystem_Assignment2.Models { public class Attendance {

	public int BookingNo { get; set; } public string DiagnosisInfo { get; set; } public string? Remark { get; set; } public string? Therapy { get; set; } } }

Booking.cs	namespace HospitalManagementSystem_Assignment2.Models
	{
	public class Booking
	{
	public int BookingNo { get; set; }
	public int EmployeeID { get; set; }
	public string DoctorName { get; set; }
	public int PatientID { get; set; }
	public string PatientName { get; set; }
	public string BookingDate { get; set; }
	public string BookingTime { get; set; }
	public int Status { get; set; } // 0: Created | 1: Completed
	public List<Booking> bookingList { get; set; }
	}
	}

Doctor.cs	namespace HospitalManagementSystem_Assignment2.Models
	{
	public class Doctor : User
	{
	public int EmployeeID { get; set; }
	public string Specialist { get; set; }
	}
	}

DoctorList.cs	namespace HospitalManagementSystem_Assignment2.Models
	{
	public class DoctorList
	{
	public int EmployeeID { get; set; }
	public string FirstName { get; set; }
	public string LastName { get; set; }
	public string FullName { get; set; }
	public string Email { get; set; }
	public string Password { get; set; }
	public string DOB { get; set; }
	public int Phone { get; set; }
	public string Gender { get; set; }
	public int AccountType { get; set; } // 0: Admin | 1: Doctor | 2: Patient
	public string Specialist { get; set; }
	}
	}

Patient.cs	namespace HospitalManagementSystem_Assignment2.Models {

	public class Patient : User
	{
	public int PatientID { get; set; }
	public string BloodGroup { get; set;}
	public string MedicalHistory { get; set;}
	public string DrugAllergy { get; set; }
	}
	}

PatientList.cs	namespace HospitalManagementSystem_Assignment2.Models
	{
	public class PatientList
	{
	public int ID { get; set; }
	public string FirstName { get; set; }
	public string LastName { get; set; }
	public string Email { get; set; }
	public string Password { get; set; }
	public string DOB { get; set; }
	public int Phone { get; set; }
	public string Gender { get; set; }
	public int AccountType { get; set; }
	// 0: Admin | 1: Doctor | 2: Patient
	public string BloodGroup { get; set; }
	public string MedicalHistory { get; set; }
	public string DrugAllergy { get; set; }
	}
	}

User.cs	namespace HospitalManagementSystem_Assignment2.Models
	{
	public class User
	{
	public int ID { get; set; }
	public string FirstName { get; set; }
	public string LastName { get; set; }
	public string Email { get; set; }
	public string Password { get; set; }
	public string DOB { get; set; }
	public int Phone { get; set; }
	public string Gender { get; set; }
	public int AccountType { get; set; } // 0: Admin | 1: Doctor | 2: Patient
	}
	}

UserList.cs	namespace HospitalManagementSystem_Assignment2.Models { public class UserList { public List<DoctorList> doctorList { get; set; } public List<PatientList> patientList { get; set; } public List<AdminList> adminList { get; set; }

	} }

Views

Admin	

AccountManagement.cshtml	@model UserList
	
	@{
	ViewBag.Title = "Account Management";
	}
	
	Create Account
	
	<h1 class="display-2">@ViewData["Title"]</h1>
	
	@{
	@if (TempData["DoctorMessage"] != null)
	{
	
	@TempData["DoctorMessage"]
	
	}
	@if (TempData["SuccessMessage"] != null)
	{
	
	@TempData["SuccessMessage"]
	
	}
	
	<h3 class="display-6">Doctor List</h3>
	<table class="table">
	<tr>
	<th >ID</th>
	<th >First Name</th>
	<th >Last Name</th>
	<th >Email</th>
	<th >Phone</th>
	<th >Password</th>
	<th >Gender</th>
	<th >DOB</th>
	<th >Account Type</th>
	<th >Specialist</th>
	<th >Action</th>
	
	</tr>

```
    @if (Model.doctorList != null)
    {
      @foreach (var doctor in Model.doctorList)
      {
        <tr>
          <td>
            @Html.DisplayFor(m => doctor.EmployeeID)
          </td>
          <td>
            @Html.DisplayFor(m => doctor.FirstName)
          </td>
          <td>
            @Html.DisplayFor(m => doctor.LastName)
          </td>
          <td>
            @Html.DisplayFor(m => doctor.Email)
          </td>
          <td>
            @Html.DisplayFor(m => doctor.Phone)
          </td>
          <td>
            @Html.DisplayFor(m => doctor.Password)
          </td>
          <td>
            @Html.DisplayFor(m => doctor.Gender)
          </td>
          <td>
            @Html.DisplayFor(m => doctor.DOB)
          </td>
          <td>
            <p>Doctor</p>
          </td>
          <td>
            @Html.DisplayFor(m => doctor.Specialist)
          </td>
          <td>
            <a href="@Url.Action("EditDoctor", "Admin", new { id =
doctor.EmployeeID })" class="btn btn-secondary">Edit</a>
          </td>
          <td>
          @using (Html.BeginForm("DeleteDoctor", "Admin", new
{ id = doctor.EmployeeID }, FormMethod.Post))
          {
            <input type="submit" value="Delete" class="btn btn-
danger" onclick="return confirm('Are you sure you want to delete
this doctor?');" />
          }
          </td>
        </tr>
      }
    }
```

```
</table>

@if (TempData["PatientMessage"] != null)
{
  <div class="alert alert-success">
    @TempData["PatientMessage"]
  </div>
}

<h3 class="display-6">Patient List</h3>
<table class="table">
  <tr>
    <th >ID</th>
    <th >First Name</th>
    <th >Last Name</th>
    <th >Email</th>
    <th >Phone</th>
    <th >Password</th>
    <th >Gender</th>
    <th >DOB</th>
    <th >Account Type</th>
    <th >Blood Group</th>
    <th >Medical History</th>
    <th >Drug Allergy</th>
    <th >Action</th>

  </tr>

  @if (Model.patientList != null)
  {
    @foreach (var patient in Model.patientList)
    {
      <tr>
        <td>
          @Html.DisplayFor(m => patient.ID)
        </td>
        <td>
          @Html.DisplayFor(m => patient.FirstName)
        </td>
        <td>
          @Html.DisplayFor(m => patient.LastName)
        </td>
        <td>
          @Html.DisplayFor(m => patient.Email)
        </td>
        <td>
          @Html.DisplayFor(m => patient.Phone)
        </td>
        <td>
          @Html.DisplayFor(m => patient.Password)
        </td>
        <td>
```

```
                @Html.DisplayFor(m => patient.Gender)
              </td>
              <td>
                @Html.DisplayFor(m => patient.DOB)
              </td>
              <td>
                <p>Patient</p>
              </td>
              <td>
                @Html.DisplayFor(m => patient.BloodGroup)
              </td>
              <td>
                @Html.DisplayFor(m => patient.MedicalHistory)
              </td>
              <td>
                @Html.DisplayFor(m => patient.DrugAllergy)
              </td>
              <td>
                <a href="@Url.Action("EditPatient", "Admin", new { id =
patient.ID })" class="btn btn-secondary">Edit</a>
              </td>
              <td>
                @using (Html.BeginForm("DeletePatient", "Admin", new
{ id = patient.ID }, FormMethod.Post))
                {
                  <input type="submit" value="Delete" class="btn btn-
danger" onclick="return confirm('Are you sure you want to delete
this patient?');" />
                }
              </td>
            </tr>
          }
        }
      </table>

      <h3 class="display-6">Admin List</h3>
      <table class="table">
        <tr>
          <th>ID</th>
          <th>First Name</th>
          <th>Last Name</th>
          <th>Email</th>
          <th>Phone</th>
          <th>Password</th>
          <th>Gender</th>
          <th>DOB</th>
          <th>Account Type</th>

        </tr>

        @if (Model.adminList != null)
        {
```

	@foreach (var admin in Model.adminList)
	{
	<tr>
	<td>
	@Html.DisplayFor(m => admin.ID)
	</td>
	<td>
	@Html.DisplayFor(m => admin.FirstName)
	</td>
	<td>
	@Html.DisplayFor(m => admin.LastName)
	</td>
	<td>
	@Html.DisplayFor(m => admin.Email)
	</td>
	<td>
	@Html.DisplayFor(m => admin.Phone)
	</td>
	<td>
	@Html.DisplayFor(m => admin.Password)
	</td>
	<td>
	@Html.DisplayFor(m => admin.Gender)
	</td>
	<td>
	@Html.DisplayFor(m => admin.DOB)
	</td>
	<td>
	 Admin
	</td>
	
