

Course Submission Cover Sheet

Module: CC6012 Data and Web Application

Deadline:

Module Leader: Rattapol Kasemrat Student ID:
E256406

PLAGIARISM

You are reminded that there exist regulations concerning plagiarism. Extracts from these regulations are printed below. Please sign below to say that you have read and understand these extracts:

Extracts from University *Regulations on*

Cheating, Plagiarism and Collusion

Section 2.3: "The following broad types of offence can be identified and are provided as indicative examples.

- (i) **Cheating including taking unauthorised material into an examination; consulting unauthorised material outside the examination hall during the examination; obtaining an unseen examination paper in advance of the examination; copying from another examinee; using an unauthorised calculator during the examination or storing unauthorised material in the memory of a programmable calculator, which is taken into the examination; copying coursework.**
- (ii) **Falsifying data in experimental results.**
- (iii) Personation, where a substitute takes an examination or test on behalf of the candidate. Both candidate and substitute may be guilty of an offence under these Regulations.
- (iv) **Bribery or attempted bribery of a person thought to have some influence on the candidate's assessment.**
- (v) Collusion to present joint work as the work solely of one individual.
- (vi) Plagiarisms, where the work or ideas of another are presented as the candidate's own.
- (vii) Other conduct calculated to secure an advantage on assessment.
- (viii) Assisting in any of the above.

Some notes on what this means for students:

1. Copying another student's work is an offence, whether from a copy on paper or from a computer file, and in whatever form the intellectual property being copied takes, including text, mathematical notation and computer programs.

2. Taking extracts from published sources *without attribution* is an offence. To quote ideas, sometimes using extracts, is generally to be encouraged. Quoting ideas is achieved by stating an author's argument and attributing it, perhaps by quoting, immediately in the text, his or her name and year of publication, e.g. " $e = mc^2$ (Einstein 1905)". A *references* section at the end of your work should then list all such references in alphabetical order of authors' surnames. (There are variations on this referencing system, which your tutors may prefer you to use.) If you wish to quote a paragraph or so from published work, then indent the quotation on both left and right margins, using an italic font where practicable, and introduce the quotation with an attribution.

Contents

Introduction.....	4
Database Generation	5
Connecting Microsoft SQL Sever Management studio and SELECT Displays in the database	10
Insert data	11
Display Data.....	13
Workflow Diagram	14
Architecture & Tech Stack.....	15
Controllers.....	17
HomeController	17
AppointmentController	18
DoctorController	20
PatientController	21
FeedBackController	22
Models.....	24
Appointment Model	24
Doctor Model	25
Patient Model	26
FeedBack Model	27
Payment Model	28
User Manual.....	29
Patient User Guide	29
Patient Dashboard.....	29
Patient Appointment Dashboard.....	30
Patient Feedback Dashboard	33

Admin User Guide	34
Doctor Account Management Page	34
Patients Accounts Management Page	37
Payment Management Page	39
Report Dashboard	40
Admin Appointment Dashboard	40
Specializations Dashboard	41
Doctor User Guide	41
Weaknesses and Future Development	43
References	44

Introduction

Background

In today's digital healthcare environment, many hospitals and clinics still rely on manual processes to manage patient appointments, resulting in scheduling conflicts, long waiting times, and inefficient record-keeping. To solve these problems, the Online Healthcare Appointment System was developed to streamline communication between patients, doctors, and administrators through a centralized web platform.

Project Purpose

The goal of this system is to allow users to easily book, amend, or cancel appointments online, while doctors and administrators can efficiently handle schedules, payments, and patient data. The solution lowers human error, increases data security, and enhances the entire patient experience.

System Overview

This project follows the Model-View-Controller (MVC) architecture using **ASP.NET MVC (C#)**, **SQL Server** as the relational database, and **Entity Framework** for ORM. Bootstrap is used for responsive design, ensuring accessibility across devices. The system includes three main roles:

- **Admin:** Manages doctors, appointments, payments, and reports.
- **Doctor:** Reviews patient details and manages appointment schedules.
- **Patient:** Registers, logs in, books appointments, and makes online payments.

Project Objectives

The primary goals of the system are to:

- Create a safe, user-friendly healthcare appointment platform.
- Reduce administrative workload and scheduling conflicts.

- Provide quick access to doctors' availability and patient records.
- Create summary reports for management and statistical analysis.

GitHub Repository

The complete source code and project documentation can be accessed at:

<https://github.com/Moses2004/Online-Healthcare-Appointment-System>

Database Generation

Add ERD, relation specs, and data dictionary here. Export diagram images and paste them below with captions.

ERD Diagram

This is the conceptual view of the database. It focuses on how the entities relate to one another rather than the exact column types.

- Admin manages doctors in the system.
- Doctor belongs to a specific Specialization and has details such as name, email, phone, consultation fee, and availability.
- Patient can register, book appointments, and give feedback.
- Appointment acts as the core link between patients and doctors, recording the date, status, and notes.
- Payment is generated after an appointment and stores payment amount, date, method, and status.
- Prescription is created by the doctor for each appointment, containing doctor notes and medicine details.
- Feedback is written by patients for doctors, including comments and ratings.

