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

namespace WebApi.Models;

public partial class AdminInfo
{
    public int Id { get; set; }

    public string? EmailId { get; set; }

    public string? Password { get; set; }
}
using System;
using System.Collections.Generic;

namespace WebApi.Models;

public partial class BlogInfo
{
    public int BlogId { get; set; }

    public string? Title { get; set; }

    public string? Subject { get; set; }

    public DateTime? DateOfCreation { get; set; }

    public string? BlogUrl { get; set; }

    public string? EmpEmailId { get; set; }
}
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=tcp:simplonatech.database.windows.net,1433;Initi
al Catalog=BlogAppDb;Persist Security Info=False;User
ID=PrashastVats;Password=Ankur23050105!;MultipleActiveResultSets=False;Encrypt=True;
TrustServerCertificate=False;Connection Timeout=30");

    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 System;
using System.Collections.Generic;

namespace WebApi.Models;

public partial class EmpInfo
{
    public int Id { get; set; }

    public string? EmailId { get; set; }

    public string? Name { get; set; }

    public DateTime? DateOfJoining { get; set; }

    public int? PassCode { get; set; }
}
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": { "CapStone":
"Server=tcp:simplonatech.database.windows.net,1433;Initial Catalog=BlogAppDb;Persist
Security Info=False;User
ID=PrashastVats;Password=Ankur23050105!;MultipleActiveResultSets=False;Encrypt=True;
TrustServerCertificate=False;Connection Timeout=30;" }
}
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();
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 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 BlogInfoesController : ControllerBase
    {
        private readonly CapStoneContext _context;

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

        // GET: api/BlogInfoes
        [HttpGet]
        public async Task<ActionResult<IEnumerable<BlogInfo>>> GetBlogInfos()
        {
          if (_context.BlogInfos == null)
          {
              return NotFound();
          }
            return await _context.BlogInfos.ToListAsync();
        }
        // GET: api/BlogInfoes/5
        [HttpGet("{id}")]
        public async Task<ActionResult<BlogInfo>> GetBlogInfo(int id)
        {
          if (_context.BlogInfos == null)
          {
              return NotFound();
          }
            var blogInfo = await _context.BlogInfos.FindAsync(id);

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

            return blogInfo;
        }

        // PUT: api/BlogInfoes/5
        // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754
        [HttpPut("{id}")]
        public async Task<IActionResult> PutBlogInfo(int id, BlogInfo blogInfo)
        {
            if (id != blogInfo.BlogId)
            {
                return BadRequest();
            }

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

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

            return NoContent();
        }

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

            return CreatedAtAction("GetBlogInfo", new { id = blogInfo.BlogId },
blogInfo);
        }

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

            _context.BlogInfos.Remove(blogInfo);
            await _context.SaveChangesAsync();

            return NoContent();
        }

        private bool BlogInfoExists(int id)
        {
            return (_context.BlogInfos?.Any(e => e.BlogId ==
id)).GetValueOrDefault();
        }
    }
}
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 EmpInfoesController : ControllerBase
    {
        private readonly CapStoneContext _context;

        public EmpInfoesController(CapStoneContext context)
        {
            _context = context;
        }
        // GET: api/EmpInfoes
        [HttpGet]
        public async Task<ActionResult<IEnumerable<EmpInfo>>> GetEmpInfos()
        {
          if (_context.EmpInfos == null)
          {
              return NotFound();
          }
            return await _context.EmpInfos.ToListAsync();
        }

        // GET: api/EmpInfoes/5
        [HttpGet("{id}")]
        public async Task<ActionResult<EmpInfo>> GetEmpInfo(int id)
        {
          if (_context.EmpInfos == null)
          {
              return NotFound();
          }
            var empInfo = await _context.EmpInfos.FindAsync(id);

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

            return empInfo;
        }

        // PUT: api/EmpInfoes/5
        // To protect from overposting attacks, see
https://go.microsoft.com/fwlink/?linkid=2123754
        [HttpPut("{id}")]
        public async Task<IActionResult> PutEmpInfo(int id, EmpInfo empInfo)
        {
            if (id != empInfo.Id)
            {
                return BadRequest();
            }

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

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

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

            return CreatedAtAction("GetEmpInfo", new { id = empInfo.Id }, empInfo);
        }