	</tr>
	}
	}
	</table>
	
	}

CreateAccount.cshtml	@model
	HospitalManagementSystem_Assignment2.Models.Account
	
	@{
	ViewData["Title"] = "Create User Account";
	}
	
	<h1 class="display-4">@ViewData["Title"]</h1>
	
	@if (TempData["ErrorMessage"] != null)
	{

```
   <div class="alert alert-
danger">@TempData["ErrorMessage"]</div>
}

@using (Html.BeginForm("CreateAccount", "Admin",
FormMethod.Post))
{
   <div class="form-group">
      @Html.LabelFor(m => m.FirstName, "First Name")
      @Html.TextBoxFor(m => m.FirstName, new { @class = "form-
control" })
      @Html.ValidationMessageFor(m => m.FirstName, null, new
{ @class = "text-danger" })
   </div>
   <div class="form-group">
      @Html.LabelFor(m => m.LastName, "Last Name")
      @Html.TextBoxFor(m => m.LastName, new { @class = "form-
control" })
      @Html.ValidationMessageFor(m => m.LastName, null, new
{ @class = "text-danger" })
   </div>
   <div class="form-group">
      @Html.LabelFor(m => m.Email)
      @Html.TextBoxFor(m => m.Email, new { @class = "form-
control" })
      @Html.ValidationMessageFor(m => m.Email, null, new { @class
= "text-danger" })
   </div>
   <div class="form-group">
      @Html.LabelFor(m => m.Phone, "Phone Number")
      @Html.TextBoxFor(m => m.Phone, new { @class = "form-
control" })
      @Html.ValidationMessageFor(m => m.Phone, null, new { @class
= "text-danger" })
   </div>
   <div class="form-group">
      @Html.LabelFor(m => m.Password)
      @Html.PasswordFor(m => m.Password, new { @class = "form-
control" })
      @Html.ValidationMessageFor(m => m.Password, null, new
{ @class = "text-danger" })
   </div>
   <div class="form-group">
      @Html.LabelFor(m => m.ConfirmPassword, "Confirm
Password")
      @Html.PasswordFor(m => m.ConfirmPassword, new { @class =
"form-control" })
      @Html.ValidationMessageFor(m => m.ConfirmPassword, null,
new { @class = "text-danger" })
   </div>
   <div class="form-group">
      @Html.Label("Gender")
```

```
      <div>
        @*  @Html.RadioButtonFor(m => m.Gender, "Male") Male
         @Html.RadioButtonFor(m => m.Gender, "Female") Female *@
         @Html.RadioButtonFor(m => m.Gender, "Male", new { id =
"Gender_Male" }) Male
         @Html.RadioButtonFor(m => m.Gender, "Female", new { id =
"Gender_Female" }) Female
      </div>
      @Html.ValidationMessageFor(m => m.Gender, null, new
{ @class = "text-danger" })
    </div>
    <div class="form-group">
      @Html.LabelFor(m => m.DOB, "Date of Birth")
      @Html.TextBoxFor(m => m.DOB, new { @class = "form-control",
@type = "date" })
      @Html.ValidationMessageFor(m => m.DOB, null, new { @class =
"text-danger" })
    </div>
    <div class="form-group">
      @Html.Label("Account Role")
      <div>
        @Html.RadioButtonFor(m => m.AccountType, 0) Admin
        @Html.RadioButtonFor(m => m.AccountType, 1) Doctor
        @Html.RadioButtonFor(m => m.AccountType, 2) Patient
      </div>
    </div>


    //hide and show for the additional field
    <div id="doctorSpec" class="form-group">
      @Html.LabelFor(m => m.Specialist)
      @Html.TextBoxFor(m => m.Specialist, new { @class = "form-
control" })
    </div>
    <br />
    <div id="patientBG" class="form-group">
      @Html.Label("Blood Group")
      <br />
     @*   @Html.RadioButtonFor(m => m.BloodGroup, "A") A
      @Html.RadioButtonFor(m => m.BloodGroup, "B") B
      @Html.RadioButtonFor(m => m.BloodGroup, "AB") AB
      @Html.RadioButtonFor(m => m.BloodGroup, "O") O *@
      @Html.RadioButtonFor(m => m.BloodGroup, "A", new { id =
"BloodGroup_A" }) A
      @Html.RadioButtonFor(m => m.BloodGroup, "B", new { id =
"BloodGroup_B" }) B
      @Html.RadioButtonFor(m => m.BloodGroup, "AB", new { id =
"BloodGroup_AB" }) AB
      @Html.RadioButtonFor(m => m.BloodGroup, "O", new { id =
"BloodGroup_O" }) O
    </div>
    <br />
```

```
    <div id="patientMH" class="form-group">
        @Html.Label("Medical History")
        @Html.TextAreaFor(m => m.MedicalHistory, new { @class =
"form-control", rows = "4" })
    </div>
    <br />
    <div id="patientDA" class="form-group">
        @Html.Label("Drug Allergy")
        @Html.TextAreaFor(m => m.DrugAllergy, new { @class = "form-
control", rows = "4" })
    </div>
    <br />
    <div class="form-group">
        <input type="submit" value="Create" class="btn btn-success" />
        <input type="reset" value="Clear" id="clear" class="btn btn-
danger" />
    </div>
    <br />
    <div class="form-group">
        <a href="@Url.Action("AccountManagement", "Admin")"
class="btn btn-secondary">Go back </a>

    </div>
    <br />
}
@section Scripts {
    <script>
        $(document).ready(function () {
            // Hide all optional fields on page load
            $("#doctorSpec, #patientBG, #patientMH, #patientDA").hide();

            // Show or hide fields based on the selected account type
            $("input[name='AccountType']").change(function () {
                if (this.value == 1) { // Doctor
                    $("#doctorSpec").show(); // Show specialization field
                    $("#patientBG, #patientMH, #patientDA").hide(); // Hide
patient fields
                } else if (this.value == 2) { // Patient
                    $("#doctorSpec").hide(); // Hide specialization field
                    $("#patientBG, #patientMH, #patientDA").show(); // Show
patient fields
                } else { // Admin or others
                    $("#doctorSpec, #patientBG, #patientMH,
#patientDA").hide(); // Hide all optional fields
                }
            });

            // Reset button to clear the form and hide all optional fields
            $("#clear").click(function () {
                $("#doctorSpec, #patientBG, #patientMH,
#patientDA").reset();
            });
```

	}); </script> }

EditDoctor.cshtml	@model
	HospitalManagementSystem_Assignment2.Models.Account
	
	@{
	ViewData["Title"] = "Edit Doctor Account";
	}
	
	<h1 class="display-2">@ViewData["Title"]</h1>
	
	<!-- Alert prompt box-->
	<div id="comingSoonModal" style="display:none; position:fixed; top:50%; left:50%; transform:translate(-50%, -50%); background:white; padding:20px; border:1px solid #333; box-shadow:0px 4px 8px rgba(0,0,0,0.2); z-index:1000; text-align:center;">
	<h3>Feature Unavailable</h3>
	 The edit function is currently unavailable. This feature is coming soon!
	<button onclick="closeModal()">Go Back</button>
	
	<!-- JavaScript for display -->
	<script type="text/javascript">
	function showComingSoonModal(event) {
	event.preventDefault(); // Prevent form submission
	document.getElementById('comingSoonModal').style.display = 'block';
	document.getElementById('overlay').style.display = 'block';
	}
	
	function closeModal() {
	document.getElementById('comingSoonModal').style.display = 'none';
	document.getElementById('overlay').style.display = 'none';
	window.history.back(); // Redirect to the previous page
	}
	</script>
	
	@using (Html.BeginForm("UpdateDoctor", "Admin", FormMethod.Post , new { onsubmit = "showComingSoonModal(event)" }))

```
{
   @Html.HiddenFor(m => m.ID)

   <div class="form-group">
      @Html.Label("First Name")
      @Html.TextBoxFor(m => m.FirstName, new { @class = "form-
control" })
      @Html.ValidationMessageFor(m => m.FirstName, "", new
{ @class = "text-danger" })
   </div>
   <br />
   <div class="form-group">
      @Html.Label("Last Name")
      @Html.TextBoxFor(m => m.LastName, new { @class = "form-
control" })
      @Html.ValidationMessageFor(m => m.LastName, "", new
{ @class = "text-danger" })
   </div>
   <br />
   <div class="form-group">
      @Html.Label("Email")
      @Html.TextBoxFor(m => m.Email, new { @class = "form-
control" })
      @Html.ValidationMessageFor(m => m.Email, "", new { @class =
"text-danger", @disabled = "disabled" })
   </div>
   <br />
   <div class="form-group">
      @Html.Label("Phone")
      @Html.TextBoxFor(m => m.Phone, new { @class = "form-
control" })
      @Html.ValidationMessageFor(m => m.Phone, "", new { @class =
"text-danger" })
   </div>
   <br />
   <div class="form-group">
      @Html.LabelFor(m => m.Password)
      @Html.PasswordFor(m => m.Password, new { @class = "form-
control"})
      @Html.ValidationMessageFor(m => m.Password, "", new
{ @class = "text-danger" })
   </div>
   <br />
   <div class="form-group">
      @Html.LabelFor(m => m.ConfirmPassword)
      @Html.PasswordFor(m => m.ConfirmPassword, new { @class =
"form-control"})
      @Html.ValidationMessageFor(m => m.ConfirmPassword, "", new
{ @class = "text-danger" })
   </div>
   <br />
   <div class="form-group">
```