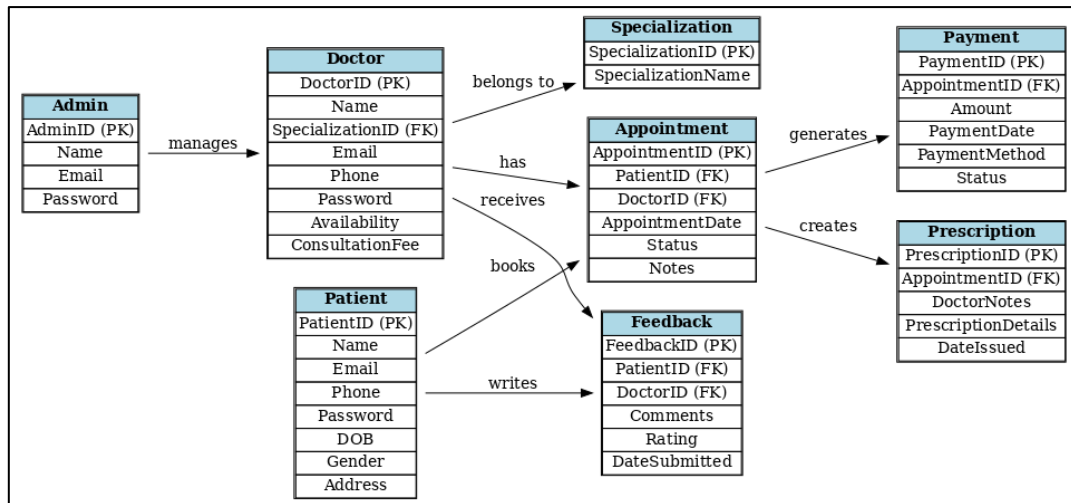


Figure: ER Diagram

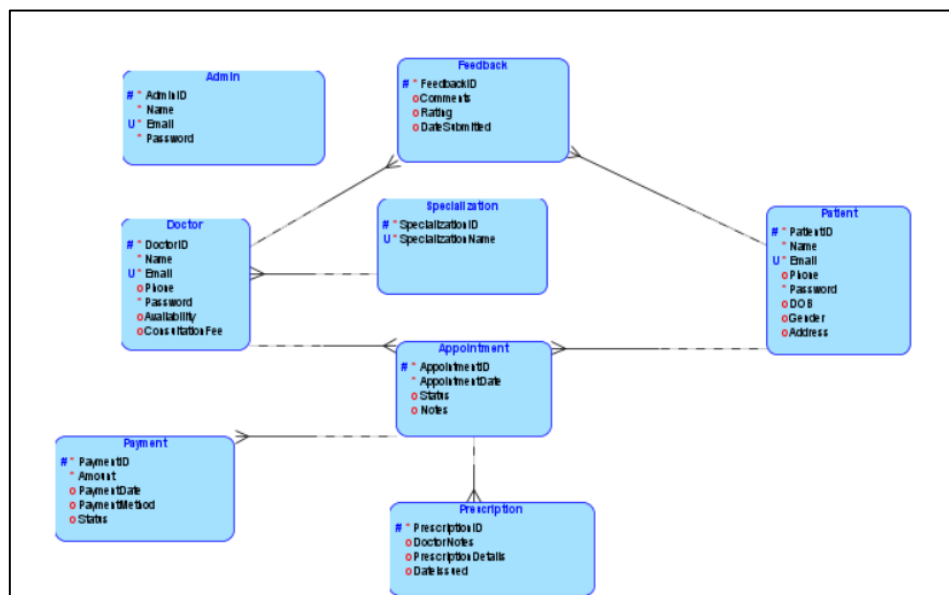


Figure: ER Diagram

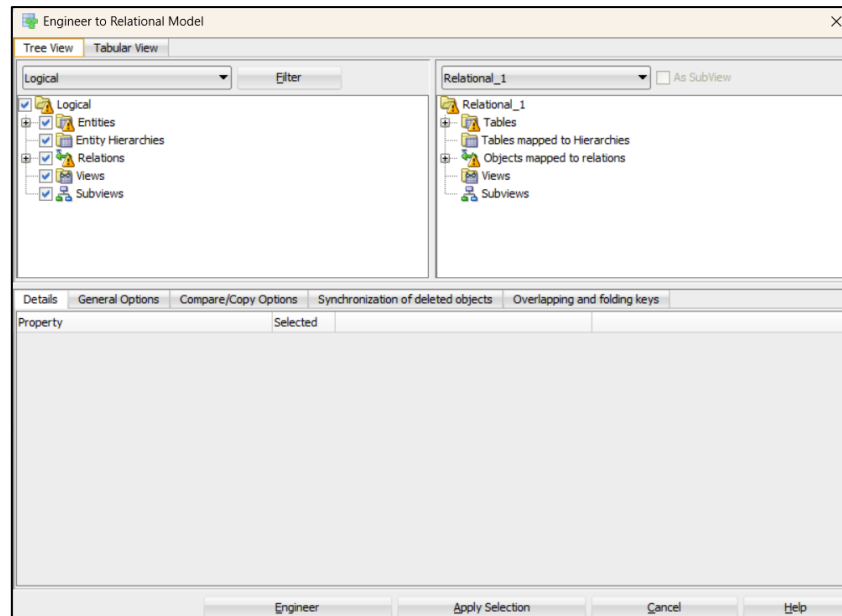


Figure: Engineer to Relational Model

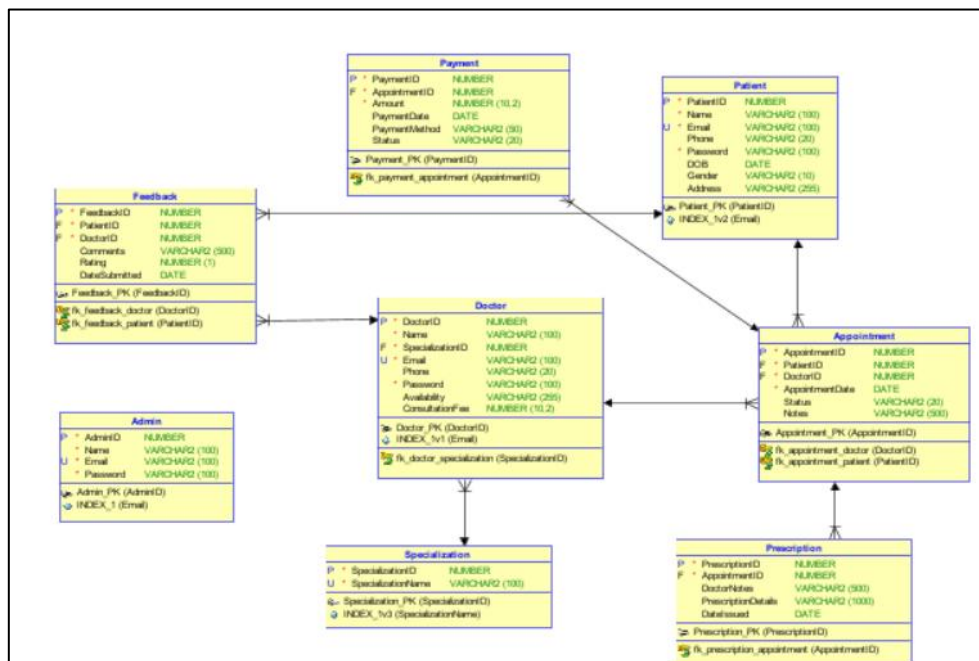


Figure: Converting ERD into DB Tables using SQL Developer Data Modeler

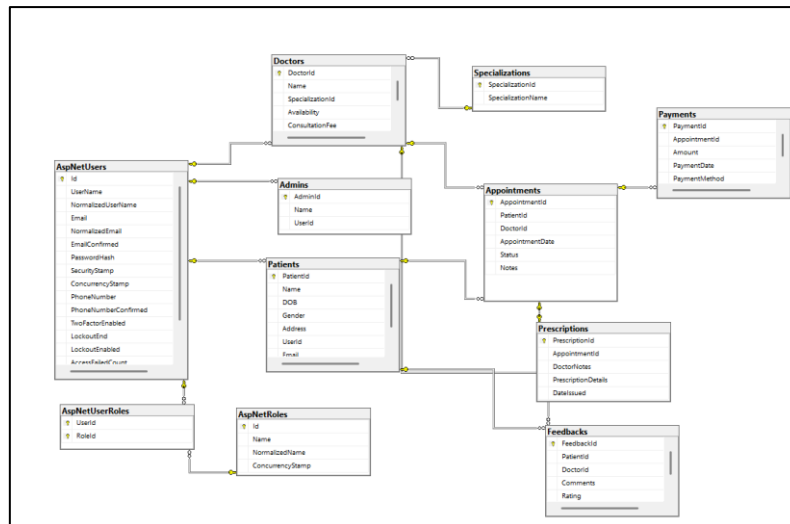


Figure: Database Diagram in Microsoft SQL Server Management Studio

Data Dictionary

Table	Column	Data Type	Constraints / Description
Admin	AdminId	Number	Primary Key, Auto Increment
	Name	VARCHAR2(100)	Not Null
	Email	VARCHAR2(100)	Unique, Not Null
	Password	VARCHAR2(100)	Not Null
Specialization	SpecializationID	NUMBER	Primary Key, Auto Increment
	SpecializationName	VARCHAR2(100)	Unique, Not Null
Doctor	DoctorID	NUMBER	Primary Key, Auto Increment
	Name	VARCHAR2(100)	Not Null
	SpecializationID	NUMBER	FK → Specialization.Specialization
	Email	VARCHAR2(100)	Unique, Not Null
	Phone	VARCHAR2(20)	Nullable
	Password	VARCHAR2(100)	Not Null
	Availability	VARCHAR2(255)	Nullable
	ConsultationFee	NUMBER(10,2)	Nullable
Patient	PatientID	NUMBER	Primary Key, Auto Increment
	Name	VARCHAR2(100)	Not Null

	Email	VARCHAR2(100)	Unique, Not Null
	Phone	VARCHAR2(100)	Nullable
	Password	VARCHAR2(100)	Not Null
	DOB	DATE	Nullable
	Gender	VARCHAR2(10)	Nullable
	Address	VARCHAR2(255)	Nullable
Appointment	AppointmentID	NUMBER	Primary Key, Auto Increment
	PatientID	NUMBER	FK → Patient.PatientID
	DoctorID	NUMBER	FK → Doctor.DoctorID
	AppointmentDate	DATE	Not Null
	Status	VARCHAR2(20)	
	Notes	VARCHAR2(500)	Nullable
Payment	PaymentID	NUMBER	Primary Key, Auto Increment
	AppointmentID	NUMBER	FK → Appointment.AppointmentID
	Amount	NUMBER(10,2)	Not Null
	PaymentDate	DATE	Default SYSDATE
	PaymentMethod	VARCHAR2(50)	Nullable
	Status	VARCHAR2(20)	Nullable
Prescription	PrescriptionID	NUMBER	Primary Key, Auto Increment
	AppointmentID	NUMBER	FK → Appointment.AppointmentID
	DoctorNotes	VARCHAR2(500)	Nullable
	PrescriptionDetails	VARCHAR2(1000)	Nullable
	DateIssued	DATE	Default SYSDATE
Feedback	FeedbackID	NUMBER	Primary Key, Auto Increment

	PatientID	NUMBER	FK → Patient.PatientID
	DoctorID	NUMBER	FK → Doctor.DoctorID
	Comments	VARCHAR2(500)	Nullable
	Rating	NUMBER(1)	Nullable
	DateSubmitted	DATE	Default SYSDATE

Connecting Microsoft SQL Server Management studio and SELECT Displays in the database

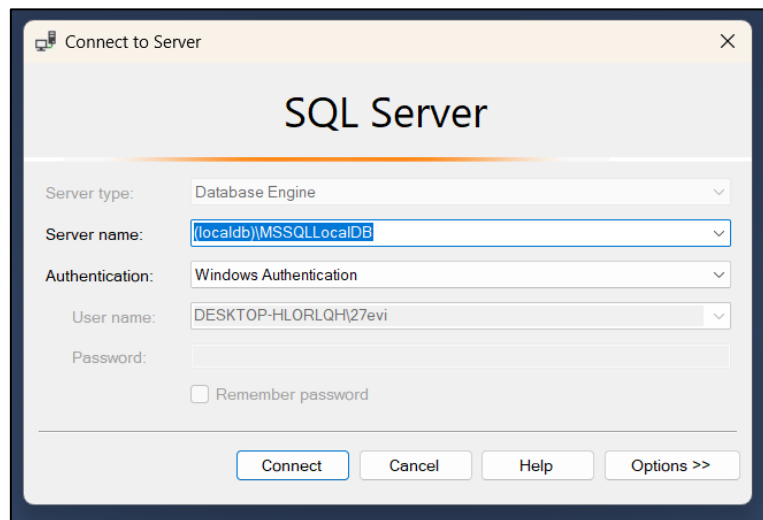


Figure: Connecting to the database

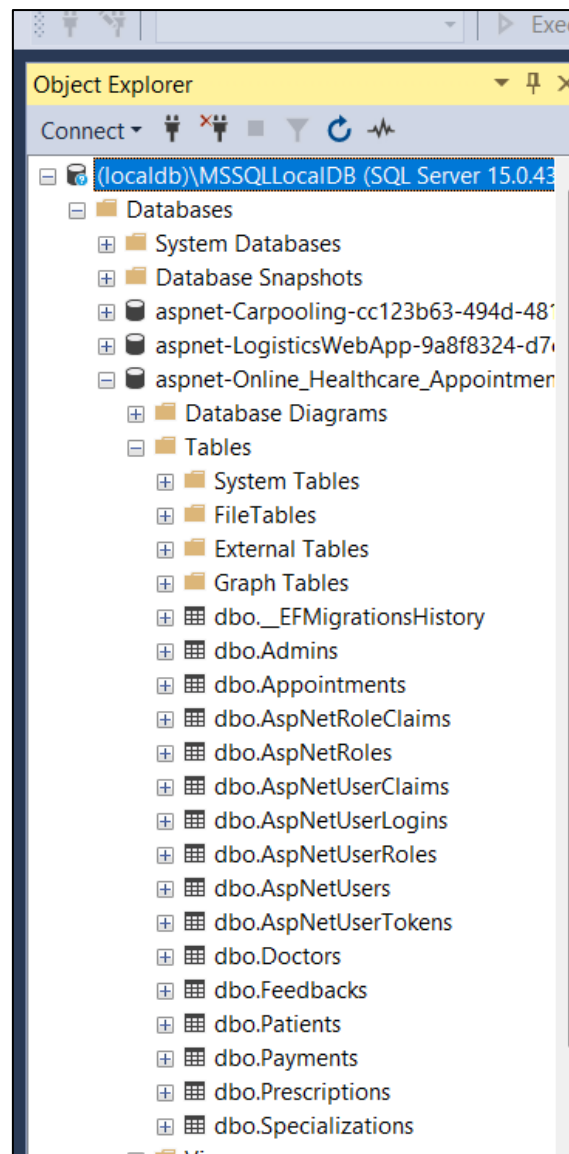


Figure: Database Creating Successful

Insert data

Inserting sample data to the **Feedbacks** table by sql command

```
“ INSERT INTO Feedbacks (PatientId, DoctorId, Comments, Rating, DateSubmitted)  
VALUES (12, 6, 'Excellent service and kind doctor.', 5, GETDATE());”
```

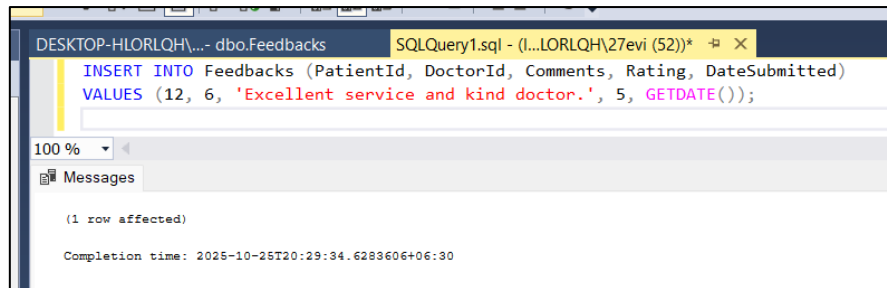


Figure: Inserting Data

The screenshot shows a SQL query window with the following text:

```
Select * from Feedbacks;
```

Below the query, the Results pane shows the following data:

	FeedbackId	PatientId	DoctorId	Comments	Rating	DateSubmitted
1	1	9	5	good	3	2025-10-03 21:13:22.9328221
2	3	14	5	Good	5	2025-10-24 22:15:35.6692589
3	4	12	6	Excellent service and kind doctor.	5	2025-10-25 20:29:34.6166667

Figure: Data inserting successful

Inserting sample data into the “**Specializations**” table.

“ INSERT INTO Specializations (SpecializationName)

VALUES

('Dermatology'),

('Neurology'),

('Pediatrics');”

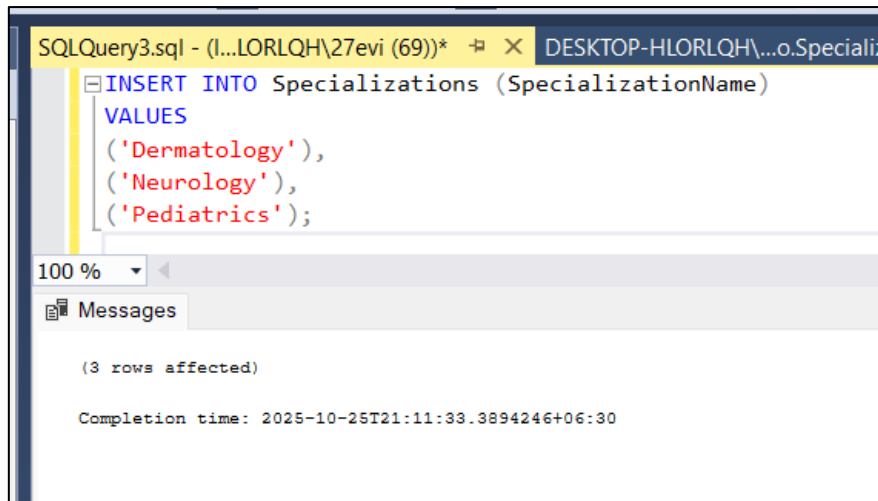


Figure: Data inserting

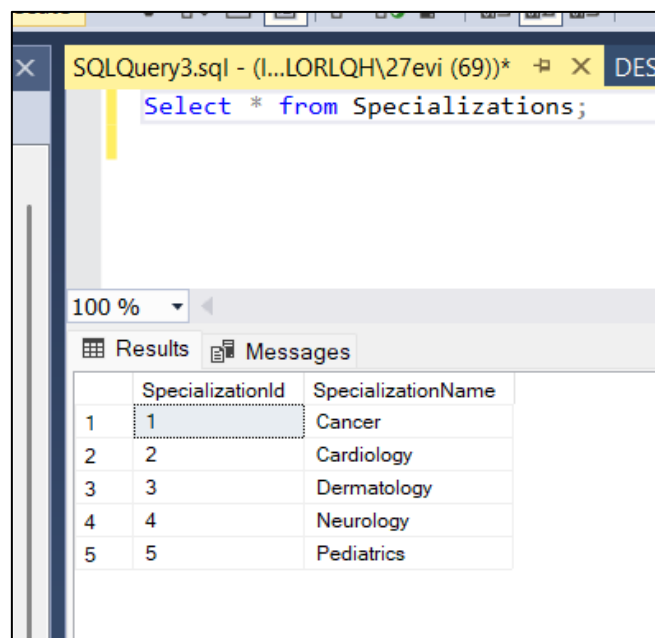


Figure: Inserting Data Successful

Display Data

Display all the data of the **Doctors** table by the following sql command:

“Select * from Doctors:”