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

            _context.EmpInfos.Remove(empInfo);
            await _context.SaveChangesAsync();

            return NoContent();
        }

        private bool EmpInfoExists(int id)
        {
            return (_context.EmpInfos?.Any(e => e.Id == id)).GetValueOrDefault();
        }
    }
}
using Microsoft.AspNetCore.Mvc;

namespace WebApi.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class WeatherForecastController : ControllerBase
    {
        private static readonly string[] Summaries = new[]
        {
        "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot",
"Sweltering", "Scorching"
```

```
    };

        private readonly ILogger<WeatherForecastController> _logger;

        public WeatherForecastController(ILogger<WeatherForecastController> logger)
        {
            _logger = logger;
        }

        [HttpGet(Name = "GetWeatherForecast")]
        public IEnumerable<WeatherForecast> Get()
        {
            return Enumerable.Range(1, 5).Select(index => new WeatherForecast
            {
                Date = DateTime.Now.AddDays(index),
                TemperatureC = Random.Shared.Next(-20, 55),
                Summary = Summaries[Random.Shared.Next(Summaries.Length)]
            })
            .ToArray();
        }
    }
}using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;

namespace MVC.Models
{
    public class AdminInfo
    {
        public int Id { get; set; }
        public string EmailId { get; set; }
        public string Password { get; set; }
    }
}using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace MVC.Models
{
    public class BlogInfo
    {
        public int BlogId { get; set; }

        public string Title { get; set; }

        public string Subject { get; set; }

        public DateTime DateOfCreation { get; set; }

        public string BlogUrl { get; set; }

        public string EmpEmailId { get; set; }
    }
}using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace MVC.Models
{
    public class EmpInfo
    {
        public int Id { get; set; }

        public string EmailId { get; set; }

        public string Name { get; set; }

        public DateTime DateOfJoining { get; set; }

        public int PassCode { get; set; }
    }
}using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;
namespace MVC.Models
{
    public class LoginInfo
    {
        [Required(ErrorMessage ="Please Enter Your EmailId")]
        public string EmailId { get; set; }
        [Required(ErrorMessage = "Please Enter Your Password")]
        public string Password { get; set; }
    }
}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 AdminController : Controller
    {
        Uri baseAddress = new Uri("http://localhost:5132/api");
        HttpClient client;

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

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

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

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

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

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

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

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

        [HttpPost]
        public ActionResult Delete(AdminInfo admins)
        {
            string data = JsonConvert.SerializeObject(admins);
            StringContent content = new StringContent(data, Encoding.UTF8,
"application/json");
            HttpResponseMessage response = client.DeleteAsync(client.BaseAddress +
"/AdminInfoes/" + admins.Id).Result;

            if (response.IsSuccessStatusCode)
            {
                return RedirectToAction("Index");
            }
            return View();
        }
    }
}
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 BlogController : Controller
    {
        Uri baseAddress = new Uri("http://localhost:5132/api");
        HttpClient client;

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

        public ActionResult Index()
        {
            List<BlogInfo> blogs = new List<BlogInfo>();
            HttpResponseMessage response = client.GetAsync(client.BaseAddress +
"/BlogInfoes").Result;
            if (response.IsSuccessStatusCode)
            {
                string data = response.Content.ReadAsStringAsync().Result;
                blogs = JsonConvert.DeserializeObject<List<BlogInfo>>(data);
            }
            return View(blogs);
        }
        [Authorize(Roles="Admin,Employee")]
        public ActionResult Create()
        {
            return View();
        }

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

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

        [HttpPost]
        public ActionResult Edit(BlogInfo blog)
        {
            try
            {
                string data = JsonConvert.SerializeObject(blog);
                StringContent content = new StringContent(data, Encoding.UTF8, 
"application/json");
                HttpResponseMessage response = client.PutAsync(client.BaseAddress + 
"/BlogInfoes/" + blog.BlogId, content).Result;