	@Html.Label("Gender") @Html.RadioButtonFor(m => m.Gender, "Male") Male @Html.RadioButtonFor(m => m.Gender, "Female") Female @Html.ValidationMessageFor(m => m.Gender, "", new { @class = "text-danger" }) @Html.Label("Date of Birth") @Html.TextBoxFor(m => m.DOB, new { @class = "form-control", placeholder = "DD/MM/YYYY", @type = "date" }) @Html.ValidationMessageFor(m => m.DOB, "", new { @class = "text-danger" }) @Html.LabelFor(m => m.Specialist) @Html.TextBoxFor(m => m.Specialist, new { @class = "form-control" }) <input type="submit" value="Update" class="btn btn-primary" /> } Go back

EditPatient.cshtml	@model HospitalManagementSystem_Assignment2.Models.Account
	
	@{
	ViewData["Title"] = "Edit Patient Account";
	}
	
	<h1 class="display-2">@ViewData["Title"]</h1>
	
	<!-- Alert prompt box-->
	<div id="comingSoonModal" style="display:none; position:fixed; top:50%; left:50%; transform:translate(-50%, -50%); background:white; padding:20px; border:1px solid #333; box-

```
shadow:0px 4px 8px rgba(0,0,0,0.2); z-index:1000; text-
align:center;">
  <h3>Feature Unavailable</h3>
  <p>The edit function is currently unavailable. This feature is
coming soon!</p>
  <button onclick="closeModal()">Go Back</button>
</div>
<div id="overlay" style="display:none; position:fixed; top:0; left:0;
width:100%; height:100%; background:rgba(0,0,0,0.5); z-
index:999;"></div>

<!-- JavaScript for display -->
<script type="text/javascript">
  function showComingSoonModal(event) {
    event.preventDefault(); // Prevent form submission
    document.getElementById('comingSoonModal').style.display =
'block';
    document.getElementById('overlay').style.display = 'block';
  }

  function closeModal() {
    document.getElementById('comingSoonModal').style.display =
'none';
    document.getElementById('overlay').style.display = 'none';
    window.history.back(); // Redirect to the previous page
  }
</script>

<!-- Form with onsubmit event to trigger l -->
@using (Html.BeginForm("UpdatePatient", "Admin",
FormMethod.Post, new { onsubmit =
"showComingSoonModal(event)" }))
{
  @Html.HiddenFor(m => m.ID)

  <div class="form-group">
    @Html.Label("First Name")
    @Html.TextBoxFor(m => m.FirstName, new { @class = "form-
control" })
  </div>
  <br />
  <div class="form-group">
    @Html.Label("Last Name")
    @Html.TextBoxFor(m => m.LastName, new { @class = "form-
control" })
    @Html.ValidationMessageFor(m => m.LastName, "", new
{ @class = "text-danger" })
  </div>
  <br />
  <div class="form-group">
    @Html.Label("Email")
```

```
    @Html.TextBoxFor(m => m.Email, new { @class = "form-
control" })
    @Html.ValidationMessageFor(m => m.Email, "", new { @class =
"text-danger", @disabled = "disabled" })
  </div>
  <br />
  <div class="form-group">
    @Html.Label("Phone")
    @Html.TextBoxFor(m => m.Phone, new { @class = "form-
control" })
    @Html.ValidationMessageFor(m => m.Phone, "", new { @class =
"text-danger" })
  </div>
  <br />
  <div class="form-group">
    @Html.LabelFor(m => m.Password)
    @Html.PasswordFor(m => m.Password, new { @class = "form-
control" })
    @Html.ValidationMessageFor(m => m.Password, "", new
{ @class = "text-danger" })
  </div>
  <br />
  <div class="form-group">
    @Html.LabelFor(m => m.ConfirmPassword)
    @Html.PasswordFor(m => m.ConfirmPassword, new { @class =
"form-control" })
    @Html.ValidationMessageFor(m => m.ConfirmPassword, "", new
{ @class = "text-danger" })
  </div>
  <br />
  <div class="form-group">
    @Html.Label("Gender")
    <br />
    @Html.RadioButtonFor(m => m.Gender, "Male") Male
    @Html.RadioButtonFor(m => m.Gender, "Female") Female
    @Html.ValidationMessageFor(m => m.Gender, "", new { @class =
"text-danger" })
  </div>
  <br />
  <div class="form-group">
    @Html.Label("Date of Birth")
    @Html.TextBoxFor(m => m.DOB, new { @class = "form-control",
placeholder = "DD/MM/YYYY", @type = "date" })
    @Html.ValidationMessageFor(m => m.DOB, "", new { @class =
"text-danger" })
  </div>
  <br />
  <div class="form-group">
    @Html.Label("Blood Group")
    <br />
    @Html.RadioButtonFor(m => m.BloodGroup, "AB") AB
    @Html.RadioButtonFor(m => m.BloodGroup, "A") A
```

	@Html.RadioButtonFor(m => m.BloodGroup, "B") B @Html.RadioButtonFor(m => m.BloodGroup, "O") O @Html.Label("Medical History") @Html.TextAreaFor(m => m.MedicalHistory, new { @class = "form-control", rows = "4" }) @Html.Label("Drug Allergy") @Html.TextAreaFor(m => m.DrugAllergy, new { @class = "form-control", rows = "4" }) <input type="submit" value="Update" class="btn btn-primary" /> Go back }
Doctor	

BokingList.cshtml	@model List<Booking>
	
	@{
	ViewBag.Title = "Booking List";
	}
	
	<h1 class="display-2">@ViewData["Title"]</h1>
	
	@{
	@if (TempData["BookingMessage"] != null)
	{
	
	@TempData["BookingMessage"]
	
	}
	
	@if (TempData["DetailMessage"] != null)
	{

```
      @TempData["DetailMessage"]
    </div>
  }
  @if (TempData["RemarkMessage"] != null)
  {
    <div class="alert alert-info">
      @TempData["RemarkMessage"]
    </div>
  }
  @if (TempData["TherapyMessage"] != null)
  {
    <div class="alert alert-info">
      @TempData["TherapyMessage"]
    </div>
  }

  <table class="table">
    <tr>
      <th scope="col">Booking No</th>
      <th scope="col">Patient Name</th>
      <th scope="col">Booking Date</th>
      <th scope="col">Booking Time</th>
      <th scope="col">Status</th>
      <th scope="col">Action</th>
    </tr>

    @if (Model != null)
    {
      @foreach (var booking in Model)
      {
        <tr>
          <td>
            @Html.DisplayFor(m => booking.BookingNo)
          </td>
          <td>
            @Html.DisplayFor(m => booking.PatientName)
          </td>
          <td>
            @Html.DisplayFor(m => booking.BookingDate)
          </td>
          <td>
            @Html.DisplayFor(m => booking.BookingTime)
          </td>
          <td>
            @if (booking.Status == 0)
            {
              <p>Pending to Diagnosis</p>
            }
            else
            {
              <p>Completed</p>
            }
```

	</td>
	<td>
	@if (booking.Status == 0)
	{
	Complete
	}
	else
	{
	@using (Html.BeginForm("ShowAttendance", "Doctor", new { id = booking.BookingNo }, FormMethod.Post))
	{
	<input type="submit" value="Detail" class="btn btn-info" />
	}
	}
	</td>
	</tr>
	}
	}
	</table>
	}

CompleteAttendance.cshtml	@model HospitalManagementSystem_Assignment2.Models.Attendance
	
	@{
	ViewData["Title"] = "Complete Attendance";
	}
	
	<h1 class="display-2">@ViewData["Title"]</h1>
	
	@if (TempData["CompleteErrorMessage"] != null)
	{
	
	@TempData["CompleteErrorMessage"]
	
	}
	
	@using (Html.BeginForm("CompleteAttendance", "Doctor", FormMethod.Post))
	{
	
	@Html.Label("Diagnosis Info")
	@Html.TextAreaFor(m => m.DiagnosisInfo, new { @class = "form-control", rows = "4", required = "required" })

	@Html.Label("Remark") @Html.TextAreaFor(m => m.Remark, new { @class = "form-control", rows = "4" }) @Html.Label("Therapy") @Html.TextAreaFor(m => m.Therapy, new { @class = "form-control", rows = "4" }) <input type="submit" value="Complete" class="btn btn-success" /> <input type="reset" value="Clear" class="btn btn-secondary" /> Go back }
Home	

About.cshtml	@{
	ViewData["Title"] = "About Us";
	}
	
	<h1 class="display-2">@ViewData["Title"]</h1>
	
	<!-- Emergency Care Section -->
	
	<**img** src="~/images/hp.png" alt="About us" class="about-img" width="400px" height="300px">
	
	 We are one of Western Australia’s leading private health campuses, providing excellence in health care for our community..
	
