public class MyCustomMiddleware

{

private readonly RequestDelegate \_next;

public MyCustomMiddleware(RequestDelegate next)

{

\_next = next;

}

public async Task InvokeAsync(HttpContext context)

{

var startTime = DateTime.Now;

await \_next(context);

var endTime = DateTime.Now;

var elapsedTime = endTime - startTime;

var logMessage = $"{context.Request.Method} {context.Request.Path} {context.Response.StatusCode} {elapsedTime.TotalMilliseconds}ms";

Console.WriteLine(logMessage);

}

}

Register Middleware

app.UseMiddleware<MyCustomMiddleware>();

Register XML Format

builder.Services.AddControllers().AddXmlSerializerFormatters();

**Model Binding Techniques**

Form Data, Query Strings, Route Data, HTTP Headers, Request Body

[Route("api/[controller]")]

[ApiController]

public class UserController : ControllerBase

{

private static List<UserModel> Users = new List<UserModel>

{

new UserModel { Id = 1, Name = "Rakesh", Department = "IT", Gender = "Male", Salary = 1000 },

new UserModel { Id = 2, Name = "Priyanka", Department = "IT", Gender = "Female", Salary = 2000 }

};

[HttpGet]

public IActionResult CreateUser(**[FromForm]** UserModel user)

public IActionResult GetProductById(**[FromRoute]** int id)

public IActionResult GetResource([FromHeader] string Authorization)

public IActionResult CreateProduct([FromBody] Product product)

public IActionResult GetProducts(**[FromQuery]** string Department)

{

// Implementation to retrieve employees based on the Department

var FilteredUsers = Users.Where(emp => emp.Department.Equals(Department, StringComparison.OrdinalIgnoreCase)).ToList();

if (FilteredUsers.Count > 0)

{

return Ok(FilteredUsers);

}

return NotFound($"No Users Found with Department: {Department}");

}

}

[Route("Emp/ById/{Id}")]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]

##### **ILog.cs**

namespace SingletoninMVC.Logger

{

public interface ILog

{

void LogException(string message);

}

}

**Log.cs**

using System;

using System.IO;

using System.Text;

namespace SingletoninMVC.Logger

{

public sealed class Log : ILog

{

private Log()

{

}

private static readonly Log LogInstance = new Log();

public static Log GetInstance()

{

return LogInstance;

}

//This Method Log the Exception Details in a Log File

public void LogException(string message)

{

//create logic to do any task

}

}

}

public class EmployeeController : Controller

{

private ILog \_ILog;

private EmployeeDBContext db = new EmployeeDBContext();

public EmployeeController()

{

\_ILog = Log.GetInstance();

}

//Whenever Any Exception Occurred, the following OnException Method will Execute

protected override void OnException(ExceptionContext filterContext)

{

//First, Log the Exception Details

\_ILog.LogException(filterContext.Exception.ToString());

}

}