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

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

        [HttpPost, ActionName("Delete")]
        public ActionResult DeleteConfirm(int id)
        {
            try
            {
                HttpResponseMessage response = client.DeleteAsync(client.BaseAddress 
+ "/BlogInfoes/" + id).Result;

                if (response.IsSuccessStatusCode)
                {
                    return RedirectToAction("Index");
                }
            }
            catch (Exception ex)
            {
                return View();
                throw;
            }
            return View();
        }
    }
}
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();
            }
        }
}
using MVC.Models;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Security;

namespace MVC.Controllers
{
    public class LoginController : Controller
    {
        public ActionResult Admin()
        {
            return View();
        }
        [HttpPost]
        public ActionResult Admin(LoginInfo loginInfo)
        {
            string connection =
ConfigurationManager.ConnectionStrings["MyConnectionString"].ConnectionString;
            SqlConnection con = new SqlConnection(connection);
            string cmd = "Select EmailId,Password from AdminInfo where 
EmailId=@Emailid and Password=@Password";
            con.Open();
            SqlCommand command = new SqlCommand(cmd, con);
            command.Parameters.AddWithValue("@EmailId", loginInfo.EmailId);
            command.Parameters.AddWithValue("@Password", loginInfo.Password);
            SqlDataReader reader = command.ExecuteReader();
            if (reader.Read())
            {
                Session["EmailId"] = loginInfo.EmailId.ToString();
                return RedirectToAction("Index", "Blog");
            }

            else
            {
                ViewData["Message"] = "Admin Login Details Failed";
            }
            con.Close();
            return View();
        }
        public ActionResult Employee()
        {
            return View();
        }

        [HttpPost]
        public ActionResult Employee(LoginInfo loginInfo)
        {
            string connection = 
ConfigurationManager.ConnectionStrings["MyConnectionString"].ConnectionString;
            SqlConnection con = new SqlConnection(connection);
            string cmd = "Select EmailId, PassCode from EmpInfo where 
EmailId=@Emailid and PassCode=@Password"; // Use PassCode column from EmpInfo table
            con.Open();
            SqlCommand command = new SqlCommand(cmd, con);
            command.Parameters.AddWithValue("@EmailId", loginInfo.EmailId);
            command.Parameters.AddWithValue("@Password", loginInfo.Password); // Use 
Password property
            SqlDataReader reader = command.ExecuteReader();
            if (reader.Read())
            {
                Session["EmailId"] = loginInfo.EmailId.ToString();
                return RedirectToAction("Index", "Blog"); // Redirect to the 
employee dashboard or the desired page
            }
            else
            {
                ViewData["Message"] = "Employee Login Details Failed";
            }
            con.Close();
            return View();
        }

        public ActionResult Logout()
        {
            FormsAuthentication.SignOut();
            Session.Clear(); // Clear the session to log out the user
```

```
                return RedirectToAction("Index", "Home"); // Redirect to the home page
or another appropriate page
            }

        }
}@model MVC.Models.LoginInfo

@{
    ViewBag.Title = "Admin";
}

<h2>Admin Login Page</h2>


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

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

        <div class="form-group">
            @Html.LabelFor(model => model.Password, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Password, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Password, "", new { @class
= "text-danger" })
            </div>
        </div>
        <br />
        <div class="form-group">
            <div class="form-actions no-color">
                <input type="submit" value="Login" class="btn btn-primary" />
            </div>
        </div>
        <hr />
        <h1>@Html.ViewData["Message"]</h1>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
@model MVC.Models.LoginInfo
```

```
@{
    ViewBag.Title = "Employee";
}

<h2>Employee Login Page</h2>


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

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

        <div class="form-group">
            @Html.LabelFor(model => model.Password, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Password, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Password, "", new { @class
= "text-danger" })
            </div>
        </div>
        <br />
        <div class="form-group">
            <div class="form-actions no-color">
                <input type="submit" value="Login" class="btn btn-primary" />
            </div>
        </div>
        <hr />
        <h1>@Html.ViewData["Message"]</h1>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - Blog Tracker App</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
    <link href="https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/4.6.0/css/bootstrap.min.css" rel="stylesheet">
    <style>
        .navbar {
            margin-bottom: 50px;
        }

        .navbar-nav .nav-link {
            padding: 10px 15px; /* Add some padding to each navigation link */
            transition: background-color 0.2s; /* Smooth transition for hover effect
*/
        }

            .navbar-nav .nav-link:hover {
                background-color: #333; /* Darken the background on hover */
            }

        .dropdown:hover .dropdown-menu {
            display: block;
        }

        .body-content {
            padding: 20px 0;
            background-color: #f4f4f4;
        }