	St John of God Murdoch Hospital is a private hospital and a division of St John of God Health Care, one of Australia's largest health care providers.

	We are one of Western Australia's leading private health campuses, providing compassionate and high quality health care for our community. Stage 2 of the hospital's redevelopment project is now complete - the last component being the renovation of the original six 1994 wards. Established in 1994, St John of God Murdoch Hospital provides: - a 24 hour Emergency Department - medical and surgical services - paediatrics - maternity - palliative care - critical and coronary care. There has been significant growth in research at the Hospital, in the areas of clinical oncology trials, orthopaedics, emergency medicine, intensive care, cardiology and nursing-led research. Many studies are delivering impressive results, with the focus on patient-centred management and innovative approaches to evaluating patient outcomes.

Contact.cshtml	@{
	ViewData["Title"] = "Contact Us";
	}
	
	<h1 class="display-2">@ViewData["Title"]</h1>
	
	<**img** src="~/images/hp2.png" alt="Hospital Image" class="img-fluid" />
	 We value your feedback and inquiries. If you have any questions, concerns, or suggestions, please feel free to reach out to us using the information below.
	
	<table class="table table-bordered">

```
<tr>
  <th>Information</th>
  <th>Details</th>
</tr>
<tr>
  <td><strong>Hospital Name:</strong></td>
  <td>St John of God Murdoch Hospital</td>
</tr>
<tr>
  <td><strong>Head Office Address:</strong></td>
  <td>St John of God Health Care
    Level 1, 556 Wellington Street
    Perth WA 6000
  </td>
</tr>
<tr>
  <td><strong>Phone:</strong></td>
  <td> 08 6116 0000</td>
</tr>
<tr>
  <td><strong>Melbourne Office Address:</strong></td>
  <td>
    St John of God Health Care
    Level 4, 360 Collins Street
    Melbourne VIC 3000
  </td>
</tr>
<tr>
  <td><strong>Phone:</strong></td>
  <td>03 9205 6500</td>
</tr>
<tr>
  <td><strong>Email:</strong></td>
  <td><a
href="mailto:stJohn@email.com">stJohn@email.com</a></td>
</tr>
<tr>
  <td><strong>Website:</strong></td>
  <td><a href="http://sghp.com">stJohn.com</a></td>
</tr>
<tr>
  <td><strong>Office Hours:</strong></td>
  <td>
    Monday to Friday: 8:00 AM - 8:00 PM<br>
    Saturday: 9:00 AM - 5:00 PM<br>
    Sunday: Closed
  </td>
</tr>
<tr>
  <td><strong>Emergency Contact:</strong></td>
  <td>+61 1988 8888</td>
</tr>
```

	</table> We are here to help you and provide the care you need.

Index.cshtml	@{
	ViewData["Title"] = "Home Page";
	}
	
	<h1 class="display-4">Welcome</h1>
	
	<**img** src="~/images/welcome.png" alt="welcome Care" width="300px" height="250px">
	
	 We are committed to delivering exceptional healthcare services tailored to meet the needs of our community. Our hospital management has been at the forefront of medical excellence, combining innovation and compassion in patient care.
	
	 We offer a wide range of services, from emergency care to specialized treatments, all backed by state-of-the-art technology and facilities.

Privacy.cshtml	@{
	ViewData["Title"] = "Privacy Policy";
	}
	 @ViewData["Title"]
	
	<!-- Privacy Section -->
	
	<**img** src="~/images/privacy.png" alt="About us" class="about-img" width="500px" height="300px">

	St John of God Health Care is committed to upholding the dignity of each person. Guided by the value of respect, we will manage all personal information in accordance with privacy legislation. This Privacy Policy applies to St John of God Health Care Inc and any related body corporate, collectively referred to as "SJGHC". SJGHC is committed to upholding the dignity of each person. Guided by the value of respect, we will manage all personal information in accordance with the Australian Privacy Principles (APPs) and the Notifiable Data Breaches Scheme (NDB Scheme) under the Privacy Act 1988 (Cth) (Privacy Act) and all other relevant Commonwealth and State legislation (together "Privacy Legislation"). This SJGHC Privacy Policy details how your personal information is collected, used, stored and disclosed by SJGHC and how you may contact us if you would like to access or correct your personal information. It also details how we respond if there is an Eligible Data Breach. In this Privacy Policy: “we”, “us” or “our” refers to SJGHC; and "you" and "your" refers to any person whose personal information we collect, except employee records. You consent to us collecting, holding, using and disclosing your personal information in accordance with this Privacy Policy.

Service.cshtml	@{
	ViewData["Title"] = "Our Service";
	}
	
	<h1 class="display-2">@ViewData["Title"]</h1>
	
	 Providing compassionate and high-quality healthcare to the community

```
    <!-- Emergency Care Section -->
    <div class="service-section">
      <div class="row">
        <div class="col-md-6">
        <img src="~/images/emergency.jpg" alt="Emergency Care"
class="service-img" width="400px" height="200px">
        </div>
        <div class="col-md-6">
          <h2 class="service-title">Emergency Care</h2>
          <p class="service-description">
            Our 24-hour Emergency Department is equipped to
handle a wide range of urgent medical conditions. With highly
trained doctors, nurses, and support staff, we ensure prompt and
efficient care in emergencies.
          </p>
        </div>
      </div>
    </div>

    <!-- Maternity Services Section -->
    <div class="service-section">
      <div class="row">
        <div class="col-md-6">
        <img src="~/images/Maternity.png" alt="Maternity Services"
class="service-img" width="400px" height="200px">
        </div>
        <div class="col-md-6">
          <h2 class="service-title">Maternity Services</h2>
          <p class="service-description">
            St. John of God Murdoch Hospital provides exceptional
maternity care, including antenatal and postnatal support, labor
and delivery services, and specialized care for both mothers and
babies. Our maternity team ensures that every new parent
experiences a smooth and safe journey.
          </p>
        </div>
      </div>
    </div>

    <!-- Cancer Care Section -->
    <div class="service-section">
      <div class="row">
        <div class="col-md-6">
        <img src="~/images/cancer.jpg" alt="Cancer Care"
class="service-img" width="400px" height="200px">
        </div>
        <div class="col-md-6">
          <h2 class="service-title">Cancer Care</h2>
          <p class="service-description">
            We offer comprehensive cancer care services, including
diagnosis, treatment, and ongoing support. Our dedicated oncology
```

```
team works closely with patients to provide personalized care that
meets their individual needs and goals.
            </p>
          </div>
        </div>
      </div>

      <!-- Orthopaedics Section -->
      <div class="service-section">
        <div class="row">
          <div class="col-md-6">
          <img src="~/images/orthopaedics.jpeg" alt="Orthopaedics"
class="service-img" width="400px" height="200px">
          </div>
          <div class="col-md-6">
            <h2 class="service-title">Orthopaedics</h2>
            <p class="service-description">
              Our orthopaedic specialists provide expert care for
musculoskeletal conditions, including joint replacement surgery,
spinal surgery, and treatment for sports injuries. We utilize the latest
technology and techniques to deliver the best outcomes for our
patients.
            </p>
          </div>
        </div>
      </div>

      <!-- Palliative Care Section -->
      <div class="service-section">
        <div class="row">
          <div class="col-md-6">
          <img src="~/images/palliative.jpeg" alt="Palliative Care"
class="service-img" width="400px" height="200px">
          </div>
          <div class="col-md-6">
            <h2 class="service-title">Palliative Care</h2>
            <p class="service-description">
              St. John of God Murdoch Hospital provides
compassionate palliative care to support patients with serious or
terminal illnesses. Our team works with patients and families to
ensure comfort, dignity, and quality of life during challenging times.
            </p>
          </div>
        </div>
      </div>
      <br/>
      <!-- Contact Section -->
      <div class="service-section text-center">
      <a href="@Url.Action("Contact", "Home")" class="btn btn-
primary btn-lg">Contact Us for More Information</a>

      </div>
```

Login	

Login.cshtml	@model HospitalManagementSystem_Assignment2.Models.User
	
	@{
	ViewData["Title"] = "Login";
	}
	
	<h1 class="display-2">@ViewData["Title"]</h1>
	
	<**form asp-action**="Login" method="post" onsubmit="return validateForm()">
	
	<label for="email">Email:</label>
	<input type="email" class="form-control" id="email" name="Email">
	
	<label for="password">Password:</label>
	<input type="password" class="form-control" id="password" name="Password">
	
	<button type="submit" class="btn btn-primary">Login</button>
	</**form**>
	
	@if (TempData["SuccessMessage"] != null)
	{
	
	@TempData["SuccessMessage"]
	
	}
	
	Please enter both Email and Password.
	
