```
using System;
using System.Collections.Generic;

namespace CustomSupport.Models;

public partial class CustLogInfo
{
    public int LogId { get; set; }

    public string? CustEmail { get; set; }

    public string? CustName { get; set; }

    public string? LogStatus { get; set; }

    public int? UserId { get; set; }

    public string? Description { get; set; }

    public virtual UserInfo? User { get; set; }
}
using System;
using System.Collections.Generic;

namespace CustomSupport.Models;

public partial class UserInfo
{
    public int UserId { get; set; }

    public string? Email { get; set; }

    public string? Password { get; set; }

    public virtual ICollection<CustLogInfo> CustLogInfos { get; set; } = new
List<CustLogInfo>();
}
using System;
using System.Collections.Generic;
using System.Reflection.Emit;
using Microsoft.EntityFrameworkCore;

namespace CustomSupport.Models;

public partial class EndDbContext : DbContext
{
    public EndDbContext()
    {
    }

    public EndDbContext(DbContextOptions<EndDbContext> options)
        : base(options)
    {
    }

    public virtual DbSet<CustLogInfo> CustLogInfos { get; set; }

    public virtual DbSet<UserInfo> UserInfos { get; set; }
```

```
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
#warning To protect potentially sensitive information in your connection string, you should move it out of source code. You can avoid scaffolding the connection string by using the Name= syntax to read it from configuration - see https://go.microsoft.com/fwlink/?linkid=2131148. For more guidance on storing connection strings, see http://go.microsoft.com/fwlink/?LinkId=723263.
        =>
optionsBuilder.UseSqlServer("Server=tcp:simplonatech.database.windows.net,1433;Initial Catalog=CustomSupportDb;User ID=PrashastVats;Password=Ankur23050105!;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;");

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<CustLogInfo>(entity =>
        {
            entity.HasKey(e => e.LogId).HasName("PK__CustLogI__5E5486482B28316E");

            entity.ToTable("CustLogInfo");

            entity.Property(e => e.LogId).ValueGeneratedNever();
            entity.Property(e => e.CustEmail).HasMaxLength(100);
            entity.Property(e => e.CustName).HasMaxLength(50);
            entity.Property(e => e.Description).HasMaxLength(50);
            entity.Property(e => e.LogStatus).HasMaxLength(50);

            entity.HasOne(d => d.User).WithMany(p => p.CustLogInfos)
                .HasForeignKey(d => d.UserId)
                .HasConstraintName("FK__CustLogIn__UserI__5EBF139D");
        });

        modelBuilder.Entity<UserInfo>(entity =>
        {
            entity.HasKey(e => e.UserId).HasName("PK__UserInfo__1788CC4C3C6A3929");

            entity.ToTable("UserInfo");

            entity.Property(e => e.UserId).ValueGeneratedNever();
            entity.Property(e => e.Email).HasMaxLength(100);
            entity.Property(e => e.Password).HasMaxLength(20);
        });

        OnModelCreatingPartial(modelBuilder);
    }

    partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using CustomSupport.Models;

namespace CustomSupport.Controllers
```

```
{
    public class UserInfoesController : Controller
    {
        private readonly EndDbContext _context;

        public UserInfoesController(EndDbContext context)
        {
            _context = context;
        }

        // GET: UserInfoes
        public async Task<IActionResult> Index()
        {
              return _context.UserInfos != null ?
                          View(await _context.UserInfos.ToListAsync()) :
                          Problem("Entity set 'EndDbContext.UserInfos'  is null.");
        }

        // GET: UserInfoes/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null || _context.UserInfos == null)
            {
                return NotFound();
            }

            var userInfo = await _context.UserInfos
                .FirstOrDefaultAsync(m => m.UserId == id);
            if (userInfo == null)
            {
                return NotFound();
            }

            return View(userInfo);
        }

        // GET: UserInfoes/Create
        public IActionResult Create()
        {
            return View();
        }

        // POST: UserInfoes/Create
        // To protect from overposting attacks, enable the specific properties you
want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("UserId,Email,Password")]
UserInfo userInfo)
        {
            if (ModelState.IsValid)
            {
                _context.Add(userInfo);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(userInfo);
```

```
        }

        // GET: UserInfoes/Edit/5
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null || _context.UserInfos == null)
            {
                return NotFound();
            }

            var userInfo = await _context.UserInfos.FindAsync(id);
            if (userInfo == null)
            {
                return NotFound();
            }
            return View(userInfo);
        }

        // POST: UserInfoes/Edit/5
        // To protect from overposting attacks, enable the specific properties you
want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id,
[Bind("UserId,Email,Password")] UserInfo userInfo)
        {
            if (id != userInfo.UserId)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(userInfo);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!UserInfoExists(userInfo.UserId))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
                return RedirectToAction(nameof(Index));
            }
            return View(userInfo);
        }

        // GET: UserInfoes/Delete/5
        public async Task<IActionResult> Delete(int? id)
        {
```

```
            if (id == null || _context.UserInfos == null)
            {
                return NotFound();
            }

            var userInfo = await _context.UserInfos
                .FirstOrDefaultAsync(m => m.UserId == id);
            if (userInfo == null)
            {
                return NotFound();
            }

            return View(userInfo);
        }

        // POST: UserInfoes/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            if (_context.UserInfos == null)
            {
                return Problem("Entity set 'EndDbContext.UserInfos'  is null.");
            }
            var userInfo = await _context.UserInfos.FindAsync(id);
            if (userInfo != null)
            {
                _context.UserInfos.Remove(userInfo);
            }

            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        private bool UserInfoExists(int id)
        {
          return (_context.UserInfos?.Any(e => e.UserId == id)).GetValueOrDefault();
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using CustomSupport.Models;

namespace CustomSupport.Controllers
{
    public class CustLogInfoesController : Controller
    {
        private readonly EndDbContext _context;

        public CustLogInfoesController(EndDbContext context)
        {
            _context = context;
```

```
        }

        // GET: CustLogInfoes
        public async Task<IActionResult> Index()
        {
            var endDbContext = _context.CustLogInfos.Include(c => c.User);
            return View(await endDbContext.ToListAsync());
        }

        // GET: CustLogInfoes/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null || _context.CustLogInfos == null)
            {
                return NotFound();
            }

            var custLogInfo = await _context.CustLogInfos
