



FIT5032 Design Report
Major Application Development
{High Distinction}

VMed MRI Imaging Group

VINCENT: 30052586

Table of Contents

1. Web Application Title and Description	3
2. User Stories and Use Case Diagram	3
3. Block/Functional Diagram.....	6
4. Your Selected Approach when Constructing the Application.....	6
5. Class Diagram or Entity Relationship Diagram.....	9
6. Data Dictionary.....	10
7. Mockup Prototypes and Implementation with User Registration and Authentication	11
8. Usability Design Review	17
9. Development Methodology	20
10. Versioning	21
11. Innovation and Research.....	23
12. Checklist of Site Functionality.....	32
References.....	33
Appendix.....	33
Appendix A	33
Appendix B	34

1. Web Application Title and Description

Title: V-Med MRI Imaging Group | Empowering Footballer's Futures Through Advanced MRI Imaging

Description: V-Med MRI Imaging Group offers a range of MRI imaging procedures including Lumbar Spine MRI, Cardiac MRI, Knee MRI, and Shoulder MRI for footballers across Australia. We are committed to providing our patients with high-resolution MRI imaging, personalized health reports, and recovery solutions that get them back on the field faster.

2. User Stories and Use Case Diagram

User Stories:

1. As a patient, I want to book an appointment with a doctor, so that I can receive the medical care and advice I need in terms of MRI imaging services.
2. As a doctor, I want to send an email to my patients, so that I can effectively communicate important information, updates, and reminders related to their MRI imaging results and healthcare.
3. As a patient, I want to navigate the location of a clinic on a map, so that I can easily find and reach the healthcare facility for my appointment or medical needs.
4. As a patient, I want to give a rating of a doctor's imaging service, so that I can provide feedback on my experience and help other patients make informed decisions about their healthcare.
5. As an admin, I want to send an email to all patients, so that I can efficiently communicate important updates, announcements, or general healthcare information to them.
6. As a patient, I want to receive a detailed and transparent breakdown of the costs associated with my MRI imaging services, so that I can make informed financial decisions regarding my healthcare and plan for any necessary expenses.
7. As a patient, I want to visually present my past appointment records, so that I can have a clear and accessible overview of my healthcare history, track my medical progress, and share this information with healthcare providers when necessary.

Use Case Diagram:

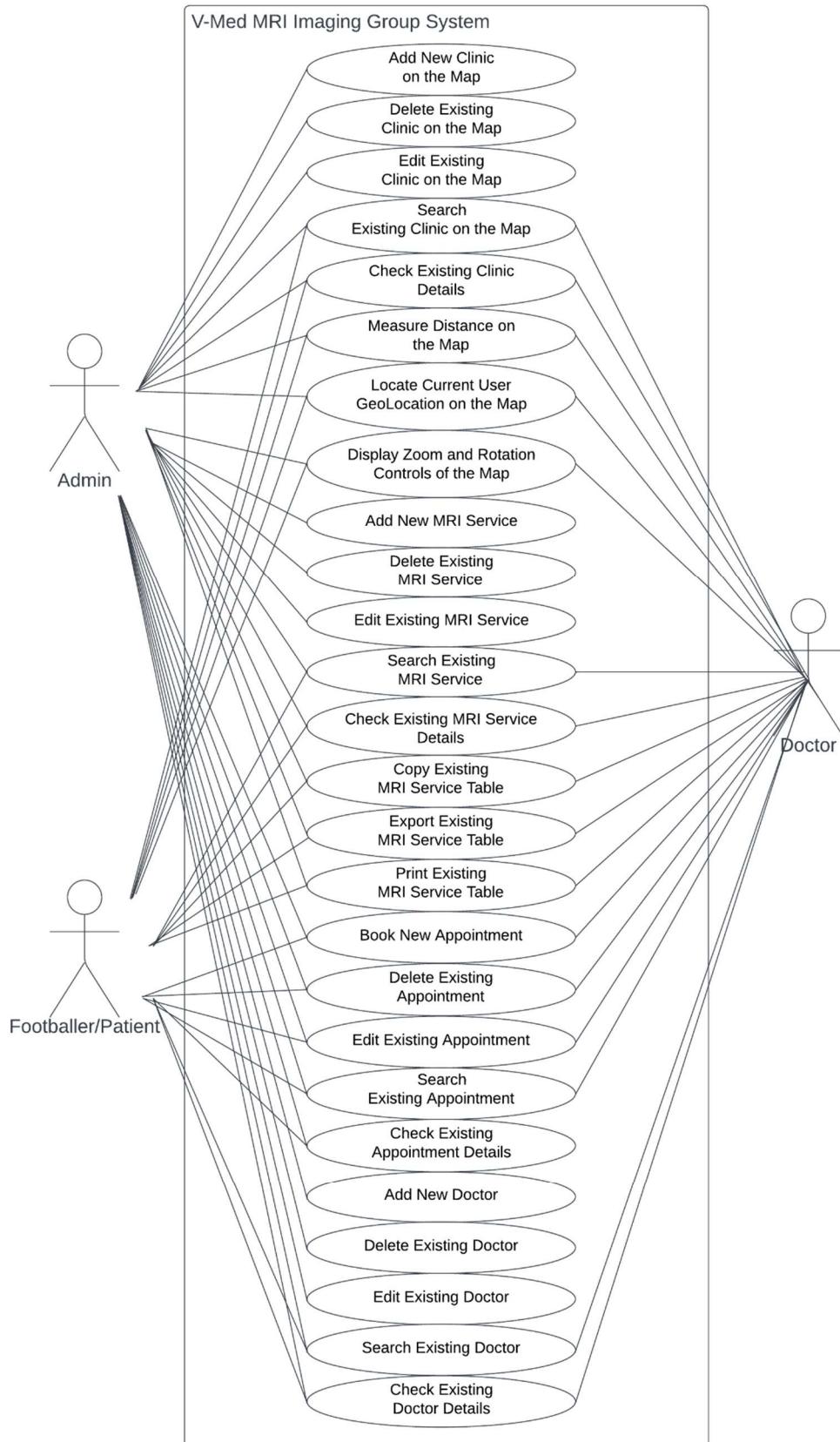


Figure 1. Use Case Diagram Part I

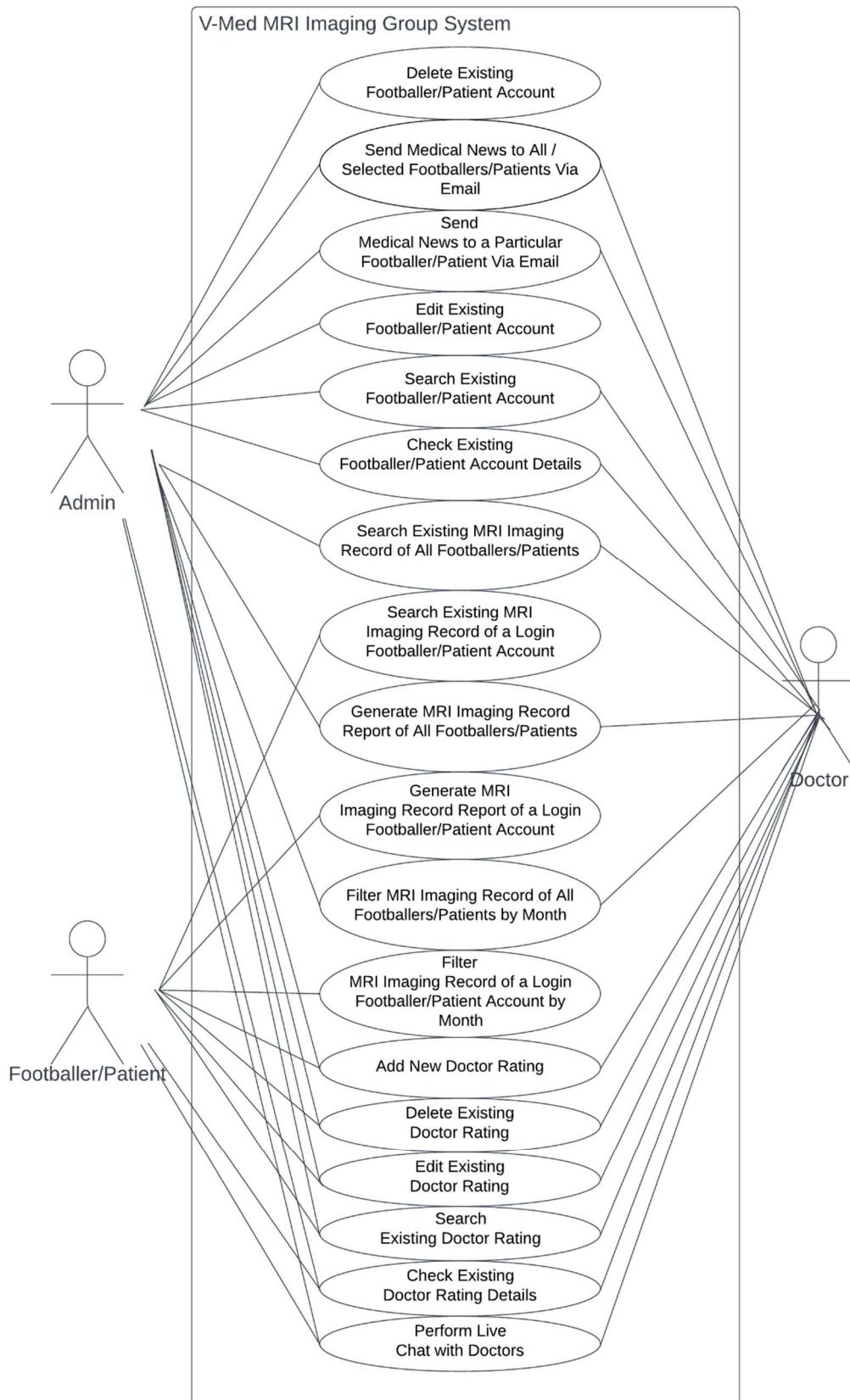


Figure 2. Use Case Diagram Part 2

3. Block/Functional Diagram

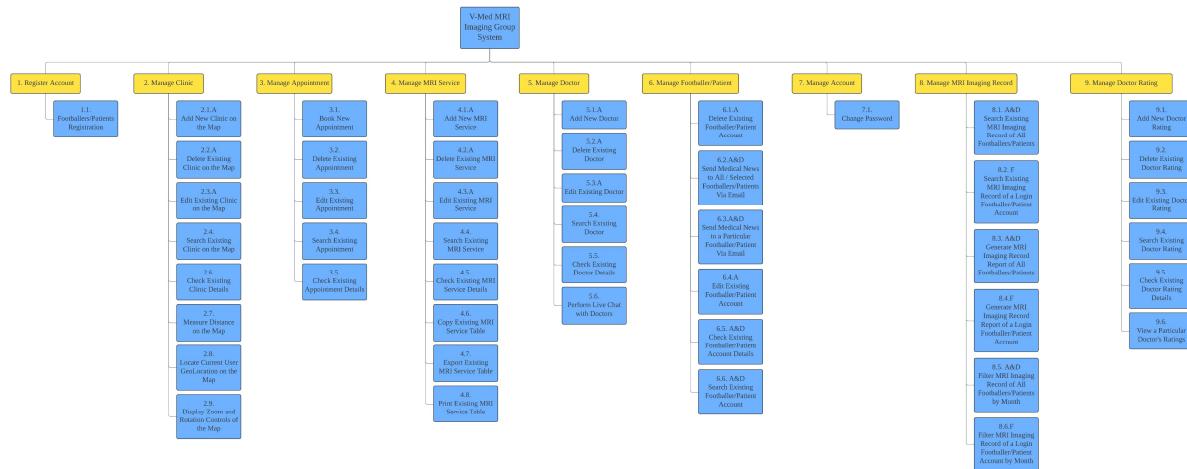


Figure 3. Block/Functional Diagram Full Screenshot

Note: A represents the function can only be performed by role Admin; D represents the function can only be performed by role Doctor; F represents the function can only be performed by role Footballer/Patient. In addition, the block/functional diagram above covers all the business requirements (BRs) up to (E.4). Since the screenshot is too small, please refer to Appendix A for the cropped version.

