```
using System;
using System.Collections.Generic;
namespace BlogTrackerWebAPI.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 BlogTrackerWebAPI.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;
namespace BlogTrackerWebAPI.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; }
}
```

```
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 BlogTrackerWebAPI.Models;
namespace BlogTrackerWebAPI.Controllers
{
 [Route("api/[controller]")]
 [ApiController]
 public class AdminInfoesController : ControllerBase
 {
 private readonly BlogdbserverContext _context;
 public AdminInfoesController(BlogdbserverContext 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 'BlogdbserverContext.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 BlogTrackerWebAPI.Models;
namespace BlogTrackerWebAPI.Controllers
{
 [Route("api/[controller]")]
 [ApiController]
 public class BlogInfoesController : ControllerBase
 {
 private readonly BlogdbserverContext _context;
 public BlogInfoesController(BlogdbserverContext 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 'BlogdbserverContext.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.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;
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 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["BlogTracker"].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", "Emp");
 }
 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["BlogTracker"].ConnectionString;
 SqlConnection con = new SqlConnection(connection);
 string cmd = "Select EmailId, PassCode from EmpInfo where
EmailId=@Emailid and PassCode=@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"] = "Employee Login Details Failed";
 }
 con.Close();
 return View();
 }
 public ActionResult Logout()
 {
 FormsAuthentication.SignOut();
 Session.Clear();
 return RedirectToAction("GuestIndex", "Blog");
 }
 }
}
```

```
@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", type =
"password" } })
 @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", type =
"password" } })
 @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")
}
```