**GLOBAL AND CUSTOMIZE EXCEPTION HANDLING**

The exception handling feature helps in dealing with the error that could appear in the code. Generally, a try-catch block is used for handling these errors in the code. But using an Exception Handling Middleware makes our work easier.

For this we add a middleware in pipeline class where using ADDMIDDLEWARE function through which the application will catch the errors in and out of the application.

**Functioning of ExceptionMiddleware:**

**In ErrorDetails.cs**

public class ErrorDetails

{

public int StatusCode { get; set; }

public string Message { get; set; }

public override string ToString()

{

return JsonSerializer.Serialize(this);

}

}

**In GlobalExceptionMiddleware.cs**

public class GlobalExceptionMiddleware

{

private RequestDelegate \_requestDelegate;

private ILogger \_logger;

public GlobalExceptionMiddleware(RequestDelegate requestDelegate, ILogger<GlobalExceptionMiddleware> logger)

{

\_requestDelegate = requestDelegate;

\_logger = logger;

}

public ILogger Get\_logger()

{

return \_logger;

}

public async Task InvokeAsync(HttpContext context)

{

try

{

await \_requestDelegate(context);

}

catch (Exception error)

{

var response = context.Response;

response.ContentType = "application/json";

switch (error)

{

case ApplicationException e:

// custom application error

response.StatusCode = (int)HttpStatusCode.BadRequest;

break;

case KeyNotFoundException e:

// not found error

response.StatusCode = (int)HttpStatusCode.NotFound;

break;

case UnauthorizedAccessException e:

// unauthorized error

response.StatusCode = (int)HttpStatusCode.Unauthorized;

break;

case MethodAccessException:

// key not found error

response.StatusCode = (int)HttpStatusCode.MethodNotAllowed;

break;

default:

// unhandled error

response.StatusCode = (int)HttpStatusCode.InternalServerError;

break;

}

var result = JsonSerializer.Serialize(new { message = error?.Message });

\_logger.LogError(result);

await response.WriteAsync(result);

}

}

}

**In ExceptionMiddlewareExtention.cs**

public static class ExceptionMiddlewareExtention

{

public static void ConfigureGlobalApplicationMiddleware(this IApplicationBuilder

applicationBuilder)

{

applicationBuilder.UseMiddleware<GlobalExceptionMiddleware>();

}

}

**In Program.cs**

//implementing Global Exception Handling

app.UseMiddleware<GlobalExceptionMiddleware>();