4. Your Selected Approach when Constructing the Application

In this project, I have adopted two approaches namely Database First approach and Code First approach with Database First approach plays most of the role. There are several reasons to choose Database First approach to finish my project. Firstly, this approach allows me to use Trigger in calculating the average rating of the doctors (BR (D.3) Rating). To illustrate, by using Trigger codes, whenever a new rating data is being added, deleted, or edited by any users, the average rating field in the Doctor entity will automatically be updated. The second reason is that time constraints are a crucial factor in this project. Since I have no experience with C# and have a better understanding in SQL query and schema, I think it is the best choice to go with Database First approach because it allows me to automatically generate model classes, controllers, and views based on the current database schema. The next reason is that I need to manipulate and report the data to fulfill BR (E.3) Chart as well as to remove all appointments related to a particular user (or other data which has a foreign key attached to it), which can be easily achieved by using ON DELETE CASCADE command. Finally, this approach also allows me to handle the versioning of my project because I have a text file that contains the SQL query to generate the entire model. Below is my SQL query to generate the entire model:

```
-- Create the Clinic Table
CREATE TABLE Clinic (
    ClinicID INT PRIMARY KEY IDENTITY(1,1),
    ClinicName VARCHAR(100) NOT NULL,
    Address VARCHAR(255) NOT NULL,
    Latitude DECIMAL(9, 6) NOT NULL,
    Longitude DECIMAL(9, 6) NOT NULL,
    ContactNumber VARCHAR(15) NOT NULL
);

-- Create the Doctor Table
CREATE TABLE Doctor (
    DoctorID INT PRIMARY KEY IDENTITY(1,1),
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Specialization VARCHAR(100) NOT NULL,
    ContactNumber VARCHAR(15) NOT NULL,
    Email VARCHAR(100) NOT NULL,
    ClinicID INT, -- Reference to Clinic
    AverageRating DECIMAL(3, 2), -- To store the average rating
    FOREIGN KEY (ClinicID) REFERENCES Clinic(ClinicID)
);

-- Create the Rating Table
CREATE TABLE Rating (
    RatingID INT PRIMARY KEY IDENTITY(1,1),
    DoctorID INT NOT NULL, -- Reference to Doctor
    RatingValue DECIMAL(3, 2) NOT NULL, -- Rating value, e.g., 4.5
    Comment TEXT, -- Optional comment or review
    FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID)
);

-- Create the Service Table
CREATE TABLE Service (
    ServiceID INT PRIMARY KEY IDENTITY(1,1),
    ServiceName VARCHAR(100) NOT NULL,
    Description VARCHAR(500),
    Cost DECIMAL(10, 2) NOT NULL
);

-- Create the Appointment Table
CREATE TABLE Appointment (
    AppointmentID INT PRIMARY KEY IDENTITY(1,1),
    DoctorID INT NOT NULL,
    ServiceID INT NOT NULL,
    AppointmentDate DATE NOT NULL,
    Notes VARCHAR(500),
    FOREIGN KEY (DoctorID) REFERENCES Doctor(DoctorID),
    FOREIGN KEY (ServiceID) REFERENCES Service(ServiceID)
);
```

```
-- Create the Trigger to calculate and update the average rating for each doctor
-- This must be in a separate batch
GO

CREATE TRIGGER UpdateDoctorAverageRating
ON Rating
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    DECLARE @DoctorID INT;

    -- Check if any records were inserted or updated
    IF EXISTS (SELECT * FROM inserted)
    BEGIN
        SELECT TOP 1 @DoctorID = DoctorID FROM inserted;

        -- Update the AverageRating for the affected doctor
        UPDATE Doctor
        SET AverageRating = (
            SELECT AVG(RatingValue)
            FROM Rating
            WHERE DoctorID = @DoctorID
        )
        WHERE DoctorID = @DoctorID;
    END
    ELSE IF EXISTS (SELECT * FROM deleted)
    BEGIN
        -- Handle DELETE operation
        -- Get the DoctorID from the first deleted record (assuming it's the same for all)
        SELECT TOP 1 @DoctorID = DoctorID FROM deleted;

        -- Update the AverageRating for the affected doctor
        UPDATE Doctor
        SET AverageRating = (
            SELECT AVG(RatingValue)
            FROM Rating
            WHERE DoctorID = @DoctorID
        )
        WHERE DoctorID = @DoctorID;
    END
END;
```

After entering the SQL codes above, I will need to manually link Appointment entity to the AspNetUsers by using UserId as the foreign key with the codes below.

[UserId] NVARCHAR (128) NOT NULL, CONSTRAINT [FK_dbo.Appointment.AspNetUsers_UserId] FOREIGN KEY ([UserId]) REFERENCES [dbo].[AspNetUsers] ([Id]) ON DELETE CASCADE
--

Note: In the SQL query above, I forgot to include ON DELETE CASCADE command in all the foreign keys. I have updated my database during the testing and checking process (i.e., this is also another reason why I choose this approach – only need to add ON DELETE CASCADE and “Update” database).

Another approach is Code First approach. This approach was adopted when I want to add more registration required fields such as FirstName, LastName, Gender, and PhoneNumber. To illustrate, I need to use migration to update my database (i.e., AspNetUsers entity) as shown in Figure 4 below.

5. Class Diagram or Entity Relationship Diagram

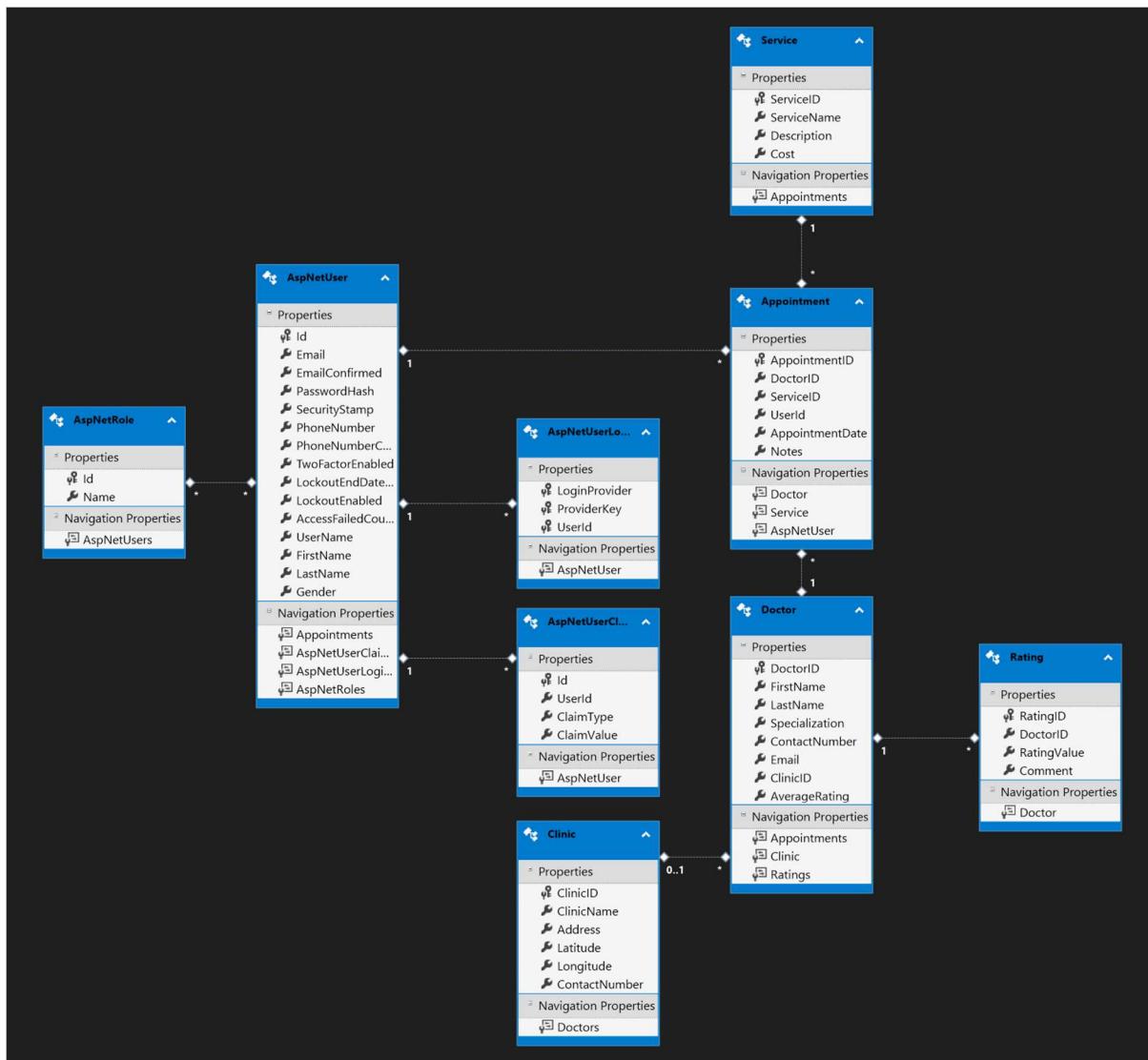


Figure 4. Entity Relationship (ER) Diagram with Crows Feet Notation Screenshot

Figure 4 above does not cover the email features. In this project, both email business requirements (BR (D.2) and (E.1)) have been achieved by using EmailController.cs, Views/Email/BulkEmail.cshtml, and Views/Email/Index.cshtml (i.e., there is no model involved). Below is the screenshot of the Email class diagram.



Figure 5. Email Class Diagram Screenshot

6. Data Dictionary

Table 1. Data Dictionary Table

Data Dictionary Table		
Entity: Service		
Field Name	Data Type	Justification
ServiceID	INT	Type INT for efficient indexing of the primary key constraint
ServiceName	VARCHAR(100)	To accommodate reasonably long service names
Description	VARCHAR(500)	To accommodate more detailed descriptions
Cost	DECIMAL(10, 2)	The cost of the service with two decimal places for precision (i.e., include cents)
Entity: Appointment		
Field Name	Data Type	Justification
AppointmentID	INT	Type INT for efficient indexing of the primary key constraint
AppointmentDate	DATE	To simplify the project because TIME will make it too complex
Notes	VARCHAR(500)	To accommodate more detailed notes
Entity: Doctor		
Field Name	Data Type	Justification
DoctorID	INT	Type INT for efficient indexing of the primary key constraint
FirstName	VARCHAR(50)	To accommodate reasonably long first names
LastName	VARCHAR(50)	To accommodate reasonably long last names
Specialization	VARCHAR(100)	To accommodate reasonably long specializations
ContactNumber	VARCHAR(10)	To accommodate standard phone numbers (e.g., 0466598833)
Email	VARCHAR(100)	To accommodate long email addresses
AverageRating	DECIMAL(3, 2)	To store the average rating of the doctor with two decimal places for precision
Entity: Rating		
Field Name	Data Type	Justification
RatingID	INT	Type INT for efficient indexing of the primary key constraint
RatingValue	DECIMAL(3, 2)	To store the rating value with two decimal places for precision
Comment	TEXT	TEXT data type is suitable for accommodating longer comments (e.g., 2 paragraphs of complaint)
Entity: Clinic		
Field Name	Data Type	Justification

ClinicID		Type INT for efficient indexing of the primary key constraint
ClinicName	VARCHAR(100)	To accommodate a reasonably long name
Address	VARCHAR(255)	To accommodate longer addresses
Latitude	DECIMAL(9, 6)	To store latitude with a high level of precision
Longitude	DECIMAL(9, 6)	To store longitude with a high level of precision
ContactNumber	VARCHAR(10)	To accommodate standard phone numbers (e.g., 0466598833)

7. Mockup Prototypes and Implementation with User Registration and Authentication

Mockup Prototypes Design:

Admin Doctor Page

V-Med MRI Imaging Group	About	Contact	Clinic	Doctors	Services	Appointment	Rating	User	Record																																																		
Doctor List																																																											
<input type="button" value="Add Doctor"/> <input type="button" value="Show"/> <input type="text" value="10"/> <input type="button" value="V entries"/>		<input type="text" value="Search"/>																																																									
Variable 1	^v	Variable 2	^v	Variable 3	^v	Variable 4	^v	Rating	^v																																																		
List of doctors with their respective attributes																																																											
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>3.3★</td><td><input checked="" type="checkbox"/></td><td></td><td></td><td></td></tr> </table>										3.3★	<input checked="" type="checkbox"/>				3.3★	<input checked="" type="checkbox"/>				3.3★	<input checked="" type="checkbox"/>				3.3★	<input checked="" type="checkbox"/>				3.3★	<input checked="" type="checkbox"/>				3.3★	<input checked="" type="checkbox"/>				3.3★	<input checked="" type="checkbox"/>				3.3★	<input checked="" type="checkbox"/>				3.3★	<input checked="" type="checkbox"/>				3.3★	<input checked="" type="checkbox"/>			
3.3★	<input checked="" type="checkbox"/>																																																										
3.3★	<input checked="" type="checkbox"/>																																																										
3.3★	<input checked="" type="checkbox"/>																																																										
3.3★	<input checked="" type="checkbox"/>																																																										
3.3★	<input checked="" type="checkbox"/>																																																										
3.3★	<input checked="" type="checkbox"/>																																																										
3.3★	<input checked="" type="checkbox"/>																																																										
3.3★	<input checked="" type="checkbox"/>																																																										
3.3★	<input checked="" type="checkbox"/>																																																										
3.3★	<input checked="" type="checkbox"/>																																																										
Showing 1 to 10 of 100 entries																																																											
©2023 - V-Med MRI Imaging Group																																																											
<input type="button" value="Previous"/> <input type="button" value="1"/> <input type="button" value="2"/> <input type="button" value="Next"/>																																																											