	@if (TempData["ErrorMessage"] != null)
	{

	@TempData["ErrorMessage"]
	
	}
	
	<script>
	function validateForm() {
	const email = document.getElementById("email").value.trim();
	const password = document.getElementById("password").value.trim();
	const errorMessage = document.getElementById("error-message");
	
	// Display error message if email or password is empty
	if (!email || !password) {
	errorMessage.style.display = "block";
	return false; // Prevent form submission
	}
	
	// Hide error message if inputs are valid
	errorMessage.style.display = "none";
	return true;
	}
	</script>

Patient	

BookingOverview.cshtml	@model List<Booking>
	
	@{
	ViewBag.Title = "My Appointment Booking";
	}
	Click! to Create a Booking
	
	<h1 class="display-2">@ViewData["Title"]</h1>
	
	@{
	@if (TempData["BookingMessage"] != null)
	{
	
	@TempData["BookingMessage"]
	
	}
	
	@if (TempData["DetailMessage"] != null)
	{

```
      @TempData["DetailMessage"]
    </div>
  }
  @if (TempData["RemarkMessage"] != null)
  {
    <div class="alert alert-info">
      @TempData["RemarkMessage"]
    </div>
  }
  @if (TempData["TherapyMessage"] != null)
  {
    <div class="alert alert-info">
      @TempData["TherapyMessage"]
    </div>
  }

  <table class="table">
    <tr>
      <th scope="col">Booking No</th>
      <th scope="col">Doctor Name</th>
      <th scope="col">Booking Date</th>
      <th scope="col">Booking Time</th>
      <th scope="col">Status</th>
      <th scope="col">Action</th>
    </tr>

    @if (Model != null)
    {
      @foreach (var booking in Model)
      {
        <tr>
          <td>
            @Html.DisplayFor(m => booking.BookingNo)
          </td>
          <td>
            @Html.DisplayFor(m => booking.DoctorName)
          </td>
          <td>
            @Html.DisplayFor(m => booking.BookingDate)
          </td>
          <td>
            @Html.DisplayFor(m => booking.BookingTime)
          </td>
          <td>
            @if (booking.Status == 0)
            {
              <p>Created</p>
            }
            else
            {
              <p>Completed</p>
            }
```

	</td>
	<td>
	@if (booking.Status == 0)
	{
	@using (Html.BeginForm("CancelBooking", "Patient", new { id = booking.BookingNo }, FormMethod.Post))
	{
	<input type="submit" value="Cancel" class="btn btn-danger" onclick="return confirm('Are you sure you want to cancel this booking?');" />
	}
	}
	else
	{
	@using (Html.BeginForm("ShowDetail", "Patient", new { id = booking.BookingNo }, FormMethod.Post))
	{
	<input type="submit" value="Detail" class="btn btn-info"/>
	}
	}
	</td>
	</tr>
	}
	}
	</table>
	}

CreateBooking.cshtml	@model HospitalManagementSystem_Assignment2.Models.Booking
	
	@{
	ViewData["Title"] = "Create a New Booking";
	}
	
	<h1 class="display-3">@ViewData["Title"]</h1>
	
	@if (TempData["ErrorMessage"] != null)
	{
	
	@TempData["ErrorMessage"]
	
	}
	
	@if (TempData["SuccessMessage"] != null)
	{
	
	@TempData["SuccessMessage"]
	
	}

```
@using (Html.BeginForm("CreateBooking", "Patient",
FormMethod.Post))
{
  <div class="form-group">
    @Html.LabelFor(m => m.BookingDate, "Booking Date")
    @Html.TextBoxFor(m => m.BookingDate, new { @class = "form-
control", @type = "date", @id = "bookingDateInput" })
    @Html.ValidationMessageFor(m => m.BookingDate, "", new
{ @class = "text-danger" })
  </div>
  <br />

  <div class="form-group">
    @Html.LabelFor(m => m.BookingTime, "Booking Time")
    @Html.DropDownListFor(m => m.BookingTime,
GetAvailableTimeSlots(), new { @class = "form-control", @id =
"bookingTimeInput" })
    @Html.ValidationMessageFor(m => m.BookingTime, "", new
{ @class = "text-danger" })
  </div>
  <br />

  <div id="doctorSelection" class="form-group">
    @Html.Label("Choose a Doctor")
    <select id="doctorDropdown" name="selectedDoctor"
class="form-control">
      <option value="">Please select a doctor</option>
    </select>
    @Html.ValidationMessage("selectedDoctor", new { @class =
"text-danger" })
  </div>
  <br />

  <div class="form-group">
    <input type="submit" value="Confirm Booking" id="submitBtn"
class="btn btn-success" />
    <a href="@Url.Action("BookingOverview", "Patient", new { id =
TempData["UserId"] })" class="btn btn-danger">Cancel</a>
  </div>
}

@section Scripts {
  <script>
    $(document).ready(function () {
      // When either BookingDate or BookingTime changes, fetch
doctors
      $('#bookingDateInput, #bookingTimeInput').change(function ()
{
        var selectedDate = $('#bookingDateInput').val();
        var selectedTime = $('#bookingTimeInput').val();
```

```
            if (!selectedDate || !selectedTime) {
                $('#doctorDropdown').empty().append('<option
value="">Please select a doctor</option>');
                return;
            }

            const selectedBookingDate = new Date(selectedDate);
            const currentDate = new Date();

            // Reset both dates' time portion to midnight for accurate
comparison
            selectedBookingDate.setHours(0, 0, 0, 0);
            currentDate.setHours(0, 0, 0, 0);

            if (selectedBookingDate < currentDate) {
                alert("The selected date cannot be in the past.");

$('#bookingDateInput').val(currentDate.toISOString().split("T")[0]); //
Reset to today's date
                $('#doctorDropdown').empty().append('<option
value="">Please select a doctor</option>');
                return;
            }

            $.getJSON('@Url.Action("GetDoctorList", "Patient")',
{ bookingDate: selectedDate, bookingTime: selectedTime })
                .done(function (data) {
                    $('#doctorDropdown').empty(); // Clear previous options
                    $('#doctorDropdown').append('<option value="">Please
select a doctor</option>');

                    // Populate the dropdown list with doctors
                    $.each(data, function (index, doctor) {
                        $('#doctorDropdown').append('<option value="' +
doctor.employeeID + '">' + doctor.doctorName + '</option>');
                    });
                })
                .fail(function (error) {
                    console.log(error);
                    alert("An error occurred while fetching the doctor list.");
                    $('#doctorDropdown').empty().append('<option
value="">Please select a doctor</option>');
                });
        });

        $('#submitBtn').click(function (e) {
            if ($("#doctorDropdown").val() === "") {
                e.preventDefault(); // Prevent form submission
                alert("Please select a doctor.");
            }
        });
    });
```

	</script>
	}
	
	@functions {
	// Function to generate time slots between 8 AM and 6 PM in 1-hour intervals
	public List<SelectListItem> GetAvailableTimeSlots()
	{
	var timeSlots = new List<SelectListItem>();
	TimeSpan start = TimeSpan.FromHours(8);
	TimeSpan end = TimeSpan.FromHours(18);
	
	while (start <= end)
	{
	timeSlots.Add(new SelectListItem { Text = start.ToString(@"hh\:mm"), Value = start.ToString(@"hh\:mm") });
	start = start.Add(TimeSpan.FromHours(1));
	}
	
