Capstone Project SourceCode

BlogTrackerAPI

AdminInfo.cs

```
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace BlogTrackerAPI.Models
{
    [Table("AdminInfo")]
    public class AdminInfo
    {
        [Key]
        public int AdminId { get; set; }
        public string? EmailId { get; set; }
        public string? Password { get; set; }
    }
}
```

EmpInfo.cs

```
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace BlogTrackerAPI.Models
{
    [Table("EmpInfo")]
    public class EmpInfo
    {
        [Key]
        public int EmpInfoId { get; set; }

        public string? EmailId { get; set; }

        public string? Name { get; set; }

        public DateTime DateOfJoining { get; set; }

        public int PassCode { get; set; }
    }
}
```

BlogInfo.cs

```
using System.ComponentModel.DataAnnotations.Schema;
using System.ComponentModel.DataAnnotations;

namespace BlogTrackerAPI.Models
```

```
{
    [Table("BlogInfo")]
    public class BlogInfo
    {
        [Key]
        public int BlogInfoId { 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; }
    }
}
```

BlogDbContext.cs

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

namespace BlogTrackerAPI.Data
{
    public class BlogDbContext : DbContext
    {
        public BlogDbContext(DbContextOptions<BlogDbContext> options)
            : base(options)
        {
        }

        public DbSet<BlogTrackerAPI.Models.BlogInfo> BlogInfo { get; set; } =
default!;

        public DbSet<BlogTrackerAPI.Models.EmpInfo>? EmpInfo { get; set; }
        public DbSet<BlogTrackerAPI.Models.AdminInfo> AdminInfo { get; set; }
    }
}
```

AdminInfoesController.cs

```
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 BlogTrackerAPI.Data;
using BlogTrackerAPI.Models;
```

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

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

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

        // GET: api/AdminInfoes/5
        [HttpGet("{id}")]
        public async Task<ActionResult<AdminInfo>> GetAdminInfo(int id)
        {
          if (_context.AdminInfo == null)
          {
              return NotFound();
          }
            var adminInfo = await _context.AdminInfo.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.AdminId)
            {
                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.AdminInfo == null)
          {
              return Problem("Entity set 'BlogDbContext.AdminInfo'  is null.");
          }
            _context.AdminInfo.Add(adminInfo);
            await _context.SaveChangesAsync();

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

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

            _context.AdminInfo.Remove(adminInfo);
            await _context.SaveChangesAsync();

            return NoContent();
        }

        private bool AdminInfoExists(int id)
        {
            return (_context.AdminInfo?.Any(e => e.AdminId ==
id)).GetValueOrDefault();
```

```
        }
    }
}
```

EmpInfoesController.cs

```
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 BlogTrackerAPI.Data;
using BlogTrackerAPI.Models;

namespace BlogTrackerAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class EmpInfoesController : ControllerBase
    {
        private readonly BlogDbContext _context;

        public EmpInfoesController(BlogDbContext context)
        {
            _context = context;
        }

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

        // GET: api/EmpInfoes/5
        [HttpGet("{id}")]
        public async Task<ActionResult<EmpInfo>> GetEmpInfo(int id)
        {
          if (_context.EmpInfo == null)
          {
              return NotFound();
          }
            var empInfo = await _context.EmpInfo.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.EmpInfoId)
            {
                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.EmpInfo == null)
          {
              return Problem("Entity set 'BlogDbContext.EmpInfo'  is null.");
          }
            _context.EmpInfo.Add(empInfo);
            await _context.SaveChangesAsync();

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

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

```
            }
            var empInfo = await _context.EmpInfo.FindAsync(id);
            if (empInfo == null)
            {
                return NotFound();
            }

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

            return NoContent();
        }

        private bool EmpInfoExists(int id)
        {
            return (_context.EmpInfo?.Any(e => e.EmpInfoId ==
id)).GetValueOrDefault();
        }
    }
}
```

BlogInfoesController.cs

```
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 BlogTrackerAPI.Data;
using BlogTrackerAPI.Models;

namespace BlogTrackerAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class BlogInfoesController : ControllerBase
    {
        private readonly BlogDbContext _context;

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

        // GET: api/BlogInfoes
        [HttpGet]
        public async Task<ActionResult<IEnumerable<BlogInfo>>> GetBlogInfo()
        {
          if (_context.BlogInfo == null)
          {
              return NotFound();
          }
            return await _context.BlogInfo.ToListAsync();
        }
```

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

```
            return Problem("Entity set 'BlogDbContext.BlogInfo'  is null.");
          }
            _context.BlogInfo.Add(blogInfo);
            await _context.SaveChangesAsync();

