MONASH University

FIT5032 Design Report

Major Application Development

{Credit/Distinction/High Distinction}

ScannerPlus

YuxiangZou:33543437

INSTRUCTIONS: Substitute all RED text with your information. DELETE all BLUE instructions before final submission (in PDF format). Feel free to edit the format of the document to improve presentation

Contents **Page**

Your design report must include the following:

Credit Level

1. Web application title and description
2. User stories and a Use case diagram
3. Block/Functional diagram
4. Your selected approach when constructing the application

Additional Distinction Level (the above and the following)

5. Class Diagram or Entity Relation Diagram
6. Mockup prototypes and Implementation
7. Data dictionary
8. Usability Design Review

Additional High Distinction Level (the above and the following)

9. Development Methodology
10. Versioning

11. Checklist of site functionality.

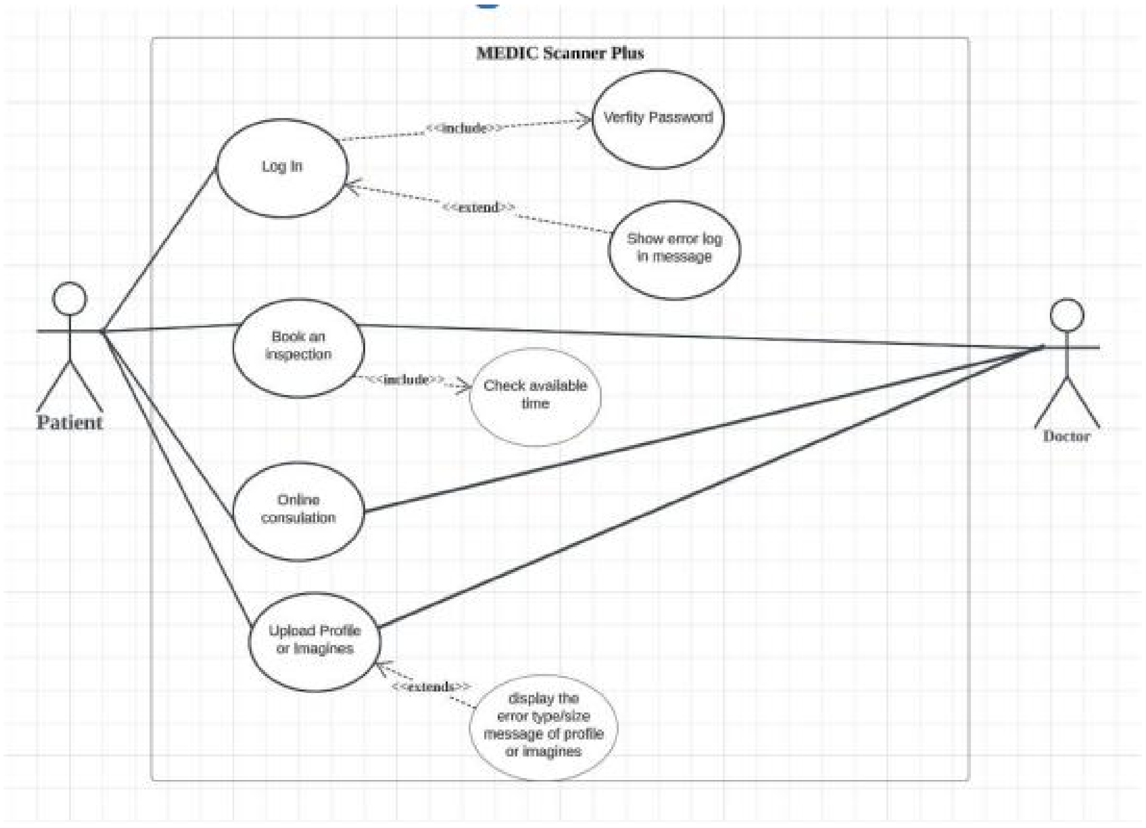
Your design report must include the following (Pass Level and Above):