	return timeSlots;
	}
	}

Controllers

AdminController.cs	using System.Xml.Linq; using Microsoft.AspNetCore.Mvc; using HospitalManagementSystem_Assignment2.Models; namespace HospitalManagementSystem_Assignment2.Controllers.Admin { public class AdminController : Controller { private static readonly string xmlFilePathForUsers = "App_Data/Users.xml"; private static readonly string xmlFilePathForDoctors = "App_Data/Doctors.xml"; private static readonly string xmlFilePathForPatients = "App_Data/Patients.xml"; public IActionResult AccountManagement() { XDocument usersDoc = XDocument.Load(xmlFilePathForUsers); XDocument doctorsDoc = XDocument.Load(xmlFilePathForDoctors); XDocument patientsDoc = XDocument.Load(xmlFilePathForPatients); List<DoctorList> doctorList = (from user in usersDoc.Descendants("User")

```
        join doctor in doctorsDoc.Descendants("Doctor") on
user.Element("ID").Value equals doctor.Element("EmployeeID").Value
        select new DoctorList
        {
          EmployeeID = int.Parse(user.Element("ID").Value),
          FirstName = user.Element("FirstName").Value,
          LastName = user.Element("LastName").Value,
          Email = user.Element("Email").Value,
          Password = user.Element("Password").Value,
          DOB = user.Element("DOB").Value,
          Phone = int.Parse(user.Element("Phone").Value),
          Gender = user.Element("Gender").Value,
          AccountType = int.Parse(user.Element("AccountType").Value),
          Specialist = doctor.Element("Specialist").Value
        }).ToList();

        List<PatientList> patientList = (
        from user in usersDoc.Descendants("User")
        join patient in patientsDoc.Descendants("Patient") on
user.Element("ID").Value equals patient.Element("PatientID").Value
        select new PatientList
        {
          ID = int.Parse(user.Element("ID").Value),
          FirstName = user.Element("FirstName").Value,
          LastName = user.Element("LastName").Value,
          Email = user.Element("Email").Value,
          Password = user.Element("Password").Value,
          DOB = user.Element("DOB").Value,
          Phone = int.Parse(user.Element("Phone").Value),
          Gender = user.Element("Gender").Value,
          AccountType = int.Parse(user.Element("AccountType").Value),
          BloodGroup = patient.Element("BloodGroup").Value,
          MedicalHistory = patient.Element("MedicalHistory").Value,
          DrugAllergy = patient.Element("DrugAllergy").Value
        }).ToList();


        List<AdminList> adminList = (
      from user in usersDoc.Descendants("User")
      where (int)user.Element("AccountType") == 0 // AccountType 0 for Admin
      select new AdminList
      {
        ID = int.Parse(user.Element("ID").Value),
        FirstName = user.Element("FirstName").Value,
        LastName = user.Element("LastName").Value,
        Email = user.Element("Email").Value,
        Password = user.Element("Password").Value,
        DOB = user.Element("DOB").Value,
        Phone = int.Parse(user.Element("Phone").Value),
        Gender = user.Element("Gender").Value
      }).ToList();
```

```
        UserList userList = new UserList
        {
          doctorList = doctorList,
          patientList = patientList,
           adminList = adminList
        };

        return View(userList);
      }

      public IActionResult CreateAccount(Account account)
      {
        if (ModelState.IsValid)
        {
          XDocument usersDoc = XDocument.Load(xmlFilePathForUsers);
          XDocument doctorsDoc = XDocument.Load(xmlFilePathForDoctors);
          XDocument patientsDoc = XDocument.Load(xmlFilePathForPatients);

          XElement checkEmail =
usersDoc.Root.Elements("User").FirstOrDefault(u =>
(string)u.Element("Email") == account.Email);
          if (checkEmail != null)
          {
            TempData["ErrorMessage"] = "Email existed, please use another
email.";
            return View();
          }

          int newUserId = GetNewUserId(usersDoc); // Get New ID

          //Add the new user to the XML documents
          XElement newUserElement = new XElement("User",
          new XElement("ID", newUserId),
            new XElement("FirstName", account.FirstName),
            new XElement("LastName", account.LastName),
            new XElement("Email", account.Email),
            new XElement("Phone", account.Phone),
            new XElement("Password", account.Password),
            new XElement("Gender", account.Gender),
            new XElement("DOB", account.DOB),
            new XElement("AccountType", account.AccountType)
          );

          usersDoc.Element("Users").Add(newUserElement);

          if (account.AccountType == 1) // Doctor
          {
            // Add doctor details to the doctorsDoc
            XElement newDoctorElement = new XElement("Doctor",
              new XElement("EmployeeID", newUserId),
              new XElement("Specialist", account.Specialist)
```

```
            );

            doctorsDoc.Element("Doctors").Add(newDoctorElement);
          }
          else if (account.AccountType == 2) // Patient
          {
            // Add patient details to the patientsDoc
            XElement newPatientElement = new XElement("Patient",
              new XElement("PatientID", newUserId),
              new XElement("BloodGroup", account.BloodGroup),
              new XElement("MedicalHistory", account.MedicalHistory),
              new XElement("DrugAllergy", account.DrugAllergy)
            );

            patientsDoc.Element("Patients").Add(newPatientElement);
          }

          // Save the updated XML documents back to their respective paths
          usersDoc.Save(xmlFilePathForUsers);
          doctorsDoc.Save(xmlFilePathForDoctors);
          patientsDoc.Save(xmlFilePathForPatients);


          // Add success message to TempData
          TempData["SuccessMessage"] = "Account created successfully.";

          // Redirect to the AccountManagement page or any other desired
page
          return RedirectToAction("AccountManagement", "Admin");
        }

        // If the model is not valid, return the same view with validation errors
        return View(account);
      }

      public IActionResult EditDoctor(int id)
      {
        XDocument usersDoc = XDocument.Load(xmlFilePathForUsers);
        XDocument doctorsDoc = XDocument.Load(xmlFilePathForDoctors);

        var doc = from user in usersDoc.Descendants("User")
                  join doctor in doctorsDoc.Descendants("Doctor")
                  on user.Element("ID").Value equals
doctor.Element("EmployeeID").Value
                  where (int)user.Element("ID") == id
                  select new DoctorList
                  {
                    EmployeeID = int.Parse(user.Element("ID").Value),
                    FirstName = user.Element("FirstName").Value,
                    LastName = user.Element("LastName").Value,
                    Email = user.Element("Email").Value,
                    Password = user.Element("Password").Value,
```

```
                DOB = user.Element("DOB").Value,
                Phone = int.Parse(user.Element("Phone").Value),
                Gender = user.Element("Gender").Value,
                AccountType = int.Parse(user.Element("AccountType").Value),
                Specialist = doctor.Element("Specialist").Value
            };

        var doctorInfo = doc.FirstOrDefault();
        var accountModel = new Account
        {
          ID = doctorInfo.EmployeeID,
          FirstName = doctorInfo.FirstName,
          LastName = doctorInfo.LastName,
          Email = doctorInfo.Email,
          Phone = doctorInfo.Phone,
          Gender = doctorInfo.Gender,
          DOB = doctorInfo.DOB,
          Specialist = doctorInfo.Specialist
        };

        return View(accountModel);
      }

      public IActionResult EditPatient(int id)
      {
        XDocument usersDoc = XDocument.Load(xmlFilePathForUsers);
        XDocument patientsDoc = XDocument.Load(xmlFilePathForPatients);

        var doc = from user in usersDoc.Descendants("User")
              join patient in patientsDoc.Descendants("Patient")
              on user.Element("ID").Value equals
patient.Element("PatientID").Value
              where (int)user.Element("ID") == id
              select new PatientList
              {
                ID = int.Parse(user.Element("ID").Value),
                FirstName = user.Element("FirstName").Value,
                LastName = user.Element("LastName").Value,
                Email = user.Element("Email").Value,
                Password = user.Element("Password").Value,
                DOB = user.Element("DOB").Value,
                Phone = int.Parse(user.Element("Phone").Value),
                Gender = user.Element("Gender").Value,
                AccountType = int.Parse(user.Element("AccountType").Value),
                BloodGroup = patient.Element("BloodGroup").Value,
                MedicalHistory = patient.Element("MedicalHistory").Value,
                DrugAllergy = patient.Element("DrugAllergy").Value
              };

        var patientInfo = doc.FirstOrDefault();
        var accountModel = new Account
        {
```

```
            ID = patientInfo.ID,
            FirstName = patientInfo.FirstName,
            LastName = patientInfo.LastName,
            Email = patientInfo.Email,
            Phone = patientInfo.Phone,
            Gender = patientInfo.Gender,
            DOB = patientInfo.DOB,
            BloodGroup = patientInfo.BloodGroup,
            MedicalHistory = patientInfo.MedicalHistory,
            DrugAllergy = patientInfo.DrugAllergy
        };

        return View(accountModel);
    }

    public IActionResult DeleteDoctor(int id)
    {
        // Load the XML document
        XDocument userXmlDoc = XDocument.Load(xmlFilePathForUsers);
        XDocument doctorXmlDoc =
XDocument.Load(xmlFilePathForDoctors);

        // Find the user element with the matching ID and remove it
        XElement userToDelete =
userXmlDoc.Root.Elements("User").FirstOrDefault(u => (int)u.Element("ID")
== id);
        if (userToDelete != null)
        {
            userToDelete.Remove();

            // Save the changes back to the XML file
            userXmlDoc.Save(xmlFilePathForUsers);
        }

        XElement doctorToDelete =
doctorXmlDoc.Root.Elements("Doctor").FirstOrDefault(u =>
(int)u.Element("EmployeeID") == id);
        if (doctorToDelete != null)
        {
            doctorToDelete.Remove();

            // Save the changes back to the XML file
            doctorXmlDoc.Save(xmlFilePathForDoctors);
        }

        TempData["DoctorMessage"] = "Doctor removed.";

        // Redirect back to the account management page after the delete
operation
        return RedirectToAction("AccountManagement", "Admin");
    }
```

	public IActionResult DeletePatient(int id)
	{
	// Load the XML document
	XDocument userXmlDoc = XDocument.Load(xmlFilePathForUsers);
	XDocument patientXmlDoc = XDocument.Load(xmlFilePathForPatients);
	
	// Find the user element with the matching ID and remove it
	XElement userToDelete = userXmlDoc.Root.Elements("User").FirstOrDefault(u => (int)u.Element("ID") == id);
	if (userToDelete != null)
	{
	userToDelete.Remove();
	
