```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using WebApi.Models;

namespace WebApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class AdminInfoesController : ControllerBase
    {
        private readonly CapStoneContext _context;

        public AdminInfoesController(CapStoneContext context)
        {
            _context = context;
        }

        // GET: api/AdminInfoes
        [HttpGet]
        public async Task<ActionResult<IEnumerable<AdminInfo>>> GetAdminInfos()
        {
          if (_context.AdminInfos == null)
          {
              return NotFound();
          }
            return await _context.AdminInfos.ToListAsync();
        }

        // GET: api/AdminInfoes/5
        [HttpGet("{id}")]
        public async Task<ActionResult<AdminInfo>> GetAdminInfo(int id)
        {
          if (_context.AdminInfos == null)
          {
              return NotFound();
          }
            var adminInfo = await _context.AdminInfos.FindAsync(id);

            if (adminInfo == null)
            {
                return NotFound();
            }

            return adminInfo;
        }

        // PUT: api/AdminInfoes/5
        // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754
        [HttpPut("{id}")]
        public async Task<IActionResult> PutAdminInfo(int id, AdminInfo adminInfo)
        {
            if (id != adminInfo.Id)
```

```
            {
                return BadRequest();
            }

            _context.Entry(adminInfo).State = EntityState.Modified;

            try
            {
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!AdminInfoExists(id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return NoContent();
        }

        // POST: api/AdminInfoes
        // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754
        [HttpPost]
        public async Task<ActionResult<AdminInfo>> PostAdminInfo(AdminInfo
adminInfo)
        {
          if (_context.AdminInfos == null)
          {
              return Problem("Entity set 'CapStoneContext.AdminInfos'  is null.");
          }
            _context.AdminInfos.Add(adminInfo);
            await _context.SaveChangesAsync();

            return CreatedAtAction("GetAdminInfo", new { id = adminInfo.Id },
adminInfo);
        }

        // DELETE: api/AdminInfoes/5
        [HttpDelete("{id}")]
        public async Task<IActionResult> DeleteAdminInfo(int id)
        {
            if (_context.AdminInfos == null)
            {
                return NotFound();
            }
            var adminInfo = await _context.AdminInfos.FindAsync(id);
            if (adminInfo == null)
            {
                return NotFound();
            }

            _context.AdminInfos.Remove(adminInfo);
```

```
            await _context.SaveChangesAsync();

            return NoContent();
        }

        private bool AdminInfoExists(int id)
        {
            return (_context.AdminInfos?.Any(e => e.Id == id)).GetValueOrDefault();
        }
    }
}


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

namespace WebApi.Models;

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

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

    public virtual DbSet<AdminInfo> AdminInfos { get; set; }

    public virtual DbSet<BlogInfo> BlogInfos { get; set; }

    public virtual DbSet<EmpInfo> EmpInfos { 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=DESKTOP-
MFQ8M0P;Database=CapStone;Trusted_Connection=True;TrustServerCertificate=True;");

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<AdminInfo>(entity =>
        {
            entity.HasKey(e => e.Id).HasName("PK__AdminInf__3214EC07B8270755");

            entity.ToTable("AdminInfo");

            entity.Property(e => e.EmailId)
                .HasMaxLength(255)
                .IsUnicode(false);
            entity.Property(e => e.Password)
                .HasMaxLength(255)
```

```
                .IsUnicode(false);
        });

        modelBuilder.Entity<BlogInfo>(entity =>
        {
            entity.HasKey(e => e.BlogId).HasName("PK__BlogInfo__54379E302BF43C34");

            entity.ToTable("BlogInfo");

            entity.Property(e => e.BlogUrl)
                .HasMaxLength(255)
                .IsUnicode(false);
            entity.Property(e => e.DateOfCreation).HasColumnType("datetime");
            entity.Property(e => e.EmpEmailId)
                .HasMaxLength(255)
                .IsUnicode(false);
            entity.Property(e => e.Subject)
                .HasMaxLength(255)
                .IsUnicode(false);
            entity.Property(e => e.Title)
                .HasMaxLength(255)
                .IsUnicode(false);
        });

        modelBuilder.Entity<EmpInfo>(entity =>
        {
            entity.HasKey(e => e.Id).HasName("PK__EmpInfo__3214EC07E9D8D724");

            entity.ToTable("EmpInfo");

            entity.HasIndex(e => e.EmailId,
"UQ__EmpInfo__7ED91ACECBE35D19").IsUnique();

            entity.Property(e => e.DateOfJoining).HasColumnType("datetime");
            entity.Property(e => e.EmailId)
                .HasMaxLength(255)
                .IsUnicode(false);
            entity.Property(e => e.Name)
                .HasMaxLength(255)
                .IsUnicode(false);
        });

        OnModelCreatingPartial(modelBuilder);
    }

    partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
}


using MVC.Models;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Text;
using System.Web.Mvc;

namespace MVC.Controllers
```

```
{
    public class EmpController : Controller
    {
        Uri baseAddress = new Uri("http://localhost:5132/api");
        HttpClient client;

        public EmpController()
        {
            client = new HttpClient();
            client.BaseAddress = baseAddress;
        }

        public ActionResult Index()
        {
            List<EmpInfo> emps = new List<EmpInfo>();
            HttpResponseMessage response = client.GetAsync(client.BaseAddress +
"/EmpInfoes").Result;
            if (response.IsSuccessStatusCode)
            {
                string data = response.Content.ReadAsStringAsync().Result;
                emps = JsonConvert.DeserializeObject<List<EmpInfo>>(data);
            }
            return View(emps);
        }

        public ActionResult Create()
        {
            return View();
        }

        [HttpPost]
        public ActionResult Create(EmpInfo emps)
        {
            string data = JsonConvert.SerializeObject(emps);
            StringContent content = new StringContent(data, Encoding.UTF8,
"application/json");
