**1. ErrorLogger Class**

Enhance the ErrorLogger class to include logging via ILogger:

csharp

Copy code

using System;

using System.Data;

using System.Diagnostics;

using System.Threading.Tasks;

using Dapper;

using Microsoft.Extensions.Logging;

public static class ErrorLogger

{

private static readonly string \_connectionString = "YourConnectionStringHere";

private static ILogger \_logger;

/// <summary>

/// Configures the logger for application logging.

/// </summary>

public static void ConfigureLogger(ILogger logger)

{

\_logger = logger;

}

/// <summary>

/// Logs an error to the SBCES.Errors table.

/// </summary>

public static async Task LogErrorToDatabaseAsync(

string userName,

string moduleName,

string functionName,

int? lineNumber,

string errorDescription,

int? errorNumber)

{

using (IDbConnection db = new System.Data.SqlClient.SqlConnection(\_connectionString))

{

string query = @"

INSERT INTO SBCES.Errors (UserName, DateTime, ModuleName, FunctionName, LineNum, ErrorDesc, ErrNum)

VALUES (@UserName, @DateTime, @ModuleName, @FunctionName, @LineNum, @ErrorDesc, @ErrNum)";

var parameters = new

{

UserName = userName,

DateTime = DateTime.Now,

ModuleName = moduleName,

FunctionName = functionName,

LineNum = lineNumber,

ErrorDesc = errorDescription,

ErrNum = errorNumber

};

await db.ExecuteAsync(query, parameters);

}

}

/// <summary>

/// Logs an exception to the database and the application logger.

/// </summary>

public static async Task LogExceptionAsync(Exception exception, string userName)

{

var stackTrace = new StackTrace(exception, true);

var frame = stackTrace.GetFrame(0);

string moduleName = frame?.GetFileName() ?? "Unknown Module";

string functionName = frame?.GetMethod()?.Name ?? "Unknown Function";

int lineNumber = frame?.GetFileLineNumber() ?? 0;

// Log to database

await LogErrorToDatabaseAsync(

userName: userName,

moduleName: moduleName,

functionName: functionName,

lineNumber: lineNumber,

errorDescription: exception.Message,

errorNumber: exception.HResult);

// Log to application logger if configured

\_logger?.LogError(exception, "Exception occurred in {ModuleName} at {FunctionName}, Line {LineNumber}", moduleName, functionName, lineNumber);

}

/// <summary>

/// Logs a general application message using the configured logger.

/// </summary>

public static void LogMessage(string message, LogLevel logLevel = LogLevel.Information)

{

if (\_logger != null)

{

\_logger.Log(logLevel, message);

}

}

}

**2. Program.cs**

Configure the application logging in Program.cs and set up middleware to use the enhanced ErrorLogger.

csharp

Copy code

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Diagnostics;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Logging;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container

builder.Services.AddControllersWithViews();

// Configure logging

builder.Logging.ClearProviders();

builder.Logging.AddConsole();

var loggerFactory = LoggerFactory.Create(logging => logging.AddConsole());

var logger = loggerFactory.CreateLogger("GlobalLogger");

// Build application

var app = builder.Build();

// Configure the ErrorLogger

ErrorLogger.ConfigureLogger(logger);

// Global exception handler middleware

app.UseExceptionHandler(errorApp =>

{

errorApp.Run(async context =>

{

var exceptionHandlerPathFeature = context.Features.Get<IExceptionHandlerPathFeature>();

if (exceptionHandlerPathFeature?.Error != null)

{

await ErrorLogger.LogExceptionAsync(

exceptionHandlerPathFeature.Error,

userName: context.User?.Identity?.Name ?? "Anonymous");

}

context.Response.StatusCode = 500;

await context.Response.WriteAsync("An internal error occurred.");

});

});

// Default routing

app.UseRouting();

app.MapDefaultControllerRoute();

app.Run();

**3. Sample Controller**

Use the ErrorLogger for both application logging and exception logging.

csharp

Copy code

using Microsoft.AspNetCore.Mvc;

public class HomeController : Controller

{

public IActionResult Index()

{

try

{

// Simulate a message log

ErrorLogger.LogMessage("Index action is invoked.");

// Simulate an exception

throw new InvalidOperationException("This is a test exception.");

}

catch (Exception ex)

{

ErrorLogger.LogExceptionAsync(ex, User?.Identity?.Name ?? "Anonymous").Wait();

return StatusCode(500, "An error occurred.");

}

}

public IActionResult About()

{

// Log a custom message

ErrorLogger.LogMessage("About action executed successfully.");

return View();

}

}

**4. View Files**

**Index.cshtml**

html

Copy code

<h1>Welcome to the Home Page</h1>

**About.cshtml**

html

Copy code

<h1>About Page</h1>