1. **Web Application Title and Description**

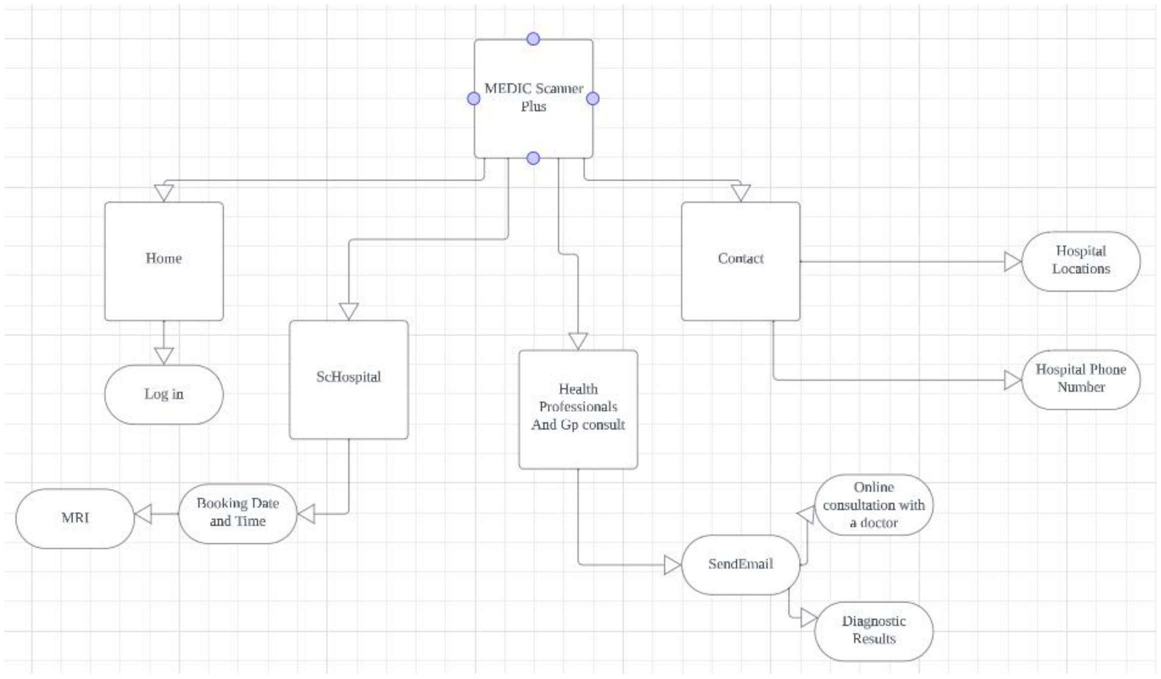
Title：MEDIC Scanner Plus Description:It is a brand-new Internet application program customized for medical imaging service companies. It is designed to provide a seamless medical imaging management and collaboration experience, as well as other business needs such as allowing patients to book time to use medical imaging services.Medical institutions can use the application to centrally manage patients' Imagines data, improve work efficiency, and ensure patient data security and compliance.Physicians can use the app to view, analyze and interpret Medic images for accurate diagnosis and treatment planning.Patients can also make appointments for Imagines examinations on this program and ask doctors about the results of Medical images online.

2. **User stories and Use case diagram**

UserStories:As a patient, I hope to have an online appointment service for medical imaging scans at the hospital, so that I can understand my physical condition in a timely manner.



