Source Code

AdminInfo.cs

```
using System.ComponentModel.DataAnnotations;

namespace BlogTracker.Models
{
   public class AdminInfo
   {
      [Key]
      public int AdminInfoId { get; set; }
      [Required]
      [EmailAddress]
      public string EmailId { get; set; }
      [Required]
      [MinLength(6)]
      public string Password { get; set; }
   }
}
```

BlogInfo.cs

```
using System.ComponentModel.DataAnnotations;

namespace BlogTracker.Models
{
   public class BlogInfo
   {
      [Key]
      public int BlogInfoId { get; set; }
      [Required]
      [StringLength(50)]
      public string Title { get; set; }
      [Required]
      [StringLength(50)]
      public string Subject { get; set; }
      [Required]
      public DateTime DateOfCreation { get; set; }
      [Required]
      [Url]
      public string BlogUrl { get; set; }
      [Required]
      [EmailAddress]
      public string EmpEmailId { get; set; }
   }
}
```

EmpInfo.cs

```
using System.ComponentModel.DataAnnotations;

namespace BlogTracker.Models
{
    public class EmpInfo
    {
        [Key]
        public int EmpInfoId { get; set; }
        [Required]
        [EmailAddress]
        public string EmailId { get; set; }
        [Required]
        [StringLength(50)]
        public string Name { get; set; }
        [Required]
        public DateTime DateOfJoining { get; set; }
        [Required]
        [Range(1000, 9999)]
        public int PassCode { get; set; }
    }
}
```

AdminInfoController.cs

```
using BlogTracker.Data;
using BlogTracker.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace BlogTracker.Controllers
{

    public class AdminInfoController : Controller
    {
        // Inject the DbContext and any other dependencies needed
        private readonly BlogTrackerDbContext _context;

        public AdminInfoController(BlogTrackerDbContext context)
        {
            _context = context;
        }

        [HttpGet]
        public IActionResult Login()
```

```
        {
            return View();
        }

        // Add an action method to handle the login POST request
        [HttpPost]
        [ValidateAntiForgeryToken]
        public IActionResult Login(AdminInfo model)
        {
            if (ModelState.IsValid)
            {
                // Perform the authentication logic here, e.g., check if the provided
credentials are valid
                var admin = _context.AdminInfo.FirstOrDefault(a => a.EmailId ==
model.EmailId && a.Password == model.Password);

                if (admin != null)
                {
                    // Authentication successful, you can set a session or cookie to mark
the user as logged in
                    // For simplicity, let's assume successful login means setting a session
variable
                    HttpContext.Session.SetString("AdminEmail", admin.EmailId);

                    return RedirectToAction("Index", "EmpInfoes"); // Redirect to the home
page or a dashboard
                }
                else
                {
                    ModelState.AddModelError(string.Empty, "Invalid login attempt.");
                }
            }

            // If the model state is not valid or authentication fails, return to the login
page with an error message
            return View(model);
        }

        // Add a logout action method if needed
        public IActionResult Logout()
        {
            // Clear the session or cookie to log the user out
            HttpContext.Session.Remove("AdminEmail");
            return RedirectToAction("Index", "BlogInfoes");
        }
    }
}
```

BlogInfoController.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using BlogTracker.Data;
using BlogTracker.Models;
using Microsoft.AspNetCore.Authorization;

namespace BlogTracker.Controllers
{

   public class BlogInfoesController : Controller
   {
      private readonly BlogTrackerDbContext _context;

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

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

      public async Task<IActionResult> EmployeeBlogIndex()
      {
         return _context.BlogInfo != null ?
                     View(await _context.BlogInfo.ToListAsync()) :
                     Problem("Entity set 'BlogTrackerDbContext.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 'BlogTrackerDbContext.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();
        }
    }
}
```

EmpInfoController.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using BlogTracker.Data;
using BlogTracker.Models;

namespace BlogTracker.Controllers
{
    public class EmpInfoesController : Controller
    {
        private readonly BlogTrackerDbContext _context;

        public EmpInfoesController(BlogTrackerDbContext context)
```

```
        {
            _context = context;
        }

        // Employee Login action
        public IActionResult EmployeeLogin()
        {
            return View();
        }

        [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("EmployeeEmail", 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 'BlogTrackerDbContext.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 'BlogTrackerDbContext.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();
        }
    }
}
```