        .jumbotron {
            padding: 1rem 2rem; /* Reduced padding for the jumbotron */
        }

        footer {
            padding: 10px 0;
            background-color: #222;
            color: white;
            text-align: center;
        }
    </style>
</head>
<body>
    <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-dark bg-dark">
        <div class="container">
            @Html.ActionLink("Blog Tracker", "Index", "Home", new { area = "" }, new
{ @class = "navbar-brand" })
            <button type="button" class="navbar-toggler" data-bs-toggle="collapse"
data-bs-target=".navbar-collapse" title="Toggle navigation" aria-
controls="navbarSupportedContent"
                    aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse d-sm-inline-flex justify-content-
between">
                <ul class="navbar-nav flex-grow-1">
                    <li>@Html.ActionLink("Home", "Index", "Blog", new { area = "" },
new { @class = "nav-link" })</li>
                    @if (Session["EmailId"] != null)
                    {
                        <li class="nav-item dropdown">
```

```
                            <a href="#" class="nav-link dropdown-toggle"
id="userDropdown" data-bs-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">
                                @Session["EmailId"]
                            </a>
                            <div class="dropdown-menu" aria-
labelledby="userDropdown">
                                <a class="dropdown-item" href="@Url.Action("Logout",
"Login")">Logout</a>
                            </div>
                        </li>
                    }
                    else
                    {
                        <li class="nav-item dropdown">
                            <a href="#" class="nav-link dropdown-toggle"
id="loginDropdown" data-bs-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">
                                Login
                            </a>
                            <div class="dropdown-menu" aria-
labelledby="loginDropdown">
                                <a class="dropdown-item" href="@Url.Action("Admin",
"Login")">Admin</a>
                                <a class="dropdown-item"
href="@Url.Action("Employee", "Login")">Employee</a>
                            </div>
                        </li>
                    }
                </ul>
            </div>
        </div>
    </nav>
    <div class="container mb-4">
        <div class="jumbotron">
            <h1>Welcome to Blog Tracker</h1>
            <p>A platform to track and manage your blogs effectively!</p>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - &reg;SimplonaTech</p>
        </footer>
    </div>
    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", required: false)
</body>
</html>
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  https://go.microsoft.com/fwlink/?LinkId=301880
  -->
<configuration>
  <configSections>
```

```
    <!-- For more information on Entity Framework configuration, visit
http://go.microsoft.com/fwlink/?LinkID=237468 -->
    <section name="entityFramework"
type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection,
EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
requirePermission="false" />
  </configSections>
  <appSettings>
    <add key="webpages:Version" value="3.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  </appSettings>
  <connectionStrings>
             <add name="MyConnectionString"
connectionString="Server=tcp:simplonatech.database.windows.net,1433;Initial
Catalog=BlogAppDb;Persist Security Info=False;User
ID=PrashastVats;Password=Ankur23050105!;MultipleActiveResultSets=False;Encrypt=True;
TrustServerCertificate=False;Connection Timeout=30;"
providerName="System.Data.SqlClient"/>
      </connectionStrings>
  <system.web>
    <compilation debug="true" targetFramework="4.7.2" />
    <httpRuntime targetFramework="4.7.2" />
  </system.web>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="Antlr3.Runtime" publicKeyToken="eb42632606e9261f" />
        <bindingRedirect oldVersion="0.0.0.0-3.5.0.2" newVersion="3.5.0.2" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Web.Infrastructure"
publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="0.0.0.0-2.0.1.0" newVersion="2.0.1.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed"
/>
        <bindingRedirect oldVersion="0.0.0.0-12.0.0.0" newVersion="12.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Optimization"
publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-1.1.0.0" newVersion="1.1.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="WebGrease" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-1.6.5135.21930"
newVersion="1.6.5135.21930" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Helpers"
publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
```

```
        <assemblyIdentity name="System.Web.WebPages"
publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Web.Mvc" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-5.2.9.0" newVersion="5.2.9.0" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
  <system.codedom>
    <compilers>
      <compiler language="c#;cs;csharp" extension=".cs"
type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.CSharpCodeProvider,
Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=2.0.1.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35" warningLevel="4"
compilerOptions="/langversion:default /nowarn:1659;1699;1701" />
      <compiler language="vb;vbs;visualbasic;vbscript" extension=".vb"
type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.VBCodeProvider,
Microsoft.CodeDom.Providers.DotNetCompilerPlatform, Version=2.0.1.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35" warningLevel="4"
compilerOptions="/langversion:default /nowarn:41008
/define:_MYTYPE=\&quot;Web\&quot; /optionInfer+" />
    </compilers>
  </system.codedom>
  <entityFramework>
    <providers>
      <provider invariantName="System.Data.SqlClient"
type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer"
/>
    </providers>
  </entityFramework>
</configuration>
```