Figure 6. Mock-up Prototype Design BR (C.3), (D.3) and (C.1) - Admin Doctor Page [Part 1]

Admin Service Page

V-Med MRI Imaging Group	About	Contact	Clinic	Doctors	Services	Appointment	Rating	User	Record																														
MRI Service List																																							
<input type="button" value="Add Service"/> <input type="button" value="Show"/> <input type="text" value="10"/> <input type="button" value="V entries"/>		<input type="text" value="Search"/>																																					
Copy	CSV	Excel	PDF	Print																																			
Variable 1	^v	Variable 2	^v	Variable 3	^v	Variable 4	^v	Variable 5	^v																														
List of doctors with their respective attributes																																							
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td><input checked="" type="checkbox"/></td><td></td><td></td></tr> </table>										<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>		
<input checked="" type="checkbox"/>																																							
<input checked="" type="checkbox"/>																																							
<input checked="" type="checkbox"/>																																							
<input checked="" type="checkbox"/>																																							
<input checked="" type="checkbox"/>																																							
<input checked="" type="checkbox"/>																																							
<input checked="" type="checkbox"/>																																							
<input checked="" type="checkbox"/>																																							
<input checked="" type="checkbox"/>																																							
<input checked="" type="checkbox"/>																																							
Showing 1 to 10 of 100 entries																																							
©2023 - V-Med MRI Imaging Group																																							
<input type="button" value="Previous"/> <input type="button" value="1"/> <input type="button" value="2"/> <input type="button" value="Next"/>																																							

Figure 7. Mock-up Prototype Design BR (C.3), (D.3), (E.4) and (C.1) - Admin Service Page [Part 2]

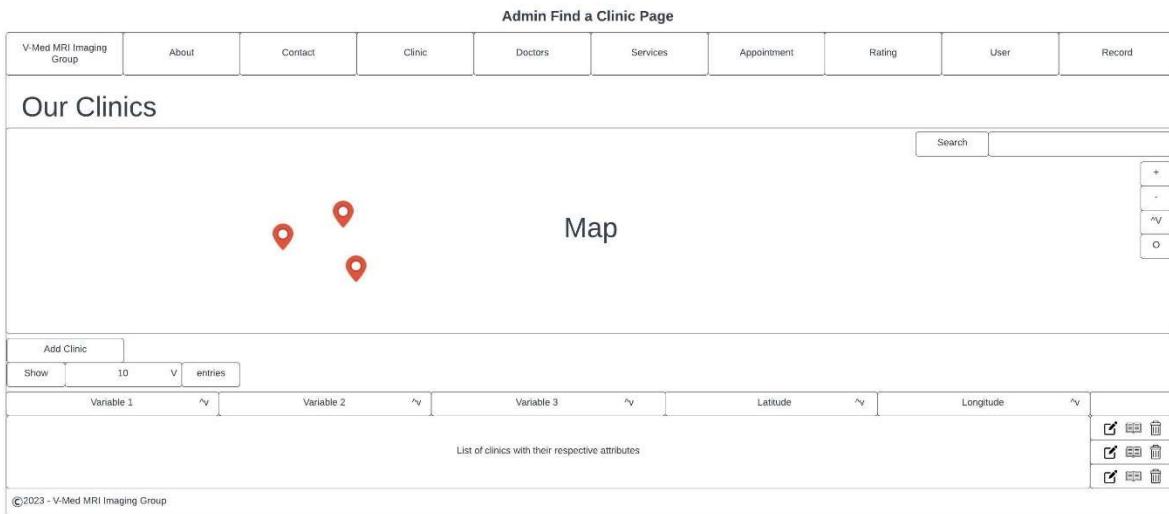


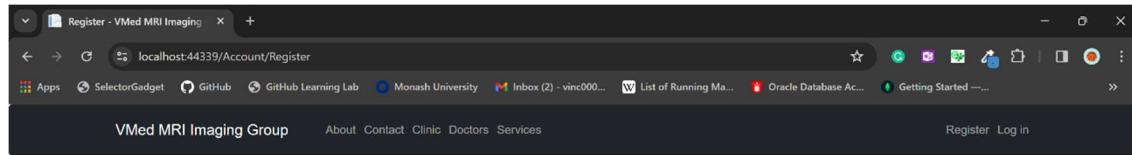
Figure 8. Mock-up Prototype Design BR (E.3) - Admin Clinic Page

Implementation with User Registration and Authentication:

In this project, all the user registration and authentication has been implemented in the AccountController.cs and Register.cshtml with Javascript.

Git URL: https://github.com/vinctechgeek/FIT5032_VMedProject

Final User Registration Implementation:



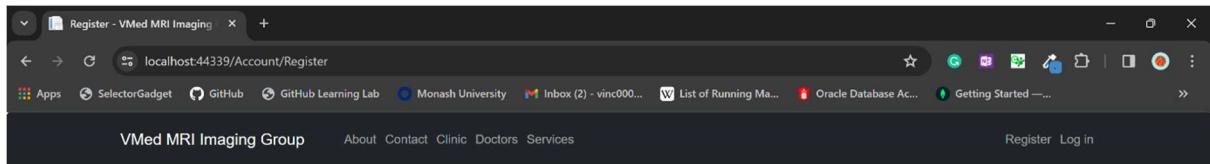
Register.

Create a new account.

FirstName	<input type="text"/>
LastName	<input type="text"/>
Gender	Select Gender
Phone Number	<input type="text"/>
Email	<input type="text"/>
Password	<input type="password"/>
Confirm password	<input type="password"/>

© 2023 - VMed MRI Imaging Group | Empowering Footballer's Futures Through Advanced MRI Imaging

Figure 9. Registration Page Screenshot Part 1



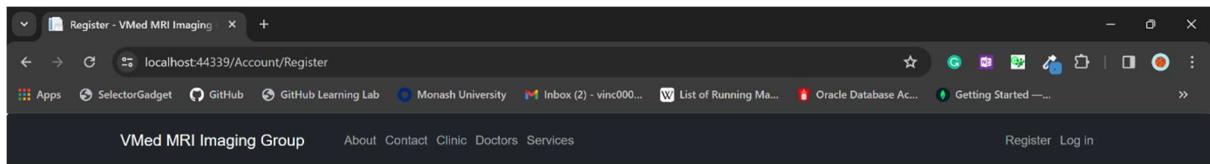
Register.

Create a new account.

FirstName	<input type="text"/>
LastName	<input type="text"/>
Gender	<input type="button" value="Select Gender"/> <div style="border: 1px solid #ccc; padding: 2px; margin-top: 2px;"> Select Gender Male Female Other </div>
Phone Number	<input type="text"/>
Email	<input type="text"/>
Password	<input type="password"/>
Confirm password	<input type="password"/>

© 2023 - VMed MRI Imaging Group | Empowering Footballer's Futures Through Advanced MRI Imaging

Figure 10. Registration Page Screenshot Part 2



Register.

Create a new account.

- Gender is required
- Please enter your phone number.
- The Email field is required.
- The Password field is required.

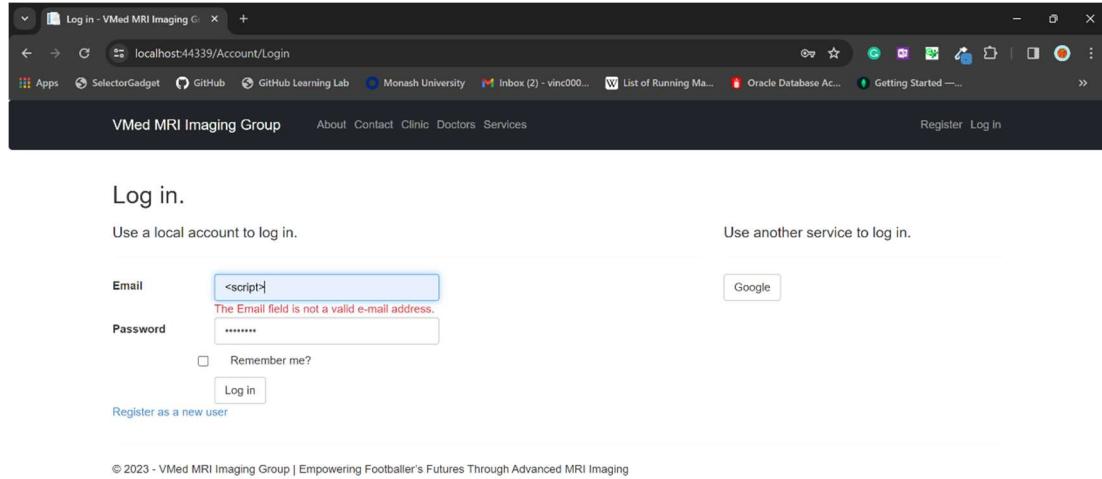
FirstName	<input type="text"/>
LastName	<input type="text"/>
Gender	<input type="button" value="Select Gender"/> <div style="border: 1px solid #ccc; padding: 2px; margin-top: 2px;"> Select Gender Gender is required </div>
Phone Number	<input type="text"/> Please enter your phone number.
Email	<input type="text"/>
Password	<input type="password"/>
Confirm password	<input type="password"/>

© 2023 - VMed MRI Imaging Group | Empowering Footballer's Futures Through Advanced MRI Imaging

Figure 11. Registration Page Screenshot Part 3 (Validation)

Note: To improve the security, I have also enabled Javascript to automatically delete any characters that are not numbers, whitespaces, and letters (i.e., symbols such as < or * will be automatically removed when typing).

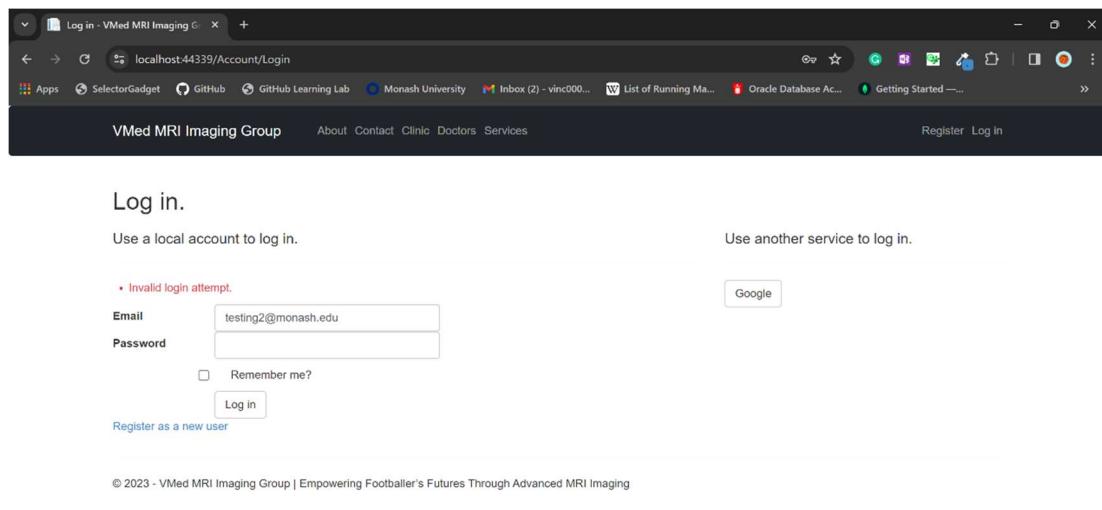
Final User Authentication Implementation:



The screenshot shows a browser window titled "Log in - VMed MRI Imaging". The URL is "localhost:44339/Account/Login". The page has a dark header with the VMed MRI Imaging Group logo and navigation links for About, Contact, Clinic, Doctors, Services, Register, and Log In.

The main content area is titled "Log in." It contains two input fields: "Email" (containing "<script>") and "Password" (containing "*****"). Below the password field is a checked "Remember me?" checkbox and a "Log in" button. To the right of the input fields is a "Google" sign-in button. A message "The Email field is not a valid e-mail address." is displayed above the "Email" field. On the far right, there is a link "Use another service to log in." and a footer note "© 2023 - VMed MRI Imaging Group | Empowering Footballer's Futures Through Advanced MRI Imaging".