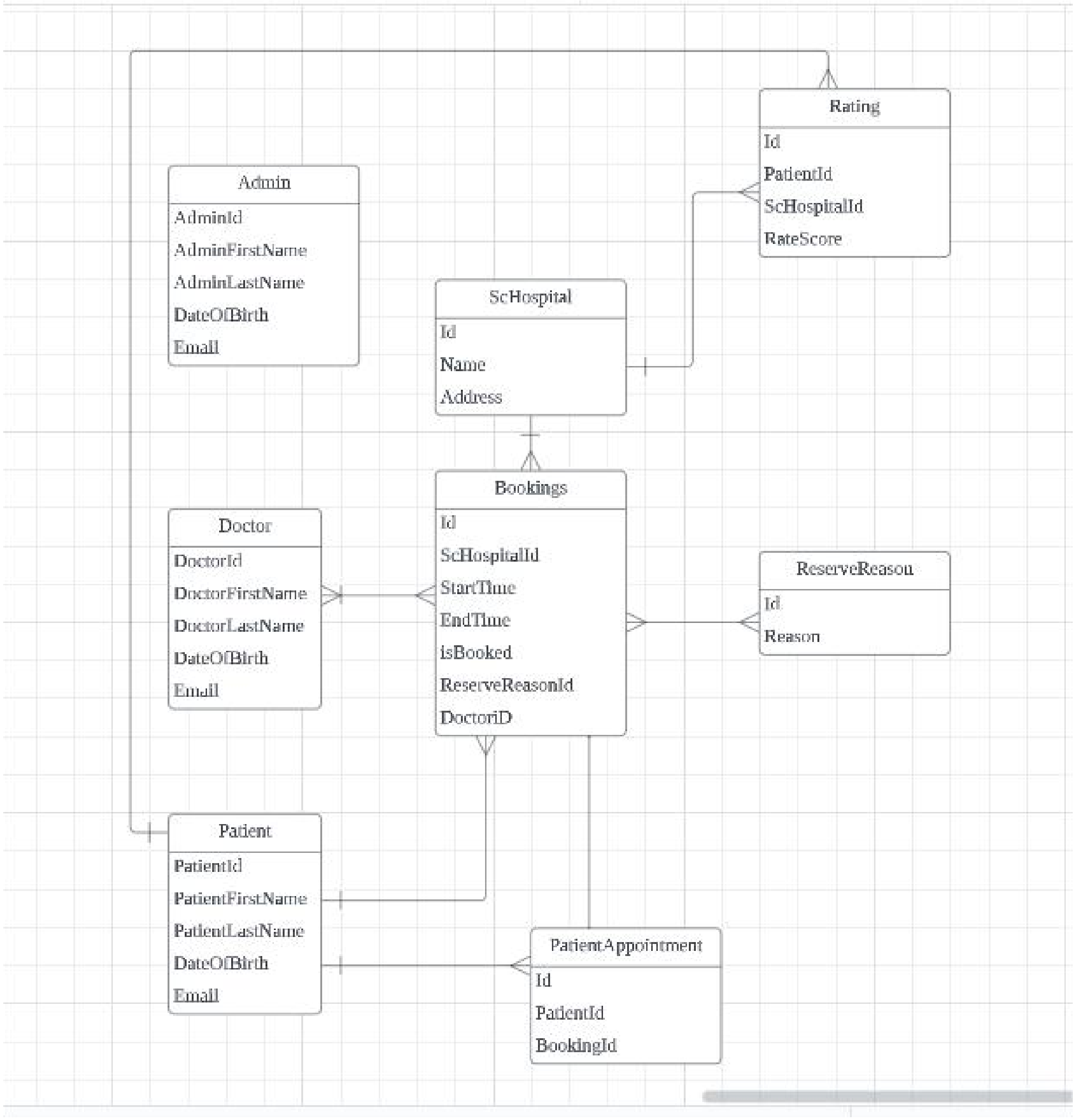
3. **Block/Functional diagram**



4. **Your selected approach when constructing the application.**
   I tried using modelfirst at first, but ended up choosing codefirst

Additional Distinction Level (the above and the following)

5. **Class Diagram or Entity Relation Diagram**



6. **Data dictionary**

AspNetUsers (Users)
Id: String (variable length). The unique identifier for users.
Email: String (variable length). The user's email address.
PasswordHash: String (maximum length). The hashed password value for the user.
UserName: String (variable length). The user's username.
FirstName: String (maximum length). The user's first name.
LastName: String (maximum length). The user's last name.

DateOfBirth: Date. The user's date of birth.

Bookings

Id: Integer. Unique identifier.
ScHospitalId: Integer. Refers to the associated hospital.
StartTime: DateTime. The start time of the appointment.
EndTime: DateTime. The end time of the appointment.
IsBooked: Bit. Indicates whether the appointment is booked.
ReserveReasonId: Integer. Refers to the reason for the reservation.
DoctorID: String (variable length). Refers to the associated doctor.

PatientAppointments

Id: Integer. Unique identifier.
PatientId: String (variable length). Refers to the associated patient.
BookingId: Integer. Refers to the associated appointment.

Ratings

Id: Integer. Unique identifier.
PatientId: String (variable length). Refers to the associated patient.
ScHospitalId: Integer. Refers to the associated hospital.
RateScore: Integer. Represents the patient's rating score.

