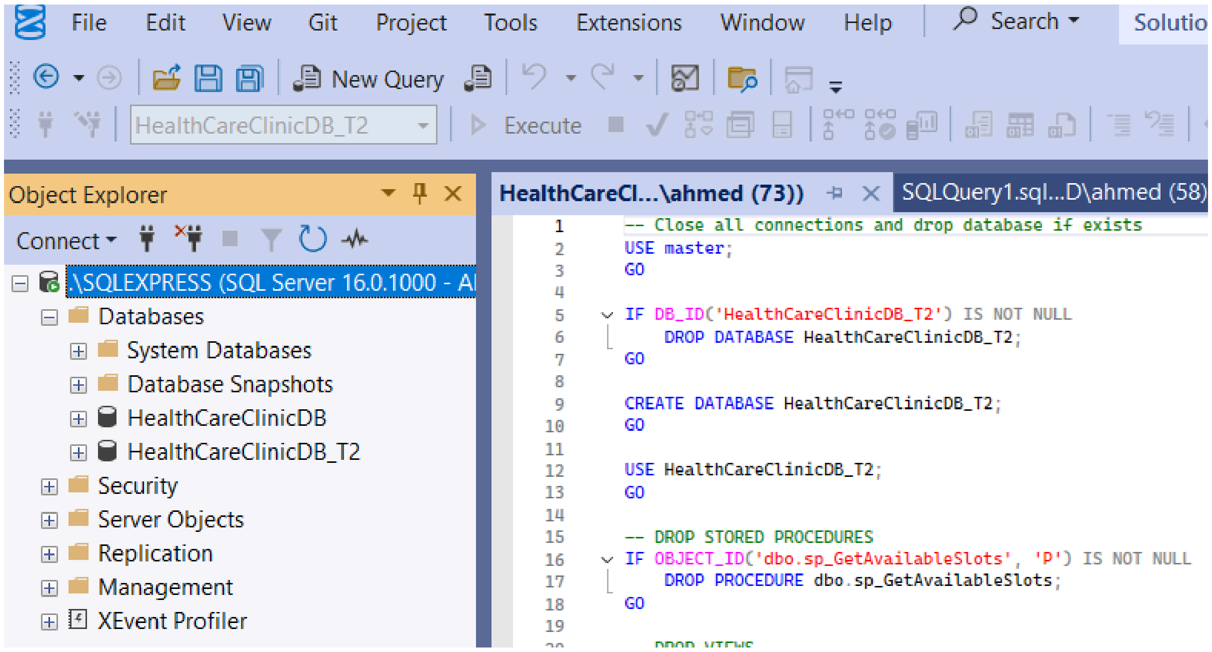
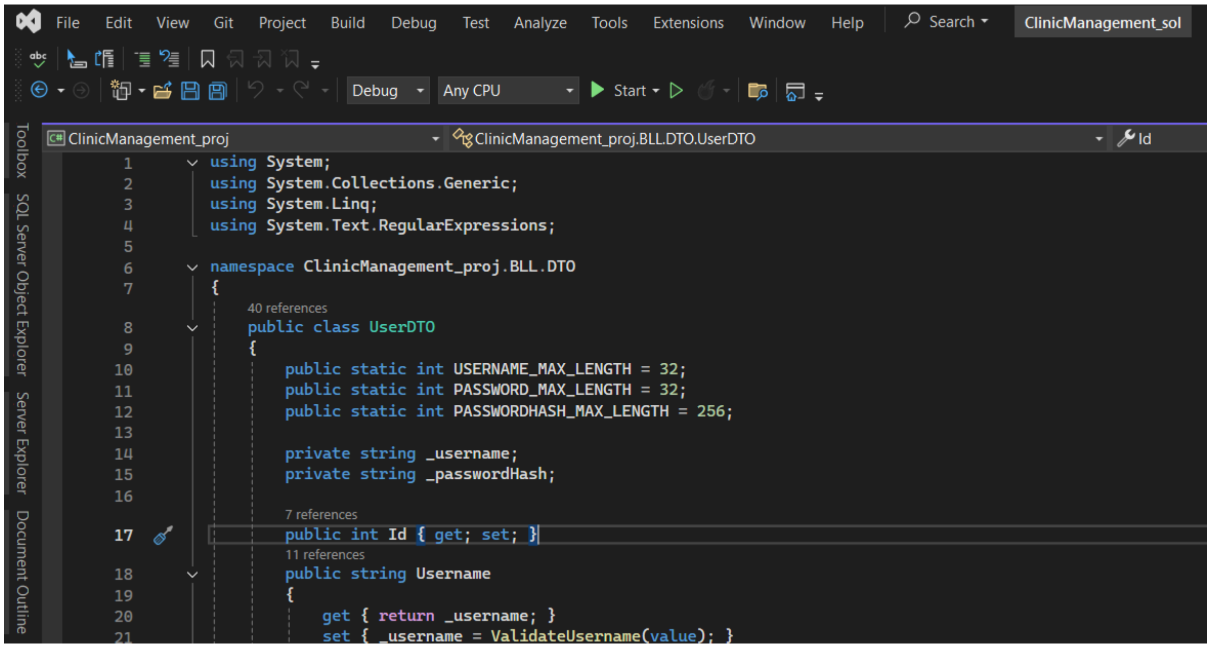
1- Software to use:

- Visual Studio 2022 (C# Language)
- Microsoft SQL Server 2022 (Server database)



```
.\SQLEXPRESS (SQL Server 16.0.1000 - A
    Databases
        System Databases
        Database Snapshots
        HealthCareClinicDB
        HealthCareClinicDB_T2
    Security
    Server Objects
    Replication
    Management
    XEvent Profiler
```

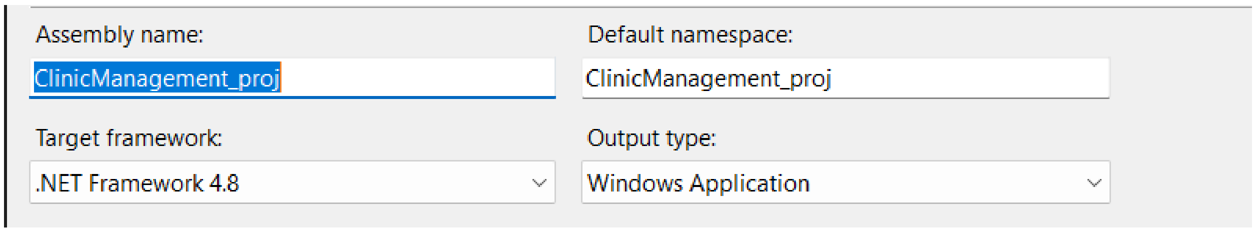
```sql
-- Close all connections and drop database if exists
USE master;
GO

IF DB_ID('HealthCareClinicDB_T2') IS NOT NULL
    DROP DATABASE HealthCareClinicDB_T2;
GO

CREATE DATABASE HealthCareClinicDB_T2;
GO

USE HealthCareClinicDB_T2;
GO

-- DROP STORED PROCEDURES
IF OBJECT_ID('dbo.sp_GetAvailableSlots', 'P') IS NOT NULL
    DROP PROCEDURE dbo.sp_GetAvailableSlots;
GO

```



```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;

namespace ClinicManagement_proj.BLL.DTO
{
    40 references
    public class UserDTO
    {
        public static int USERNAME_MAX_LENGTH = 32;
        public static int PASSWORD_MAX_LENGTH = 32;
        public static int PASSWORDHASH_MAX_LENGTH = 256;

        private string _username;
        private string _passwordHash;

        7 references
        public int Id { get; set; }
        11 references
        public string Username
        {
            get { return _username; }
            set { _username = ValidateUsername(value); }
```

2- Project Template to be used:

Windows Forms Application (.Net Framework)



Assembly name: ClinicManagement_proj

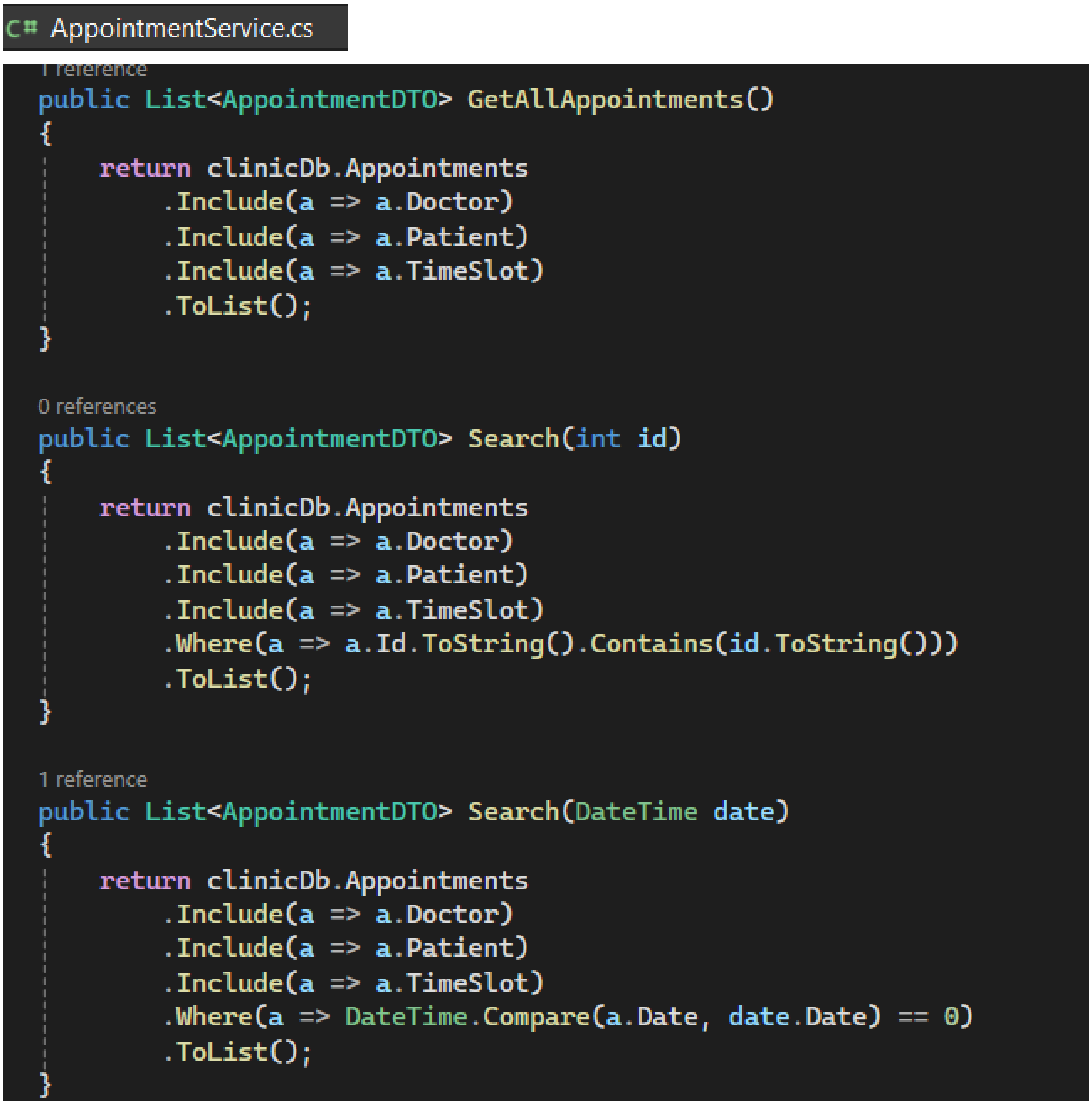
Default namespace: ClinicManagement_proj