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

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

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

            return NoContent();
        }

        private bool BlogInfoExists(int id)
        {
            return (_context.BlogInfo?.Any(e => e.BlogInfoId ==
id)).GetValueOrDefault();
        }
    }
}
```

Appsettings.js

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "ConnectionStrings": {
    "Constr":
"Server=tcp:blogtracker.database.windows.net,1433;initialCatalog=BlogTrackerDb;Persi
stSecurityInfo=False;UserID=priyanka;Password=Workfolder@00987;MultipleActiveResultS
ets=False;Encrypt=True;TrustServerCertificate=True;"
  "AllowedHosts":
}
}
```

Program.cs

```
using BlogTrackerAPI.Data;
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at
https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.AddDbContext<BlogDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("Constr")));

var app = builder.Build();

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

app.UseAuthorization();

app.MapControllers();

app.Run();
```

BlogTrackerMVC

Login.cs

```
using System.ComponentModel.DataAnnotations;

namespace BlogTrackerMVC.Models
{
    public class Login
    {
        [Required]
        [EmailAddress]
        public string? Email { get; set; }

        [Required]
        [DataType(DataType.Password)]
        public string? Password { get; set; }
    }
}
```

EmployeeLogin.cs

```
using System.ComponentModel.DataAnnotations;

namespace BlogTrackerMVC.Models
{
    public class EmployeeLogin
    {
        [Required]
        [EmailAddress]
        public string? EmailId { get; set; }

        [Required]
        [DataType(DataType.Password)]
        public int PassCode { get; set; }
    }
}
```

AdminInfoesController.cs

```
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 BlogTrackerAPI.Data;
using BlogTrackerAPI.Models;
using BlogTrackerMVC.Models;

namespace BlogTrackerMVC.Controllers
{
    public class AdminInfoesController : Controller
    {
        private readonly BlogDbContext _context;

        public AdminInfoesController(BlogDbContext context)
        {
            _context = context;
        }
        [HttpGet]
        public IActionResult Login()
        {
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Login(Login model)
        {
            if (ModelState.IsValid)
            {
                // Check if the provided credentials are valid
                var user = await _context.AdminInfo
```

```
                    .FirstOrDefaultAsync(u => u.EmailId == model.Email && u.Password
== model.Password);

                if (user != null)
                {
                    // Redirect to a dashboard or another page after successful
login
                    return RedirectToAction("Index", "EmpInfoes");
                }
                else
                {
                    TempData["ErrorMessage"] = "Invalid login attempt";
                }
                ModelState.AddModelError(string.Empty, "Invalid login attempt");
            }

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

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

            var adminInfo = await _context.AdminInfo
                .FirstOrDefaultAsync(m => m.AdminId == id);
            if (adminInfo == null)
            {
                return NotFound();
            }

            return View(adminInfo);
        }

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

        // POST: AdminInfoes/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("AdminId,EmailId,Password")]
AdminInfo adminInfo)
```

```
        {
            if (ModelState.IsValid)
            {
                _context.Add(adminInfo);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(adminInfo);
        }

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

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

        // POST: AdminInfoes/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("AdminId,EmailId,Password")] AdminInfo adminInfo)
        {
            if (id != adminInfo.AdminId)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(adminInfo);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!AdminInfoExists(adminInfo.AdminId))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
```

```
                return RedirectToAction(nameof(Index));
            }
            return View(adminInfo);
        }

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

            var adminInfo = await _context.AdminInfo
                .FirstOrDefaultAsync(m => m.AdminId == id);
            if (adminInfo == null)
            {
                return NotFound();
            }

            return View(adminInfo);
        }

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

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

        private bool AdminInfoExists(int id)
        {
          return (_context.AdminInfo?.Any(e => e.AdminId ==
id)).GetValueOrDefault();
        }
    }
}
```