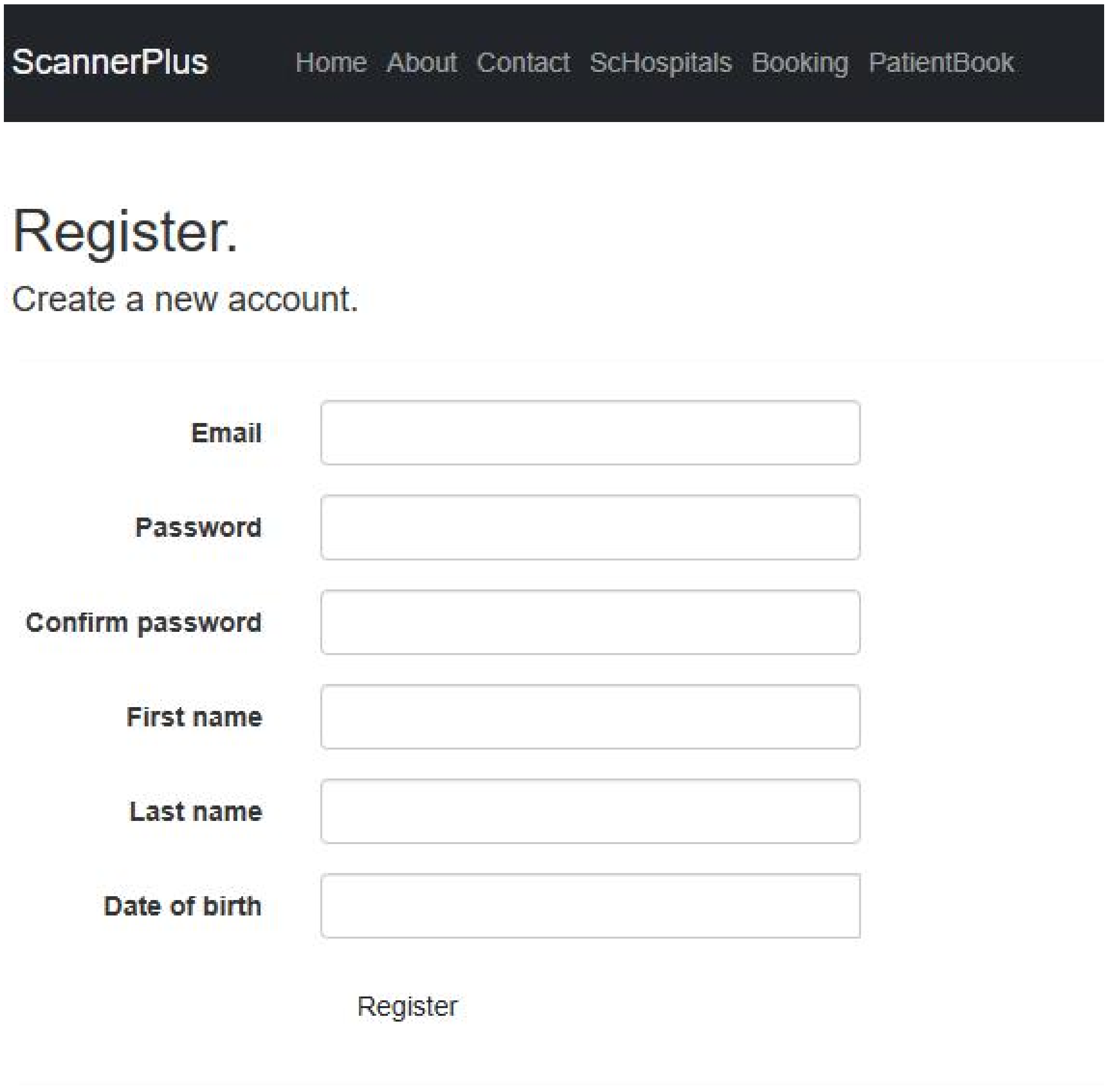
ReserveReasons

Id: Integer. Unique identifier.
Reason: String (maximum length). Description of the reservation reason.

ScHospitals

Id: Integer. Unique identifier.
Name: String (maximum length). The name of the hospital.
Address: String (maximum length). The address of the hospital.