Target framework: .NET Framework 4.8

Output type: Windows Application

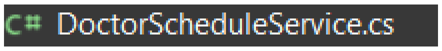
4- Project Code Source to be used:

LINQ-EF (LINQ To Entity)

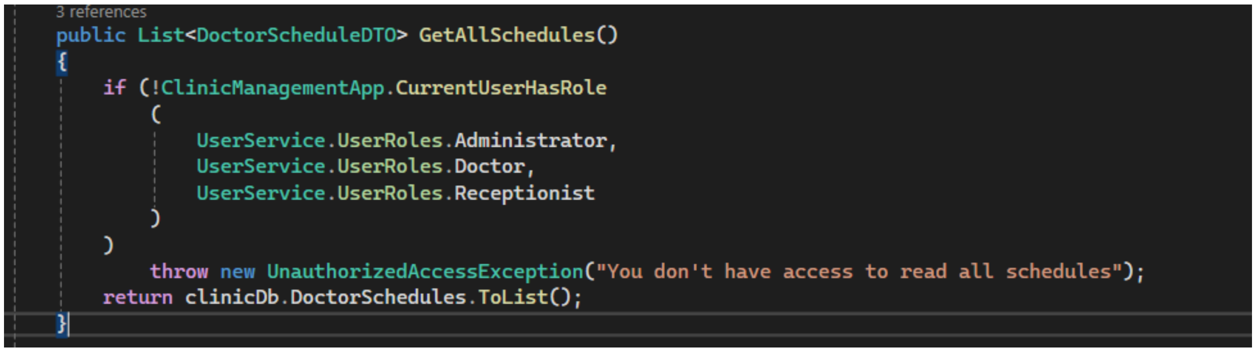


C# AppointmentService.cs

```
1 reference
public List<AppointmentDTO> GetAllAppointments()
{
    return clinicDb.Appointments
        .Include(a => a.Doctor)
        .Include(a => a.Patient)
        .Include(a => a.TimeSlot)
        .ToList();
}

0 references
public List<AppointmentDTO> Search(int id)
{
    return clinicDb.Appointments
        .Include(a => a.Doctor)
        .Include(a => a.Patient)
        .Include(a => a.TimeSlot)
        .Where(a => a.Id.ToString().Contains(id.ToString()))
        .ToList();
}

1 reference
public List<AppointmentDTO> Search(DateTime date)
{
    return clinicDb.Appointments
        .Include(a => a.Doctor)
        .Include(a => a.Patient)
        .Include(a => a.TimeSlot)
        .Where(a => DateTime.Compare(a.Date, date.Date) == 0)
        .ToList();
}
```

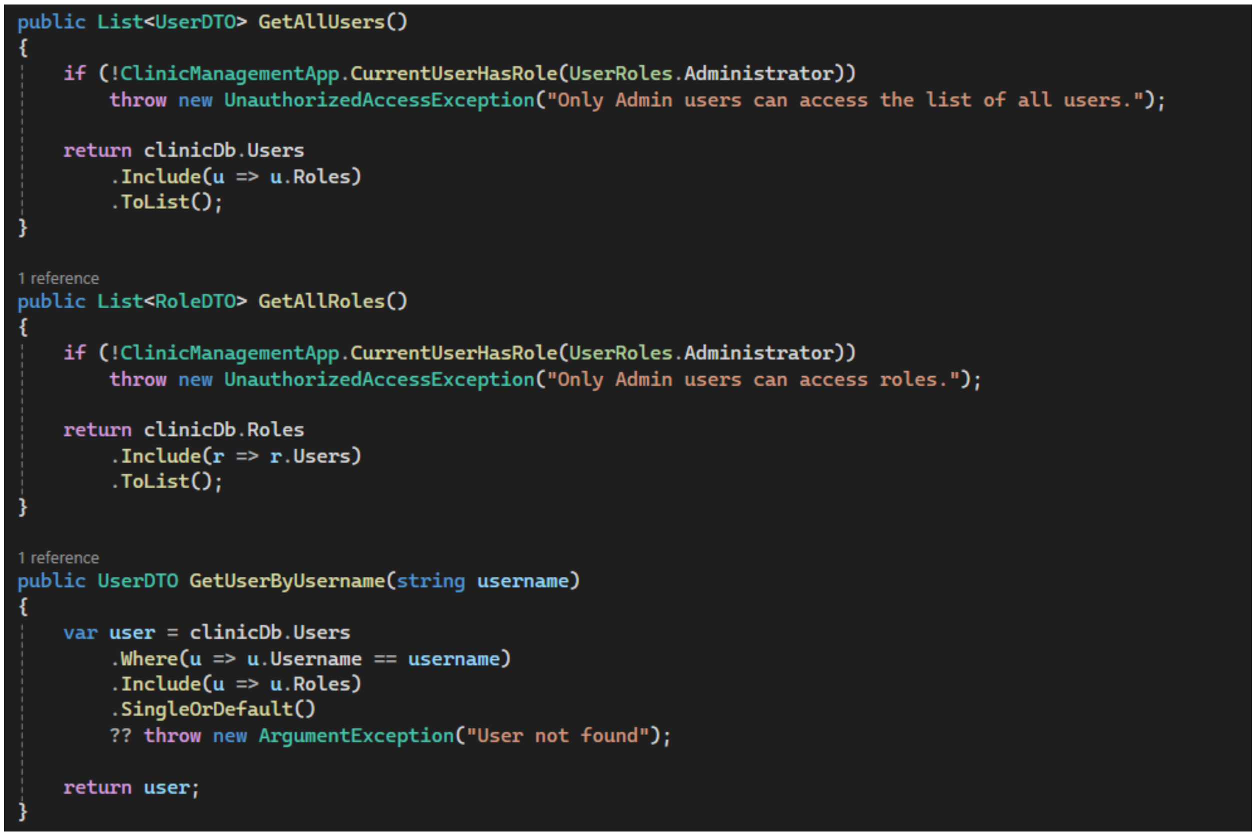


C# DoctorScheduleService.cs

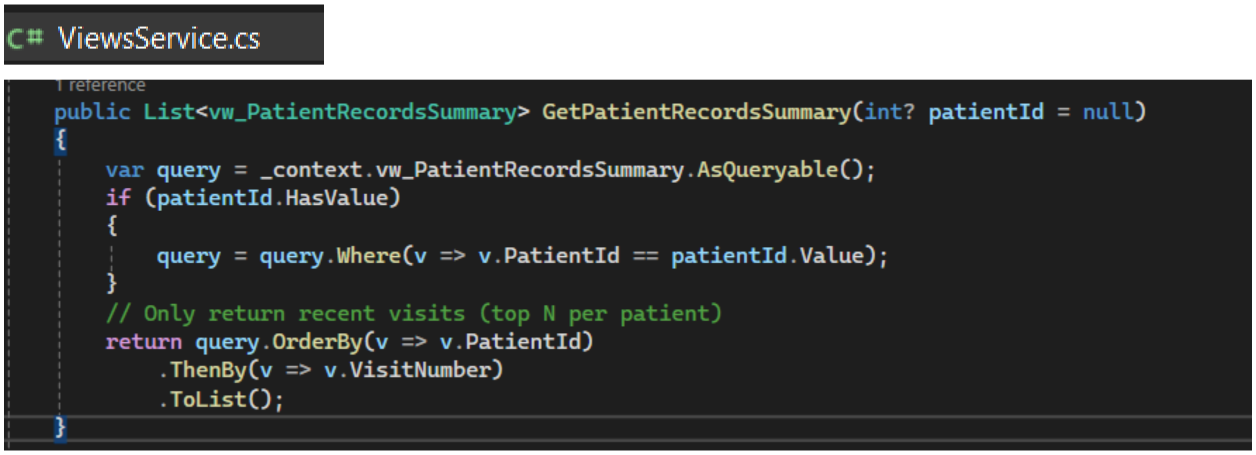


```
3 references
public List<DoctorScheduleDTO> GetAllSchedules()
{
    if (!ClinicManagementApp.CurrentUserHasRole
        (
            UserService.UserRoles.Administrator,
            UserService.UserRoles.Doctor,
            UserService.UserRoles.Receptionist
        )
    )
        throw new UnauthorizedAccessException("You don't have access to read all schedules");
    return clinicDb.DoctorSchedules.ToList();
}
```