EmpInfoesController.cs

```
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 BlogTrackerAPI.Data;
using BlogTrackerAPI.Models;
using BlogTrackerMVC.Models;

namespace BlogTrackerMVC.Controllers
{
    public class EmpInfoesController : Controller
    {
        private readonly BlogDbContext _context;

        public EmpInfoesController(BlogDbContext context)
        {
            _context = context;
        }
        [HttpGet]
        public IActionResult Login()
        {
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Login(EmployeeLogin model)
        {
            if (ModelState.IsValid)
            {
                // Check if the provided credentials are valid
                var user = await _context.EmpInfo
                    .FirstOrDefaultAsync(u => u.EmailId == model.EmailId &&
u.PassCode == model.PassCode);

                if (user != null)
                {
                    // Redirect to a dashboard or another page after successful
login
                    return RedirectToAction("Index", "BlogInfoes");
                }
                else
                {
                    TempData["ErrorMessage"] = "Invalid login attempt";
                }
                ModelState.AddModelError(string.Empty, "Invalid login attempt");
            }

            return View(model);
        }
        // Employee Login action


        [HttpPost]
        public IActionResult EmployeeLogin(string emailId, int passCode)
        {
            // Implement authentication logic here
            var employee = _context.EmpInfo.FirstOrDefault(e => e.EmailId == emailId
&& e.PassCode == passCode);
```

```
            if (employee != null)
            {
                // Set a session or cookie to mark the employee as logged in
                HttpContext.Session.SetString("EmailId", employee.EmailId);

                return RedirectToAction("EmployeeBlogIndex", "BlogInfoes");
            }
            else
            {
                ModelState.AddModelError(string.Empty, "Invalid login attempt.");
                return View();
            }
        }

        // Employee Logout action
        public IActionResult Logout()
        {
            // Clear the session or cookie to log the employee out
            HttpContext.Session.Remove("EmployeeEmail");
            return RedirectToAction("Index", "BlogInfoes");
        }
        // GET: EmpInfoes
        public async Task<IActionResult> Index()
        {
              return _context.EmpInfo != null ?
                          View(await _context.EmpInfo.ToListAsync()) :
                          Problem("Entity set 'BlogDbContext.EmpInfo'  is null.");
        }

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

            var empInfo = await _context.EmpInfo
                .FirstOrDefaultAsync(m => m.EmpInfoId == id);
            if (empInfo == null)
            {
                return NotFound();
            }

            return View(empInfo);
        }

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

        // POST: EmpInfoes/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("EmpInfoId,EmailId,Name,DateOfJoining,PassCode")] EmpInfo empInfo)
        {
            if (ModelState.IsValid)
            {
                _context.Add(empInfo);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(empInfo);
        }

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

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

        // POST: EmpInfoes/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("EmpInfoId,EmailId,Name,DateOfJoining,PassCode")] EmpInfo empInfo)
        {
            if (id != empInfo.EmpInfoId)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(empInfo);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!EmpInfoExists(empInfo.EmpInfoId))
                    {
                        return NotFound();
                    }
                    else
                    {
```

```
                    throw;
                }
            }
            return RedirectToAction(nameof(Index));
        }
        return View(empInfo);
    }

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

        var empInfo = await _context.EmpInfo
            .FirstOrDefaultAsync(m => m.EmpInfoId == id);
        if (empInfo == null)
        {
            return NotFound();
        }

        return View(empInfo);
    }

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

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

    private bool EmpInfoExists(int id)
    {
      return (_context.EmpInfo?.Any(e => e.EmpInfoId ==
id)).GetValueOrDefault();
    }
}
}
```

BlogInfoesController.cs

```
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 BlogTrackerAPI.Data;
using BlogTrackerAPI.Models;

namespace BlogTrackerMVC.Controllers
{
    public class BlogInfoesController : Controller
    {
        private readonly BlogDbContext _context;

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

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

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

            var blogInfo = await _context.BlogInfo
                .FirstOrDefaultAsync(m => m.BlogInfoId == id);
            if (blogInfo == null)
            {
                return NotFound();
            }

            return View(blogInfo);
        }

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

        // POST: BlogInfoes/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("BlogInfoId,Title,Subject,DateOfCreation,BlogUrl,EmpEmailId")] BlogInfo
blogInfo)
        {
            if (ModelState.IsValid)
            {
                _context.Add(blogInfo);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(blogInfo);
        }

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

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

        // POST: BlogInfoes/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("BlogInfoId,Title,Subject,DateOfCreation,BlogUrl,EmpEmailId")] BlogInfo
blogInfo)
        {
            if (id != blogInfo.BlogInfoId)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(blogInfo);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!BlogInfoExists(blogInfo.BlogInfoId))
                    {
                        return NotFound();
                    }
                    else
```

```
                {
                    throw;
                }
            }
            return RedirectToAction(nameof(Index));
        }
        return View(blogInfo);
    }

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

        var blogInfo = await _context.BlogInfo
            .FirstOrDefaultAsync(m => m.BlogInfoId == id);
        if (blogInfo == null)
        {
            return NotFound();
        }

        return View(blogInfo);
    }

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

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

    private bool BlogInfoExists(int id)
    {
      return (_context.BlogInfo?.Any(e => e.BlogInfoId ==
id)).GetValueOrDefault();
    }
}
}
```

Appsettings.js

```
{
```

```
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "ConnectionStrings: {
    "Constr": "Server=tcp:blogtracker.database.windows.net,1433;initial
Catalog=BlogTrackerDb;Persist Security Info=False;User
ID=priyanka;Password=Workfolder@00987;MultipleActiveResultSets=False;Encrypt=True;Tr
ustServerCertificate=True;"
  "AllowedHosts": "*"
}
```

Program.cs

```
using BlogTrackerAPI.Data;
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();
builder.Services.AddDbContext<BlogDbContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("Constr")));

var app = builder.Build();

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

app.UseRouting();

app.UseAuthorization();

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

app.Run();
```