**SOLUTION APPROACH DOCUMENT**

**Seri Log Logging in .Net 6.0**

***Report Submitted***

***by***

ANIKET KUMAR (VTU16970)

S .SRI SINDHU (VTU12405 )

MANU KUMARI (VTU16872 )

NITIN DAS (VTU16807)

***Under the guidance of***

Ms. S Shanmuga priya(Assistant Professor)

Dr. R.Lotus (Assistant Professor)

Ms. P.Yashika(Assistant Professor)

****

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING  
SCHOOL OF COMPUTING**

**VEL TECH RANGARAJAN Dr. SAGUNTHALA R&D  
INSTITUTE OF SCIENCE AND TECHNOLOGY**

**ABSTRACT**

One of the most fundamental requirements for every programmer is logging. Troubleshooting any application issues is essential.

By using straightforward parameters, logging frameworks make it simple to deliver your logs to many locations. Without altering your code, Serilog employs objects referred to as sinks to transmit your logs to a text file, database, log management programmed, or potentially dozens of additional locations.

Serilog Sinks can write log events in a variety of forms. Sinks are another term for the type of data store we use to define where we want to save the log data for our application. The most typical and well-liked sinks are files or consoles because they are simple to set up. There are additional sinks that let the storing of name properties and their values and support structured logging.

**Keywords:** SeriLog, SeriLog Sinks, log, properties, structures

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| Fig 2.1 | Component Design of SeriLog Component | 2 |
| Fig 2.2 | Sequence Diagram of SeriLog Component | 3 |
| Fig 2.3 | Class Diagram of SeriLog Component | 4 |
|  |  |  |

**TABLE OF CONTENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ABSTRACT** | | | | **ii** |
| **LIST OF FIGURES** | | | | **iii** |
| **ABBREVIATIONS** | | | | **iv** |
| **1** | **INTRODUCTION** | | |  |
|  | 1.1 | ABOUT THE DOCUMENT | |  |
|  |  | 1.1.1 | PURPOSE & SCOPE | 1 |
|  |  | 1.1.2 | OVERVIEW | 1 |
| **2** | **COMPONENT DESIGN** | | |  |
|  | 2.1 | COMPONENT DESIGN DIAGRAM | |  |
|  |  | 2.1.1 | OVERALL WORKFLOW | 2 |
|  |  | 2.1.2 | SEQUENCE DIAGRAM | 3 |
|  |  | 2.1.3 | LOW LEVEL DESIGN | 4 |
|  |  | 2.1.4 | ACTIVITY DIAGRAM | 5 |
| **3** | **TECHNOLOGY AND FRAMEWORK** | | |  |
|  | 3.1 | Dot NET | |  |
|  |  |  | .NET CORE 6.0 | 6 |
|  | 3.2 | DATABASE | |  |
|  |  |  | SQL | 7 |
|  | 3.3 | TOOLS USED | |  |
|  |  | 3.3.1 VISUAL STUDIO CODE | | 8 |
| **4** | **SOLUTION APPROACH** | | |  |
|  | 4.1 | APPROACH DESCRIPTION | | 9 |

**CHAPTER 1**

**INTRODUCTION**

* 1. **ABOUT THIS DOCUMENT** 
     1. **PURPOSE & SCOPE**

The main purpose of this component is to store logs of an application which has been retrieved from SeriLogs.

It takes the api calls on behalf of SeriLogs, and sends those calls to SeriLog for retrieving the logs while transforming it to be of actual use for the database.

It is essential for efficient logging due it’s versatility of dealing with different types of logs in different ways.

* + 1. **OVERVIEW**

Logging is a major game-changer in the long term, even though it's not one of the first things we often set up when starting a new application. If we have enough logging in place, we can learn more about the environment in which an error occurred so that we can quickly pinpoint its root cause.

In the case of distributed systems, it will be quite difficult to locate the precise logs connected with that specific request if you need to trace them throughout the systems. This gives you the option to add any special identification so you can follow the log message between platforms. Tracking logs for distributed transactions across many database systems would be another excellent example.