Figure 12. Email Validation Screenshot



This screenshot is identical to Figure 12, showing the VMed MRI Imaging Group login page. The "Email" field now contains "testing2@monash.edu" instead of "<script>". A red asterisk and the text "Invalid login attempt." are displayed next to the "Email" field, indicating an error. All other elements, including the "Password" field, "Remember me?" checkbox, "Log in" button, "Google" sign-in button, and the footer text, remain the same.

Figure 13. Invalid Login Input Validation (i.e., Username or Password is wrong)

Implementation (Codes):

```

public class RegisterViewModel
{
    [Required]
    [EmailAddress]
    [Display(Name = "Email")]
    public string Email { get; set; }

    [Required]
    [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength = 6)]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    public string Password { get; set; }

    [DataType(DataType.Password)]
    [Display(Name = "Confirm password")]
    [Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]
    public string ConfirmPassword { get; set; }

    [Required(ErrorMessage = "Please enter your phone number.")]
    [Display(Name = "Phone Number")]
    [RegularExpression(@"^\d{10}$", ErrorMessage = "Phone number must be 10 digits.")]
    public string PhoneNumber { get; set; }

    [Required(ErrorMessage = "First name is required")]
    [StringLength(50, MinimumLength = 2, ErrorMessage = "First name must be between 2 and 50 characters")]
    public string FirstName { get; set; }

    [Required(ErrorMessage = "Last name is required")]
    [StringLength(50, MinimumLength = 2, ErrorMessage = "Last name must be between 2 and 50 characters")]
    public string LastName { get; set; }

    [Required(ErrorMessage = "Gender is required")]
    [Display(Name = "Gender")]
    public string Gender { get; set; }
}

```

Figure 14. User Registration Model Validation (AccountViewModels.cs)

```

public async Task<ActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = new ApplicationUser
        {
            UserName = model.Email,
            Email = model.Email,
            PhoneNumber = model.PhoneNumber,
            FirstName = model.FirstName,
            LastName = model.LastName,
            Gender = model.Gender
        };

        var result = await UserManager.CreateAsync(user, model.Password);
        if (result.Succeeded)
        {
            await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);
            return RedirectToAction("Index", "Home");
        }
        AddErrors(result);
    }
}

public async Task<ActionResult> ConfirmEmail(string userId, string code)
{
}

```

Figure 15. User Registration with Additional Fields Request (AccountController.cs)

```

1 <model> FIT5032_VMedProject.Models.RegisterViewModel
2     ViewBag.Title = "Register";
3
4     <main aria-labelledby="title">
5         <h2 id="title">@ViewBag.Title</h2>
6
7         <using> (Html.BeginForm("Register", "Account", FormMethod.Post, new { role = "form" }))</using>
8
9             <Html.AntiForgeryToken()>
10            <h4>Create a new account.</h4>
11            <hr />
12            <Html.ValidationSummary("", new { @class = "text-danger" })>
13
14             <div class="row">
15                 <Html.LabelFor(m => m.FirstName, new { @class = "col-md-2 col-form-label" })>
16                     <div class="col-md-10">
17                         <input type="text" class="form-control" id="FirstName" name="FirstName" />
18                         <Html.ValidationMessageFor(m => m.FirstName, "", new { @class = "text-danger" })>
19                     </div>
20             </div>
21
22             <div class="row">
23                 <Html.LabelFor(m => m.LastName, new { @class = "col-md-2 col-form-label" })>
24                     <div class="col-md-10">
25                         <input type="text" class="form-control" id="LastName" name="LastName" />
26                         <Html.ValidationMessageFor(m => m.LastName, "", new { @class = "text-danger" })>
27                     </div>
28             </div>
29
30             <div class="row">
31                 <Html.LabelFor(m => m.Gender, new { @class = "col-md-2 col-form-label" })>
32                     <div class="col-md-10">
33                         <Html.DropDownListFor(m => m.Gender, new SelectList(new []
34                         {
35                             new { Value = "Male", Text = "Male" },
36                             new { Value = "Female", Text = "Female" },
37                             new { Value = "Other", Text = "Other" }
38                         }, "Text"), "Select Gender", new { @class = "form-control" })>
39                         <Html.ValidationMessageFor(m => m.Gender, "", new { @class = "text-danger" })>
40                     </div>
41             </div>
42
43             <div class="row">
44                 <Html.LabelFor(m => m.PhoneNumber, new { @class = "col-md-2 col-form-label" })>
45                     <div class="col-md-10">
46                         <Html.TextBoxFor(m => m.PhoneNumber, new { @class = "form-control" })>
47                         <Html.ValidationMessageFor(m => m.PhoneNumber, "", new { @class = "text-danger" })>
48                     </div>
49             </div>
50
51             <div class="row">
52                 <Html.LabelFor(m => m.Email, new { @class = "col-md-2 col-form-label" })>
53                     <div class="col-md-10">
54                         <Html.TextBoxFor(m => m.Email, new { @class = "form-control" })>
55                         <Html.ValidationMessageFor(m => m.Email, "", new { @class = "text-danger" })>
56                     </div>
57             </div>
58
59             <div class="row">
60                 <Html.LabelFor(m => m.Password, new { @class = "col-md-2 col-form-label" })>
61                     <div class="col-md-10">
62                         <Html.PasswordFor(m => m.Password, new { @class = "form-control" })>
63                     </div>
64             </div>
65
66             <br />
67             <div class="row">
68                 <div class="offset-md-2 col-md-10">
69                     <input type="submit" value="Register" class="btn btn-default" />
70                 </div>
71             </div>
72         </main>
73
74         <section Scripts>
75             <Scripts.Render("~/bundles/jqueryval")>
76             <script>
77                 $(document).ready(function () {
78                     // Attach keyup event handlers to the text input fields
79                     $('#FirstName', '#LastName').on('keyup', function () {
80                         var inputValue = $(this).val();
81                     });
82                 });
83             </script>
84         </section>
85     </body>
86 </html>
87
88 %>
```

No issues found

Output

Show output from: Debug
The program '[12128] iisexpress.exe' has exited with code 4294967295 (0xffffffff).

Figure 16. User Registration New View Part 1 (Register.cshtml)

```

37             <input type="password" class="form-control" id="ConfirmPassword" name="ConfirmPassword" />
38             <Html.ValidationMessageFor(m => m.ConfirmPassword, "", new { @class = "text-danger" })>
39         </div>
40
41         <div class="row">
42             <Html.LabelFor(m => m.Email, new { @class = "col-md-2 col-form-label" })>
43                 <div class="col-md-10">
44                     <Html.TextBoxFor(m => m.Email, new { @class = "form-control" })>
45                     <Html.ValidationMessageFor(m => m.Email, "", new { @class = "text-danger" })>
46                 </div>
47             </div>
48
49             <div class="row">
50                 <Html.LabelFor(m => m.Password, new { @class = "col-md-2 col-form-label" })>
51                     <div class="col-md-10">
52                         <Html.PasswordFor(m => m.Password, new { @class = "form-control" })>
53                         <Html.ValidationMessageFor(m => m.Password, "", new { @class = "text-danger" })>
54                     </div>
55             </div>
56
57             <div class="row">
58                 <div class="offset-md-2 col-md-10">
59                     <input type="submit" value="Register" class="btn btn-default" />
60                 </div>
61             </div>
62
63             <br />
64             <div class="row">
65                 <div class="offset-md-2 col-md-10">
66                     <input type="submit" value="Register" class="btn btn-default" />
67                 </div>
68             </div>
69
70         </main>
71
72         <section Scripts>
73             <Scripts.Render("~/bundles/jqueryval")>
74             <script>
75                 $(document).ready(function () {
76                     // Attach keyup event handlers to the text input fields
77                     $('#FirstName', '#LastName').on('keyup', function () {
78                         var inputValue = $(this).val();
79                     });
80                 });
81             </script>
82         </section>
83     </body>
84 </html>
85
86 %>
```

No issues found

Output

Show output from: Debug
The program '[12128] iisexpress.exe' has exited with code 4294967295 (0xffffffff).

Figure 17. User Registration New View Part 2 (Register.cshtml)

```

49 <div class="col-md-10">
50   <div class="col-md-2 col-form-label">
51     @Html.LabelFor(m => m.Email, new { @class = "col-md-2 col-form-label" })
52   </div>
53   <div class="col-md-8">
54     @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })
55   </div>
56 </div>
57 <div class="col-md-10">
58   <div class="col-md-2 col-form-label">
59     @Html.LabelFor(m => m.Password, new { @class = "col-md-2 col-form-label" })
60   </div>
61   <div class="col-md-8">
62     @Html.PasswordFor(m => m.Password, new { @class = "form-control" })
63   </div>
64 </div>
65 <br />
66 <div class="row">
67   <div class="col-md-10">
68     <div class="offset-md-2 col-md-10">
69       <input type="submit" value="Register" class="btn btn-default" />
70     </div>
71   </div>
72 </div>
73 </main>
74
75 <section Scripts>
76   <Scripts.Render("~/bundles/jqueryval")>
77     <script>
78       $(document).ready(function () {
79         // Attach keyup event handlers to the text input fields
80         $('#firstname, #lastname').on('keyup', function () {
81           var inputValue = $(this).val();
82           // Remove characters that do not match [\w\d]
83           inputValue = inputValue.replace(/[^{\w\ds}]/g, '');
84           // Update the input value
85           $(this).val(inputValue);
86         });
87       });
88     </script>
89   </Scripts>

```

No issues found

Show output from: Debug
The program '[12128] iisexpress.exe' has exited with code 4294967295.

Figure 18. User Registration New View Part 3 – Javascript Validation as Mentioned in the Note Above (Register.cshtml)

8. Usability Design Review

The usability evaluation is based on Donald Norman principles.

Visibility

Visibility in design refers to making important elements, controls, and information easily visible and apparent to users (Norman, 1988; Eduative, 2023). In this web application, there will be three roles and each role will have their respective visible elements, controls, and information. In addition, “Administrator” role will have more visibility than “Footballer” and “Doctor” roles. For example, “Administrator” role can add, delete, edit, and view details of MRI services whereas “Footballer” and “Doctor” roles can only view the listed MRI services.

MRI Service List				Cost	
				Cost	
Service Name	Description	Cost			
Ankle MRI	MRI of the ankle for sports injuries	325.00			

Figure 19. Visibility Principle (Norman)

Feedback

Feedback is about providing users with clear and timely information about the results of their actions (Norman, 1988; Educative, 2023). Whenever an administrator or doctor managed to send either email with attachment (BR (D.2)) or bulk emails (BR (E.1)), there will be a message of “Email sent successfully!” as shown in Figure 20 below.

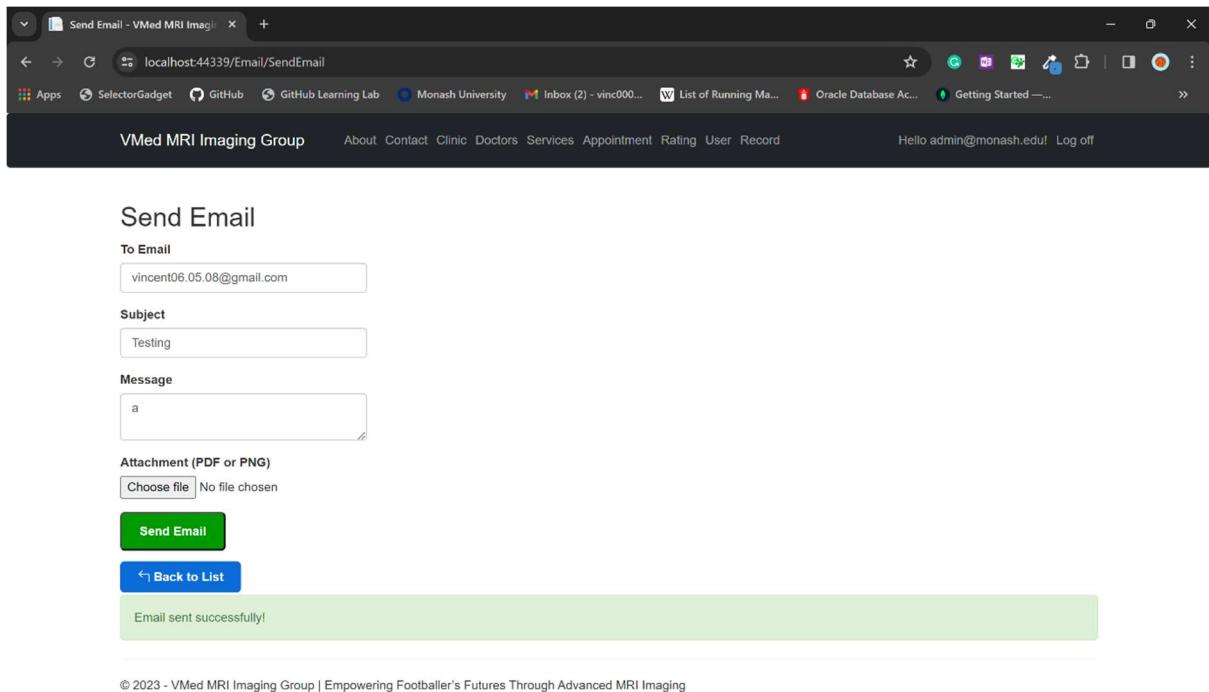


Figure 20. Feedback Principle (Norman)

Affordance

Affordance refers to the perceived or intuitive properties of an object or interface that suggest how it should be used (Norman, 1988; Educative, 2023). All the clickable buttons will be designed in such a way that can be perceived by users such as “Book Appointment” button in the Home page.