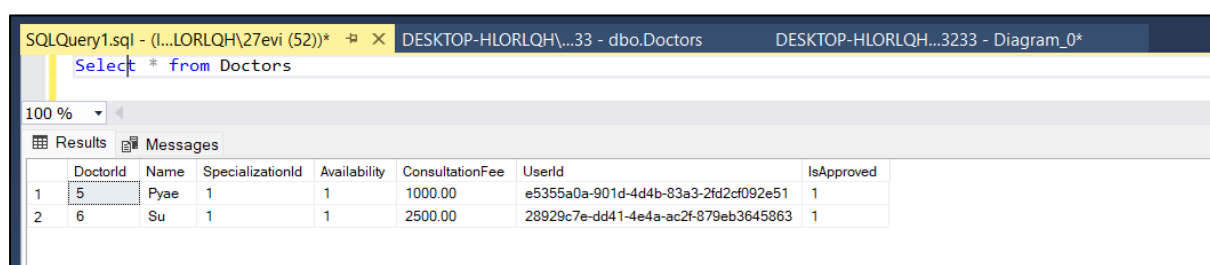
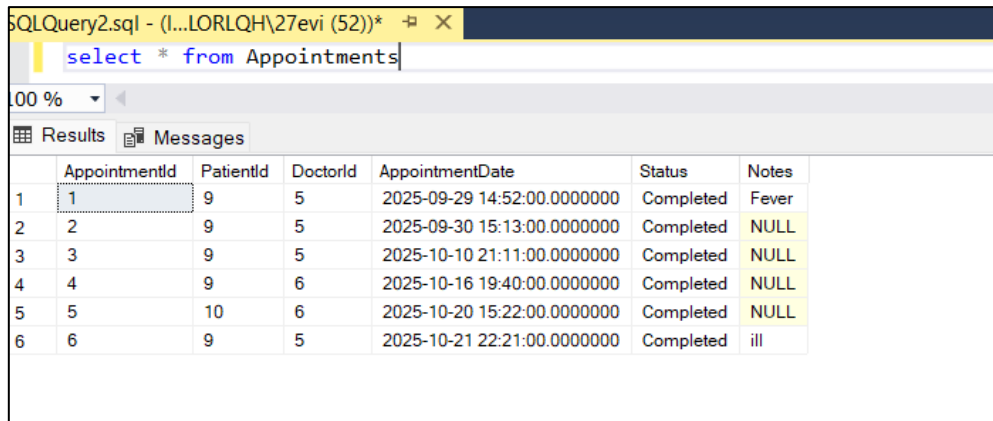


Figure: Display all data in doctor table

Display all the data of the **Appointments** table by the following sql command:

“ Select * from Appointments:”



SQLQuery2.sql - (L...LORLQH\27evi (52))*

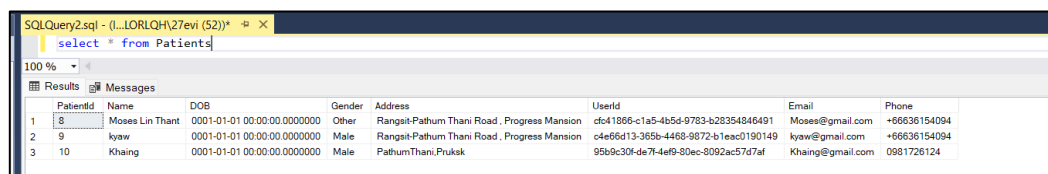
```
select * from Appointments
```

100 %

Results Messages

	AppointmentId	PatientId	DoctorId	AppointmentDate	Status	Notes
1	1	9	5	2025-09-29 14:52:00.0000000	Completed	Fever
2	2	9	5	2025-09-30 15:13:00.0000000	Completed	NULL
3	3	9	5	2025-10-10 21:11:00.0000000	Completed	NULL
4	4	9	6	2025-10-16 19:40:00.0000000	Completed	NULL
5	5	10	6	2025-10-20 15:22:00.0000000	Completed	NULL
6	6	9	5	2025-10-21 22:21:00.0000000	Completed	ill

Figure: Display all data in appointment table



SQLQuery2.sql - (L...LORLQH\27evi (52))*

```
select * from Patients
```

100 %

Results Messages

	PatientId	Name	DOB	Gender	Address	Userid	Email	Phone
1	8	Moses Lin Thant	0001-01-01 00:00:00.0000000	Other	Rangsit-Pathum Thani Road , Progress Mansion	cf41866-c1a5-4b5d-9783-b28354846491	Moses@gmail.com	+66636154094
2	9	Kyaw	0001-01-01 00:00:00.0000000	Male	Rangsit-Pathum Thani Road , Progress Mansion	c4e6d13-365b-4468-9872-b1eac0190149	Kyaw@gmail.com	+66636154094
3	10	Khaing	0001-01-01 00:00:00.0000000	Male	PathumThani,Pruksa	95b9c30f-de7f-4ef9-80ec-8092ac57d7af	Khaing@gmail.com	0981726124

Figure: Display all data in Patient table

Workflow Diagram

The following diagram is created by using draw.io software

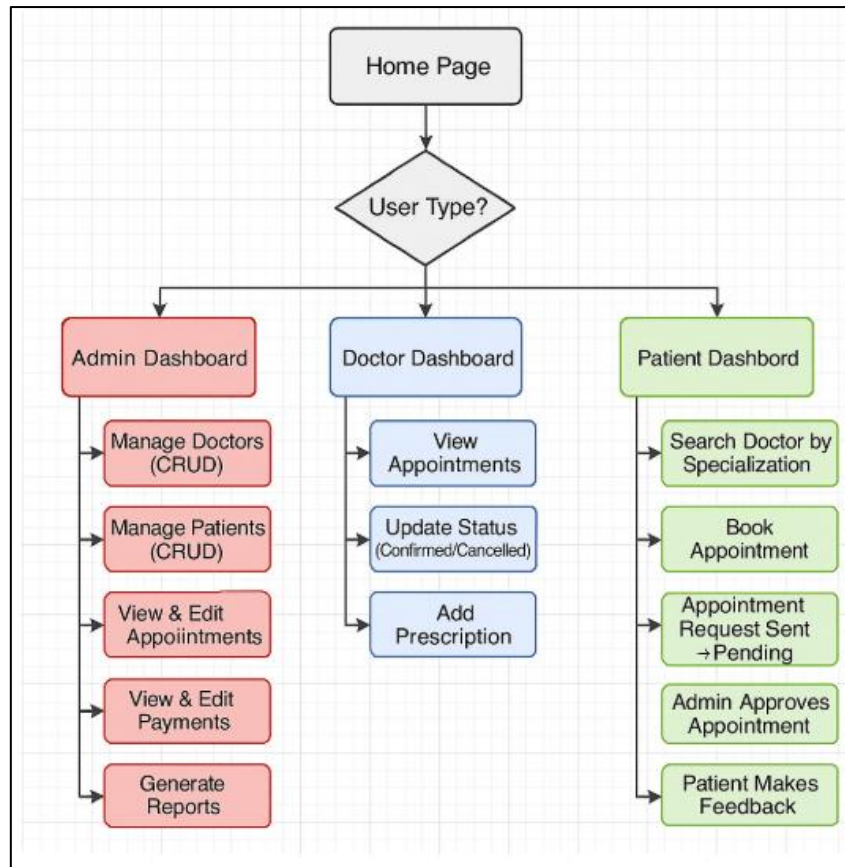
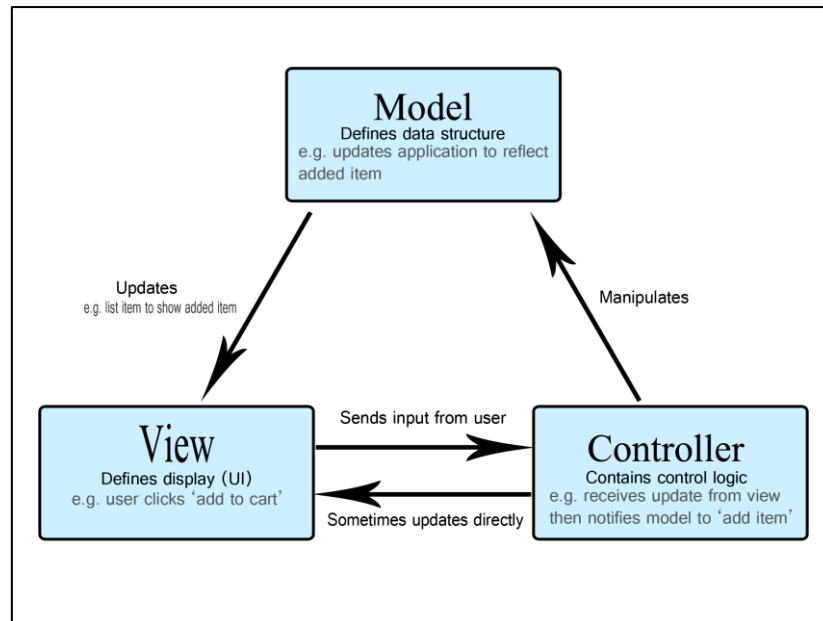


Figure: Workflow Diagram for HealthCare Application System

Architecture & Tech Stack

The Online Healthcare Appointment System is created using the ASP.NET MVC architecture, which separates the application into three communicating layers — Model, View, and Controller. This design pattern facilitates the development of more scalable, maintainable, and modular applications. The Models articulate data and business logic, the Controllers handle requests and orchestrate the flow, and the Views provide the users with Razor-based interfaces for information through their interactions. The use of Entity Framework and Identity is to manage database communication and role-based authentication respectively.



Model Layer (Data and Business Logic)

- Represents the data structure and logic of the application.
- Includes classes like Doctor, Patient, Appointment, and Payment inside the Models folder.
- Uses Entity Framework for ORM (Object-Relational Mapping), handling database operations through C# objects instead of raw SQL.
- The model defines relationships, validation, and database context (ApplicationDbContext.cs).

View Layer (User Interface)

- Contains Razor View (.cshtml) files organized under the Views folder.
- Each folder (e.g., Views/Doctor, Views/Appointment, Views/Patient) corresponds to a controller.
- Responsible for displaying data to users using HTML, Bootstrap, and Razor syntax such as @Model and @Html.DisplayFor.

Controller Layer (Application Logic)

- Found in the Controllers folder (e.g., DoctorController.cs, AppointmentController.cs, PaymentController.cs).
- Handles incoming HTTP requests, interacts with Models, and returns Views or JSON responses.

- Example: The DoctorController retrieves doctor records from the database and passes them to the Index view.

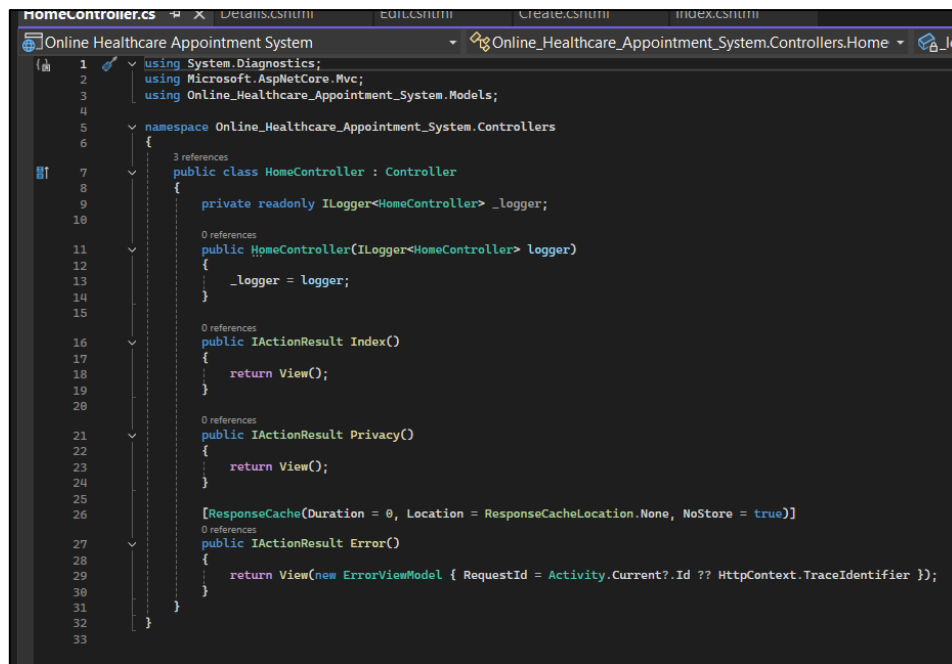
Supporting Components

- Entity Framework Core: Manages database communication via LINQ queries.
- ASP.NET Identity: Provides authentication and authorization for Admin, Doctor, and Patient roles.
- SQL Server: Used as the primary database for storing relational data.
- Bootstrap & jQuery: Handle front-end layout and dynamic interactivity.
- Dependency Injection (DI): Registers and manages services such as ApplicationDbContext within Startup.cs or Program.cs.

Controllers

HomeController

The HomeController.cs is the starting point of the website. It controls what users see when they open the system. This controller has three main actions: Index(), Privacy(), and Error(). The Index() method loads the homepage, Privacy() shows the privacy policy, and Error() displays error details using the ErrorViewModel. It helps keep the website organized and ensures users are directed to the correct pages or shown an error message when something goes wrong.



```
1 using System.Diagnostics;
2 using Microsoft.AspNetCore.Mvc;
3 using Online_Healthcare_Appointment_System.Models;
4
5 namespace Online_Healthcare_Appointment_System.Controllers
6 {
7     public class HomeController : Controller
8     {
9         private readonly ILogger<HomeController> _logger;
10
11         public HomeController(ILogger<HomeController> logger)
12         {
13             _logger = logger;
14         }
15
16         public IActionResult Index()
17         {
18             return View();
19         }
20
21         public IActionResult Privacy()
22         {
23             return View();
24         }
25
26         [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
27         public IActionResult Error()
28         {
29             return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
30         }
31     }
32 }
33
```

Figure: HomeController.cs

AppointmentController

The **AppointmentsController** manages all actions related to appointments between doctors and patients. It makes sure that each user can only see or change what they are allowed to. It uses the database context `_context` to get or save data.