            HttpResponseMessage responce = client.PostAsync(client.BaseAddress +
"/EmpInfoes", content).Result;
            if (responce.IsSuccessStatusCode)
            {
                return RedirectToAction("Index");
            }
            return View();
        }

        [HttpGet]
        public ActionResult Edit(int id)
        {
            EmpInfo emps = new EmpInfo();
            HttpResponseMessage response = client.GetAsync(client.BaseAddress +
"/EmpInfoes/" + id).Result;
            if (response.IsSuccessStatusCode)
            {
                string data = response.Content.ReadAsStringAsync().Result;
                emps = JsonConvert.DeserializeObject<EmpInfo>(data);
            }
            return View(emps);
        }
```

```
        [HttpPost]
        public ActionResult Edit(EmpInfo emp)
        {
            try
            {
                string data = JsonConvert.SerializeObject(emp);
                StringContent content = new StringContent(data, Encoding.UTF8,
"application/json");
                HttpResponseMessage response = client.PutAsync(client.BaseAddress +
"/EmpInfoes/" + emp.Id, content).Result;

                if (response.IsSuccessStatusCode)
                {
                    return RedirectToAction("Index");
                }
                else
                {
                    ModelState.AddModelError(string.Empty, "Error updating emp.");
                    return View(emp);
                }
            }
            catch (Exception ex)
            {
                ModelState.AddModelError(string.Empty, "An error occurred: " +
ex.Message);
                return View(emp);
            }
        }

        [HttpGet]
        public ActionResult Delete(int id)
        {
            try
            {
                EmpInfo emps = new EmpInfo();
                HttpResponseMessage response = client.GetAsync(client.BaseAddress +
"/EmpInfoes/" + id).Result;
                if (response.IsSuccessStatusCode)
                {
                    string data = response.Content.ReadAsStringAsync().Result;
                    emps = JsonConvert.DeserializeObject<EmpInfo>(data);
                }
                return View(emps);
            }
            catch (Exception ex)
            {
                return View();
            }
            return View();
        }

        [HttpPost, ActionName("Delete")]
        public ActionResult DeleteConfirm(int id)
        {
            try
            {
```

```
                HttpResponseMessage response = client.DeleteAsync(client.BaseAddress
+ "/EmpInfoes/" + id).Result;

                if (response.IsSuccessStatusCode)
                {
                    return RedirectToAction("Index");
                }
            }
            catch (Exception ex)
            {
                return View();
                throw;
            }
            return View();
        }
    }
}


@model IEnumerable<MVC.Models.BlogInfo>

@{
    ViewBag.Title = "Index";
}

<h2>List Of Blog</h2>

<p>
        @Html.ActionLink("Create New", "Create", null, new { @class = "btn btn-
success" })
</p>

<div class="blog-container">
    @foreach (var item in Model)
    {
        <div class="blog-item">
            <h3 style="display: inline;">@Html.DisplayFor(modelItem =>
item.Title)</h3>
            <span><h5> by </h5></span>
            <p style="display: inline;"><h5>@Html.DisplayFor(modelItem =>
item.EmpEmailId)</h5></p>
            <p>Date : <span style="display: inline;">@Html.DisplayFor(modelItem =>
item.DateOfCreation)</span></p>
            <p>Subject : <span style="display: inline;">@Html.DisplayFor(modelItem
=> item.Subject)</span></p>
            <p>
                <a href="@item.BlogUrl" class="btn btn-primary" target="_blank">View
Blog</a>
                    @Html.ActionLink("Edit", "Edit", new { id = item.BlogId }, new {
@class = "btn btn-info" })
                    @Html.ActionLink("Delete", "Delete", new { id = item.BlogId },
new { @class = "btn btn-danger" })
            </p>
        </div>
    }
</div>

<style>
```

```
    .blog-container {
        padding: 20px;
        margin: 20px;
        background-color: none;
        display: flex;
        flex-wrap: wrap;
    }

    .blog-item {
        border: 1px solid #ccc;
        padding: 10px;
        margin: 10px;
        background-color: #fff;
        width: calc(50% - 20px); /* 50% width with margin on both sides */
        box-sizing: border-box;
    }
</style>


@model MVC.Models.BlogInfo

@{
    ViewBag.Title = "Create";
}

<h2>Add Blog Details</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>BlogInfo</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })

        <div class="form-group">
            @Html.LabelFor(model => model.Title, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Title, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Title, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Subject, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Subject, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Subject, "", new { @class
= "text-danger" })
            </div>
        </div>
```

```
        <div class="form-group">
            @Html.LabelFor(model => model.DateOfCreation, htmlAttributes: new {
@class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.DateOfCreation, new { htmlAttributes
= new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.DateOfCreation, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.BlogUrl, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.BlogUrl, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.BlogUrl, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.EmpEmailId, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.EmpEmailId, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.EmpEmailId, "", new {
@class = "text-danger" })
            </div>
        </div>
        <hr />
        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Add Blog Details" class="btn btn-
success" />
            </div>
        </div>
    </div>
}


@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}


using Microsoft.EntityFrameworkCore;
using WebApi.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();
builder.Services.AddDbContext<CapStoneContext>(options =>
```

```
    options.UseSqlServer(builder.Configuration.GetConnectionString("Capstone") ??
throw new InvalidOperationException("Connection string 'CapStone' not found.")));

// Learn more about configuring Swagger/OpenAPI at
https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseAuthorization();

app.MapControllers();

app.Run();
```