C# UserService.cs



```
public List<UserDTO> GetAllUsers()
{
    if (!ClinicManagementApp.CurrentUserHasRole(UserRoles.Administrator))
        throw new UnauthorizedAccessException("Only Admin users can access the list of all users.");

    return clinicDb.Users
        .Include(u => u.Roles)
        .ToList();
}

1 reference
public List<RoleDTO> GetAllRoles()
{
    if (!ClinicManagementApp.CurrentUserHasRole(UserRoles.Administrator))
        throw new UnauthorizedAccessException("Only Admin users can access roles.");

    return clinicDb.Roles
        .Include(r => r.Users)
        .ToList();
}

1 reference
public UserDTO GetUserByUsername(string username)
{
    var user = clinicDb.Users
        .Where(u => u.Username == username)
        .Include(u => u.Roles)
        .SingleOrDefault()
        ?? throw new ArgumentException("User not found");

    return user;
}
```


C# ViewsService.cs

```
1 reference
public List<vw_PatientRecordsSummary> GetPatientRecordsSummary(int? patientId = null)
{
    var query = _context.vw_PatientRecordsSummary.AsQueryable();
    if (patientId.HasValue)
    {
        query = query.Where(v => v.PatientId == patientId.Value);
    }
    // Only return recent visits (top N per patient)
    return query.OrderBy(v => v.PatientId)
        .ThenBy(v => v.VisitNumber)
        .ToList();
}
```

5- Multi-Tiers Architecture

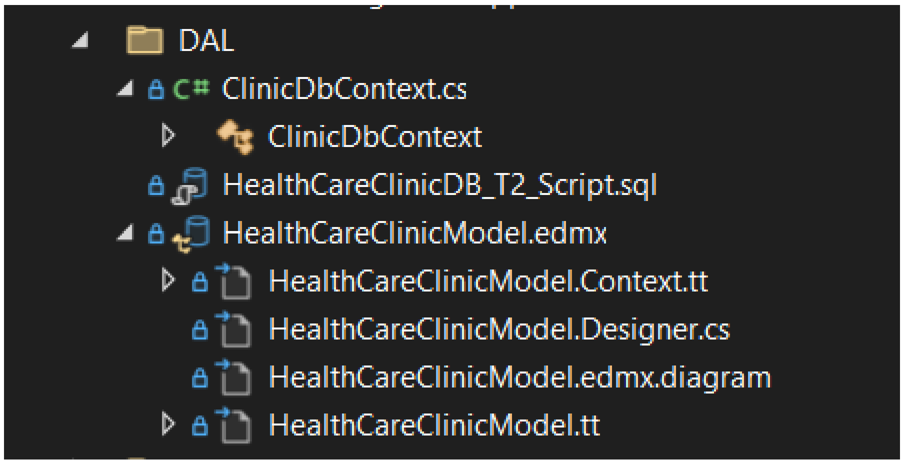
The Application must be structured as follows:

5.1- <u>DAL</u> (Data Access Layer)

- DbContext:

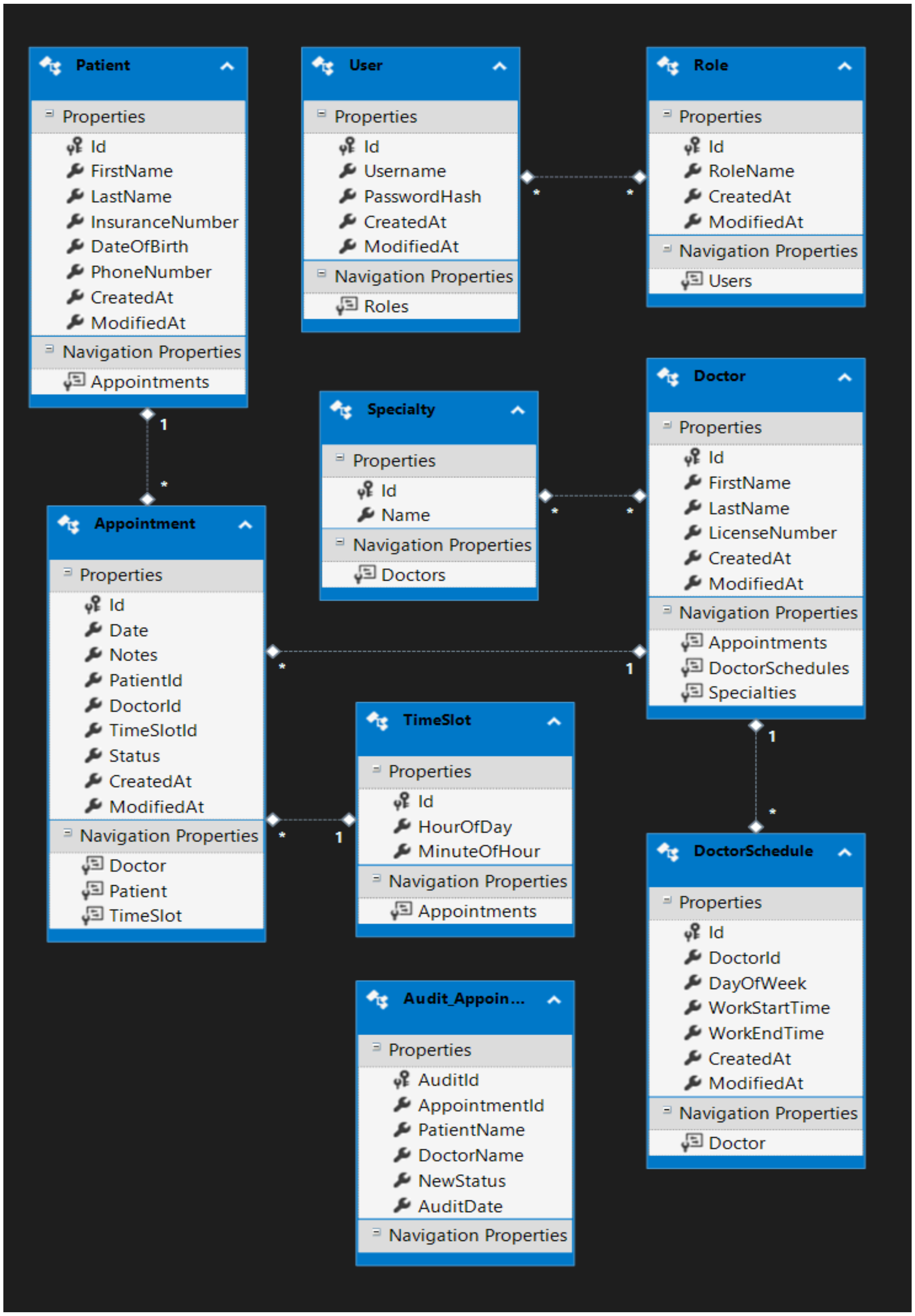
It contains the DbContext, which is used by the BLL to provide database connection.

Data access classes must be generated by the scaffolding.


```
DAL
    ClinicDbContext.cs
        ClinicDbContext
    HealthCareClinicDB_T2_Script.sql
    HealthCareClinicModel.edmx
        HealthCareClinicModel.Context.tt
        HealthCareClinicModel.Designer.cs
        HealthCareClinicModel.edmx.diagram
        HealthCareClinicModel.tt
```

- Database Schema:

It contains the database schema



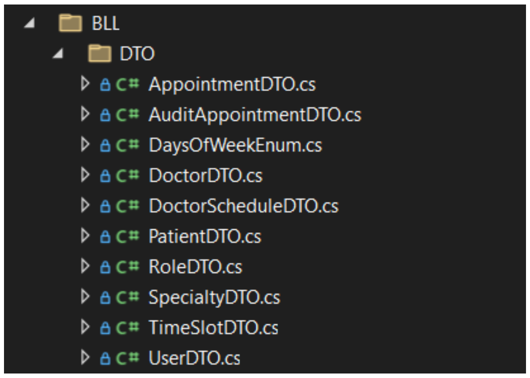
5.2- BLL (Business Logic Layer)

The back-end of the application, It must contain all the data exchange services

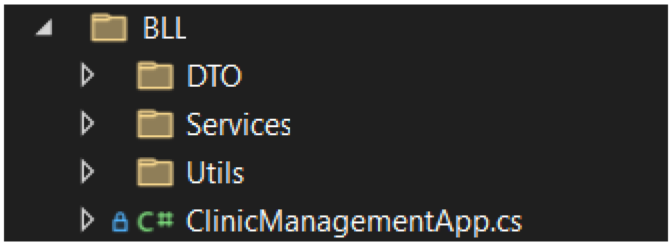
- DTO (Data Transfer Object)

Defines a common object shape that is used across the application-back-end to front-end

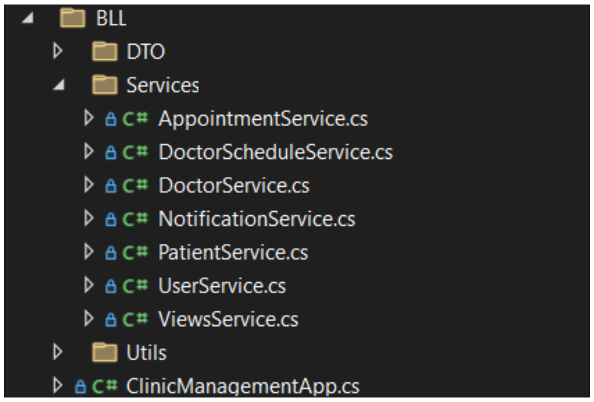
A common data exchange language



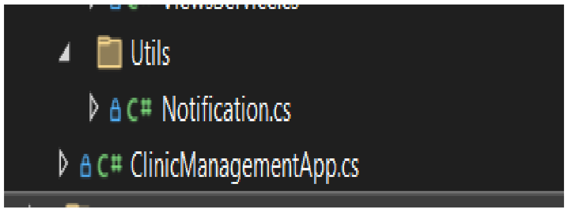
```
BLL
  DTO
    AppointmentDTO.cs
    AuditAppointmentDTO.cs
    DaysOfWeekEnum.cs
    DoctorDTO.cs
    DoctorScheduleDTO.cs
    PatientDTO.cs
    RoleDTO.cs
    SpecialtyDTO.cs
    TimeSlotDTO.cs
    UserDTO.cs
```



```
BLL
  DTO
  Services
  Utils
  ClinicManagementApp.cs
```