The **Index()** method shows all appointments.

- If the user is **Admin**, they can see every appointment and search by doctor or patient name.
- If the user is a **Doctor**, they only see their own appointments.
- If the user is a **Patient**, they only see their personal appointments.

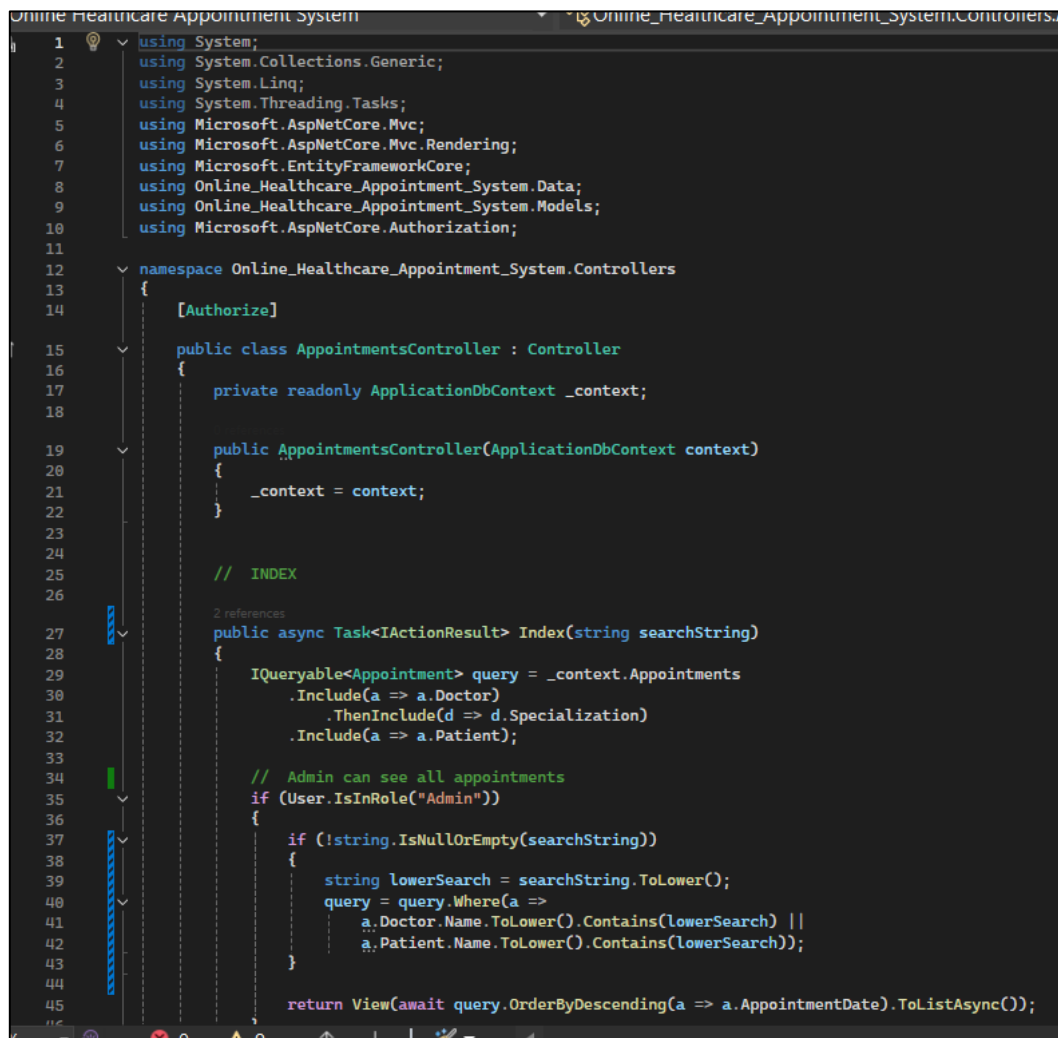
The **Details()** method shows full information about one appointment. It checks the user role to make sure only the doctor, the patient, or the admin can view it.

The **Create()** methods let a patient make a new appointment. The system automatically fills in the patient's information and sets the status to "Pending." Only approved doctors appear in the dropdown list.

The **Edit()** methods allow doctors or admins to edit appointments, such as changing the date, status, or notes. Patients are not allowed to edit.

The **Delete()** methods let doctors or admins remove appointments from the system. Patients are not allowed to delete.

Finally, the **AppointmentExists()** method checks if an appointment still exists in the database.



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Threading.Tasks;
5 using Microsoft.AspNetCore.Mvc;
6 using Microsoft.AspNetCore.Mvc.Rendering;
7 using Microsoft.EntityFrameworkCore;
8 using Online_Healthcare_Appointment_System.Data;
9 using Online_Healthcare_Appointment_System.Models;
10 using Microsoft.AspNetCore.Authorization;
11
12 namespace Online_Healthcare_Appointment_System.Controllers
13 {
14     [Authorize]
15
16     public class AppointmentController : Controller
17     {
18         private readonly ApplicationDbContext _context;
19
20         public AppointmentController(ApplicationDbContext context)
21         {
22             _context = context;
23         }
24
25         // INDEX
26
27         public async Task<IActionResult> Index(string searchString)
28         {
29             IQueryable<Appointment> query = _context.Appointments
30                 .Include(a => a.Doctor)
31                 .ThenInclude(d => d.Specialization)
32                 .Include(a => a.Patient);
33
34             // Admin can see all appointments
35             if (User.IsInRole("Admin"))
36             {
37                 if (!string.IsNullOrEmpty(searchString))
38                 {
39                     string lowerSearch = searchString.ToLower();
40                     query = query.Where(a =>
41                         a.Doctor.Name.ToLower().Contains(lowerSearch) ||
42                         a.Patient.Name.ToLower().Contains(lowerSearch));
43                 }
44
45                 return View(await query.OrderByDescending(a => a.AppointmentDate).ToListAsync());
46             }
47         }
48     }
49 }
```

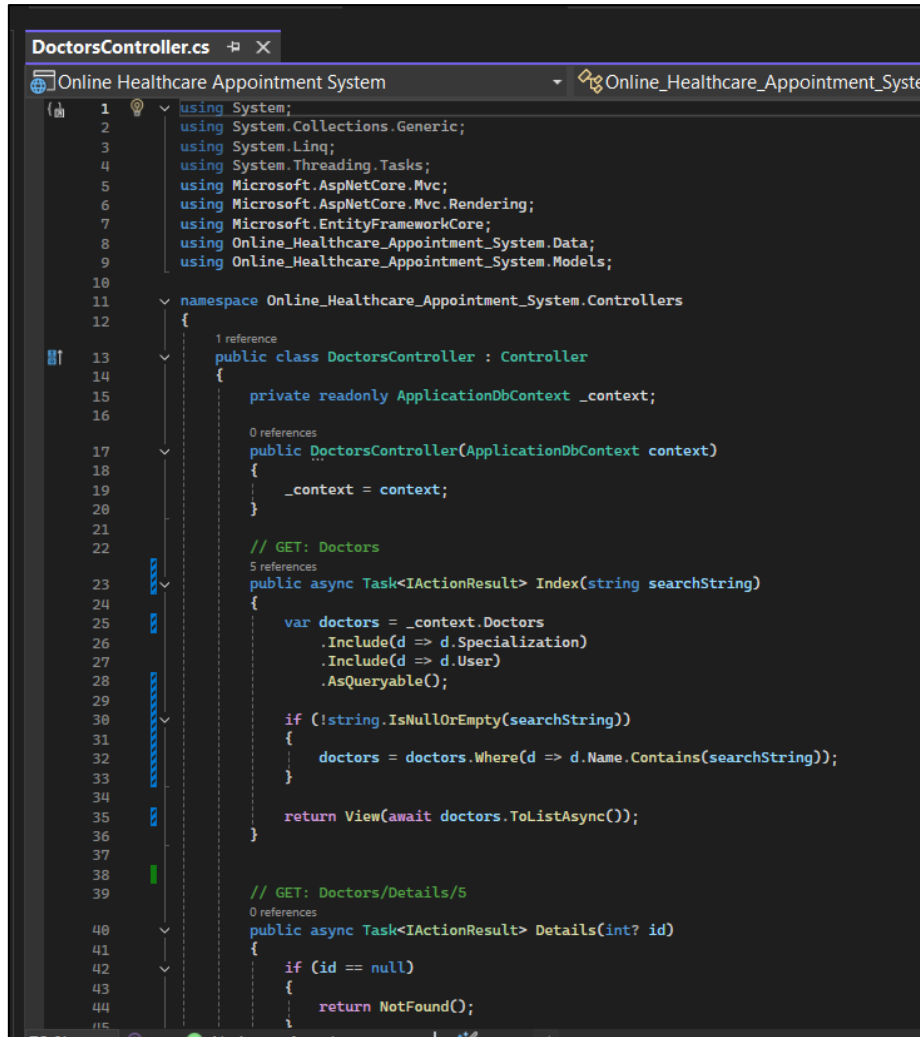
Figure: AppointmentContoller.cs

DoctorController

The **DoctorsController.cs** manages everything related to doctors in the system. It connects to the database using **ApplicationDbContext** to display, add, edit, or delete doctor information.

- The **Index()** method shows all doctors and allows searching by name.
- The **Details()** method shows full information of one doctor.
- The **Create()** methods let the admin add new doctors with their specialization, availability, and consultation fee.
- The **Edit()** methods are used to update doctor details.
- The **Delete()** methods remove a doctor from the system.
- There are also **Approve()** and **Remove()** methods to approve or delete a doctor easily.

It helps the admin manage doctor accounts and keeps the data organized in the database.



```

DoctorsController.cs
Online Healthcare Appointment System
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Threading.Tasks;
5 using Microsoft.AspNetCore.Mvc;
6 using Microsoft.AspNetCore.Mvc.Rendering;
7 using Microsoft.EntityFrameworkCore;
8 using Online_Healthcare_Appointment_System.Data;
9 using Online_Healthcare_Appointment_System.Models;
10
11 namespace Online_Healthcare_Appointment_System.Controllers
12 {
13     1 reference
14     public class DoctorsController : Controller
15     {
16         private readonly ApplicationDbContext _context;
17
18         0 references
19         public DoctorsController(ApplicationDbContext context)
20         {
21             _context = context;
22         }
23
24         // GET: Doctors
25         5 references
26         public async Task<IActionResult> Index(string searchString)
27         {
28             var doctors = _context.Doctors
29                 .Include(d => d.Specialization)
30                 .Include(d => d.User)
31                 .AsQueryable();
32
33             if (!string.IsNullOrEmpty(searchString))
34             {
35                 doctors = doctors.Where(d => d.Name.Contains(searchString));
36             }
37
38             return View(await doctors.ToListAsync());
39         }
40
41         // GET: Doctors/Details/5
42         0 references
43         public async Task<IActionResult> Details(int? id)
44         {
45             if (id == null)
46             {
47                 return NotFound();
48             }
49         }
50     }
51 }

```

Figure: DoctorController.cs

PatientController

The **PatientsController.cs** manages all actions related to patients in the system

- The **Index()** method shows a list of all patients, which only admins and doctors can see.
- The **Details()** method shows full details of one patient.
- The **Create()** methods allow the admin to add a new patient with their name, gender, date of birth, and address.
- The **Edit()** methods let the admin update patient information.
- The **Delete()** methods remove a patient from the system.

- There's also a **ManageProfile()** method that redirects patients to their profile management page.

It helps the admin and doctors manage patient records safely and easily.

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using Online_Healthcare_Appointment_System.Data;
using Online_Healthcare_Appointment_System.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;

namespace Online_Healthcare_Appointment_System.Controllers
{
    [Authorize]
    public class PatientsController : Controller
    {
        private readonly ApplicationDbContext _context;

        public PatientsController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: Patients
        [Authorize(Roles = "Admin,Doctor")]
        public async Task<IActionResult> Index()
        {
            var applicationDbContext = _context.Patients.Include(p => p.User);
            return View(await applicationDbContext.ToListAsync());
        }

        // Redirect to Identity Manage Page
        public IActionResult ManageProfile()
        {
            return Redirect("/Identity/Account/Manage");
        }

        // GET: Patients/Details/5
        [Authorize(Roles = "Admin,Doctor")]
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }
        }
    }
}
```

Figure: PatientController.cs

FeedbackController

The **FeedbacksController.cs** manages all feedback actions in the system. It connects to the database to show, add, edit, or delete patient feedback for doctors.

