Phase End Project – 1 Source

Code

```
using System;
using System.Collections.Generic; namespace CustomerSupportLogger.Models; public

partial class UserInfo

{

public int UserId { get; set; }


public string Email { get; set; } = null!; public string Password { get; set; } = null!;

    public virtual ICollection<CustLogInfo> CustLogInfos { get; set; } = new
List<CustLogInfo>();

}
```

CustLogInfo

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

namespace CustomerSupportLogger.Models; public partial class CustLogInfo
{
public int LogId { get; set; }


public string CustEmail { get; set; } = null!; public string CustName { get; set; } = null!;

public string LogStatus { get; set; } = null!; public int? UserId { get; set; } public string

Description { get; set; } = null!;


public virtual UserInfo? User { get; set; }
}
```

DbContext

using System;
using System.Collections.Generic; using Microsoft.EntityFrameworkCore;

namespace CustomerSupportLogger.Models; public partial class

CustomerSupportLoggerDbContext : DbContext

{

public CustomerSupportLoggerDbContext()
{

}

public
CustomerSupportLoggerDbContext(DbContextOptions<CustomerSupportLoggerDbC
ontext> 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:newserver3058.database.windows.net,143
3;Initial Catalog=CustomerSupportLoggerDB;User
ID=admin123;Password=vasanth@123;Encrypt=True;TrustServerCertificate=False;");

protected override void OnModelCreating(ModelBuilder modelBuilder)

```
{
modelBuilder.Entity<CustLogInfo>(entity =>
{
entity.HasKey(e => e.LogId).HasName("PK__CustLogI__5E548648B316D002");

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__398D8EEE");




modelBuilder.Entity<UserInfo>(entity =>
{
entity.HasKey(e => e.UserId).HasName("PK__UserInfo__1788CC4C769353B1");

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;
```

CustLogInfoesController

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

namespace CustomerSupportLogger.Controllers
{

public class CustLogInfoesController : Controller

{

private readonly CustomerSupportLoggerDbContext _context;
```

```
public CustLogInfoesController(CustomerSupportLoggerDbContext context)
{
_context = context;
}

// GET: CustLogInfoes

public async Task<IActionResult> Index()
{
        var customerSupportLoggerDbContext = _context.CustLogInfos.Include(c =>
c.User);
return View(await customerSupportLoggerDbContext.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 'CustomerSupportLoggerDbContext.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 System;
```

UserInFoesController

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

namespace CustomerSupportLogger.Controllers
{
public class UserInfoesController : Controller
{
private readonly CustomerSupportLoggerDbContext _context;

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

// GET: UserInfoes
public async Task<IActionResult> Index()
{
```

```
return _context.UserInfos != null ?
View(await _context.UserInfos.ToListAsync()) :
Problem("Entity set 'CustomerSupportLoggerDbContext.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)

{

var user = await _context.UserInfos

                .FirstOrDefaultAsync(u => u.UserId == userInfo.UserId && u.Email == userInfo.Email && u.Password == userInfo.Password);


if (user != null)

{ return RedirectToAction("Create", "CustLogInfoes"); // Redirect to the Create action in CustLogInfoesController

}

else

{

ModelState.AddModelError("", "Incorrect UserID, Email or Password");

}

}

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 'CustomerSupportLoggerDbContext.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();
}
```

```
}
}
```

Test with NUnit and Moq using
NUnit.Framework; using Moq;
using CustomerSupportLogger.Controllers; using CustomerSupportLogger.Models;
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic; using System.Linq; using
System.Threading.Tasks;

```
namespace CustomerSupportLogger.Tests
{
[TestFixture] public class
UserInfoesControllerTests
{
[Test]
public void UserInfo_GetUserId_ReturnsUserId()
{
// Arrange var userInfo = new UserInfo {
UserId = 1 };

// Act
int userId = userInfo.UserId;

// Assert Assert.AreEqual(1, userId);
}

[Test]
public void UserInfo_SetUserId_CanSetUserId()
{
```

```
// Arrange var userInfo = new
UserInfo();

// Act userInfo.UserId = 2;

// Assert
Assert.AreEqual(2, userInfo.UserId);
}

[Test]
public void CustLogInfo_GetLogId_ReturnsLogId()
{
// Arrange var custLogInfo = new CustLogInfo {
LogId = 1 };

// Act int logId =
custLogInfo.LogId;

// Assert Assert.AreEqual(1, logId);
}

[Test]
public void CustLogInfo_SetLogId_CanSetLogId()
{
// Arrange var custLogInfo = new
CustLogInfo();

// Act custLogInfo.LogId = 2;

// Assert
```

```
Assert.AreEqual(2, custLogInfo.LogId);
}
}
}
```

Jenkinsfile

```
pipeline { agent any

stages { stage('Checkout')
{ steps {
checkout scm
}
}

stage('Build') { steps {
bat 'dotnet build'
}
}

stage('Test') { steps {

bat 'dotnet test'
}
}

stage('Publish') { steps {
bat 'dotnet publish -c Release -o ./publish'
```

}

}

}

post { failure { emailext (

subject: "Pipeline Failed",

body: "There was an error in the Jenkins pipeline. Please investigate.", to: "sandeepprasad@gmail.com"

)

}

}

}

Dockerfile

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

FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build WORKDIR /src

COPY ["CustomerSupportLogger/CustomerSupportLogger.csproj", "CustomerSupportLogger/"]

RUN dotnet restore "CustomerSupportLogger/CustomerSupportLogger.csproj" COPY . .

```
WORKDIR "/src/CustomerSupportLogger"

RUN dotnet build "CustomerSupportLogger.csproj" -c Release -o /app/build

FROM build AS publish

RUN dotnet publish "CustomerSupportLogger.csproj" -c Release -o /app/publish
/p:UseAppHost=false

FROM base AS final WORKDIR /app

COPY --from=publish /app/publish .

ENTRYPOINT ["dotnet", "CustomerSupportLogger.dll"]
```