BlogTrackerDbContext.cs

```
using Microsoft.EntityFrameworkCore;

namespace BlogTracker.Data
```

```
{
    public class BlogTrackerDbContext : DbContext
    {
        public BlogTrackerDbContext (DbContextOptions<BlogTrackerDbContext>
options)
            : base(options)
        {
        }

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

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

Program.cs

```
using Microsoft.EntityFrameworkCore;
using BlogTracker.Data;
using BlogTracker.Models;

var builder = WebApplication.CreateBuilder(args);
builder.Services.AddDbContext<BlogTrackerDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("BlogTrackerDbCo
ntext") ?? throw new InvalidOperationException("Connection string
'BlogTrackerDbContext' not found.")));

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

var app = builder.Build();

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

app.UseRouting();

app.UseAuthorization();
app.UseSession();
```

```
using (var scope = app.Services.CreateScope())
{
    var services = scope.ServiceProvider;
    var dbContext = services.GetRequiredService<BlogTrackerDbContext>();

    // Check if there are any existing admin records
    if (!dbContext.AdminInfo.Any())
    {
        // Create a new AdminInfo instance with email and password
        var admin = new AdminInfo
        {
            EmailId = "admin@example.com",
            Password = "adminpassword" // You should hash the password in a real
application
        };

        // Add the admin record to the database
        dbContext.AdminInfo.Add(admin);
        dbContext.SaveChanges();
    }
}

app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
        name: "adminLogin",
        pattern: "admin/login",
        defaults: new { controller = "AdminInfo", action = "Login" });

    endpoints.MapControllerRoute(
        name: "employeeLogin",
        pattern: "employee/login",
        defaults: new { controller = "EmpInfo", action = "EmployeeLogin" });

    endpoints.MapControllerRoute(
        name: "employeeLogout",
        pattern: "employee/logout",
        defaults: new { controller = "EmpInfo", action = "Logout" });

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

app.Run();
```

appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "BlogTrackerDbContext":
"Server=tcp:newsqlserver3058.database.windows.net,1433;Initial
Catalog=BlogTrackerDb;User
ID=admin123;Password=admin@123;Encrypt=True;TrustServerCertificate=true;"
  }
}
```

Web API Source Code

BlogInfoController.cs

```
using BlogTracker.Data;
using BlogTracker.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace AppServiceLayer.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class BlogInfoController : ControllerBase
    {
        private readonly BlogTrackerDbContext _dbContext;

        public BlogInfoController(BlogTrackerDbContext dbContext)
        {
            _dbContext = dbContext;
        }

        // GET: api/BlogInfo
        [HttpGet]
        public async Task<ActionResult<IEnumerable<BlogInfo>>> GetBlogInfo()
        {
            return await _dbContext.BlogInfo.ToListAsync();
        }

        // GET: api/BlogInfo/5
```

```
        [HttpGet("{id}")]
        public async Task<ActionResult<BlogInfo>> GetBlogInfo(int id)
        {
            var blogInfo = await _dbContext.BlogInfo.FindAsync(id);

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

            return blogInfo;
        }

        // POST: api/BlogInfo
        [HttpPost]
        public async Task<ActionResult<BlogInfo>> PostBlogInfo(BlogInfo blogInfo)
        {
            _dbContext.BlogInfo.Add(blogInfo);
            await _dbContext.SaveChangesAsync();

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

        // PUT: api/BlogInfo/5
        [HttpPut("{id}")]
        public async Task<IActionResult> PutBlogInfo(int id, BlogInfo blogInfo)
        {
            if (id != blogInfo.BlogInfoId)
            {
                return BadRequest();
            }

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

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