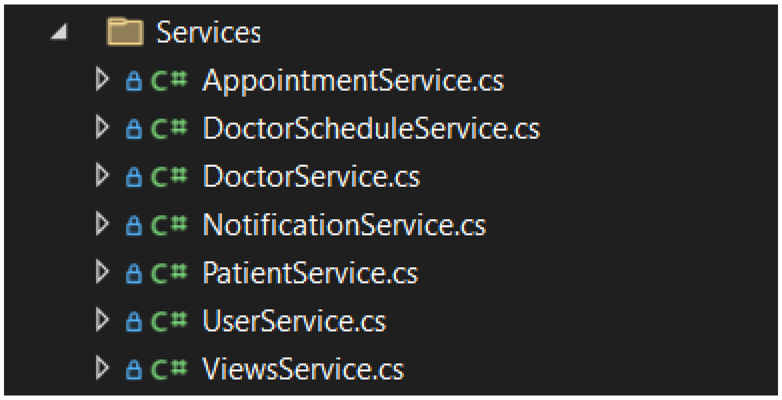
```
BLL
  DTO
  Services
    AppointmentService.cs
    DoctorScheduleService.cs
    DoctorService.cs
    NotificationService.cs
    PatientService.cs
    UserService.cs
    ViewsService.cs
  Utils
  ClinicManagementApp.cs
```



```
  Utils
    Notification.cs
  ClinicManagementApp.cs
```

- Services:

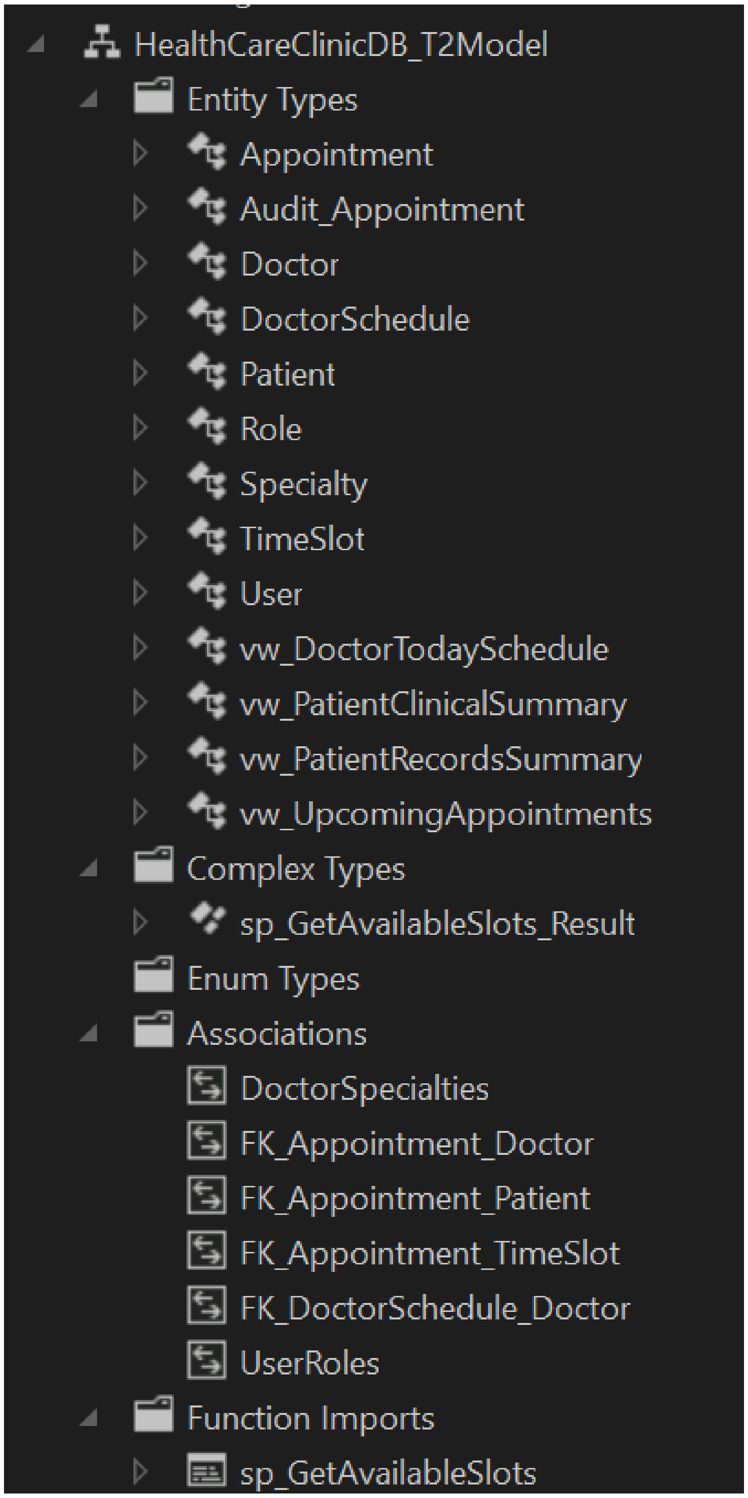
The Services define operations that are available to the application front-end and user
It also contains validation and data transformation logic



```
Services
    AppointmentService.cs
    DoctorScheduleService.cs
    DoctorService.cs
    NotificationService.cs
    PatientService.cs
    UserService.cs
    ViewsService.cs
```

- Models

Model classes implementation



```
HealthCareClinicDB_T2Model
  Entity Types
    Appointment
    Audit_Appointment
    Doctor
    DoctorSchedule
    Patient
    Role
    Specialty
    TimeSlot
    User
    vw_DoctorTodaySchedule
    vw_PatientClinicalSummary
    vw_PatientRecordsSummary
    vw_UpcomingAppointments
  Complex Types
    sp_GetAvailableSlots_Result
  Enum Types
  Associations
    DoctorSpecialties
    FK_Appointment_Doctor
    FK_Appointment_Patient
    FK_Appointment_TimeSlot
    FK_DoctorSchedule_Doctor
    UserRoles
  Function Imports
    sp_GetAvailableSlots
```

- Documentation:

It contains the technical documentation

We realized that if we write doc blocks on top of every function and class and use the docfx tool we can generate extensive technical documentation for the entire application,

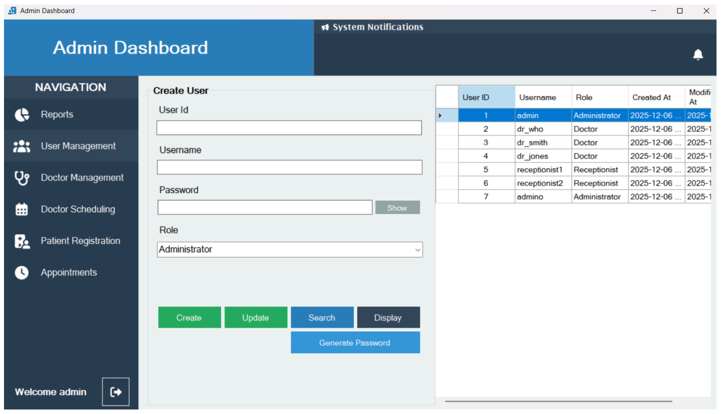
The file is presented in Documents/TechnicalDocumentation.pdf
But there is another one the way you intended in readme.txt and the same one just better styled in TechnicalDocumentation.md

\

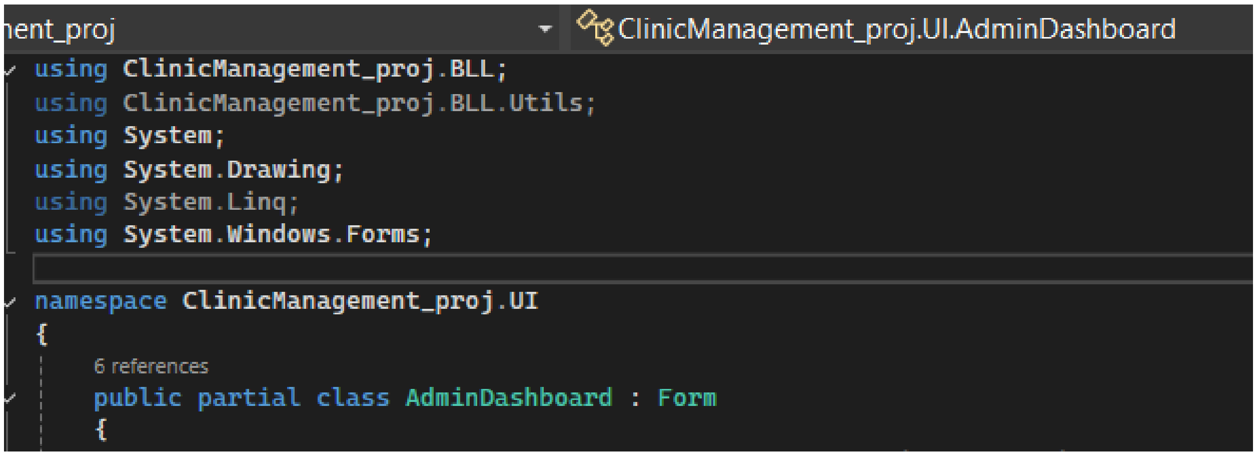
5.3- UI (Graphic User Interface Layer)