We're open and welcoming patients

VMed MRI Imaging Group facilities located throughout Victoria, Western Australia, and South Australia - We are dedicated to supporting your medical imaging needs.

[Book Appointment >](#) [Location >](#)

Our Doctors

VMed MRI Imaging Group is dedicated to providing high-quality MRI imaging services for healthcare professionals and patients.

[Learn more »](#)

Our Services

Discover our wide range of MRI imaging services, including advanced diagnostic imaging and expert radiology consultations.

[Learn more »](#)

Contact Us

Contact us today to schedule an appointment or inquire about our imaging solutions and services.

[Contact »](#)

© 2023 - VMed MRI Imaging Group | Empowering Footballer's Futures Through Advanced MRI Imaging

Figure 21. "Book Appointment" Button Satisfies Affordance Principle (Norman)

Constraint

Constraints are limitations or restrictions placed on a design to prevent users from making errors or taking unintended actions (Norman, 1988; Educative, 2023). As briefly discussed in “Visibility” section, the “Footballer” role will have a lot of constraint especially in terms of CRUD activity within the web application.

MRI Service List			
Add Service Search: <input type="text"/>			
Service Name	Description	Cost	
Ankle MRI	MRI of the ankle for sports injuries	325.00	

Figure 22. Administrator Screen: Constraint Principle (Norman) Part 1

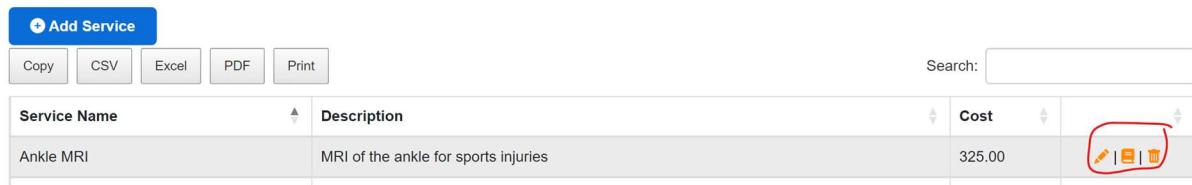
MRI Service List			
Search: <input type="text"/>			
Service Name	Description	Cost	
Ankle MRI	MRI of the ankle for sports injuries	325.00	

Figure 23. Footballer Screen: Constraint Principle (Norman) Part 1

Consistency

Consistency in design ensures that elements, actions, and terminology remain the same or follow predictable patterns across an interface (Norman, 1988; Eduative, 2023). It will be an icon that represents deleting (i.e., bin) and editing (i.e., pencil and paper) when the “Administrator” role wants to perform a specific CRUD activity. All of these icons are consistent with other web applications.

MRI Service List



MRI Service List		
+ Add Service Copy CSV Excel PDF Print Search: <input style="width: 150px; border: 1px solid #ccc; padding: 2px;" type="text"/>		
Service Name	Description	Cost
Ankle MRI	MRI of the ankle for sports injuries	325.00

Figure 24. Consistent Icons - Consistency Principle (Norman)

9. Development Methodology

For this portfolio submission, I used the Code & Fix development methodology. This approach involves an iterative process of coding, testing, and refining to achieve the desired outcomes. To illustrate, in week 8, I just noticed in the studio class that I have done a wrong Database First approach implementation in which I have create two separate databases. Due to this mistake, my web application cannot connect the UserId in the AspNetUsers entity to my Appointment entity. Therefore, I need to redo the SQL query steps to generate the database schema or model.

Furthermore, as the Code & Fix model encourages rapid iterations, I continuously tested and evaluated the code as I progressed, making improvements and adjustments as needed. This iterative process allowed me to identify and address issues early in the development cycle. To illustrate, in the process of completing DataTable and Mapbox, the provided examples in Moodle are outdated in which all CDN versions cannot be used anymore as well as some parts of the provided codes (e.g., location.js in the Map Post Activity). Therefore, I need to continuously evaluate the Javascript code to achieve my desired visual representation of the clinics on a map and interactive DataTables.

Finally, while the Code & Fix model does not emphasize comprehensive documentation upfront, I made sure to document the changes, updates, and rationale behind decisions made

during the development process in a separate Word document, which also help me in handling the versioning of my project as discussed in the next section (available upon request).

10. Versioning

For this project, I utilized Git as my version control system. Git is a widely recognized and versatile tool for tracking changes in code and collaborating with others. Throughout the development process, I regularly committed changes to the Git repository. Each commit was accompanied by a descriptive message explaining the purpose and context of the modifications made which allows for a clear and organized version history.

Proof of using Git:

Git URL: https://github.com/vinctechgeek/FIT5032_VMedProject

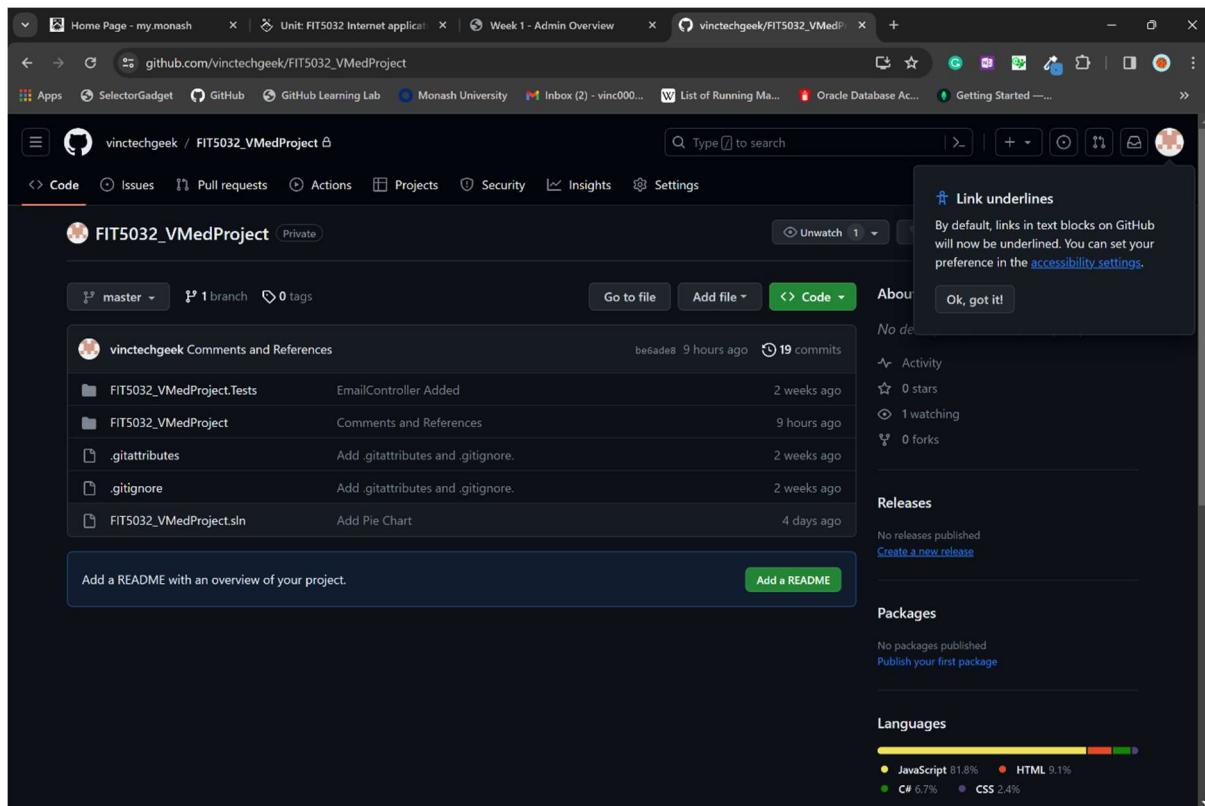


Figure 25. Proof of using Git Screenshot

The second approach is to store the foundation codes in a separate location. To illustrate, I have a folder called Database materials to store all the SQL queries to generate my whole model. The schema SQL query can be observed in section 4 above.

Proof:

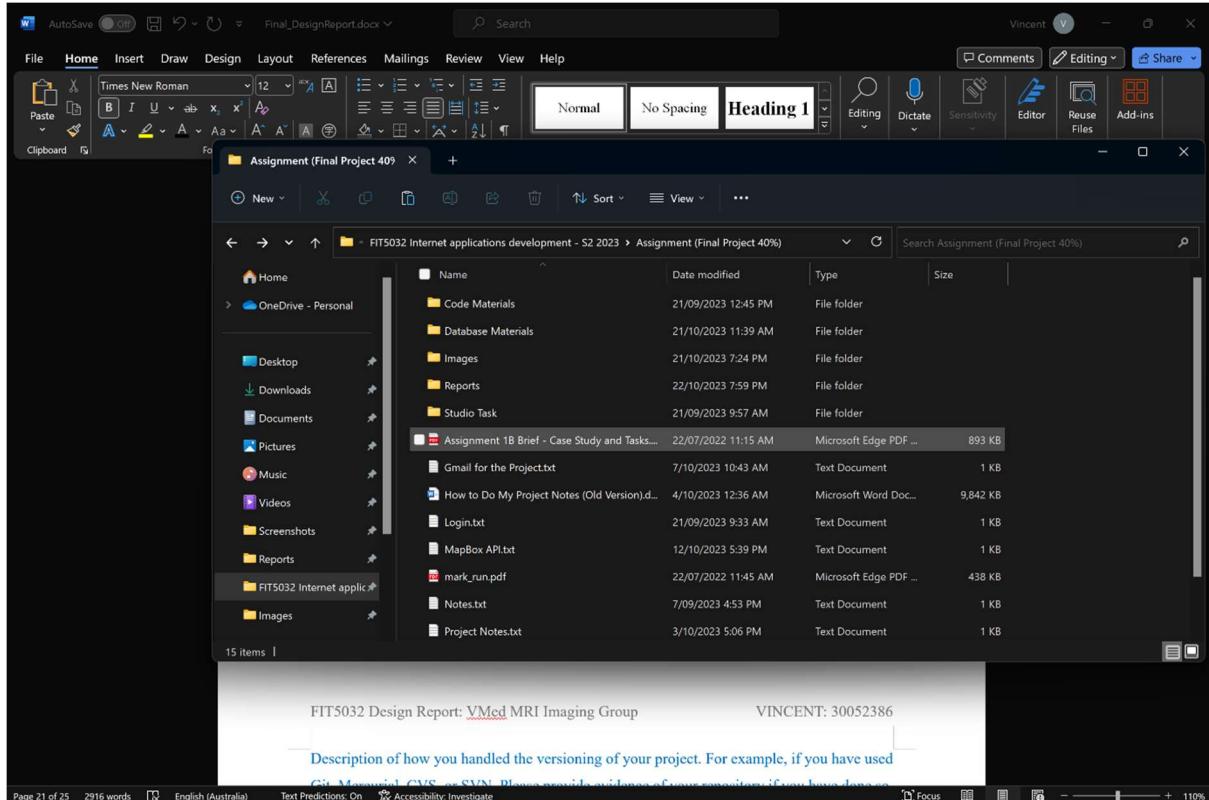


Figure 26. Foundation Codes Local Repository Screenshot

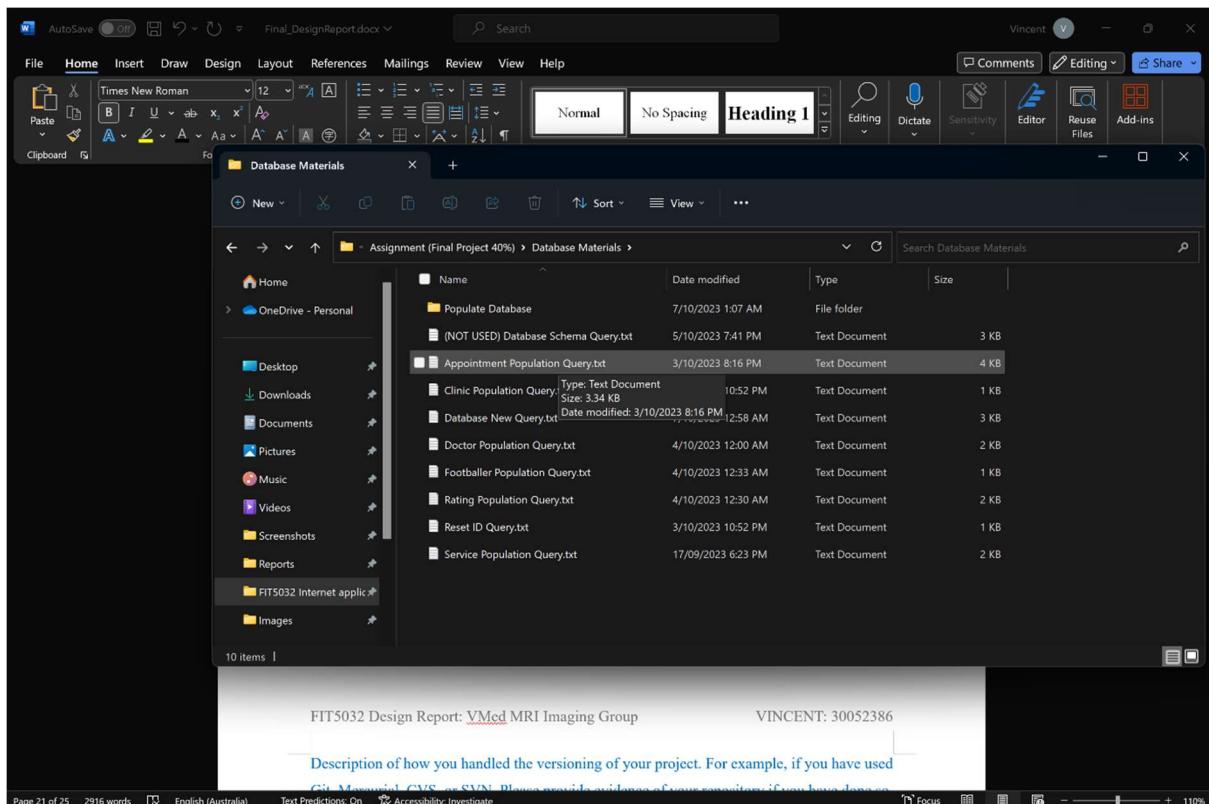


Figure 27. Database Materials Folder Screenshot

Finally, as discussed in previous section, I also created a Word document to document the changes, updates, and rationale behind decisions made.

Proof:

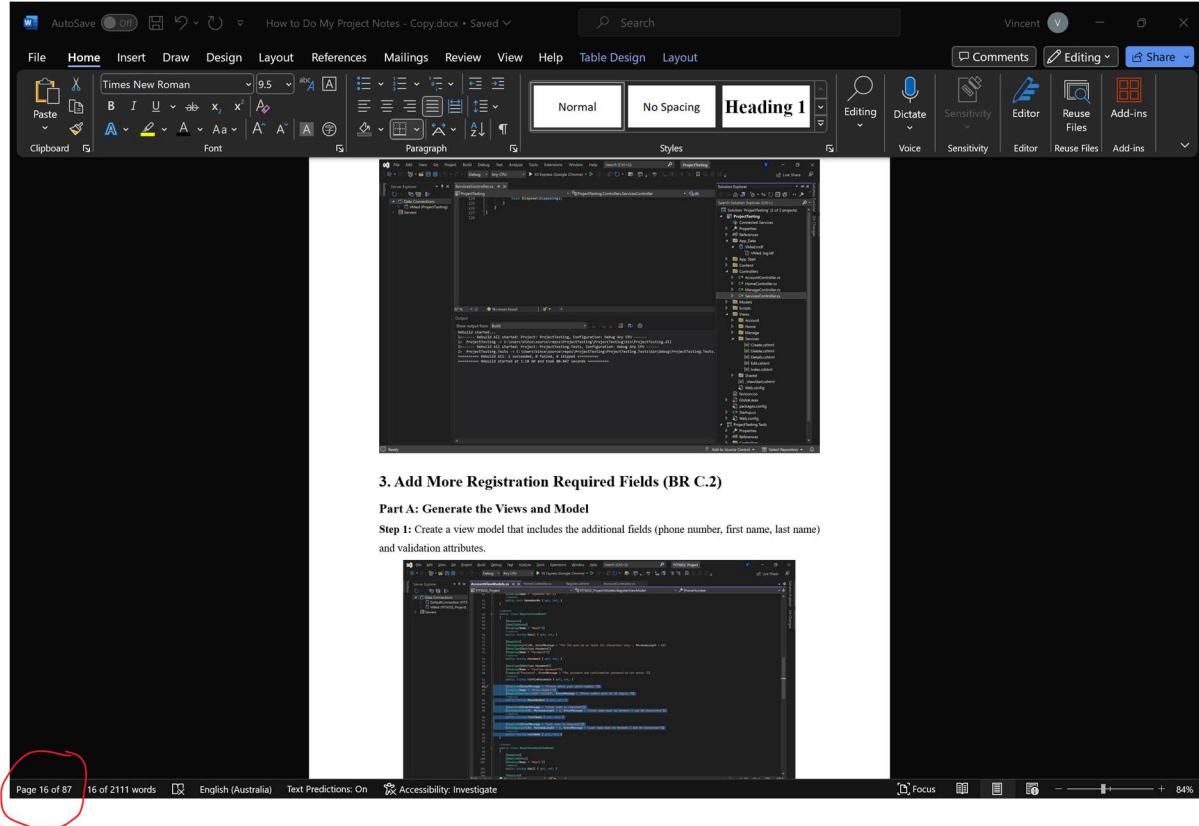


Figure 28. Word Document Notes Screenshot

11. Innovation and Research

User Story: [Enhancement of BR (D.3) Rating]

As a patient, I want to provide a review of a doctor's MRI service from my personal experience, so that I can help the community in making informed decisions about their choice of a doctor to consult.

Description:

For the implementation of the new web application, a significant innovation aimed at improving the user experience centres around the enhancement of the "Doctor MRI Service Reviews" feature (by enhancing BR (D.3) Rating). This enhancement was designed to encourage patients to provide comprehensive and insightful reviews of a doctor's MRI service based on their personal experiences. The primary goal of this innovation is to facilitate

informed decision-making within the community regarding the selection of a doctor to consult for MRI services.

Implementation:

Git URL: https://github.com/vinctechgeek/FIT5032_VMedProject

Final Presentation:

Full Name	Clinic Name	Specialization	Average Rating
Christopher White	VMed Richmond Family Clinic	Gynecologist	2.25 ★
David Jones	VMed Fitzroy Community Health	Pediatrician	4.13 ★
Emily Brown	VMed St Kilda Medical Group	Gynecologist	2.50 ★
John Smith	VMed Victoria Medical Center	Cardiologist	3.40 ★
Laura Martinez	VMed Victoria Medical Center	Orthopedic Surgeon	3.70 ★
Michael Williams	VMed Southbank Health Hub	Orthopedic Surgeon	4.07 ★
Sarah Johnson	VMed Richmond Family Clinic	Dermatologist	2.90 ★
Thomas Muller	VMed Richmond Family Clinic	Pediatrician	2.35 ★

Showing 1 to 8 of 8 entries

© 2023 - VMed MRI Imaging Group

Figure 29. Review Implementation Screenshot [Part 1]

Specific Doctor Ratings

Full Name	RatingValue	Comment
Christopher White	1.50 ★	Terrible MRI scan, avoid this doctor.
Christopher White	3.00 ★	Not recommended for MRI scans.

Showing 1 to 2 of 2 entries

[← Back to List](#)

© 2023 - VMed MRI Imaging Group

Figure 30. Review Implementation Screenshot [Part 2]

Controller: Full Implementation Refer to DoctorsController.cs and RatingsController.cs in the Git Repository.

```

using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using FIT5032_VMedProject.Models;

namespace FIT5032_VMedProject.Controllers
{
    public class RatingsController : Controller
    {
        private Entities2 db = new Entities2();

        // GET: Ratings
        public ActionResult Index()
        {
            var ratings = db.Ratings.Include(r => r.Doctor);
            return View(ratings.ToList());
        }

        // GET: Ratings/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Rating rating = db.Ratings.Find(id);
            if (rating == null)
            {
                return HttpNotFound();
            }
            return View(rating);
        }

        // GET: Ratings/ViewRatingsDetails/5
        public ActionResult ViewRatingsDetails(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Rating rating = db.Ratings.Find(id);
            if (rating == null)
                return HttpNotFound();
            return View(rating);
        }
    }
}

```

Figure 31. RatingsController.cs Screenshot [Part 1]

```

    // GET: Ratings/ViewRatingsDetails/{id}
    public ActionResult ViewRatingsDetails(int? id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        Rating rating = db.Ratings.Find(id);
        if (rating == null)
        {
            return HttpNotFound();
        }
        return View(rating);
    }

    // GET: AllRatings
    public ActionResult AllRatings()
    {
        // Redirect to the "Index" action in the "DoctorsController"
        return RedirectToAction("Index", "Doctors");
    }

    // GET: Ratings of a specific doctor
    public ActionResult ViewRatings(int doctorId)
    {
        // Retrieve the ratings for the specified doctorId
        var ratings = db.Ratings.Where(r => r.DoctorID == doctorId).ToList();
        // You can also retrieve the doctor's information here if needed
        return View(ratings);
    }

    // GET: Ratings/Create
    public ActionResult Create()
    {
        ViewBag.DoctorID = new SelectList(db.Doctors, "DoctorID", "FirstName");
        return View();
    }

    // POST: Ratings/Create
    // To protect from overposting attacks, enable the specific properties you want to bind to, for
    // more details see https://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Create([Bind(Include = "RatingID,DoctorID,RatingValue,Comment")] Rating rating)
    {
        if (ModelState.IsValid)
        {
            db.Ratings.Add(rating);
            db.SaveChanges();
            return RedirectToAction("Index");
        }
        ViewBag.DoctorID = new SelectList(db.Doctors, "DoctorID", "FirstName", rating.DoctorID);
        return View(rating);
    }

    // GET: Ratings/ViewRatingsCreate/{doctorId}
    public ActionResult ViewRatingsCreate(int? doctorId)
    {
        ViewBag.DoctorID = new SelectList(db.Doctors, "DoctorID", "FirstName");
        // Set the doctorId in ViewBag
        ViewBag.SelectedDoctorID = doctorId;
        return View();
    }

    // POST: Ratings/ViewRatingsCreate
    // To protect from overposting attacks, enable the specific properties you want to bind to, for
    // more details see https://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult ViewRatingsCreate([Bind(Include = "RatingID,DoctorID,RatingValue,Comment")] Rating rating)
    {
        // Access the doctorId from the rating object
        int doctorId = rating.DoctorID;

        if (ModelState.IsValid)
        {
            db.Ratings.Add(rating);
            db.SaveChanges();
            return RedirectToAction("ViewRatings", new { doctorId = doctorId });
        }
    }

```

Figure 32. RatingsController.cs Screenshot [Part 2]

```

    // POST: Ratings/Create
    // To protect from overposting attacks, enable the specific properties you want to bind to, for
    // more details see https://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Create([Bind(Include = "RatingID,DoctorID,RatingValue,Comment")] Rating rating)
    {
        if (ModelState.IsValid)
        {
            db.Ratings.Add(rating);
            db.SaveChanges();
            return RedirectToAction("Index");
        }
        ViewBag.DoctorID = new SelectList(db.Doctors, "DoctorID", "FirstName", rating.DoctorID);
        return View(rating);
    }

    // GET: Ratings/ViewRatingsCreate/{doctorId}
    public ActionResult ViewRatingsCreate(int? doctorId)
    {
        ViewBag.DoctorID = new SelectList(db.Doctors, "DoctorID", "FirstName");
        // Set the doctorId in ViewBag
        ViewBag.SelectedDoctorID = doctorId;
        return View();
    }

    // POST: Ratings/ViewRatingsCreate
    // To protect from overposting attacks, enable the specific properties you want to bind to, for
    // more details see https://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult ViewRatingsCreate([Bind(Include = "RatingID,DoctorID,RatingValue,Comment")] Rating rating)
    {
        // Access the doctorId from the rating object
        int doctorId = rating.DoctorID;

        if (ModelState.IsValid)
        {
            db.Ratings.Add(rating);
            db.SaveChanges();
            return RedirectToAction("ViewRatings", new { doctorId = doctorId });
        }
    }