```
            }
        }

        return NoContent();
    }

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

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

        return NoContent();
    }

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

EmpIngoController.cs

```
using BlogTracker.Data;
using BlogTracker.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace AppServiceLayer.Controllers
{
  [Route("api/[controller]")]
  [ApiController]
  public class EmpInfoController : ControllerBase
  {
    private readonly BlogTrackerDbContext _dbContext;

    public EmpInfoController(BlogTrackerDbContext dbContext)
    {
      _dbContext = dbContext;
```

```
        }

        // GET: api/EmpInfo
        [HttpGet]
        public async Task<ActionResult<IEnumerable<EmpInfo>>> GetEmpInfo()
        {
            return await _dbContext.EmpInfo.ToListAsync();
        }

        // GET: api/EmpInfo/5
        [HttpGet("{id}")]
        public async Task<ActionResult<EmpInfo>> GetEmpInfo(int id)
        {
            var empInfo = await _dbContext.EmpInfo.FindAsync(id);

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

            return empInfo;
        }

        // POST: api/EmpInfo
        [HttpPost]
        public async Task<ActionResult<EmpInfo>> PostEmpInfo(EmpInfo empInfo)
        {
            _dbContext.EmpInfo.Add(empInfo);
            await _dbContext.SaveChangesAsync();

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

        // PUT: api/EmpInfo/5
        [HttpPut("{id}")]
        public async Task<IActionResult> PutEmpInfo(int id, EmpInfo empInfo)
        {
            if (id != empInfo.EmpInfoId)
            {
                return BadRequest();
            }

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

            try
            {
                await _dbContext.SaveChangesAsync();
```

```
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!EmpInfoExists(id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }

            return NoContent();
        }

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

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

            return NoContent();
        }

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

Program.cs

```
using BlogTracker.Data;
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);
```

```
// Add services to the container.

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

options.UseSqlServer(builder.Configuration.GetConnectionString("BlogTrackerDbCo
ntext") ?? throw new InvalidOperationException("Connection string
'BlogTrackerDbContext' not found.")));

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

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}
app.UseCors(options =>
options.WithOrigins("http://localhost:5072").AllowAnyMethod().AllowAnyHeader());
app.UseAuthorization();

app.MapControllers();

app.Run();
```

TestClass.cs

```
using BlogTracker.Controllers;
using BlogTracker.Data;
using BlogTracker.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Moq;
using NUnit.Framework;

namespace BlogTracker
{
    [TestFixture]
    public class TestClass
    {
        [Test]
```

```
public void BlogInfoTest()
{
    //Arrange
    var blogid = new BlogInfo { BlogInfoId = 1 };
    var blogTitle = new BlogInfo { Title = "Test" };

    //Act
    var blogIdTest = blogid.BlogInfoId;
    var blogTitleTest = blogTitle.Title;

    //Assert
    Assert.AreEqual(1, blogIdTest);
    Assert.AreEqual("Test", blogTitleTest);
}

[Test]
public void EmpInfoTest()
{
    //Arrange
    var empid = new EmpInfo { EmpInfoId = 1 };
    var empName = new EmpInfo { Name = "Name" };

    //Act
    var empIdTest = empid.EmpInfoId;
    var empNameTest = empName.Name;

    //Assert
    Assert.AreEqual(1, empIdTest);
    Assert.AreEqual("Name", empNameTest);
}

//Moq Test
[Test]
public async Task Create_ValidEmpInfo_RedirectsToIndex()
{
    // Arrange
    var options = new DbContextOptionsBuilder<BlogTrackerDbContext>()
        .UseInMemoryDatabase(databaseName: "TestDatabase")
        .Options;

    var mockContext = new Mock<BlogTrackerDbContext>(options);
    var controller = new EmpInfoesController(mockContext.Object);

    var empInfo = new EmpInfo
    {
        EmpInfoId = 1,
        EmailId = "test@example.com",
        Name = "John Doe",
```

```
            DateOfJoining = DateTime.Now,
            PassCode = 1234
        };

        // Act
        var result = await controller.Create(empInfo) as RedirectToActionResult;

        // Assert
        Assert.IsNotNull(result);
        Assert.AreEqual("Index", result.ActionName);
    }

  }
}
```