	// Save the changes back to the XML file
	userXmlDoc.Save(xmlFilePathForUsers);
	}
	
	XElement patientToDelete = patientXmlDoc.Root.Elements("Patient").FirstOrDefault(u => (int)u.Element("PatientID") == id);
	if (patientToDelete != null)
	{
	patientToDelete.Remove();
	
	// Save the changes back to the XML file
	patientXmlDoc.Save(xmlFilePathForPatients);
	}
	
	TempData["PatientMessage"] = "Patient removed.";
	
	// Redirect back to the account management page after the delete operation
	return RedirectToAction("AccountManagement", "Admin");
	}
	
	private int GetNewUserId(XDocument usersDoc)
	{
	var lastUser = usersDoc.Descendants("User").LastOrDefault();
	if (lastUser != null && int.TryParse(lastUser.Element("ID").Value, out int lastUserId))
	{
	return lastUserId + 1;
	}
	
	return 1; // Default to 1 if there are no existing users
	}
	}
	}

DoctorController.cs	using System.Xml.Linq;

```
using Microsoft.AspNetCore.Mvc;
using HospitalManagementSystem_Assignment2.Models;

namespace HospitalManagementSystem_Assignment2.Controllers
{
  public class DoctorController : Controller
  {
    private static readonly string xmlFilePathForBookings =
"App_Data/Bookings.xml";
    private static readonly string xmlFilePathForAttendances =
"App_Data/Attendances.xml";
    private static readonly string xmlFilePathForUsers =
"App_Data/Users.xml";

    public IActionResult BookingList(string id)
    {
      XDocument bookingsDoc = XDocument.Load(xmlFilePathForBookings);
      string userId = string.IsNullOrEmpty(id) ? TempData["UserId"] as string :
id;
      TempData["UserId"] = userId;
      List<Booking> bookingList = (
        from book in bookingsDoc.Descendants("Booking")
        where book.Element("EmployeeID")?.Value == userId
        select new Booking
        {
          BookingNo = int.Parse(book.Element("BookingNo").Value),
          PatientID = int.Parse(book.Element("PatientID").Value),
          PatientName =
GetPatientNameById(int.Parse(book.Element("PatientID").Value)),
          BookingDate = book.Element("BookingDate").Value,
          BookingTime = book.Element("BookingTime").Value,
          Status = int.Parse(book.Element("Status").Value)
        }).ToList();

      return View(bookingList);
    }

    public IActionResult CompleteAttendance(Attendance attendance, int id)
    {
      XDocument bookingsXmlDoc =
XDocument.Load(xmlFilePathForBookings);
      XDocument attendancesXmlDoc =
XDocument.Load(xmlFilePathForAttendances);

      // new redirect to this page
      if (attendance.DiagnosisInfo == null && attendance.Remark == null &&
attendance.Therapy == null)
      {
        return View();
      }

      XElement checkBooking = bookingsXmlDoc.Descendants("Booking")
```

```
                    .FirstOrDefault(u =>
                      u.Element("BookingNo").Value == id.ToString());
        if (checkBooking == null)
        {
          TempData["CompleteErrorMessage"] = "Booking not found.";
          return View();
        }

        // Find the booking element based on the booking no
        XElement bookingToUpdate =
bookingsXmlDoc.Root.Elements("Booking").FirstOrDefault(u =>
(int)u.Element("BookingNo") == id);
        if (bookingToUpdate != null)
        {
          bookingToUpdate.Element("Status").Value = "1"; // update to
complete

          // Save the changes back to the XML file
          bookingsXmlDoc.Save(xmlFilePathForBookings);
        }

        //Add the new attendance to the XML documents
        XElement newAttendanceElement = new XElement("Attendance",
        new XElement("BookingNo", id),
          new XElement("DiagnosisInfo", attendance.DiagnosisInfo),
          new XElement("Remark", attendance.Remark),
          new XElement("Therapy", attendance.Therapy)
        );


attendancesXmlDoc.Element("Attendances").Add(newAttendanceElement);

        // Save the updated XML documents back to their respective paths
        attendancesXmlDoc.Save(xmlFilePathForAttendances);

        TempData["BookingMessage"] = "Booking No: " + id + " completed.";

        string userId = GetEmployeeIdByBookingNo(id);

        // Redirect to the AccountManagement page or any other desired page
        return RedirectToAction("BookingList", "Doctor", new { id = userId });
      }

      public IActionResult ShowAttendance(int id)
      {
        // Load the XML document
        XDocument attendanceXmlDoc =
XDocument.Load(xmlFilePathForAttendances);

        // Find the user element with the matching ID and remove it
```

```
        XElement bookingDetail =
attendanceXmlDoc.Root.Elements("Attendance").FirstOrDefault(u =>
(int)u.Element("BookingNo") == id);
        if (bookingDetail != null)
        {
          TempData["DetailMessage"] = "Diagnosis Detail : " +
bookingDetail.Element("DiagnosisInfo").Value;
          TempData["RemarkMessage"] = "Remark : " +
(string.IsNullOrEmpty(bookingDetail.Element("Remark").Value) ? "N/A" :
bookingDetail.Element("Remark").Value);
          TempData["TherapyMessage"] = "Therapy required : " +
(string.IsNullOrEmpty(bookingDetail.Element("Therapy").Value) ? "N/A" :
bookingDetail.Element("Therapy").Value);
        }

        string userId = GetEmployeeIdByBookingNo(id);

        // Redirect to the AccountManagement page or any other desired page
        return RedirectToAction("BookingList", "Doctor", new { id = userId });
      }

      private string GetPatientNameById(int id)
      {
        // Load the XML file
        XDocument doc = XDocument.Load(xmlFilePathForUsers);

        // Find the user element with the specified ID
        XElement userElement = doc.Descendants("User")
                        .FirstOrDefault(u => u.Element("ID").Value == id.ToString());

        // If the user with the specified ID exists, get the Name
        if (userElement != null)
        {
          return userElement.Element("FirstName").Value + " " +
userElement.Element("LastName").Value;
        }

        return "";
      }

      private string GetEmployeeIdByBookingNo(int id)
      {
        // Load the XML file
        XDocument doc = XDocument.Load(xmlFilePathForBookings);

        // Find the user element with the specified ID
        XElement bookingElement = doc.Descendants("Booking")
                        .FirstOrDefault(u => u.Element("BookingNo").Value ==
id.ToString());

        // If the user with the specified ID exists, get the Name
        if (bookingElement != null)
```

	{
	return bookingElement.Element("EmployeeID").Value;
	}
	
	return "";
	}
	}
	}

HomeController.cs	using System.Diagnostics;
	using Microsoft.AspNetCore.Mvc;
	using HospitalManagementSystem_Assignment2.Models;
	
	namespace HospitalManagementSystem_Assignment2.Controllers
	{
	public class HomeController : Controller
	{
	private readonly ILogger<HomeController> _logger;
	
	public HomeController(ILogger<HomeController> logger)
	{
	_logger = logger;
	}
	
	public IActionResult Index()
	{
	return View();
	}
	
	public IActionResult Privacy()
	{
	return View();
	}
	
	public IActionResult About()
	{
	return View();
	}
	
	public IActionResult Contact()
	{
	return View();
	}
	
	public IActionResult Service()
	{
	return View();
	}
	
	[ResponseCache(Duration = 0, Location =
	ResponseCacheLocation.None, NoStore = true)]
	public IActionResult Error()

	{
	return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
	}
	}
	}

LoginController.cs	using Microsoft.AspNetCore.Mvc;
	using System.Xml.Linq;
	using HospitalManagementSystem_Assignment2.Models;
	
	namespace HospitalManagementSystem_Assignment2.Controllers
	{
	public class LoginController : Controller
	{
	private static readonly string xmlFilePath = "App_Data/Users.xml"; // Path to the XML file storing user info
	
	public IActionResult Login(User model)
	{
	// Clear previous messages each login attempt
	TempData.Clear();
	
	//input validate
	if (!string.IsNullOrWhiteSpace(model.Email) && !string.IsNullOrWhiteSpace(model.Password))
	{
	// Load the XML file
	XDocument doc = XDocument.Load(xmlFilePath);
	
	// Look for a user element
	XElement matchingUser = doc.Descendants("User")
	.FirstOrDefault(user =>
	(string)user.Element("Email") == model.Email &&
	(string)user.Element("Password") == model.Password);
	
	// If a matching user was found in the XML file
	if (matchingUser != null)
	{
	string accountType = (string)matchingUser.Element("AccountType"); // Read the account type
	TempData["UserId"] = (string)matchingUser.Element("ID"); // Store user ID in TempData for later use
	
	// Redirect account type
	if (accountType == "1")
	{
	return RedirectToAction("BookingList", "Doctor"); // For doctors
	}
	else if (accountType == "2")
	{
	return RedirectToAction("BookingOverview", "Patient"); // For patients