```

Figure 33. RatingsController.cs Screenshot [Part 3]

```

104     }
105
106     // POST: Ratings/ViewRatingsCreate
107     // To protect from overposting attacks, enable the specific properties you want to bind to, for
108     // more details see https://go.microsoft.com/fwlink/?LinkId=317598.
109     [HttpPost]
110     [ValidateAntiForgeryToken]
111     public ActionResult ViewRatingsCreate([Bind(Include = "RatingID,DoctorID,RatingValue,Comment")] Rating rating)
112     {
113         // Access the doctorId from the rating object
114         int doctorId = rating.DoctorID;
115
116         if (ModelState.IsValid)
117         {
118             db.Ratings.Add(rating);
119             db.SaveChanges();
120             return RedirectToAction("ViewRatings", new { doctorId = doctorId });
121         }
122
123         ViewBag.DoctorID = new SelectList(db.Doctors, "DoctorID", "FirstName", rating.DoctorID);
124         return View(rating);
125     }
126
127     // GET: Ratings/Edit/5
128     public ActionResult Edit(int? id)
129     {
130         if (id == null)
131         {
132             return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
133         }
134         Rating rating = db.Ratings.Find(id);
135         if (rating == null)
136         {
137             return HttpNotFound();
138         }
139         ViewBag.DoctorID = new SelectList(db.Doctors, "DoctorID", "FirstName", rating.DoctorID);
140         return View(rating);
141     }
142
143     // POST: Ratings/Edit/5
144     // To protect from overposting attacks, enable the specific properties you want to bind to, for
145     // more details see https://go.microsoft.com/fwlink/?LinkId=317598.
146     [HttpPost]
147     [ValidateAntiForgeryToken]
148     public ActionResult Edit([Bind(Include = "RatingID,DoctorID,RatingValue,Comment")] Rating rating)
149     {
150         if (ModelState.IsValid)
151         {
152             db.Entry(rating).State = EntityState.Modified;
153             db.SaveChanges();
154             return RedirectToAction("Index");
155         }
156         ViewBag.DoctorID = new SelectList(db.Doctors, "DoctorID", "FirstName", rating.DoctorID);
157         return View(rating);
158     }
159
160     // GET: Ratings/ViewRatingsEdit/5
161     public ActionResult ViewRatingsEdit(int? id)
162     {
163         if (id == null)
164         {
165             return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
166         }
167         Rating rating = db.Ratings.Find(id);
168         if (rating == null)
169         {
170             return HttpNotFound();
171         }
172         ViewBag.DoctorID = new SelectList(db.Doctors, "DoctorID", "FirstName", rating.DoctorID);
173         return View(rating);
174     }
175
176     // POST: Ratings/ViewRatingsEdit/5
177     // To protect from overposting attacks, enable the specific properties you want to bind to, for
178     // more details see https://go.microsoft.com/fwlink/?LinkId=317598.
179     [HttpPost]
180     [ValidateAntiForgeryToken]
181     public ActionResult ViewRatingsEdit([Bind(Include = "RatingID,DoctorID,RatingValue,Comment")] Rating rating)
182     {
183         if (ModelState.IsValid)
184         {
185             db.Entry(rating).State = EntityState.Modified;
186             db.SaveChanges();
187             return RedirectToAction("Index");
188         }
189     }

```

Figure 34. RatingsController.cs Screenshot [Part 4]

```

141     }
142
143     // POST: Ratings/Edit/5
144     // To protect from overposting attacks, enable the specific properties you want to bind to, for
145     // more details see https://go.microsoft.com/fwlink/?LinkId=317598.
146     [HttpPost]
147     [ValidateAntiForgeryToken]
148     public ActionResult Edit([Bind(Include = "RatingID,DoctorID,RatingValue,Comment")] Rating rating)
149     {
150         if (ModelState.IsValid)
151         {
152             db.Entry(rating).State = EntityState.Modified;
153             db.SaveChanges();
154             return RedirectToAction("Index");
155         }
156         ViewBag.DoctorID = new SelectList(db.Doctors, "DoctorID", "FirstName", rating.DoctorID);
157         return View(rating);
158     }
159
160     // GET: Ratings/ViewRatingsEdit/5
161     public ActionResult ViewRatingsEdit(int? id)
162     {
163         if (id == null)
164         {
165             return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
166         }
167         Rating rating = db.Ratings.Find(id);
168         if (rating == null)
169         {
170             return HttpNotFound();
171         }
172         ViewBag.DoctorID = new SelectList(db.Doctors, "DoctorID", "FirstName", rating.DoctorID);
173         return View(rating);
174     }
175
176     // POST: Ratings/ViewRatingsEdit/5
177     // To protect from overposting attacks, enable the specific properties you want to bind to, for
178     // more details see https://go.microsoft.com/fwlink/?LinkId=317598.
179     [HttpPost]
180     [ValidateAntiForgeryToken]
181     public ActionResult ViewRatingsEdit([Bind(Include = "RatingID,DoctorID,RatingValue,Comment")] Rating rating)
182     {
183         if (ModelState.IsValid)
184         {
185             db.Entry(rating).State = EntityState.Modified;
186             db.SaveChanges();
187             return RedirectToAction("Index");
188         }
189     }

```

Figure 35. RatingsController.cs Screenshot [Part 5]

The screenshot shows the Microsoft Visual Studio IDE interface. The top menu bar includes File, Edit, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a Search bar. The left sidebar has a 'Toolbox' tab and a 'Server Explorer' tab with sections for Data Connections, Entities2 (FIT5032_VMedProject), and Servers. The main code editor window displays the 'RatingsController.cs' file under the 'FIT5032_VMedProject.Controllers' namespace. The file contains C# code for handling rating operations, including methods for editing and deleting ratings. The bottom status bar shows '89 %' completion, 'No issues found', and navigation keys like Ln: 18, Ch: 36, SPC, and CRLF.

```
// POST: Ratings/ViewRatingsEdit/5
// To protect from overposting attacks, enable the specific properties you want to bind to, for
// more details see https://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult ViewRatingsEdit([Bind(Include = "RatingID,DoctorID,RatingValue,Comment")] Rating rating)
{
    if (ModelState.IsValid)
    {
        db.Entry(rating).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("ViewRatings", new { doctorId = rating.DoctorID });
    }
    ViewBag.DoctorID = new SelectList(db.Doctors, "DoctorID", "FirstName", rating.DoctorID);
    return View(rating);
}

// GET: Ratings/Delete/5
public ActionResult Delete(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Rating rating = db.Ratings.Find(id);
    if (rating == null)
    {
        return HttpNotFound();
    }
    return View(rating);
}

// POST: Ratings/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public ActionResult DeleteConfirmed(int id)
{
    Rating rating = db.Ratings.Find(id);
    db.Ratings.Remove(rating);
    db.SaveChanges();
    return RedirectToAction("Index");
}
```

Figure 36. RatingsController.cs Screenshot [Part 6]

The screenshot shows the Microsoft Visual Studio IDE interface. The top menu bar includes File, Edit, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and Search (Ctrl+Q). The toolbar contains icons for file operations like Open, Save, and Print. The title bar displays "FIT5032_VMedProject". The left sidebar has a "Server Explorer" section with "Data Connections" and "Entities2 (FIT5032_VMedProj)" listed under "Servers". The main code editor window shows the "RatingsController.cs" file with the following content:

```
// POST: Ratings/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public ActionResult DeleteConfirmed(int id)
{
    Rating rating = db.Ratings.Find(id);
    db.Ratings.Remove(rating);
    db.SaveChanges();
    return RedirectToAction("Index");
}

// GET: Ratings/ViewRatingDelete/5
public ActionResult ViewRatingDelete(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Rating rating = db.Ratings.Find(id);
    if (rating == null)
    {
        return HttpNotFound();
    }
    return View(rating);
}

// POST: Ratings/ViewRatingDeleteConfirmed/5
[HttpPost, ActionName("ViewRatingDelete")]
[ValidateAntiForgeryToken]
public ActionResult ViewRatingDeleteConfirmed(int id)
{
    Rating rating = db.Ratings.Find(id);
    db.Ratings.Remove(rating);
    db.SaveChanges();
    return RedirectToAction("ViewRating", new { doctorId = rating.DoctorID });
}

protected override void Dispose(bool disposing)
{
    if (disposing)
    {

```

Figure 37. RatingsController.cs Screenshot [Part 7]

```

223     {
224         return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
225     }
226     Rating rating = db.Ratings.Find(id);
227     if (rating == null)
228     {
229         return HttpNotFound();
230     }
231     return View(rating);
232 }
233
234 // POST: Ratings/ViewRatingsDeleteConfirmed/
235 [HttpPost, ActionName("ViewRatingsDelete")]
236 [ValidateAntiForgeryToken]
237 public ActionResult ViewRatingsDeleteConfirmed(int id)
238 {
239     Rating rating = db.Ratings.Find(id);
240     db.Ratings.Remove(rating);
241     db.SaveChanges();
242     return RedirectToAction("ViewRatings", new { doctorId = rating.DoctorID });
243 }
244
245 protected override void Dispose(bool disposing)
246 {
247     if (disposing)
248     {
249         db.Dispose();
250     }
251     base.Dispose(disposing);
252 }
253 }
254
255

```

Figure 38. RatingsController.cs Screenshot [Part 8]

```

15
16
17
18     private Entities2 db = new Entities2();
19
20     // GET: Doctors
21     public ActionResult Index()
22     {
23         var doctors = db.Doctors.Include(d => d.Clinic);
24         return View(doctors.ToList());
25     }
26
27     // GET: Doctors/Details/
28     public ActionResult Details(int? id)
29     {
30         if (id == null)
31         {
32             return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
33         }
34         Doctor doctor = db.Doctors.Find(id);
35         if (doctor == null)
36         {
37             return HttpNotFound();
38         }
39         return View(doctor);
40     }
41
42     // GET: Doctors/DoctorRatings/
43     public ActionResult DoctorRatings(int id)
44     {
45         // Redirect to the "ViewRatings" action in the "RatingsController"
46         return RedirectToAction("ViewRatings", "Ratings", new { doctorId = id });
47     }
48
49     // GET: Doctors/Create
50     public ActionResult Create()
51     {
52         ViewBag.ClinicID = new SelectList(db.Clinics, "ClinicID", "ClinicName");
53         return View();
54     }
55
56     // POST: Doctors/Create
57     // To protect from overposting attacks, enable the specific properties you want to bind to, for
58     // more details see https://go.microsoft.com/fwlink/?LinkId=317598.
59     [HttpPost]
60

```

Figure 39. Redirect Action Method from Doctors/Index.cshtml to ViewRatings.cshtml in DoctorsController.cs Screenshot

View: Full Implementation Refer to Views/Doctors/Index.cshtml and Views/Ratings/ViewRatings[Create/Delete/Details/Edit].cshtml in the Git Repository.

```

4 <h2>Specific Doctor Ratings</h2>
5 <br />
6 <p><a href=@Url.Action("ViewRatingsCreate", new { doctorId = Model.First().DoctorID })>Add Rating</a>
7 </p>
8 <table class="table">
9   <thead>
10    <tr>
11      <th>Full Name</th>
12      <th>Rating Value</th>
13      <th>Comment</th>
14    </tr>
15  </thead>
16  <tbody>
17    <foreach var item in Model>
18      <tr>
19          <td>@Html.DisplayFor(modelItem => item.Doctor.FirstName)<br/>@Html.DisplayFor(modelItem => item.Doctor.LastName)</td>
20          <td>@Html.DisplayFor(modelItem => item.RatingValue)<br/><i class="fa fa-star" aria-hidden="true" style="color: gold;"></i><br/>@Html.DisplayFor(modelItem => item.Comment)</td>
21          <td style="text-align: center; vertical-align: middle;">
22              <a href=@Url.Action("ViewRatingsEdit", new { id = item.RatingID })>Edit</a>
23              <a href=@Url.Action("ViewRatingsDetails", new { id = item.RatingID })>Details</a>
24              <a href=@Url.Action("ViewRatingsDelete", new { id = item.RatingID })>Delete</a>
25          </td>
26      </tr>
27    </foreach>
28  </tbody>
29 </table>
30 <br />
31 
```

