CROSS | OVER

Preconditions

1. You should work on your local machine.

2. Use the following tools:

    1. Visual Studio 2017 (Community, Professional or Enterprise)

    2. MongoDB as database

Objective

Work on an application solution based on REST to log incoming messages from 3rd party applications. The solution should handle a simple log message as json and store it for data mining purpose.

We already started to work on this application, now we want you to do a code review and implement some new requirements.

Tasks to do

Please download the current source code and the API documentation here.

Part 1 - Perform reviews:

The given code needs to be setup, configured and run properly for you to review or develop it further. Please do so locally. If you find any issues to do so, please fix them, but no need to fix any issues you find in the reviews as described below.

The given code was implemented in a hurry. It has several functional, technical design and coding flaws. Based on the industry best practices and standards these flaws must be identified. You need to perform reviews, find most of the flaws, log them, suggest how best to fix them, and give your reasoning and rationale. No need to fix the issues you find.

The source code provided contains a folder called “Docs” which contains Excel sheets that can be used for logging review remarks. If you do not have the proper software to work on them, then use any file format and make sure the following info is captured for each review remark in the form of a table (Create two separate files, one for functional+design review and one for code review):

- Review remark: Description of the issue found or suggestion for improvement.

- Artifact / Location of issue: Which file(s) and method(s) the remark is related to?
- Why it is an issue? Why do you think you are making this remark? What is the rationale/ basis of the remark? Any references you would like to quote?
- How can it be best fixed? What are the changes you suggest to fix the issue or make an improvement? And why do you think so?

Part 2 - Add new features:

Requirements <u>already implemented by our team</u>:

The following requirements are already implemented on the source code provided:

1. A registration endpoint that allows clients to register their applications. (See API documentation)
2. An authentication endpoint that allows applications to authenticate before sending log messages. (See API documentation)

Requirements <u>to be implemented by you</u>:

1. The endpoint to send log messages (See API documentation):
   1. All log requests must be handled asynchronously.
   2. This endpoint requires authentication through an access token (provided by the authentication endpoint already implemented - See previous section).
2. Implement a rate-limiting request to avoid possible attacks, allowing only 60 requests per minute by application. If you use up your 60 api calls then you will receive “rate limit exceeded” message and must wait 5 minutes before can make requests again (See API documentation).
3. Write unit tests, even for the code that we’ve already implemented.

IMPORTANT!
We use an automated tool to do the evaluation of your code, so:

1. You must strictly follow the API documentation provided, because our automated tests are based on this documentation.
   For example, there are an endpoint “/log” expecting some parameters: If you implement this endpoint as “/logger”, or change the name of any parameter, our tests on this endpoint will fail and this will affect your overall score.
2. The source code already have a test project. You must implement your unit tests on the test project provided, tests created in any other project will be ignored.
3. Any extra package you need must be added to the project using the NuGet Package Manager, and our tool will download them before build and run your code. Also, please do not remove any package from the original source code, even if it’s not been used (maybe our automated tool will use it).

Part 3 - Readme Document:

The source code provided contains a file called “Readme.txt”, where you can provide more information, if needed:

1. Any assumptions made and missing requirements that are not covered in the requirements.
2. Any feedback you wish to give about improving the assignment.

Delivery / What to submit

Please, read and follow this section carefully.
Any delivery that does not follow this section will score much less or simply won’t be evaluated:

- You must deliver a zip file with exactly the same structure of the file with the current source code we provided to you.
- It’s simple: Download the file, unzip it, work on the documents from the “Docs” folder and on the code from the “Src” folder, edit the “Readme.txt” (if needed) and them zip it again.

You just have to do one change: Add your name at the beginning of the zip filename.
For example: Our zip file is called “MessageLogger_DotnetCore.zip”, you must rename your file as “<your_name>_MessageLogger_.DotnetCore.zip”