The **Index()** method displays feedback based on user roles: admins can see all, doctors can see feedback about them, and patients can see only their own.

The **Create()** methods let patients write and submit feedback for approved doctors.

The **Edit()** methods allow patients to update their own feedback.

The **Delete()** methods remove a feedback entry from the system.

It helps manage doctor reviews and ensures each user only sees or edits their allowed data.

```
1 using Microsoft.AspNetCore.Authorization;
2 using Microsoft.AspNetCore.Mvc;
3 using Microsoft.AspNetCore.Mvc.Rendering;
4 using Microsoft.EntityFrameworkCore;
5 using Online_Healthcare_Appointment_System.Data;
6 using Online_Healthcare_Appointment_System.Models;
7 using System;
8 using System.Collections.Generic;
9 using System.Linq;
10 using System.Threading.Tasks;

11 namespace Online_Healthcare_Appointment_System.Controllers
12 {
13     public class FeedbacksController : Controller
14     {
15         private readonly ApplicationDbContext _context;
16
17         // GET: Feedbacks
18         [Authorize]
19         public async Task<IActionResult> Index()
20         {
21             IQueryable<Feedback> feedbacks = _context.Feedbacks
22                 .Include(f => f.Doctor)
23                 .Include(f => f.Patient);
24
25             if (User.IsInRole("Admin"))
26             {
27                 //Admin can see all feedbacks
28                 return View(await feedbacks
29                     .OrderByDescending(f => f.DateSubmitted)
30                     .ToListAsync());
31             }
32             else if (User.IsInRole("Patient"))
33             {
34                 // Patient sees only their own feedbacks
35                 var userEmail = User.Identity.Name;
36                 var patient = await _context.Patients
37                     .Include(p => p.User)
38                     .FirstOrDefaultAsync(p => p.User.Email == userEmail);
39             }
40         }
41     }
42 }
```

Figure: Feedbackcontroller.cs

Models

Appointment Model

The Appointment model is used to store each booking between a patient and a doctor in healthcare system.

- **AppointmentId** → This is the main ID (primary key). It gives each appointment a unique number.
- **PatientId** → This connects the appointment to the patient who made it.
- **DoctorId** → This connects the appointment to the doctor who will meet the patient.
- **AppointmentDate** → This stores the date and time when the appointment happens.
- **Status** → Shows if the appointment is “Pending”, “Confirmed”, “Cancelled”, etc.
- **Notes** → Optional. The doctor or admin can write extra information about the appointment.
- **Patient** and **Doctor** → These are navigation properties. They help the system link the appointment with real patient and doctor details.
- **Payments** → This shows that one appointment can have one or more payments.
- **Prescription** → One appointment can have one prescription


```

using System;

namespace Online_Healthcare_Appointment_System.Models
{
    18 references
    public class Appointment
    {
        23 references
        public int AppointmentId { get; set; } // PK
        21 references
        public int PatientId { get; set; } // FK
        33 references
        public int DoctorId { get; set; } // FK
        40 references
        public DateTime AppointmentDate { get; set; }
        31 references
        public string Status { get; set; }
        9 references
        public string ? Notes { get; set; }

        // Navigation
        24 references
        public Patient Patient { get; set; }
        16 references
        public Doctor Doctor { get; set; }

        3 references
        public ICollection<Payment> Payments { get; set; } = new List<Payment>();

        public Prescription Prescription { get; set; } // one-to-one back reference
    }
}

```

Figure: Appointment Model

Doctor Model

The Doctor model keeps all the main information about each doctor in the system.

- **DoctorId** → The unique ID for each doctor.
- **Name** → The doctor's full name.
- **SpecializationId** → Connects the doctor to their medical field .
- **Availability** → Shows if the doctor is available or not.
- **ConsultationFee** → The fee that patients must pay for an appointment
- **UserId** → Links the doctor's account to the main login system (**AspNetUsers table**).
- **User** → Connects the doctor with the user profile (email, password, etc.).
- **Specialization** → Connects the doctor to their medical department or skill area.

```
namespace Online_Healthcare_Appointment_System.Models
{
    public class Doctor
    {
        public int DoctorId { get; set; }

        public string Name { get; set; }
        15 references

        public int SpecializationId { get; set; }
        17 references

        public bool Availability { get; set; }
        23 references

        public decimal ConsultationFee { get; set; }

        // Identity Link
        14 references
        public string UserId { get; set; } // FK toAspNetUsers
        24 references
        public ApplicationUser User { get; set; }

        // Navigation
        21 references
        public Specialization Specialization { get; set; }

        12 references
        public bool IsApproved { get; set; } = false;
    }
}
```

Figure: Doctor.cs

Patient Model

The **Patient** model stores personal information about each patient who uses the system.

- **PatientId** → The unique ID for each patient.
- **Name** → The patient's full name.
- **Email** → The patient's email address.
- **Phone** → The patient's phone number.
- **DOB** → The patient's date of birth.
- **Gender** → The patient's gender (Male/Female).
- **Address** → The patient's home address.
- **UserId** → Links the patient account to the login system (**AspNetUsers** table).
- **User** → Stores login details and connects the patient to their user profile.

```
1 using System;
2
3 namespace Online_Healthcare_Appointment_System.Models
4 {
5     public class Patient
6     {
7         public int PatientId { get; set; }
8         public string Name { get; set; }
9         public string Email { get; set; }
10
11         public string Phone { get; set; }
12         public DateTime DOB { get; set; }
13         public string Gender { get; set; }
14         public string Address { get; set; }
15         // Identity Link
16         public string UserId { get; set; }
17         public ApplicationUser User { get; set; }
18     }
19 }
20 }
```

Figure: Patient.cs

FeedBack Model

The **Feedback** model stores comments and ratings that patients give to doctors after their appointments.

- **FeedbackId** → The unique ID for each feedback.
- **PatientId** → Connects the feedback to the patient who wrote it.
- **DoctorId** → Connects the feedback to the doctor who received it.
- **Comments** → The message or opinion the patient writes.
- **Rating** → A number that shows how satisfied the patient is (for example, 1–5).
- **DateSubmitted** → The date when the feedback was sent.
- **Patient** → Connects the feedback to the patient's information.
- **Doctor** → Connects the feedback to the doctor's information.

```
namespace Online_Healthcare_Appointment_System.Models
{
    public class Feedback
    {
        public int FeedbackId { get; set; } // PK
        public int PatientId { get; set; } // FK
        public int DoctorId { get; set; } // FK
        public string Comments { get; set; }
        public int Rating { get; set; }
        public DateTime DateSubmitted { get; set; }

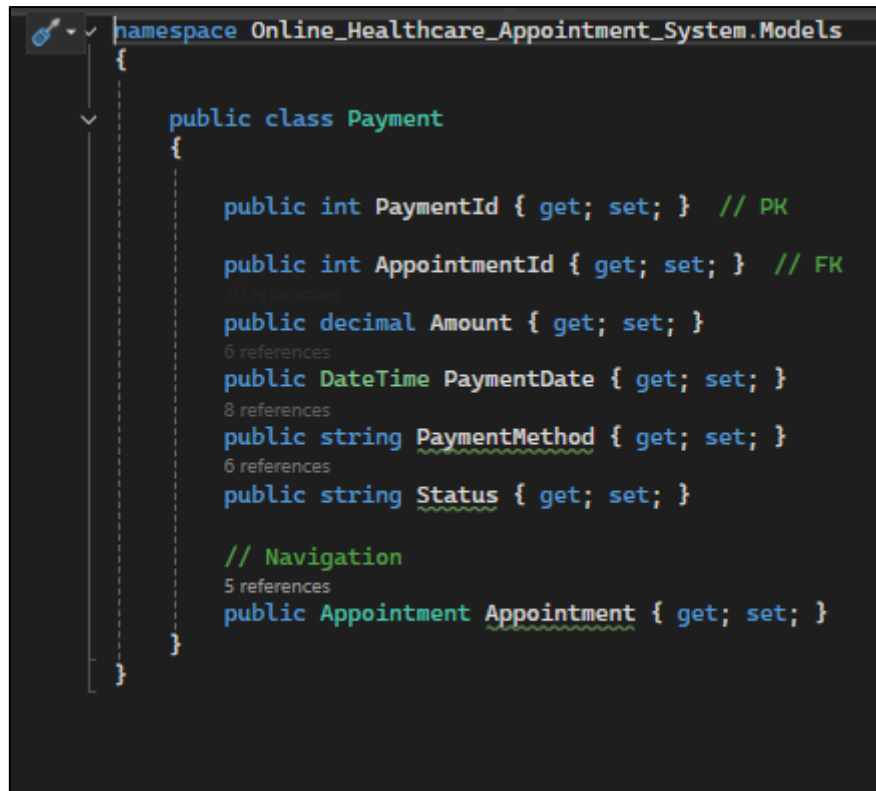
        // Navigation
        public Patient Patient { get; set; }
        public Doctor Doctor { get; set; }
    }
}
```

Figure: FeeBack.cs

Payment Model

The Payment model stores details about the money a patient pays for an appointment.

- **PaymentId** → The unique ID for each payment.
- **AppointmentId** → Connects the payment to the specific appointment.
- **Amount** → The total money the patient paid.
- **PaymentDate** → The date when the payment was made.
- **PaymentMethod** → The way the payment was done (like cash, card, or online).
- **Status** → Shows if the payment is “Completed”, “Pending”, or “Failed”.
- **Appointment** → Connects the payment record to its related appointment.



```
namespace Online_Healthcare_Appointment_System.Models
{
    public class Payment
    {
        public int PaymentId { get; set; } // PK
        public int AppointmentId { get; set; } // FK
        public decimal Amount { get; set; }
        6 references
        public DateTime PaymentDate { get; set; }
        8 references
        public string PaymentMethod { get; set; }
        6 references
        public string Status { get; set; }

        // Navigation
        5 references
        public Appointment Appointment { get; set; }
    }
}
```

Figure: Payment.cs

User Manual

Patient User Guide

Patient Dashboard

In patient dashboard, there are five categories. They are appointments, prescription, payments, my profile and feedback.

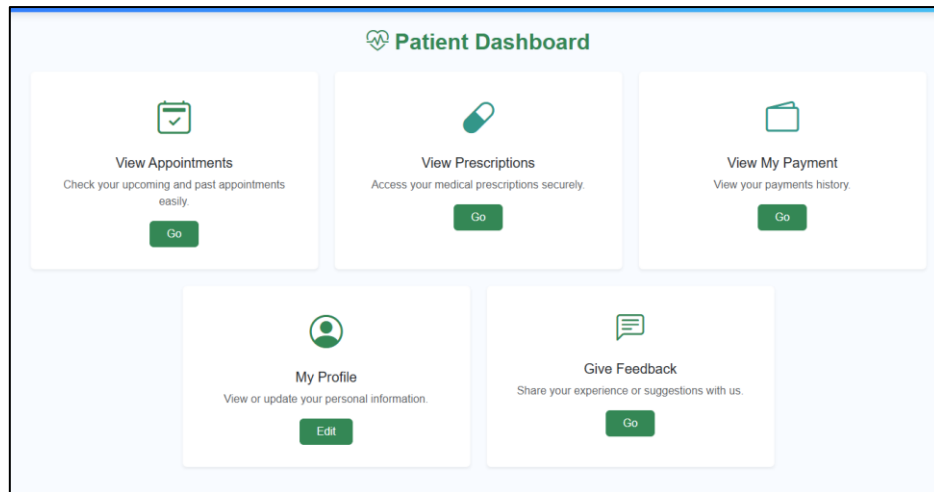


Figure: Patient Dashboard

Patient Appointment Dashboard

When a patient click “Go” in the view appointments box, the browser will lead the patient to the appointment dashboard. He can create new appointment. A patient can choose a doctor and his or her desired date time. A patient can also check all doctors information in doctors page. A patient cannot edit or delete doctors’ information. In prescription page, a patient can check all his prescription history. Also the same in payment page.

Doctor Accounts

Search

[← Back to Dashboard](#)

Name	Availability	Consultation Fee	Specialization	Approved	Actions
Pyae	Available	\$1,000.00	Cancer	Approved	Details
Su	Available	\$2,500.00	Cancer	Approved	Details
Nyein	Available	\$3,000.00	Cancer	Approved	Details

① Your patient account will be automatically linked.