- It consumes DTO
- It creates instances of Services to perform data operations

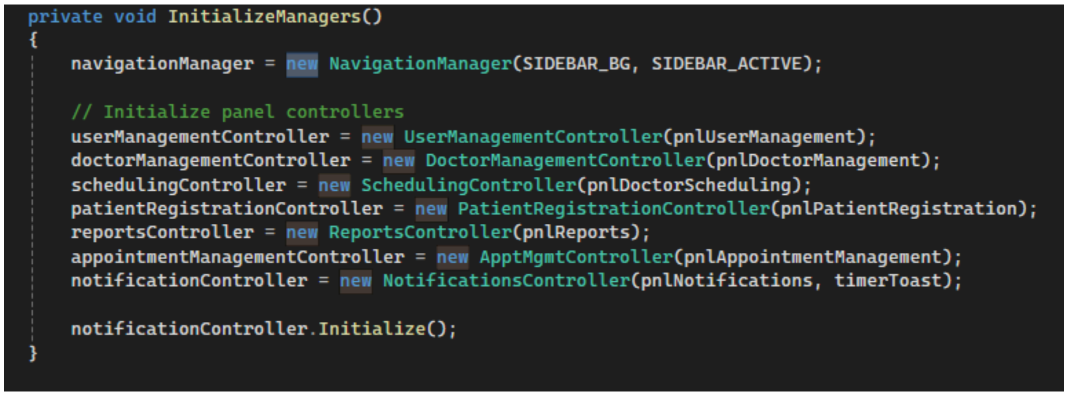
UI Layer must be built as multi-forms graphic application



User ID	Username	Role	Created At	Modifi At
1	admin	Administrator	2025-12-06 ...	2025-
2	dr_who	Doctor	2025-12-06 ...	2025-1
3	dr_smith	Doctor	2025-12-06 ...	2025-1
4	dr_jones	Doctor	2025-12-06 ...	2025-1
5	receptionist1	Receptionist	2025-12-06 ...	2025-1
6	receptionist2	Receptionist	2025-12-06 ...	2025-1
7	admino	Administrator	2025-12-06 ...	2025-1



```
using ClinicManagement_proj.BLL;
using ClinicManagement_proj.BLL.Utils;
using System;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace ClinicManagement_proj.UI
{
    6 references
    public partial class AdminDashboard : Form
    {
```



```
private void InitializeManagers()
{
    navigationManager = new NavigationManager(SIDEBAR_BG, SIDEBAR_ACTIVE);

    // Initialize panel controllers
    userManagementController = new UserManagementController(pnlUserManagement);
    doctorManagementController = new DoctorManagementController(pnlDoctorManagement);
    schedulingController = new SchedulingController(pnlDoctorScheduling);
    patientRegistrationController = new PatientRegistrationController(pnlPatientRegistration);
    reportsController = new ReportsController(pnlReports);
    appointmentManagementController = new ApptMgmtController(pnlAppointmentManagement);
    notificationController = new NotificationsController(pnlNotifications, timerToast);

    notificationController.Initialize();
}
```

Explanation

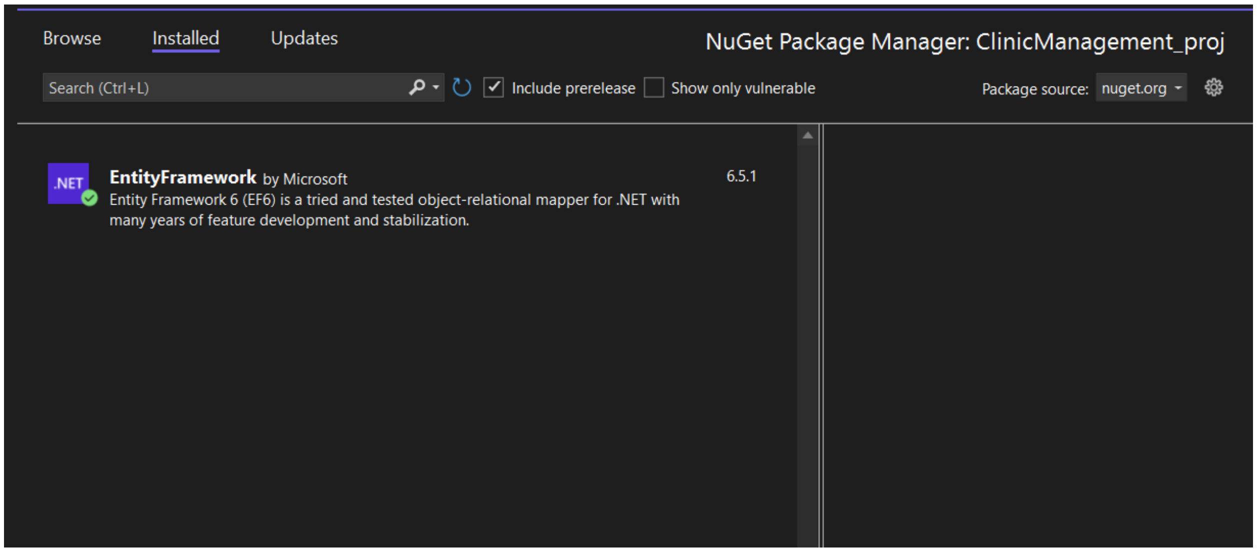
- We use a modular design that handles the panels that are shown and hidden away on button click, controllers to handle events such as OnHide, OnShow, Initialize to not repeat the code of each panel on each dashboard

Questions:

Q1 : Prepare the development environment.

1- Install required NuGet Packages.

Use Package Manager Console to install the Entity Framework package.



Browse · Installed · Updates

NuGet Package Manager: ClinicManagement_proj

Search (Ctrl+L) · ✓ Include prerelease · Show only vulnerable · Package source: nuget.org

EntityFramework by Microsoft 6.5.1
Entity Framework 6 (EF6) is a tried and tested object-relational mapper for .NET with many years of feature development and stabilization.

2- Configure Connection String (App.config) to connect the entity data model to the SQL Server database

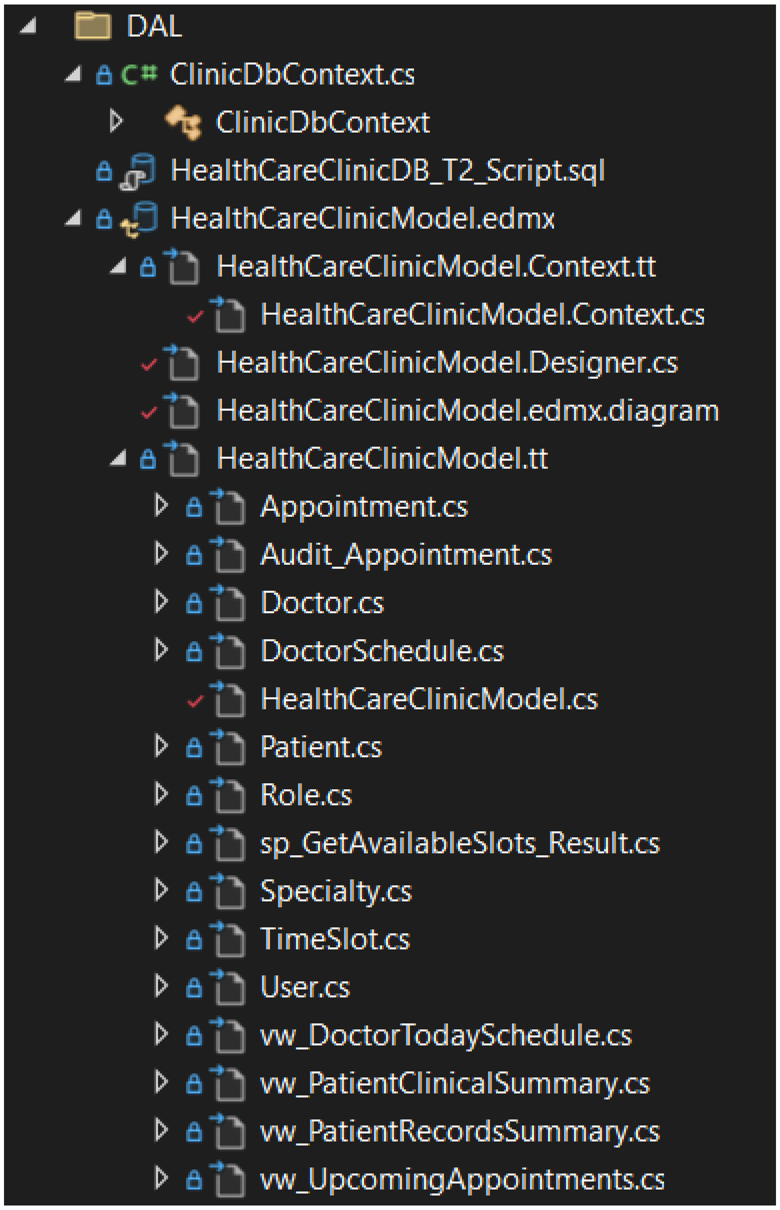
```
<connectionStrings>
      - - - -
      - - - - -
 </connectionStrings>
```

```
<connectionStrings>
<add name="HealthCareClinicDB_T2Entities"
connectionString="metadata=res://*/DAL.HealthCareClinicModel.csdl|res://*/DAL.HealthCareCli
nicModel.ssdl|res://*/DAL.HealthCareClinicModel.msl;provider=System.Data.SqlClient;provider
connection string=&quot;data source=.\SQLEXPRESS;initial
catalog=HealthCareClinicDB_T2;integrated
security=True;multipleactiveresultsets=True;encrypt=False;application
name=EntityFramework&quot;" providerName="System.Data.EntityClient" />
</connectionStrings>
```