                .Include(c => c.User)
                .FirstOrDefaultAsync(m => m.LogId == id);
            if (custLogInfo == null)
            {
                return NotFound();
            }

            return View(custLogInfo);
        }

        // GET: CustLogInfoes/Create
        public IActionResult Create()
        {
            ViewData["UserId"] = new SelectList(_context.UserInfos, "UserId",
"UserId");
            return View();
        }

        // POST: CustLogInfoes/Create
        // To protect from overposting attacks, enable the specific properties you
want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult>
Create([Bind("LogId,CustEmail,CustName,LogStatus,UserId,Description")] CustLogInfo
custLogInfo)
        {
            if (ModelState.IsValid)
            {
                _context.Add(custLogInfo);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            ViewData["UserId"] = new SelectList(_context.UserInfos, "UserId",
"UserId", custLogInfo.UserId);
            return View(custLogInfo);
        }

        // GET: CustLogInfoes/Edit/5
```

```
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null || _context.CustLogInfos == null)
            {
                return NotFound();
            }

            var custLogInfo = await _context.CustLogInfos.FindAsync(id);
            if (custLogInfo == null)
            {
                return NotFound();
            }
            ViewData["UserId"] = new SelectList(_context.UserInfos, "UserId",
"UserId", custLogInfo.UserId);
            return View(custLogInfo);
        }

        // POST: CustLogInfoes/Edit/5
        // To protect from overposting attacks, enable the specific properties you
want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id,
[Bind("LogId,CustEmail,CustName,LogStatus,UserId,Description")] CustLogInfo
custLogInfo)
        {
            if (id != custLogInfo.LogId)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(custLogInfo);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!CustLogInfoExists(custLogInfo.LogId))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
                return RedirectToAction(nameof(Index));
            }
            ViewData["UserId"] = new SelectList(_context.UserInfos, "UserId",
"UserId", custLogInfo.UserId);
            return View(custLogInfo);
        }

        // GET: CustLogInfoes/Delete/5
```

```
        public async Task<IActionResult> Delete(int? id)
        {
            if (id == null || _context.CustLogInfos == null)
            {
                return NotFound();
            }

            var custLogInfo = await _context.CustLogInfos
                .Include(c => c.User)
                .FirstOrDefaultAsync(m => m.LogId == id);
            if (custLogInfo == null)
            {
                return NotFound();
            }

            return View(custLogInfo);
        }

        // POST: CustLogInfoes/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            if (_context.CustLogInfos == null)
            {
                return Problem("Entity set 'EndDbContext.CustLogInfos'  is null.");
            }
            var custLogInfo = await _context.CustLogInfos.FindAsync(id);
            if (custLogInfo != null)
            {
                _context.CustLogInfos.Remove(custLogInfo);
            }

            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        private bool CustLogInfoExists(int id)
        {
          return (_context.CustLogInfos?.Any(e => e.LogId ==
id)).GetValueOrDefault();
        }
    }
}
using CustomSupport.Models;
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();

builder.Services.AddDbContext<EndDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("CustomerConStr")));

var app = builder.Build();
```

```
// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}
app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": { "CustomerConStr":
"Server=tcp:simplonatech.database.windows.net,1433;Initial
Catalog=CustomSupportDb;User
ID=PrashastVats;Password=Ankur23050105!;Encrypt=True;TrustServerCertificate=False;Co
nnection Timeout=30;" }
}
#See https://aka.ms/customizecontainer to learn how to customize your debug
container and how Visual Studio uses this Dockerfile to build your images for faster
debugging.

FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443

FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
WORKDIR /src
COPY ["CustomSupport/CustomSupport.csproj", "CustomSupport/"]
RUN dotnet restore "CustomSupport/CustomSupport.csproj"
COPY . .
WORKDIR "/src/CustomSupport"
RUN dotnet build "CustomSupport.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "CustomSupport.csproj" -c Release -o /app/publish
/p:UseAppHost=false

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "CustomSupport.dll"]<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - CustomSupport</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
    <link rel="stylesheet" href="~/CustomSupport.styles.css" asp-append-
version="true" />
    <style>
        /* General Styles */
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background-color: #fafafa;
            color: #4a4a4a;
        }

        .container {
            background-color: #ffffff;
            border-radius: 4px;
            padding: 20px;
            box-shadow: 0px 3px 10px rgba(0, 0, 0, 0.05);
        }

        h1, h2, h3, h4, h5, h6 {
            color: #2c3e50;
        }

        a:hover {
            text-decoration: none;
        }

        /* Navbar Styles */
        .navbar {
            background-color: #3498db;
        }

            .navbar a.navbar-brand, .navbar a.nav-link {
                color: #ecf0f1;
            }

                .navbar a.navbar-brand:hover, .navbar a.nav-link:hover {
                    color: #e74c3c;
                }

        /* Footer Styles */
        .footer {
            background-color: #2c3e50;
            color: #ecf0f1;
            padding: 10px 0;
        }

            .footer a {
                color: #ecf0f1;
            }

                .footer a:hover {
                    color: #3498db;
                }
```

```
    </style>

</head>
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-
white border-bottom box-shadow mb-3">
            <div class="container-fluid">
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-
action="Index">CustomSupport</a>
                <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target=".navbar-collapse" aria-
controls="navbarSupportedContent"
                        aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="navbar-collapse collapse d-sm-inline-flex justify-
content-between">
                    <ul class="navbar-nav flex-grow-1">
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-
controller="Home" asp-action="Index">Home</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-
controller="Home" asp-action="Privacy">Privacy</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-
controller="UserInfoes" asp-action="Index">Executive Login</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-
controller="CustLogInfoes" asp-action="Index">Complaint information</a>
                        </li>
                    </ul>
                </div>
            </div>
        </nav>
    </header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="border-top footer text-muted">
        <div class="container">
            &copy; 2023 - CustomSupport - <a asp-area="" asp-controller="Home" asp-
action="Privacy">Privacy</a>
        </div>
    </footer>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