Doctor

Nyein (Cancer) ▼

Date & Time

28/10/2025 14:56

Notes

Fever

⬅ Back to List

➡ Create Appointment

Figure: Creating appointment

Doctor Accounts

Search doctor by name... Search

← Back to Dashboard

Name	Availability	Consultation Fee	Specialization	Approved	Actions
Pyae	Available	\$1,000.00	Cancer	Approved	Details
Su	Available	\$2,500.00	Cancer	Approved	Details
Nyein	Available	\$3,000.00	Cancer	Approved	Details

Figure Doctors information page

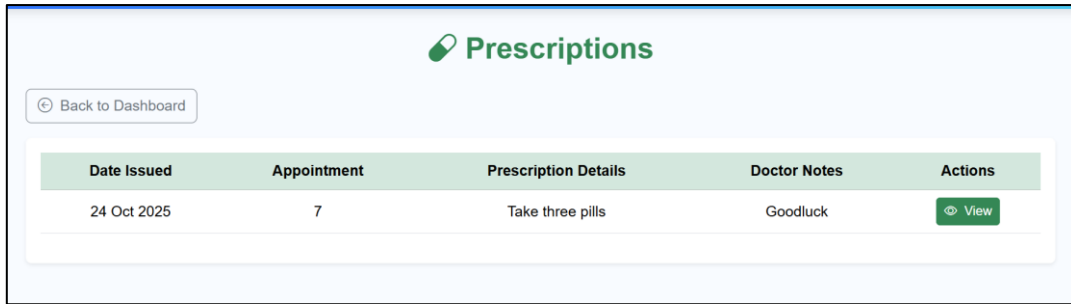
After a patient created an appointment, an admin have to approve his appointment. After an admin approve the appointment, a doctor can give prescription and the customer can pay. After the customer complete payment, the appointment status change to completed.

⬅ Back to Dashboard

➡ Create Appointment

Appointment ID	Patient	Doctor	Date	Status	Notes	Actions
7	Zoe	Nyein	30/10/2025 16:03	Pending	Fever	View

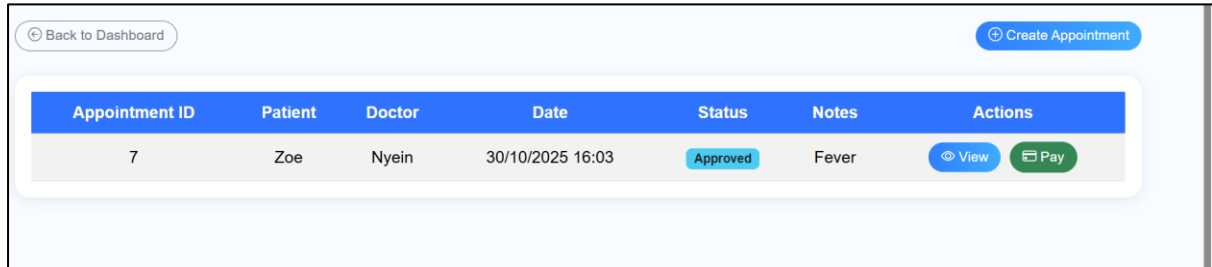
Figure: Create appointment



The screenshot shows a 'Prescriptions' page with a light blue header and a green pill icon. A 'Back to Dashboard' button is in the top left. Below is a table with columns: Date Issued, Appointment, Prescription Details, Doctor Notes, and Actions. One row is visible with the date '24 Oct 2025', appointment '7', details 'Take three pills', notes 'Goodluck', and a 'View' button.

Date Issued	Appointment	Prescription Details	Doctor Notes	Actions
24 Oct 2025	7	Take three pills	Goodluck	View

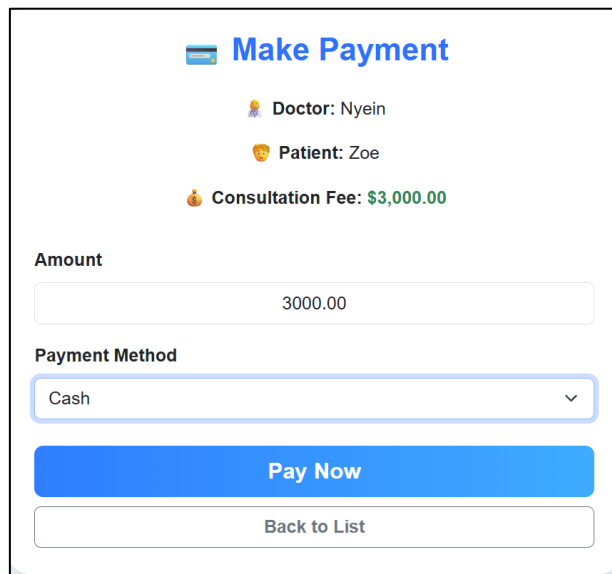
Figure: Patient Prescription



The screenshot shows an appointment list with a 'Back to Dashboard' button and a 'Create Appointment' button. The table has columns: Appointment ID, Patient, Doctor, Date, Status, Notes, and Actions. One row is shown for appointment ID '7', patient 'Zoe', doctor 'Nyein', date '30/10/2025 16:03', status 'Approved', and notes 'Fever'. The actions column contains 'View' and 'Pay' buttons.

Appointment ID	Patient	Doctor	Date	Status	Notes	Actions
7	Zoe	Nyein	30/10/2025 16:03	Approved	Fever	View Pay

Figure: After a doctor gave prescription



The screenshot shows a 'Make Payment' form with a credit card icon. It displays the doctor as 'Nyein' and the patient as 'Zoe'. The consultation fee is '\$3,000.00'. There is a text input for the amount with '3000.00' entered. The payment method is set to 'Cash' in a dropdown menu. At the bottom are 'Pay Now' and 'Back to List' buttons.

Make Payment

Doctor: Nyein

Patient: Zoe

Consultation Fee: \$3,000.00

Amount

3000.00

Payment Method

Cash

[Pay Now](#)

[Back to List](#)

Figure: Make Payment

Payment successful! Appointment marked as Completed.

Appointment Details

← Back

Patient	Zoe
Doctor	Nyein
Date	30 Oct 2025 - 04:03 pm
Status	Completed

Edit

Back

Figure: Payment Successful

Patient Feedback Dashboard

A patient can pay feedback rate the doctors with starts from 1 to five. A patient can view only his feedback but an admin can view all feedback.

Submit Feedback

Doctor

Pyae (Cancer)

Rating (1-5)

3

Comments

Good

Submit

Cancel

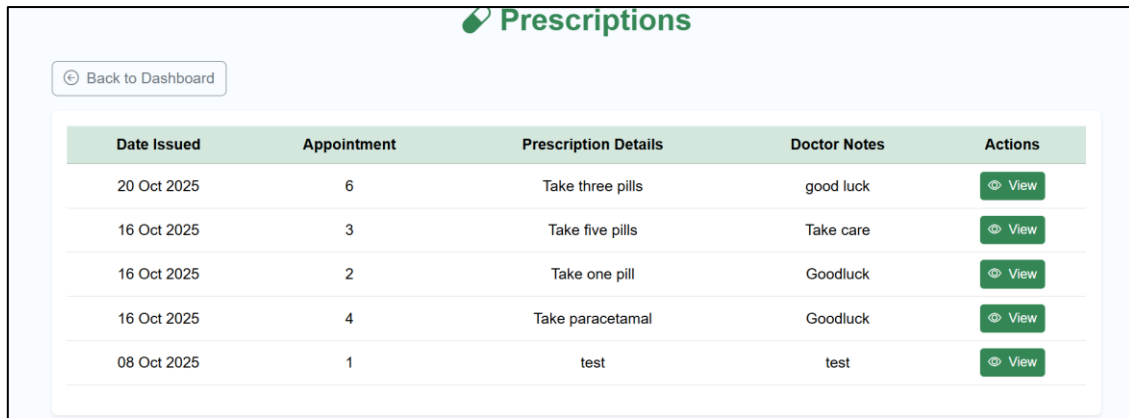
Feedback List

Add Feedback

Feedback ID	Patient	Doctor	Rating	Comments	Date	Actions
3	Zoe	Pyae	★★★★★	Good	2025-10-24	<div><div></div><div></div><div></div></div>

← Back to Dashboard

Figure: Feedback Dashboard



The screenshot shows a web application interface for 'Prescriptions'. At the top, there is a green header with a pill icon and the word 'Prescriptions'. Below the header is a 'Back to Dashboard' button. The main content is a table with five columns: 'Date Issued', 'Appointment', 'Prescription Details', 'Doctor Notes', and 'Actions'. The table contains five rows of data. Each row has a 'View' button in the 'Actions' column.

Date Issued	Appointment	Prescription Details	Doctor Notes	Actions
20 Oct 2025	6	Take three pills	good luck	View
16 Oct 2025	3	Take five pills	Take care	View
16 Oct 2025	2	Take one pill	Goodluck	View
16 Oct 2025	4	Take paracetamal	Goodluck	View
08 Oct 2025	1	test	test	View

Figure: Patient Prescription Page

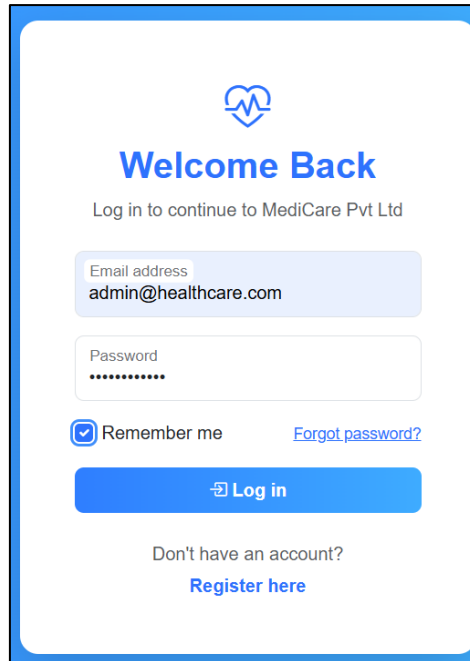
Admin User Guide

Doctor Account Management Page

There are three roles in the system. They are Admin, Doctor and Patient roles. Each role has their own privileges and restrictions. Admin default login email is “**admin@healthcare.com**” and password is “Admin123!@#”.

```
// Create default admin if not exists
string adminEmail = "admin@healthcare.com";
string adminPassword = "Admin123!@#";
```

Figure: Creating default admin Email and Pw



The login page features a heart icon with a pulse line. Below it, the text 'Welcome Back' is displayed in a large, bold, blue font. Underneath, it says 'Log in to continue to MediCare Pvt Ltd'. There are two input fields: 'Email address' with the value 'admin@healthcare.com' and 'Password' with masked characters. A 'Remember me' checkbox is checked, and a 'Forgot password?' link is present. A blue 'Log in' button is at the bottom. Below the button, it says 'Don't have an account?' with a 'Register here' link.

Figure: Login Page

After logging in as an admin, the browser will lead to you to the admin dashboard. In admin dashboard, admin can view general information of the system such as , how many patients, doctors and appointments. An admin can also go to page and can edit, create, delete processes. An can go to the pages by using the tags top bar or buttons in the page.

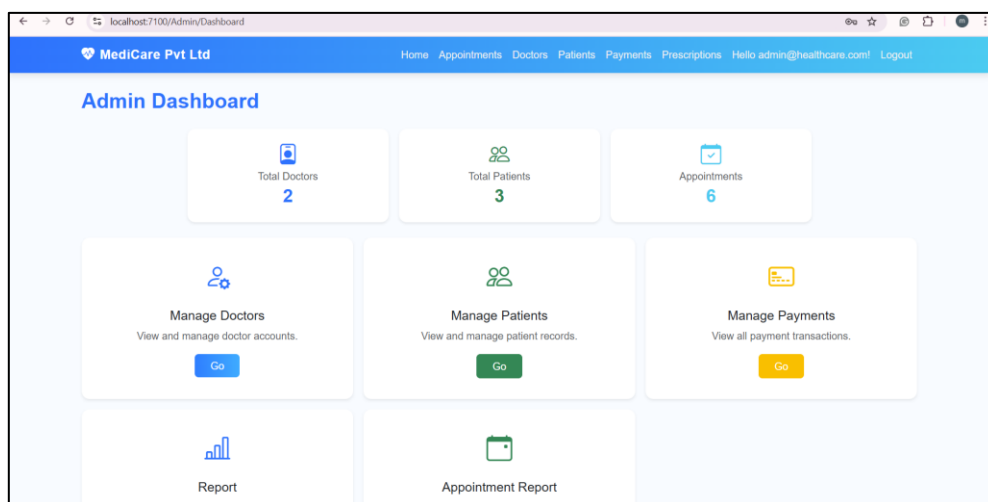
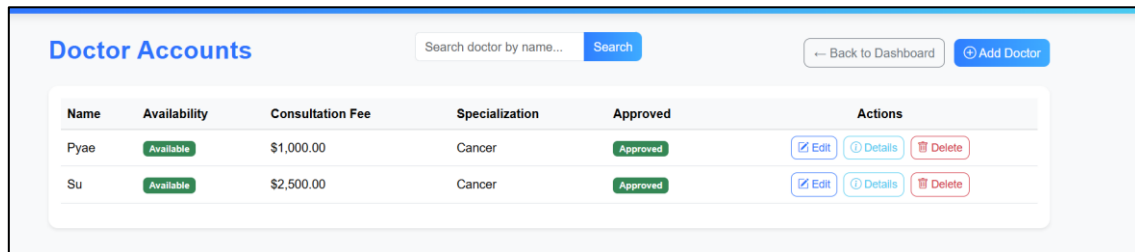