Q2: Data Access Class Development

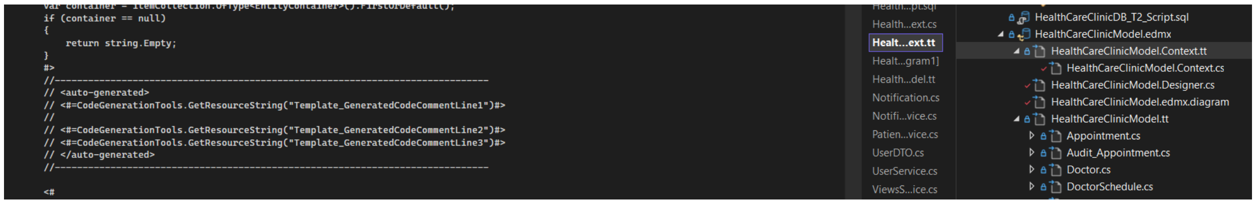
Technical Requirements :

- All data access classes must be created in a folder named « DAL ».



```
DAL
  ClinicDbContext.cs
    ClinicDbContext
  HealthCareClinicDB_T2_Script.sql
  HealthCareClinicModel.edmx
    HealthCareClinicModel.Context.tt
      HealthCareClinicModel.Context.cs
    HealthCareClinicModel.Designer.cs
    HealthCareClinicModel.edmx.diagram
    HealthCareClinicModel.tt
      Appointment.cs
      Audit_Appointment.cs
      Doctor.cs
      DoctorSchedule.cs
      HealthCareClinicModel.cs
      Patient.cs
      Role.cs
      sp_GetAvailableSlots_Result.cs
      Specialty.cs
      TimeSlot.cs
      User.cs
      vw_DoctorTodaySchedule.cs
      vw_PatientClinicalSummary.cs
      vw_PatientRecordsSummary.cs
      vw_UpcomingAppointments.cs
```

- Data access classes must be generated by the **scaffolding**



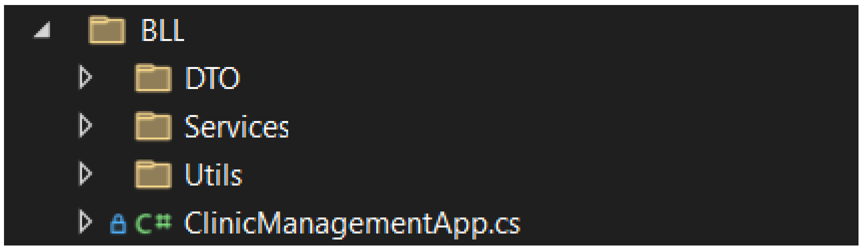
```
if (container == null)
{
    return string.Empty;
}
#>
//------------------------------------------------------------------------------
// <auto-generated>
// <#=CodeGenerationTools.GetResourceString("Template_GeneratedCodeCommentLine1")#>
//
// <#=CodeGenerationTools.GetResourceString("Template_GeneratedCodeCommentLine2")#>
// <#=CodeGenerationTools.GetResourceString("Template_GeneratedCodeCommentLine3")#>
// </auto-generated>
//------------------------------------------------------------------------------

<#
```

```
HealthCareClinicDB_T2_Script.sql
HealthCareClinicModel.edmx
  HealthCareClinicModel.Context.tt
    HealthCareClinicModel.Context.cs
  HealthCareClinicModel.Designer.cs
  HealthCareClinicModel.edmx.diagram
  HealthCareClinicModel.tt
    Appointment.cs
    Audit_Appointment.cs
    Doctor.cs
    DoctorSchedule.cs
```

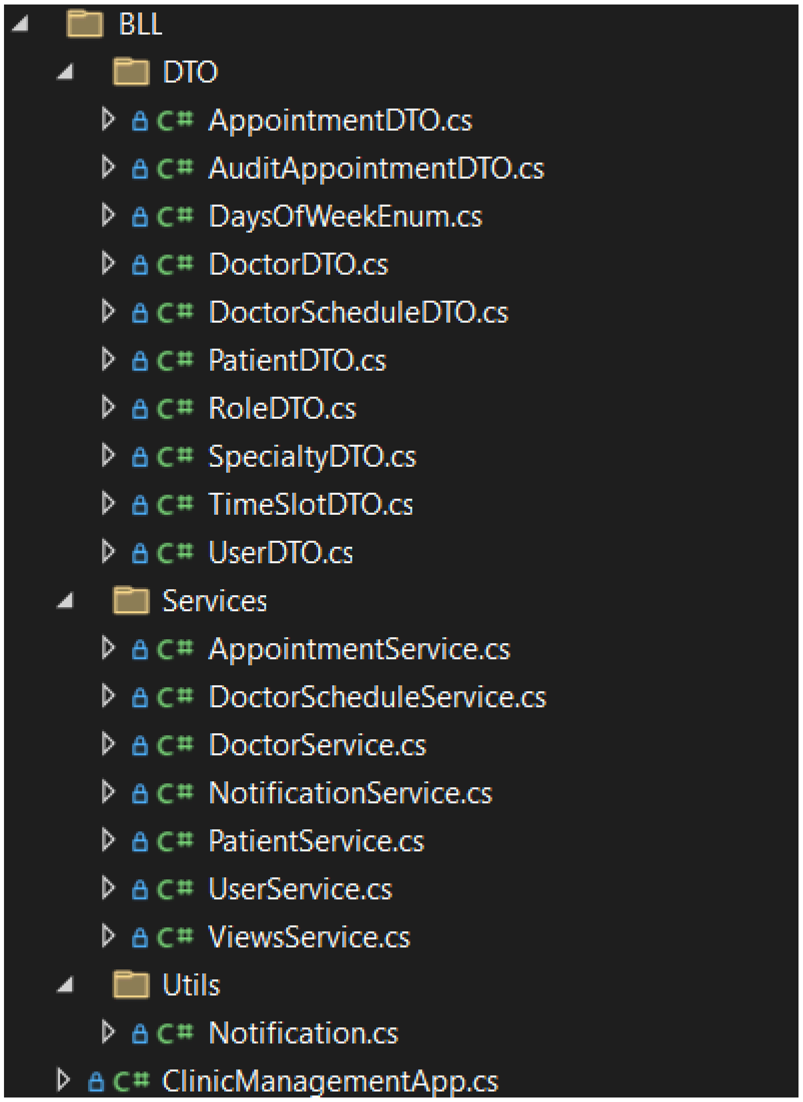
Q3: Business Classes Development

Technical Requirements :

- Business service classes use data access classes.



```
◢ BLL
  ▷ DTO
  ▷ Services
  ▷ Utils
  ▷ ClinicManagementApp.cs
```



```
◢ BLL
  ◢ DTO
    ▷ AppointmentDTO.cs
    ▷ AuditAppointmentDTO.cs
    ▷ DaysOfWeekEnum.cs
    ▷ DoctorDTO.cs
    ▷ DoctorScheduleDTO.cs
    ▷ PatientDTO.cs
    ▷ RoleDTO.cs
    ▷ SpecialtyDTO.cs
    ▷ TimeSlotDTO.cs
    ▷ UserDTO.cs
  ◢ Services
    ▷ AppointmentService.cs
    ▷ DoctorScheduleService.cs
    ▷ DoctorService.cs
    ▷ NotificationService.cs
    ▷ PatientService.cs
    ▷ UserService.cs
    ▷ ViewsService.cs
  ◢ Utils
    ▷ Notification.cs
  ▷ ClinicManagementApp.cs
```

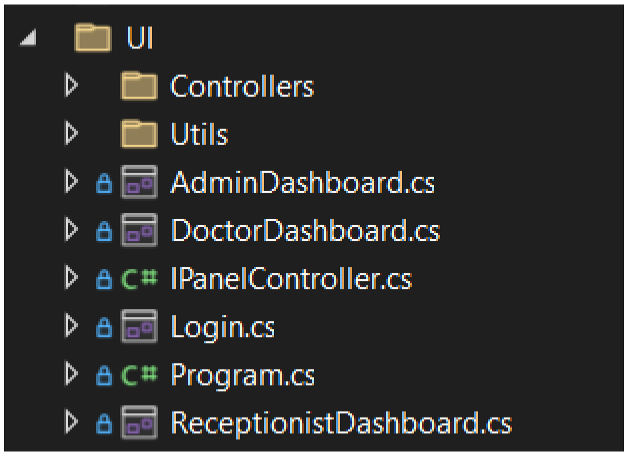
- Database connection information must already be added to the application configuration file.

```
<connectionStrings>
<add name="HealthCareClinicDB_T2Entities"
connectionString="metadata=res://*/DAL.HealthCareClinicModel.csdl|res://*/DAL.HealthCareCli
nicModel.ssdl|res://*/DAL.HealthCareClinicModel.msl;provider=System.Data.SqlClient;provider
connection string=&quot;data source=.\SQLEXPRESS;initial
catalog=HealthCareClinicDB_T2;integrated
security=True;multipleactiveresultsets=True;encrypt=False;application
name=EntityFramework&quot;" providerName="System.Data.EntityClient" />
</connectionStrings>
```

Q4: Presentation classes development

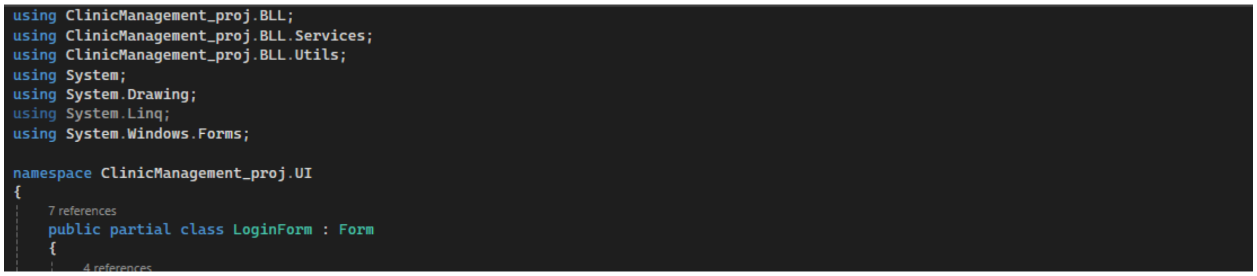
Technical Requirements :

- All presentation classes must be placed in a folder named « UI ».