```
namespace DAL
{
    public class UserInfo
    {
        public int UserId { get; set; }
        public string Email { get; set; }
        public string Password { get; set; }
    }
}

namespace DAL
{
    public class CustLogInfo
    {
        public int LogId { get; set; }
        public string CustEmail { get; set; }
        public string CustName { get; set; }
        public string LogStatus { get; set; }
        public int UserId { get; set; }
        public string Description { get; set; }
    }
}

using DAL;
using Moq;
using NUnit.Framework;

namespace DALTest
{
    [TestFixture]
    public partial class CustLogInfoTest
    {
        [Test]
        public void GetAllLogInfoesTest()
        {
            // Arrange
            var mockCustLogInfoRepository = new Mock<ICustLogInfoRepository>();
            // Set up mock data or behavior if needed

            // Act
            var result = mockCustLogInfoRepository.Object.GetAllLogInfoes();

            // Assert
            // Add assertions to validate the result
        }

        [Test]
        public void SaveCustLogInfoTest()
        {
            // Arrange
            var mockCustLogInfoRepository = new Mock<ICustLogInfoRepository>();
            // Set up mock data or behavior if needed

            // Act
            var result = mockCustLogInfoRepository.Object.SaveCustLogInfo(new
CustLogInfo());

            // Assert
```

```
            // Add assertions to validate the result
        }

        public interface ICustLogInfoRepository
        {
            object SaveCustLogInfo(CustLogInfo custLogInfo);
            object GetAllLogInfoes();
        }
    }
}

// UserInfoTest.cs

using DAL;
using Moq;
using NUnit.Framework;

namespace DALTest
{
    [TestFixture]
    public class UserInfoTest
    {
        [Test]
        public void ValidateUserTest()
        {
            // Arrange
            var mockUserInfoRepository = new Mock<IUserInfoRepository>();
            // Set up mock data or behavior if needed

            // Act
            var result = mockUserInfoRepository.Object.ValidateUser(1, "Maximus");

            // Assert
            // Add assertions to validate the result
        }
    }

    public interface IUserInfoRepository
    {
        object ValidateUser(int i, string Maximus);
    }
}

using DAL;

namespace DALTest
{
    internal interface ICustLogInfoRepository
    {
        object SaveCustLogInfo(CustLogInfo custLogInfo);
        object GetAllLogInfoes();
    }
}
```