Figure: Admin Dashboard

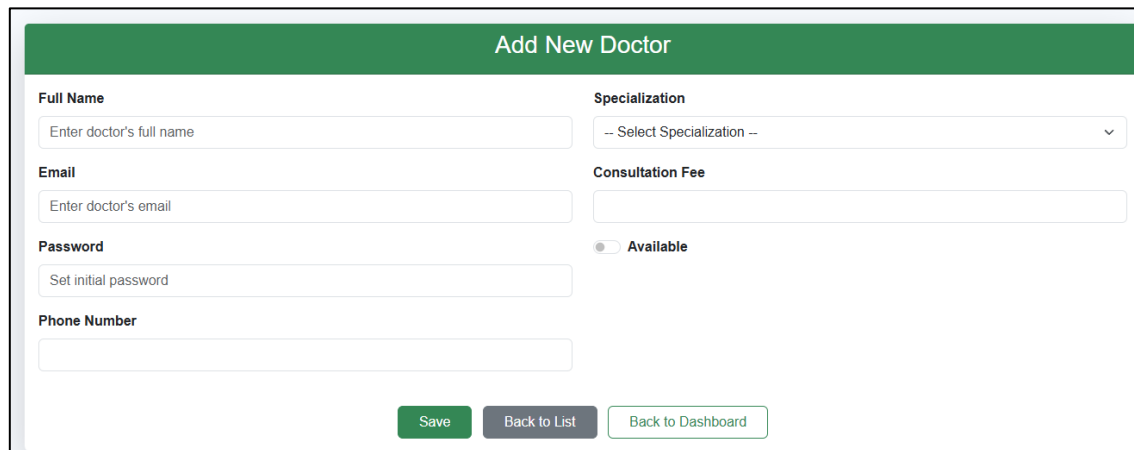
In this dashboard, admin can create new doctors and edit doctors' information such as Specilization, consultation fee etc. An admin can also delete and edit doctors' data. An admin can also search a doctor by his name.



The screenshot shows a 'Doctor Accounts' dashboard. At the top, there is a search bar with the text 'Search doctor by name...' and a 'Search' button. To the right of the search bar are two buttons: '← Back to Dashboard' and '+ Add Doctor'. Below the search bar is a table with the following columns: Name, Availability, Consultation Fee, Specialization, Approved, and Actions. The table contains two rows of data. The first row is for a doctor named 'Pyae' with an 'Available' status, a consultation fee of '\$1,000.00', and a specialization of 'Cancer'. The second row is for a doctor named 'Su' with an 'Available' status, a consultation fee of '\$2,500.00', and a specialization of 'Cancer'. Both doctors are marked as 'Approved'. The 'Actions' column for each row contains three buttons: 'Edit' (with a pencil icon), 'Details' (with an eye icon), and 'Delete' (with a trash can icon).

Name	Availability	Consultation Fee	Specialization	Approved	Actions
Pyae	Available	\$1,000.00	Cancer	Approved	Edit Details Delete
Su	Available	\$2,500.00	Cancer	Approved	Edit Details Delete

Figure: Doctors management dashboard



The screenshot shows the 'Add New Doctor' form. It has a green header with the text 'Add New Doctor'. The form contains several input fields and a dropdown menu. The 'Full Name' field has a placeholder 'Enter doctor's full name'. The 'Email' field has a placeholder 'Enter doctor's email'. The 'Password' field has a placeholder 'Set initial password'. The 'Phone Number' field is empty. The 'Specialization' field is a dropdown menu with the text '-- Select Specialization --'. The 'Consultation Fee' field is empty. There is a checkbox labeled 'Available' which is currently unchecked. At the bottom of the form are three buttons: 'Save' (green), 'Back to List' (grey), and 'Back to Dashboard' (white with a green border).

Add New Doctor

Full Name
Enter doctor's full name

Email
Enter doctor's email

Password
Set initial password

Phone Number

Specialization
-- Select Specialization --

Consultation Fee

☐ Available

[Save](#) [Back to List](#) [Back to Dashboard](#)

Figure: Creating a new doctor

Edit Doctor

Name

Specialization

ConsultationFee

☒ Availability

☒ IsApproved

Delete Doctor

Are you sure you want to delete this doctor?

Name	Pyae
Specialization	Cancer
Consultation Fee	1000.00

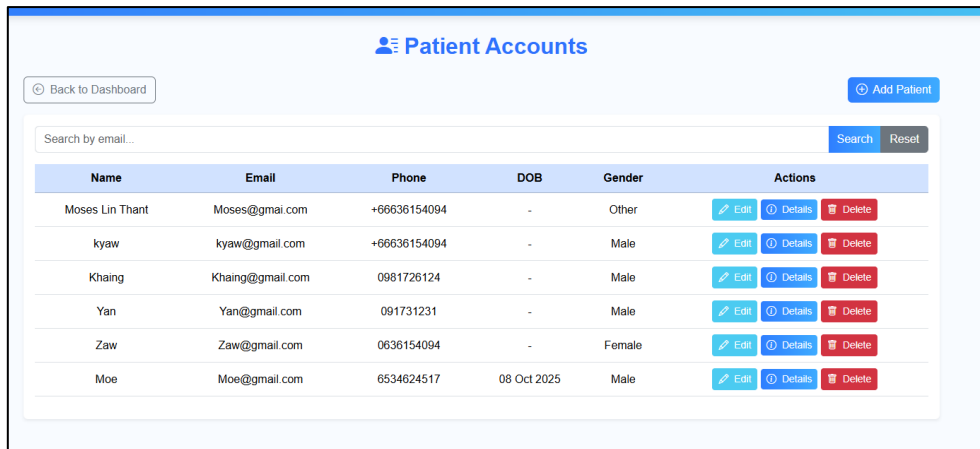
Figure: Edit and Delete Pages

Doctor Details	
Name	Su
Specialization	Cancer
Consultation Fee	\$2,500.00
Availability	True
Approved	<input checked="" type="button" value="Approved"/>
Phone Number	N/A
Email	su@gmail.com
<input type="button" value="Edit"/> <input type="button" value="Back"/>	

Figure: Doctor Detail Page

Patients Accounts Management Page

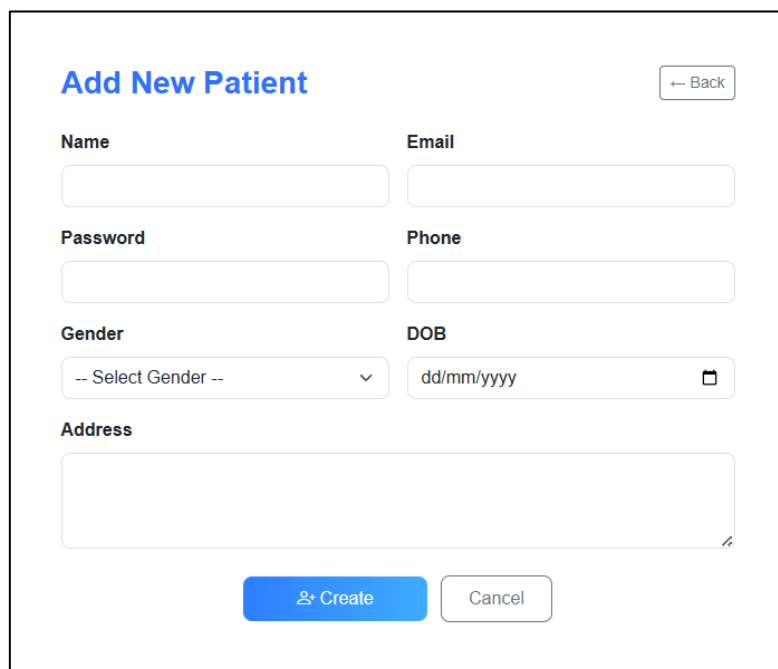
Similar to doctors account management page, an admin can do CRUD processes and find patient accounts by using search bar.



The interface shows a 'Patient Accounts' section with a 'Back to Dashboard' button and an 'Add Patient' button. A search bar labeled 'Search by email...' is present with 'Search' and 'Reset' buttons. Below is a table listing patient accounts with columns for Name, Email, Phone, DOB, Gender, and Actions. The table contains six entries: Moses Lin Thant, kyaw, Khaing, Yan, Zaw, and Moe. Each entry has corresponding 'Edit', 'Details', and 'Delete' buttons in the Actions column.

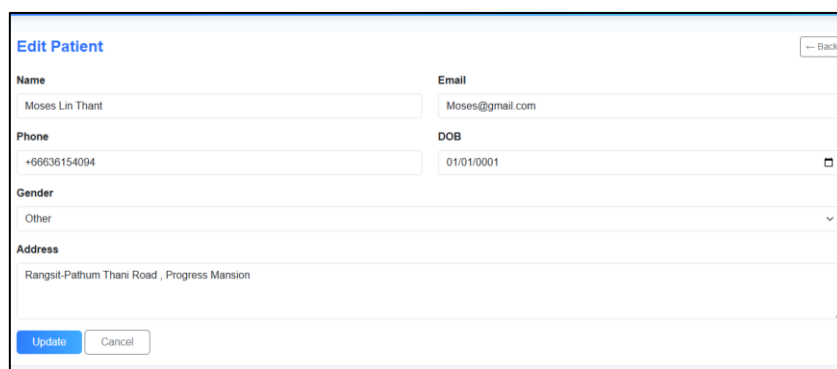
Name	Email	Phone	DOB	Gender	Actions
Moses Lin Thant	Moses@gmail.com	+66636154094	-	Other	Edit Details Delete
kyaw	kyaw@gmail.com	+66636154094	-	Male	Edit Details Delete
Khaing	Khaing@gmail.com	0981726124	-	Male	Edit Details Delete
Yan	Yan@gmail.com	091731231	-	Male	Edit Details Delete
Zaw	Zaw@gmail.com	0636154094	-	Female	Edit Details Delete
Moe	Moe@gmail.com	6534624517	08 Oct 2025	Male	Edit Details Delete

Figure: Patient Account Management



The 'Add New Patient' form includes a 'Back' button and input fields for Name, Email, Password, Phone, Gender (a dropdown menu), and DOB (a date picker). There is also a large text area for Address. At the bottom, there are 'Create' and 'Cancel' buttons.

Figure: Create Patient Form



The 'Edit Patient' form includes a 'Back' button and pre-filled input fields for Name (Moses Lin Thant), Email (Moses@gmail.com), Phone (+66636154094), and DOB (01/01/0001). The Gender dropdown is set to 'Other'. The Address field contains 'Rangsit-Pathum Thani Road , Progress Mansion'. At the bottom, there are 'Update' and 'Cancel' buttons.

Figure: Patient Edit Form

Patient Details

Name	kyaw
Email	kyaw@gmail.com
Phone	+66636154094
DOB	01 Jan 0001
Gender	Male
Address	Rangsit-Pathum Thani Road , Progress Mansion

Edit
Back

Delete Patient

Are you sure you want to delete this patient?

Name	Moses Lin Thant
Email	Moses@gmail.com
Phone	+66636154094

Delete
Cancel

Figure: Patient Edit and Delete Pages

Payment Management Page

In the payment management page, admin can view and edit payment information but he cannot create new payments.

Back to Dashboard

Payment ID	Appointment	Amount	Method	Date	Status	Actions
1	1	\$1,000.00	Cash	04 Oct 2025	Paid	View Edit Delete
4	2	\$5,000.00	Cash	17 Oct 2025	Paid	View Edit Delete
2	3	\$3,000.00	Cash	17 Oct 2025	Paid	View Edit Delete
3	4	\$3,000.00	Cash	17 Oct 2025	Paid	View Edit Delete
5	5	\$4,000.00	Cash	20 Oct 2025	Paid	View Edit Delete
6	6	\$1,000.00	Mobile Banking	20 Oct 2025	Paid	View Edit Delete

Figure: Payment Management Dashboard

Payment ID	1
Appointment	1
Amount	\$1,000.00
Payment Method	Cash
Date	04 Oct 2025
Status	Paid

Amount	1000.00
Payment Method	Cash

Figure: Detail and Edit payments page

Report Dashboard

In report dashboard, an can see the overall situation of the business. For example, how much income by month, patient age group, patient group by gender. An admin can download these report as a pdf file.