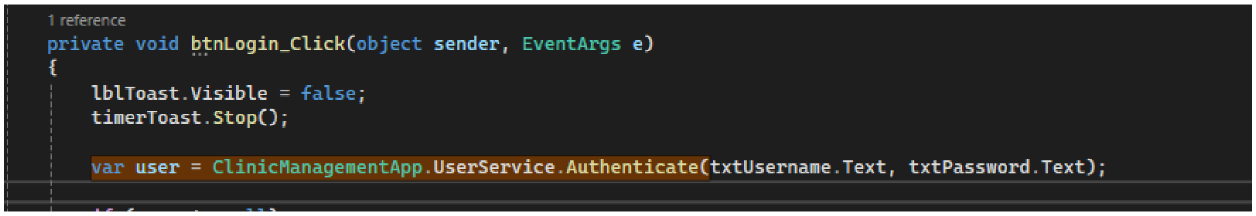


```
UI
    Controllers
    Utils
    AdminDashboard.cs
    DoctorDashboard.cs
    IPanelController.cs
    Login.cs
    Program.cs
    ReceptionistDashboard.cs
```

- Use of services classes and DTO classes in presentation classes.

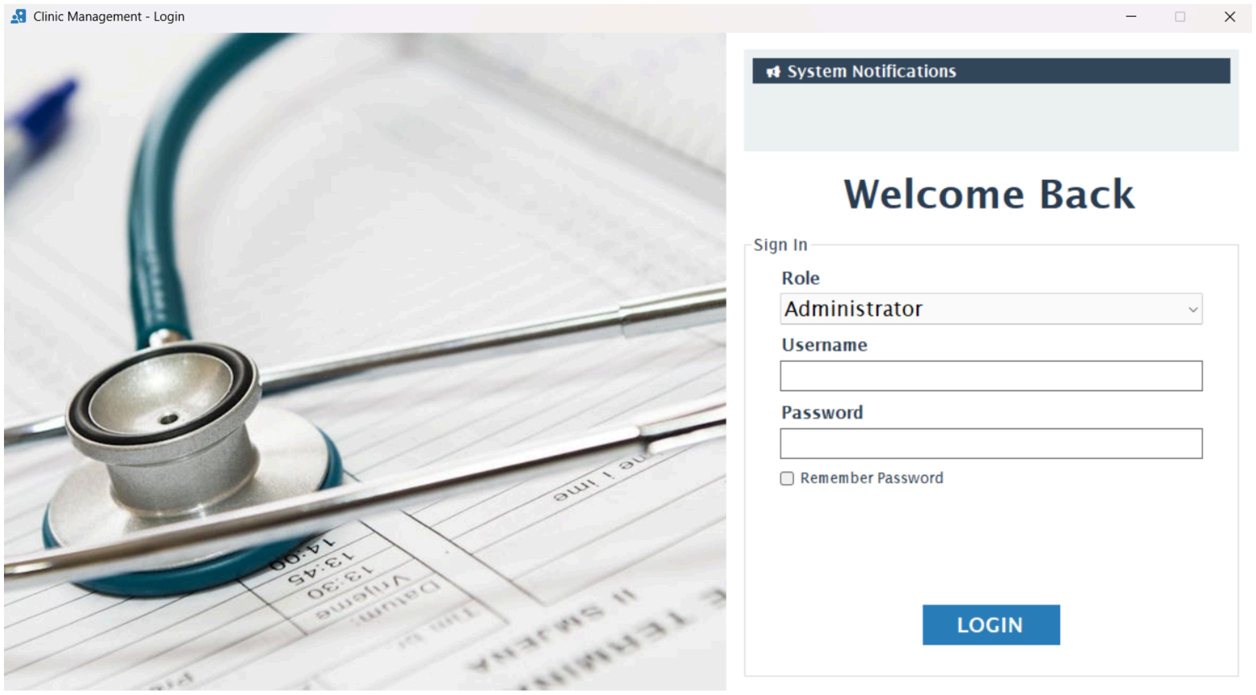


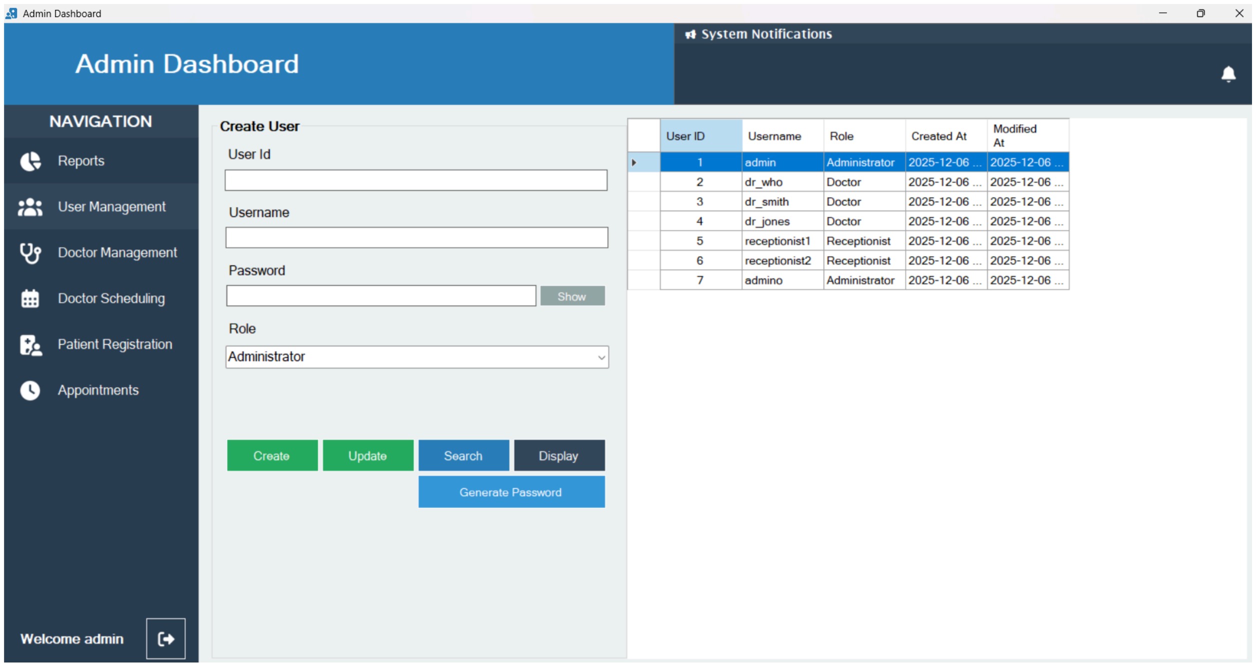
```
using ClinicManagement_proj.BLL;
using ClinicManagement_proj.BLL.Services;
using ClinicManagement_proj.BLL.Utils;
using System;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace ClinicManagement_proj.UI
{
    7 references
    public partial class LoginForm : Form
    {
        4 references
```



```
    1 reference
    private void btnLogin_Click(object sender, EventArgs e)
    {
        lblToast.Visible = false;
        timerToast.Stop();

        var user = ClinicManagementApp.UserService.Authenticate(txtUsername.Text, txtPassword.Text);
```

- Use of appropriate Windows Forms controls to create a user-friendly interface





User ID	Username	Role	Created At	Modified At
1	admin	Administrator	2025-12-06 ...	2025-12-06 ...
2	dr_who	Doctor	2025-12-06 ...	2025-12-06 ...
3	dr_smith	Doctor	2025-12-06 ...	2025-12-06 ...
4	dr_jones	Doctor	2025-12-06 ...	2025-12-06 ...
5	receptionist1	Receptionist	2025-12-06 ...	2025-12-06 ...
6	receptionist2	Receptionist	2025-12-06 ...	2025-12-06 ...
7	admino	Administrator	2025-12-06 ...	2025-12-06 ...

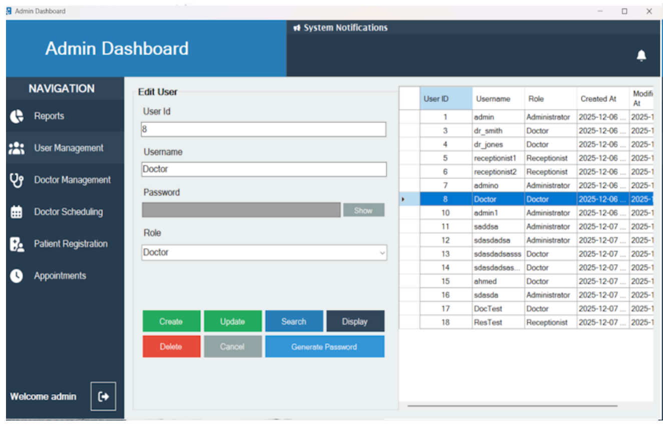
Q5. Control the quality of the application

Technical Requirements :

- *Test this 3-tier architecture application and report the test results in a file named: «* Health-Clinic_Test_Plan».
- Test all update operations at the interface level, then verify that these updates are effective in the database under SQL Server.

The TestPlan.pdf file delves in depth for all the test cases

The admin updated user ID 8, changing the name from 'Doctor' to 'Doctortwo'.