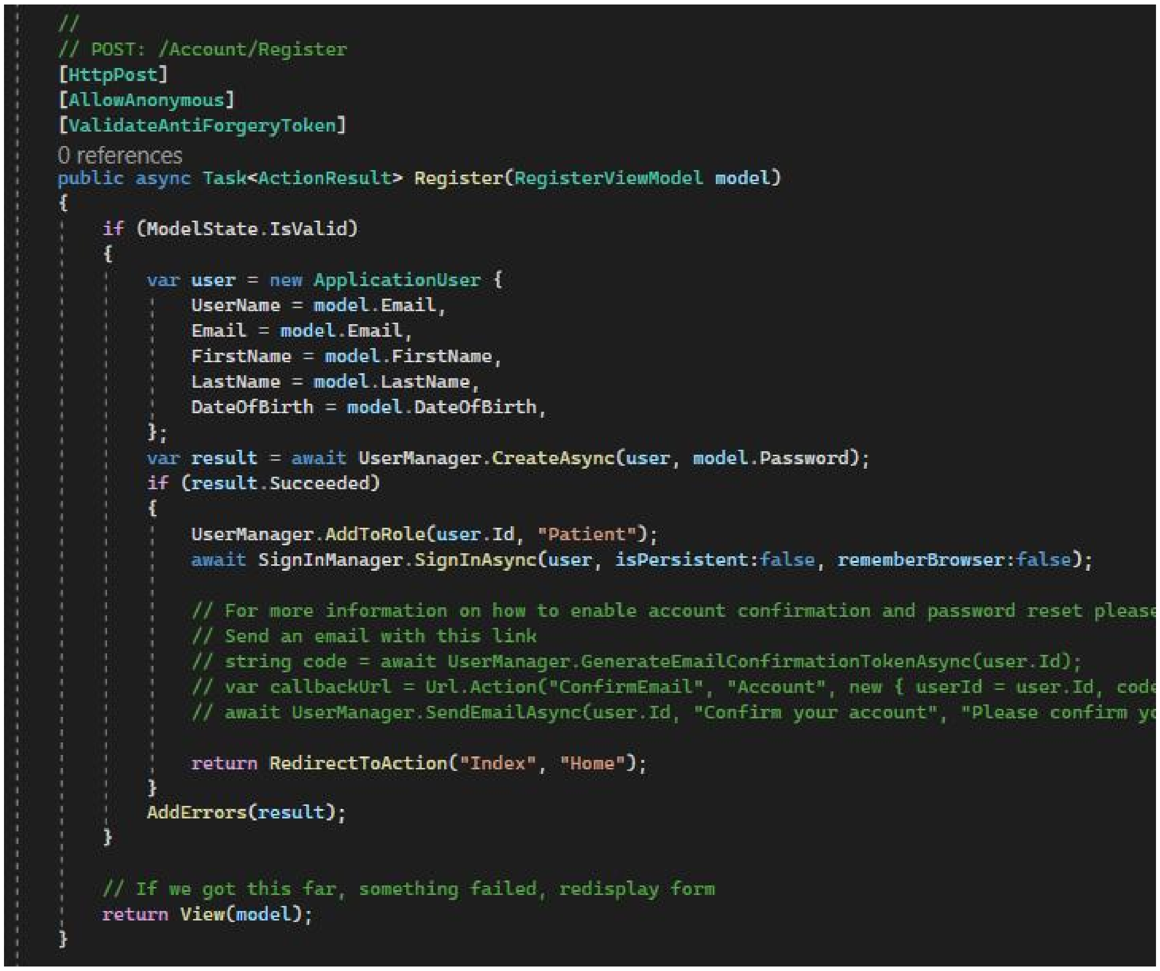
7. Mockup prototypes and implementation with user registration and authentication



Register.

Create a new account.

Email	
Password	
Confirm password	
First name	
Last name	
Date of birth	



```
//
// POST: /Account/Register
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
0 references
public async Task<ActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = new ApplicationUser {
            UserName = model.Email,
            Email = model.Email,
            FirstName = model.FirstName,
            LastName = model.LastName,
            DateOfBirth = model.DateOfBirth,
        };
        var result = await UserManager.CreateAsync(user, model.Password);
        if (result.Succeeded)
        {
            UserManager.AddToRole(user.Id, "Patient");
            await SignInManager.SignInAsync(user, isPersistent:false, rememberBrowser:false);

            // For more information on how to enable account confirmation and password reset please
            // Send an email with this link
            // string code = await UserManager.GenerateEmailConfirmationTokenAsync(user.Id);
            // var callbackUrl = Url.Action("ConfirmEmail", "Account", new { userId = user.Id, code
            // await UserManager.SendEmailAsync(user.Id, "Confirm your account", "Please confirm yo

            return RedirectToAction("Index", "Home");
        }
        AddErrors(result);
    }

    // If we got this far, something failed, redisplay form
    return View(model);
}
```

8. **Usability Design Review**

I follow DonNorman's principles. My web application's interface elements have obvious visibility. Users should be able to immediately understand how to interact with the application.

Check the menus and buttons to achieve the desired operation. By setting up an application feedback mechanism, error messages will be prompted to allow users to perform correct operations. In terms of app functionality, they are intuitive and will reduce user confusion.
I also checked for consistency in the application's fonts, colors, buttons, and other elements to ensure that the overall look and feel of the user interface was coordinated.

9. Checklist of site functionality

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