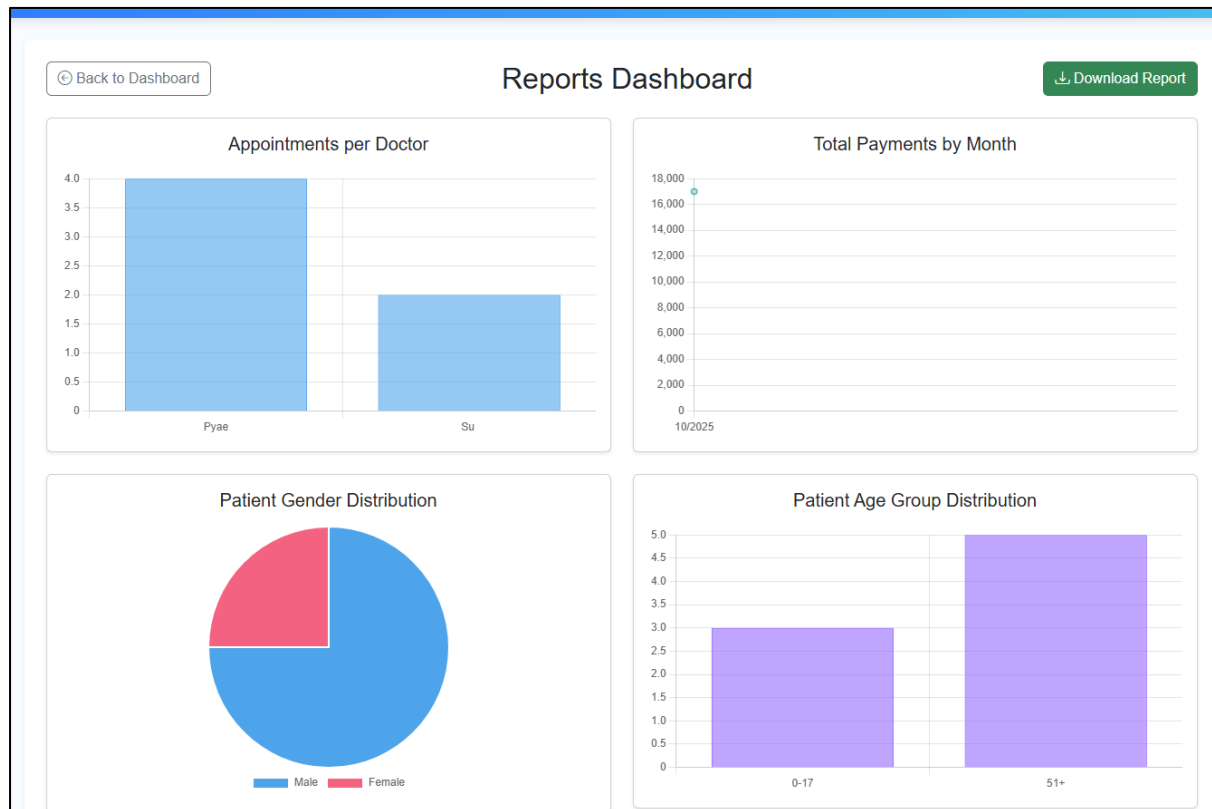


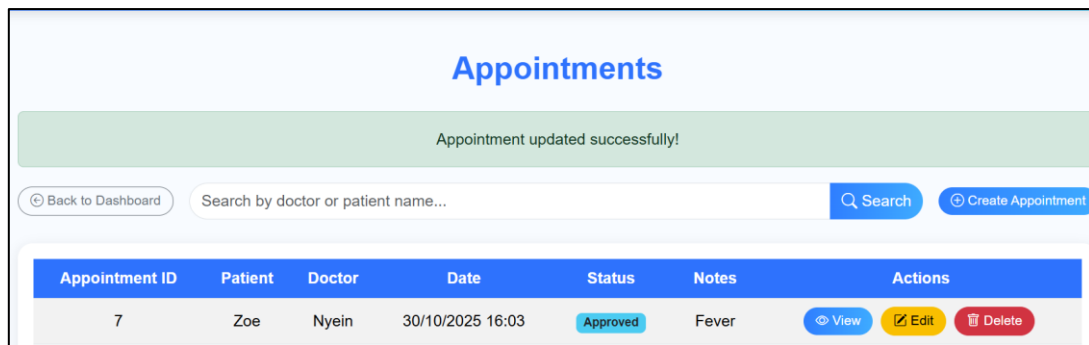
Figure: Report Dashboard

Admin Appointment Dashboard

In appoint dashboard, an admin's duty is to check the appointment details and decide to approve or not. After an admin decided to give an approvment, the status of appointment became "Approved". Only after that a doctor can see this appointment.

Back to Dashboard <input type="text" value="Search by doctor or patient name..."/> <input type="button" value="Search"/> <input type="button" value="Create Appointment"/>						
Appointment ID	Patient	Doctor	Date	Status	Notes	Actions
7	Zoe	Nyein	30/10/2025 16:03	Pending	Fever	View Edit Delete
6	kyaw	Pyae	21/10/2025 22:21	Completed	ill	View Edit Delete
5	Khaing	Su	20/10/2025 15:22	Completed		View Edit Delete
4	kyaw	Su	16/10/2025 19:40	Completed		View Edit Delete
3	kyaw	Pyae	10/10/2025 21:11	Completed		View Edit Delete
2	kyaw	Pyae	30/09/2025 15:13	Completed		View Edit Delete
1	kyaw	Pyae	29/09/2025 14:52	Completed	Fever	View Edit Delete

Figure: Before approve



Appointments

Appointment updated successfully!

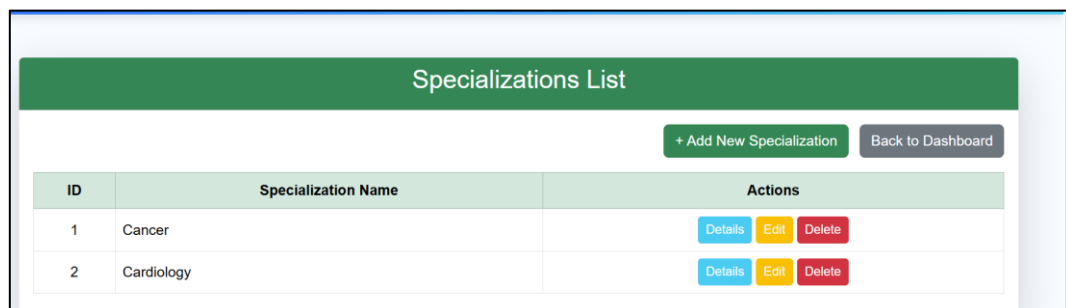
[Back to Dashboard](#) Search by doctor or patient name... [Search](#) [Create Appointment](#)

Appointment ID	Patient	Doctor	Date	Status	Notes	Actions
7	Zoe	Nyein	30/10/2025 16:03	Approved	Fever	View Edit Delete

Figure: After Approve

Specializations Dashboard

This dashboard is nothing special but only an admin can view, create and edit.



Specializations List

[+ Add New Specialization](#) [Back to Dashboard](#)

ID	Specialization Name	Actions
1	Cancer	Details Edit Delete
2	Cardiology	Details Edit Delete

Figure: Specialization Dashboard

Doctor User Guide

In doctor dashboard, a doctor can see today appointment and this week's appointment. A doctor can view his appointments, patients and give prescriptions to the patients. Doctors are restricted to view only their patients and appointments. Only the appointments which are approved by admin can be used to give prescription by a doctor.

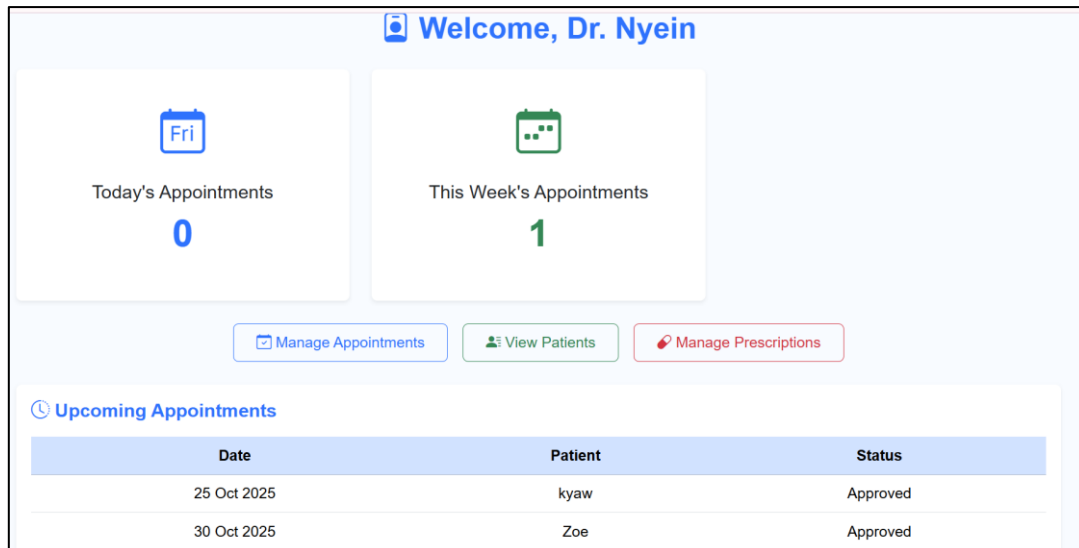


Figure: Doctor Dashboard

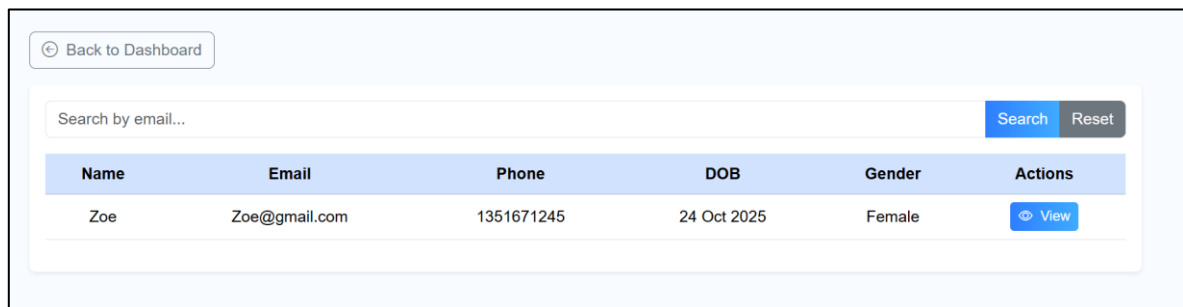


Figure: Current appointment

Create Prescription

Appointment

Appt #7 — 30/10/2025 16:03 — Zoe@gmail.com

DateIssued

24/10/2025

PrescriptionDetails

Take Three pills

DoctorNotes

Goodluck

Save Cancel

Figure: Create Prescription for the patient

Date Issued	Appointment	Prescription Details	Doctor Notes	Actions
24 Oct 2025	7	Take three pills	Goodluck	Edit Details Delete

Figure: Creating Prescription successful

Name	Email	Phone	DOB	Gender	Actions
kyaw	kyaw@gmail.com	+66636154094	-	Male	View
Zoe	Zoe@gmail.com	1351671245	24 Oct 2025	Female	View

Figure: A doctor can only view their patients

Appointment ID	Patient	Doctor	Date	Status	Notes	Actions
7	Zoe	Nyein	30/10/2025 16:03	Completed	Fever	View Edit
8	kyaw	Nyein	25/10/2025 19:40	Approved	Stomach	View Edit

Figure: A doctor can only view his appointments

Weaknesses and Future Development

The Online Healthcare Appointment System works well for small healthcare businesses but there are some issues with big businesses. There are some weak points that can be improved in the future.

One weakness is the lack of real-time notifications. When an appointment is created or cancelled, the system does not send a message or alert to the doctor or patient. Adding automatic notifications through email, SMS or pop-up alerts would make communication faster

and reduce missed appointments. The system also does not have automatic schedule checking and the system does not stop double bookings. A future version should include a smart scheduling future that checks for time conflicts and only allows available slots to be booked.

In terms of security , the system uses ASP.NET identity for login and authentication. But sensitive data like patient records and payment information are not encrypted. Adding advanced data encryption algorithms, HTTPS and following privacy standards such as GDPR or HIPAA will make the system more secure and trustworthy. And also the user interface (UI) can also be improved. Some pages, like the form pages, are too simple and not well arranged. The layout is not very friendly for mobile users.

Lastly, the system is not yet deployed online. In the future, it can be hosted on a cloud platform like Microsoft Azure or AWS so it can be used by real users and can grow easily as more people use it.

References

ASP.NET MVC Pattern | .NET. (n.d.). Microsoft. <https://dotnet.microsoft.com/en-us/apps/aspnet/mvc>

Contributors, M. O. J. T. a. B. (n.d.). *Get started with Bootstrap.*
<https://getbootstrap.com/docs/5.3/getting-started/introduction/>

rwestMSFT. (n.d.). *SQL Server Management Studio.* Microsoft Learn.
<https://learn.microsoft.com/en-us/ssms/>

Jcjiang. (n.d.). *Entity Framework documentation hub.* Microsoft Learn.
<https://learn.microsoft.com/en-us/ef/>

OpenJS Foundation - openjsf.org. (n.d.). *JQuery.* <https://jquery.com/>

studio96 & HEALTHCARE ENTERPRISES. (2024, November 4). *Home - Healthcare enterprises.* Healthcare Enterprises. <https://healthcareent.co.th/>