Figure 40. ViewRatings.cshtml Screenshot [Part 1]

```

20 </thead>
21 <tbody>
22   <foreach var item in Model>
23     <tr>
24       <td>@Html.DisplayFor(modelItem => item.Doctor.FirstName)<br/>@Html.DisplayFor(modelItem => item.Doctor.LastName)</td>
25       <td>@Html.DisplayFor(modelItem => item.RatingValue)<br/><i class="fa fa-star" aria-hidden="true" style="color: gold;"></i><br/>@Html.DisplayFor(modelItem => item.Comment)</td>
26       <td style="text-align: center; vertical-align: middle;">
27           <a href=@Url.Action("ViewRatingsEdit", new { id = item.RatingID })>Edit</a>
28           <a href=@Url.Action("ViewRatingsDetails", new { id = item.RatingID })>Details</a>
29           <a href=@Url.Action("ViewRatingsDelete", new { id = item.RatingID })>Delete</a>
30       </td>
31     </tr>
32   </foreach>
33 </tbody>
34 </table>
35 <link href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css" rel="stylesheet" />
36 <script src="https://code.jquery.com/jquery-3.7.0.js"></script>
37 <script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js"></script>
38 <script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js"></script>
39 <script>
40   $(document).ready(function () {
41     $('.table').DataTable();
42   });
43 </script>
44 <br />
45 
```

Figure 41. ViewRatings.cshtml Screenshot [Part 2]

```
40 <a href="#" style="color: white; text-decoration: none;">
41 |   <i class="bi bi-pencil-fill" aria-hidden="true" style="color: darkorange;"></i>
42 | <a href="#" style="color: white; text-decoration: none;">
43 |   <i class="fa fa-book" aria-hidden="true" style="color: darkorange;"></i>
44 | <a href="#" style="color: white; text-decoration: none;">
45 |   <i class="bi bi-trash-fill" aria-hidden="true" style="color: darkorange;"></i>
46 </td>
47 </tr>
48 </tbody>
49 </table>
50 <link href="https://cdn.datatables.net/1.13.6/css/dataTables.bootstrap5.min.css" rel="stylesheet" />
51
52 @section Scripts {
53
54   <script src="https://code.jquery.com/jquery-3.7.0.js"></script>
55   <script src="https://cdn.datatables.net/1.13.6/js/jquery.dataTables.min.js"></script>
56   <script src="https://cdn.datatables.net/1.13.6/js/dataTables.bootstrap5.min.js"></script>
57   <script>
58     $(document).ready(function () {
59       $('.table').DataTable();
60     });
61   </script>
62
63   <br />
64   <p>
65     <a href="#" class="add-button">
66       <i class="bi bi-arrow-90deg-left" aria-hidden="true"></i> Back to List
67     </a>
68   </p>
69
70
71
72
73
74
75
76
77
78
79
```

Figure 42. ViewRatings.cshtml Screenshot [Part 3]

12. Checklist of Site Functionality

	TICK if complete
1. (Layout Page)	
Good Design	V
Stylesheet	V
JavaScript	V
Menu	V
2. (Home page)	
Design and content	V
Banner Image	V
3. (User Log in)	
Web form and validation controls	V
Formatted data entry display	V
Overall page design	V
4. (Customised Views and Controllers)	
Customised Views	V
Customised Controllers	V
Other customisations	V
5. (Documentation)	
Code Comments	V
Attribution of Source of any code used	V
6 Business Requirements	
BR(A1): for P	V
BR(A2): for P	V
BR(B1): for C to C+	V
BR(B2): for C to C+	V
BR(C1): for C+ to C++	V
BR(C2): for C+ to C++	V
BR(C3): for C+ to C++	V
BR(C4): for C+ to C++	V
BR(D1): for D to D++	V
BR(D2): for D to D++	V
BR(D3): for D to D++	V
BR(D4): for D to D++	V
BR(E1): for HD to HD+	V
BR(E2): for HD to HD+	V
BR(F1): for HD+ to HD++	V
Audit	
No breaking of copyright	

Note: For the computer, OS, and Visual Studio specifications, refer to Appendix B.

References

- Norman, D. A. (1988). *The psychology of everyday things*. Basic books.
- Educative. (2023). *What are Norman's design principles?*
- <https://www.educative.io/answers/what-are-normans-design-principles>

Appendix

Appendix A

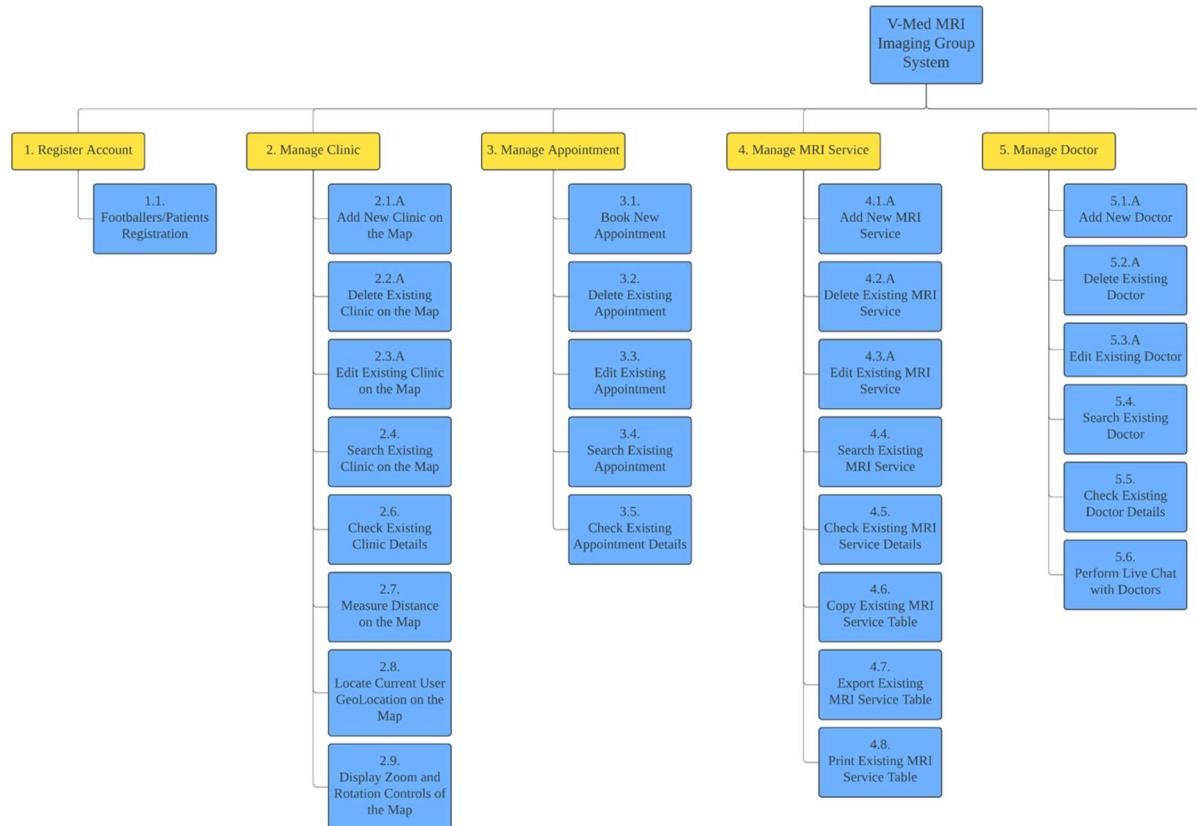


Figure 43. Block/Function Diagram Part 1

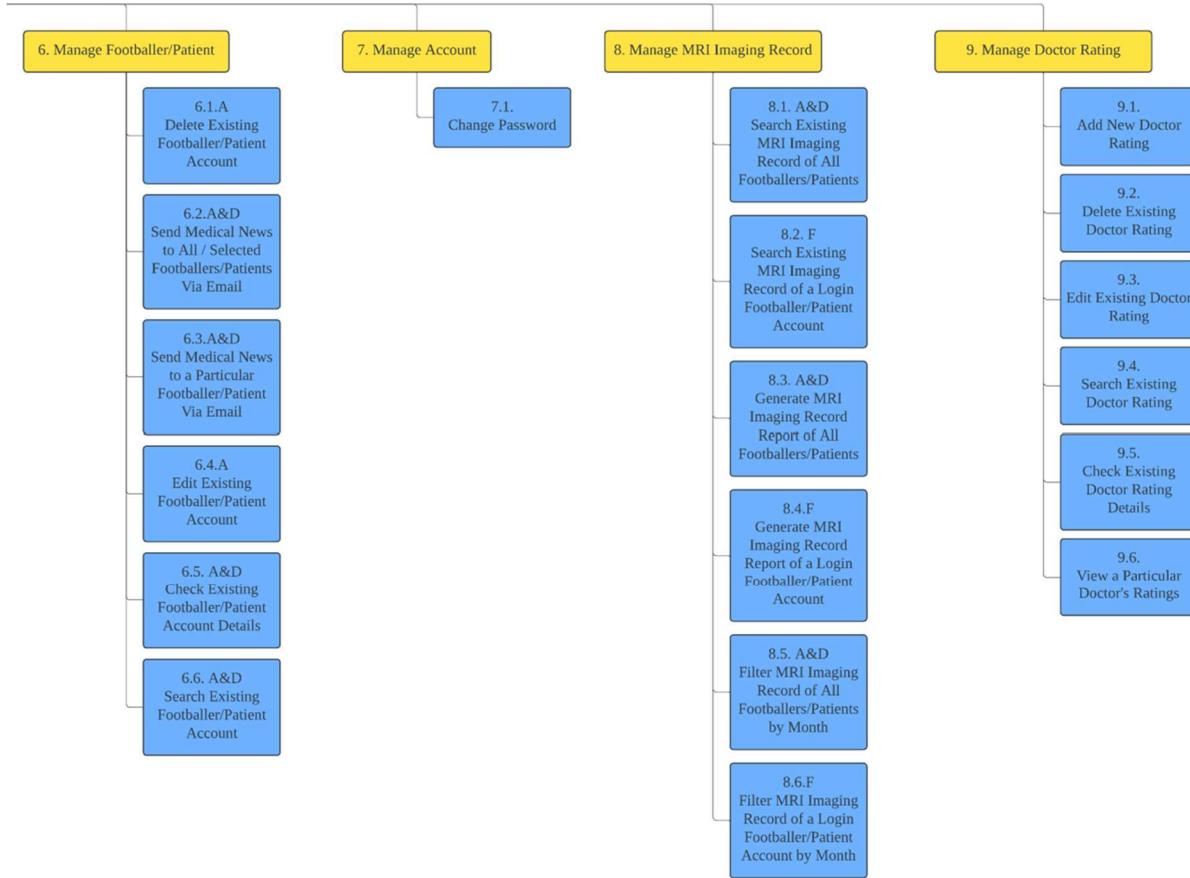


Figure 44. Block/Functional Diagram Part 2

Appendix B

Laptop: Surface Pro 7

OS: Windows 11

Visual Studio:

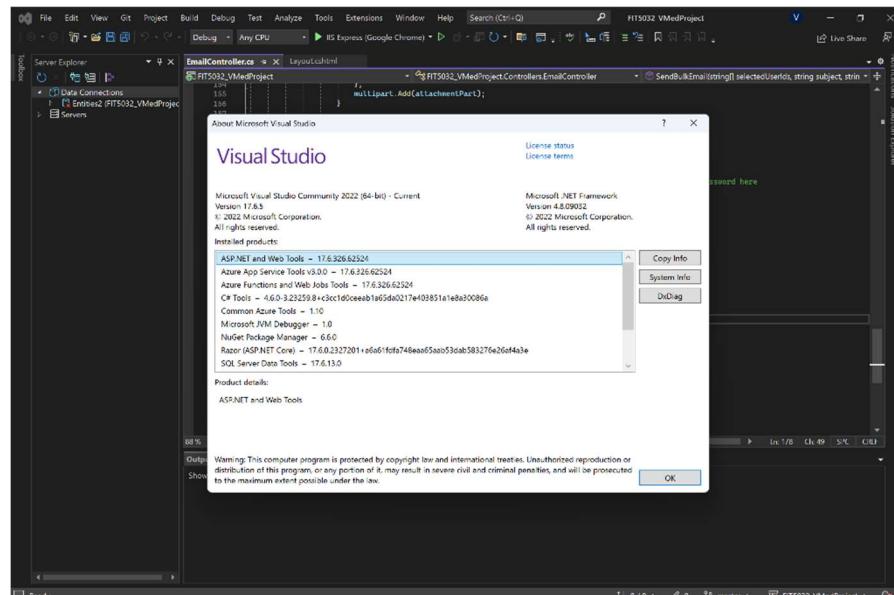


Figure 45. Visual Studio Version Screenshot