	}
	else
	{
	return RedirectToAction("AccountManagement", "Admin"); // For admins
	}
	}
	else
	{
	// No matching user found, show an error
	TempData["ErrorMessage"] = "No such user and password found";
	}
	}
	
	// Return to the login view with any error message
	return View();
	}
	}
	}

PatientController.cs	using System.Globalization;
	using System.Numerics;
	using System.Security.Principal;
	using System.Text.Json;
	using System.Xml.Linq;
	using Microsoft.AspNetCore.Http;
	using Microsoft.AspNetCore.Mvc;
	using Microsoft.AspNetCore.Mvc.Rendering;
	using Microsoft.VisualBasic;
	using HospitalManagementSystem_Assignment2.Models;
	
	namespace HospitalManagementSystem_Assignment2.Controllers
	{
	public class PatientController : Controller
	{
	private static readonly string xmlFilePathForBookings = "App_Data/Bookings.xml";
	private static readonly string xmlFilePathForUsers = "App_Data/Users.xml";
	private static readonly string xmlFilePathForDoctors = "App_Data/Doctors.xml";
	private static readonly string xmlFilePathForAttendances = "App_Data/Attendances.xml";
	public IActionResult BookingOverview(string id)
	{
	XDocument bookingsDoc = XDocument.Load(xmlFilePathForBookings);
	string userId = string.IsNullOrEmpty(id) ? TempData["UserId"] as string : id;
	TempData["UserId"] = userId;
	List<Booking> bookingList = (
	from book in bookingsDoc.Descendants("Booking")
	where book.Element("PatientID")?.Value == userId

```
            select new Booking
            {
                BookingNo = int.Parse(book.Element("BookingNo").Value),
                PatientID = int.Parse(book.Element("PatientID").Value),
                DoctorName =
GetDoctorNameById(int.Parse(book.Element("EmployeeID").Value)),
                BookingDate = book.Element("BookingDate").Value,
                BookingTime = book.Element("BookingTime").Value,
                Status = int.Parse(book.Element("Status").Value)
            }).ToList();

        return View(bookingList);
    }

    public IActionResult CreateBooking(Booking booking, int selectedDoctor,
int id)
    {
        XDocument bookingsXmlDoc =
XDocument.Load(xmlFilePathForBookings);
        TempData["UserId"] = id;
        // If bookingList is null, initialize it to an empty list
        if (booking.bookingList == null)
        {
            booking.bookingList = new List<Booking>();
        }

        // new redirect to this page
        if(booking.BookingDate == null && booking.BookingTime == null)
        {
            return View();
        }

        XElement checkBooking = bookingsXmlDoc.Descendants("Booking")
                        .FirstOrDefault(u =>
                            u.Element("BookingDate").Value == booking.BookingDate
&&
                            u.Element("BookingTime").Value ==
booking.BookingTime
                        );
        if (checkBooking != null)
        {
            TempData["AddBookingErrorMessage"] = "You had booked a same
time slot on the day. Please book another time slot.";
            return View();
        }

        int newBookingNo = GetNewBookingNo(bookingsXmlDoc); // Get New
ID

        //Add the new user to the XML documents
        XElement newBookingElement = new XElement("Booking",
        new XElement("BookingNo", newBookingNo),
```

```
            new XElement("EmployeeID", selectedDoctor),
            new XElement("PatientID", id),
            new XElement("BookingDate", booking.BookingDate),
            new XElement("BookingTime", booking.BookingTime),
            new XElement("Status", "0") // 0: created
        );

        bookingsXmlDoc.Element("Bookings").Add(newBookingElement);

        // Save the updated XML documents back to their respective paths
        bookingsXmlDoc.Save(xmlFilePathForBookings);

        TempData["BookingMessage"] = "Booking No: " + newBookingNo + "
was created.";
        // Redirect to the AccountManagement page or any other desired page
        return RedirectToAction("BookingOverview", "Patient", new { id = id });
    }

    public IActionResult CancelBooking(int id)
    {
        // Load the XML document
        XDocument bookingsXmlDoc =
XDocument.Load(xmlFilePathForBookings);

        // Find the user element with the matching ID and remove it
        XElement bookingToCancel =
bookingsXmlDoc.Root.Elements("Booking").FirstOrDefault(u =>
(int)u.Element("BookingNo") == id);
        if (bookingToCancel != null)
        {
            bookingToCancel.Remove();

            // Save the changes back to the XML file
            bookingsXmlDoc.Save(xmlFilePathForBookings);
        }

        TempData["BookingMessage"] = "Booking Cancelled.";
        string userid = bookingToCancel.Element("PatientID").Value;

        return RedirectToAction("BookingOverview", "Patient", new { id =
userid });
    }

    public IActionResult ShowDetail(int id)
    {
        // Load the XML document
        XDocument attendanceXmlDoc =
XDocument.Load(xmlFilePathForAttendances);

        // Find the user element with the matching ID and remove it
```

```
        XElement bookingDetail =
attendanceXmlDoc.Root.Elements("Attendance").FirstOrDefault(u =>
(int)u.Element("BookingNo") == id);
        if (bookingDetail != null)
        {
          TempData["DetailMessage"] = "Diagnosis Detail : " +
bookingDetail.Element("DiagnosisInfo").Value;
          TempData["RemarkMessage"] = "Remark from Doctor : " +
(string.IsNullOrEmpty(bookingDetail.Element("Remark").Value) ? "N/A" :
bookingDetail.Element("Remark").Value);
          TempData["TherapyMessage"] = "Therapy required : " +
(string.IsNullOrEmpty(bookingDetail.Element("Therapy").Value) ? "N/A" :
bookingDetail.Element("Therapy").Value);
        }
        string userid = GetPatientIdByBookingNo(id);
        return RedirectToAction("BookingOverview", "Patient", new { id =
userid });
    }

    public IActionResult GetDoctorList(string bookingDate, string
bookingTime)
    {
      // Create a Booking object with the selected date and time
      Booking booking = new Booking
      {
        BookingDate = bookingDate,
        BookingTime = bookingTime
      };

      // Get the list of available doctors
      List<Booking> availableDoctors =
GetAvailableDoctorsForBooking(booking);

      // Return the data as JSON
      return Json(availableDoctors);
    }

    private string GetDoctorNameById(int id)
    {
      // Load the XML file
      XDocument doc = XDocument.Load(xmlFilePathForUsers);

      // Find the user element with the specified ID
      XElement userElement = doc.Descendants("User")
                    .FirstOrDefault(u => u.Element("ID").Value == id.ToString());

      // If the user with the specified ID exists, get the Name
      if (userElement != null)
      {
        return userElement.Element("FirstName").Value + " " +
userElement.Element("LastName").Value;
      }
```

```
        return "";
    }

    private List<Booking> GetAvailableDoctorsForBooking(Booking booking)
    {
        // Load the XML data for doctors and bookings
        XDocument doctorsDoc = XDocument.Load(xmlFilePathForDoctors);
        XDocument bookingsDoc = XDocument.Load(xmlFilePathForBookings);

        // Create a string representation of the selected date and time
        string selectedDateTime = booking.BookingDate + " " +
booking.BookingTime;

        // Query the XML data to get a list of doctors who do not have a booking
at the selected date and time
        var availableDoctors = (
          from doctor in doctorsDoc.Descendants("Doctor")
          where !bookingsDoc.Descendants("Booking")
            .Any(bookingElement =>
              bookingElement.Element("EmployeeID").Value ==
doctor.Element("EmployeeID").Value &&
              (bookingElement.Element("BookingDate").Value + " " +
bookingElement.Element("BookingTime").Value == selectedDateTime)
            )
          select new Booking
          {
            EmployeeID = int.Parse(doctor.Element("EmployeeID")?.Value),
            DoctorName =
GetDoctorNameById(int.Parse(doctor.Element("EmployeeID")?.Value))
          }
        ).ToList();

        return availableDoctors;
    }

    private int GetNewBookingNo(XDocument bookingXml)
    {
        var lastBooking = bookingXml.Descendants("Booking").LastOrDefault();
        if (lastBooking != null &&
int.TryParse(lastBooking.Element("BookingNo").Value, out int lastBookingNo))
        {
          return lastBookingNo + 1;
        }

        return 1; // Default to 1 if there are no existing users
    }

    private string GetPatientIdByBookingNo(int id)
    {
        // Load the XML file
        XDocument doc = XDocument.Load(xmlFilePathForBookings);
```

```
        // Find the user element with the specified ID
        XElement bookingElement = doc.Descendants("Booking")
                        .FirstOrDefault(u => u.Element("BookingNo").Value ==
id.ToString());

        // If the user with the specified ID exists, get the Name
        if (bookingElement != null)
        {
           return bookingElement.Element("PatientID").Value;
        }

        return "";
     }

  }
}
```