Edit User

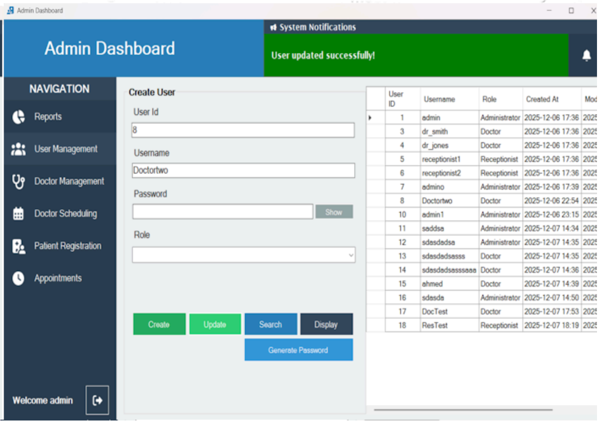
User Id: 8

Username: Doctor

Password:

Role: Doctor

User ID	Username	Role	Created At	Modifi At
1	admin	Administrator	2025-12-06 ...	2025-1
3	dr_smith	Doctor	2025-12-06 ...	2025-1
4	dr_jones	Doctor	2025-12-06 ...	2025-1
5	receptionist1	Receptionist	2025-12-06 ...	2025-1
6	receptionist2	Receptionist	2025-12-06 ...	2025-1
7	admino	Administrator	2025-12-06 ...	2025-1
8	Doctor	Doctor	2025-12-06 ...	2025-1
10	admin1	Administrator	2025-12-06 ...	2025-1
11	saddsa	Administrator	2025-12-07 ...	2025-1
12	sdasdadsa	Administrator	2025-12-07 ...	2025-1
13	sdasdadsasss	Doctor	2025-12-07 ...	2025-1
14	sdasdadsas...	Doctor	2025-12-07 ...	2025-1
15	ahmed	Doctor	2025-12-07 ...	2025-1
16	sdasda	Administrator	2025-12-07 ...	2025-1
17	DocTest	Doctor	2025-12-07 ...	2025-1
18	ResTest	Receptionist	2025-12-07 ...	2025-1



User updated successfully!

Create User

User Id: 8

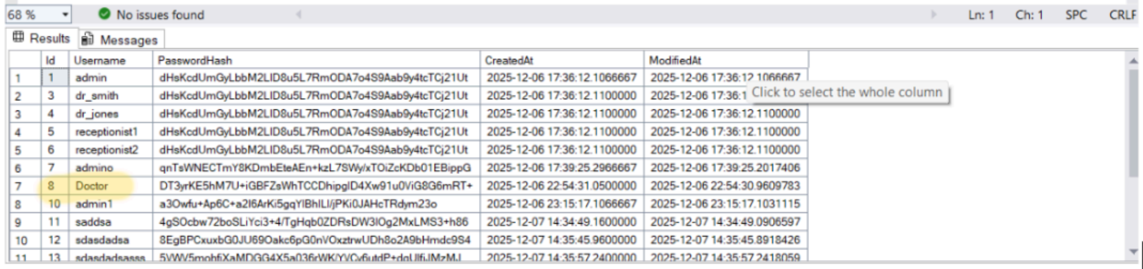
Username: Doctortwo

Password:

Role:

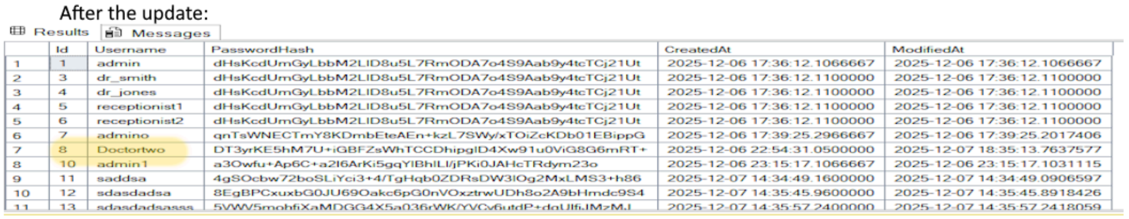
User ID	Username	Role	Created At	Mod
1	admin	Administrator	2025-12-06 17:36	2025
3	dr_smith	Doctor	2025-12-06 17:36	2025
4	dr_jones	Doctor	2025-12-06 17:36	2025
5	receptionist1	Receptionist	2025-12-06 17:36	2025
6	receptionist2	Receptionist	2025-12-06 17:36	2025
7	admino	Administrator	2025-12-06 17:39	2025
8	Doctortwo	Doctor	2025-12-06 22:54	2025
10	admin1	Administrator	2025-12-06 23:15	2025
11	saddsa	Administrator	2025-12-07 14:34	2025
12	sdasdadsa	Administrator	2025-12-07 14:35	2025
13	sdasdadsasss	Doctor	2025-12-07 14:35	2025
14	sdasdadsasssaaa	Doctor	2025-12-07 14:36	2025
15	ahmed	Doctor	2025-12-07 14:39	2025
16	sdasda	Administrator	2025-12-07 14:50	2025
17	DocTest	Doctor	2025-12-07 17:53	2025
18	ResTest	Receptionist	2025-12-07 18:19	2025

Before the update:



	Id	Username	PasswordHash	CreatedAt	ModifiedAt
1	1	admin	dHsKcdUmGyLbbM2LID8u5L7RmODA7o4S9Aab9y4tcTCj21Ut	2025-12-06 17:36:12.1066667	2025-12-06 17:36:12.1066667
2	3	dr_smith	dHsKcdUmGyLbbM2LID8u5L7RmODA7o4S9Aab9y4tcTCj21Ut	2025-12-06 17:36:12.1100000	2025-12-06 17:36:1
3	4	dr_jones	dHsKcdUmGyLbbM2LID8u5L7RmODA7o4S9Aab9y4tcTCj21Ut	2025-12-06 17:36:12.1100000	2025-12-06 17:36:12.1100000
4	5	receptionist1	dHsKcdUmGyLbbM2LID8u5L7RmODA7o4S9Aab9y4tcTCj21Ut	2025-12-06 17:36:12.1100000	2025-12-06 17:36:12.1100000
5	6	receptionist2	dHsKcdUmGyLbbM2LID8u5L7RmODA7o4S9Aab9y4tcTCj21Ut	2025-12-06 17:36:12.1100000	2025-12-06 17:36:12.1100000
6	7	admino	qnTsWNECTmY8KDmbEteAEn+kzL7SWy/xTOiZcKDb01EBippG	2025-12-06 17:39:25.2966667	2025-12-06 17:39:25.2017406
7	8	Doctor	DT3yrKE5hM7U+iGBFZsWhTCCDhipgID4Xw91u0ViG8G6mRT+	2025-12-06 22:54:31.0500000	2025-12-06 22:54:30.9609783
8	10	admin1	a3Owfu+Ap6C+a2I6ArKi5gqYIBhILI/jPKi0JAHcTRdym23o	2025-12-06 23:15:17.1066667	2025-12-06 23:15:17.1031115
9	11	saddsa	4gSOcbw72boSLiYci3+4/TgHqb0ZDRsDW3IOg2MxLMS3+h86	2025-12-07 14:34:49.1600000	2025-12-07 14:34:49.0906597
10	12	sdasdadsa	8EgBPCxuxbG0JU69Oakc6pG0nVOxztrwUDh8o2A9bHmdc9S4	2025-12-07 14:35:45.9600000	2025-12-07 14:35:45.8918426
11	13	sdasdadsasss	5VWV5mohfiXaMDGG4X5a036rWK/YVCv6utdP+dqUlfiJMzMJ	2025-12-07 14:35:57.2400000	2025-12-07 14:35:57.2418059

After the update:



	Id	Username	PasswordHash	CreatedAt	ModifiedAt
1	1	admin	dHsKcdUmGyLbbM2LID8u5L7RmODA7o4S9Aab9y4tcTCj21Ut	2025-12-06 17:36:12.1066667	2025-12-06 17:36:12.1066667
2	3	dr_smith	dHsKcdUmGyLbbM2LID8u5L7RmODA7o4S9Aab9y4tcTCj21Ut	2025-12-06 17:36:12.1100000	2025-12-06 17:36:12.1100000
3	4	dr_jones	dHsKcdUmGyLbbM2LID8u5L7RmODA7o4S9Aab9y4tcTCj21Ut	2025-12-06 17:36:12.1100000	2025-12-06 17:36:12.1100000
4	5	receptionist1	dHsKcdUmGyLbbM2LID8u5L7RmODA7o4S9Aab9y4tcTCj21Ut	2025-12-06 17:36:12.1100000	2025-12-06 17:36:12.1100000
5	6	receptionist2	dHsKcdUmGyLbbM2LID8u5L7RmODA7o4S9Aab9y4tcTCj21Ut	2025-12-06 17:36:12.1100000	2025-12-06 17:36:12.1100000
6	7	admino	qnTsWNECTmY8KDmbEteAEn+kzL7SWy/xTOiZcKDb01EBippG	2025-12-06 17:39:25.2966667	2025-12-06 17:39:25.2017406
7	8	Doctortwo	DT3yrKE5hM7U+iGBFZsWhTCCDhipgID4Xw91u0ViG8G6mRT+	2025-12-06 22:54:31.0500000	2025-12-07 18:35:13.7637577
8	10	admin1	a3Owfu+Ap6C+a2I6ArKi5gqYIBhILI/jPKi0JAHcTRdym23o	2025-12-06 23:15:17.1066667	2025-12-06 23:15:17.1031115
9	11	saddsa	4gSOcbw72boSLiYci3+4/TgHqb0ZDRsDW3IOg2MxLMS3+h86	2025-12-07 14:34:49.1600000	2025-12-07 14:34:49.0906597
10	12	sdasdadsa	8EgBPCxuxbG0JU69Oakc6pG0nVOxztrwUDh8o2A9bHmdc9S4	2025-12-07 14:35:45.9600000	2025-12-07 14:35:45.8918426
11	13	sdasdadsasss	5VWV5mohfiXaMDGG4X5a036rWK/YVCv6utdP+dqUlfiJMzMJ	2025-12-07 14:35:57.2400000	2025-12-07 14:35:57.2418059