**CHAPTER 2**

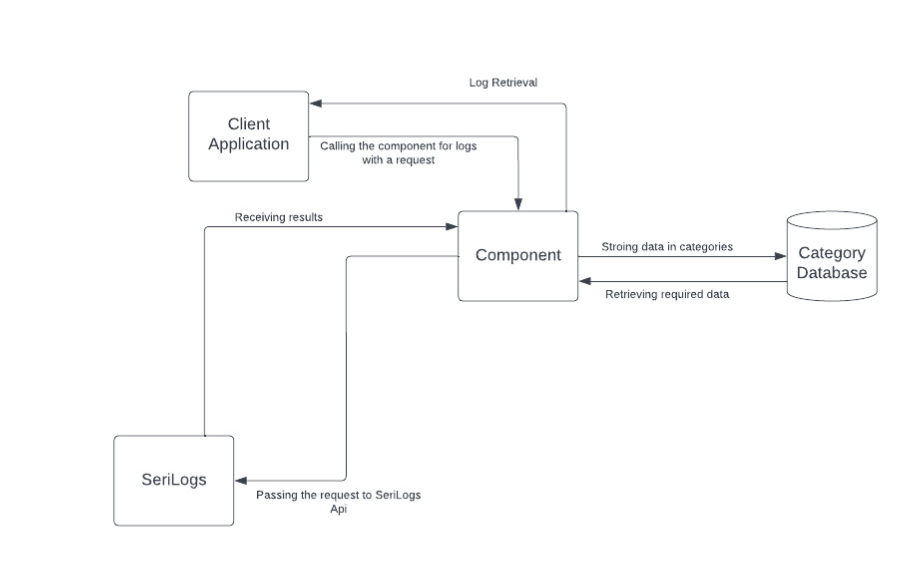
**COMPONENT DESIGN**

**2.1 COMPONENT DESIGN DIAGRAM**

This section describes about the component design and the specific way of designing.

**2.1.1 OVERALL WORKFLOW**

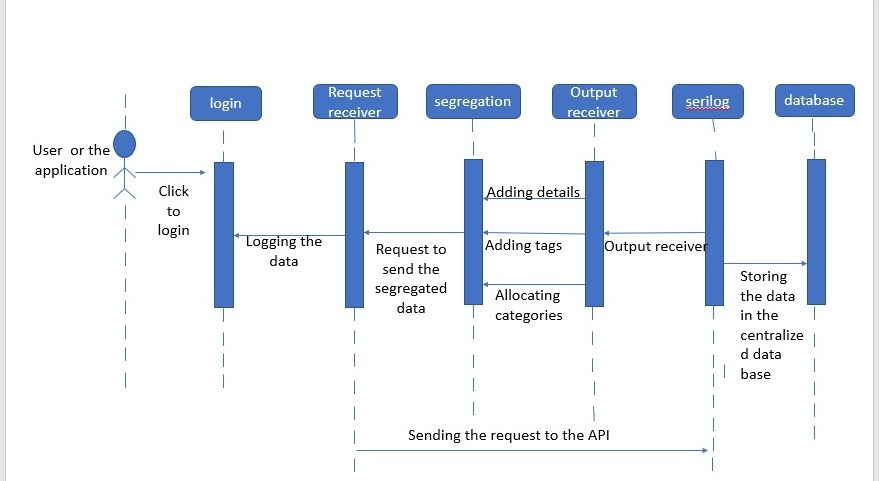
This component workflow diagram describes the complete working process of the interdependent component. Initially a logging request is made to the component instead of Seri Log. The component then sends the request to Seri Log Api and receives a result. This result is first transformed and edited to fit the provided categories and then stored in the data base. Meanwhile, the original result is sent to the client application after being received. This component acts as middle-man between Client Application and the Seri Log Api.



**Fig 2.1 :** *Component Design of Seri Log Component*

**2.1.2 SEQUENCE DIAGRAM**

This sequence of diagrams displays the sequence of actions performed during the functions of the application. As explained in the above diagram. The client makes a request to the component which then transfers the request to Seri Log. This request is handled by SeriLog and the result is then sent back to the component. The component then perform two operations. One it sends the retrieved result to the Client and other it transforms and categorizes the result into the available categories in the database and stores them.

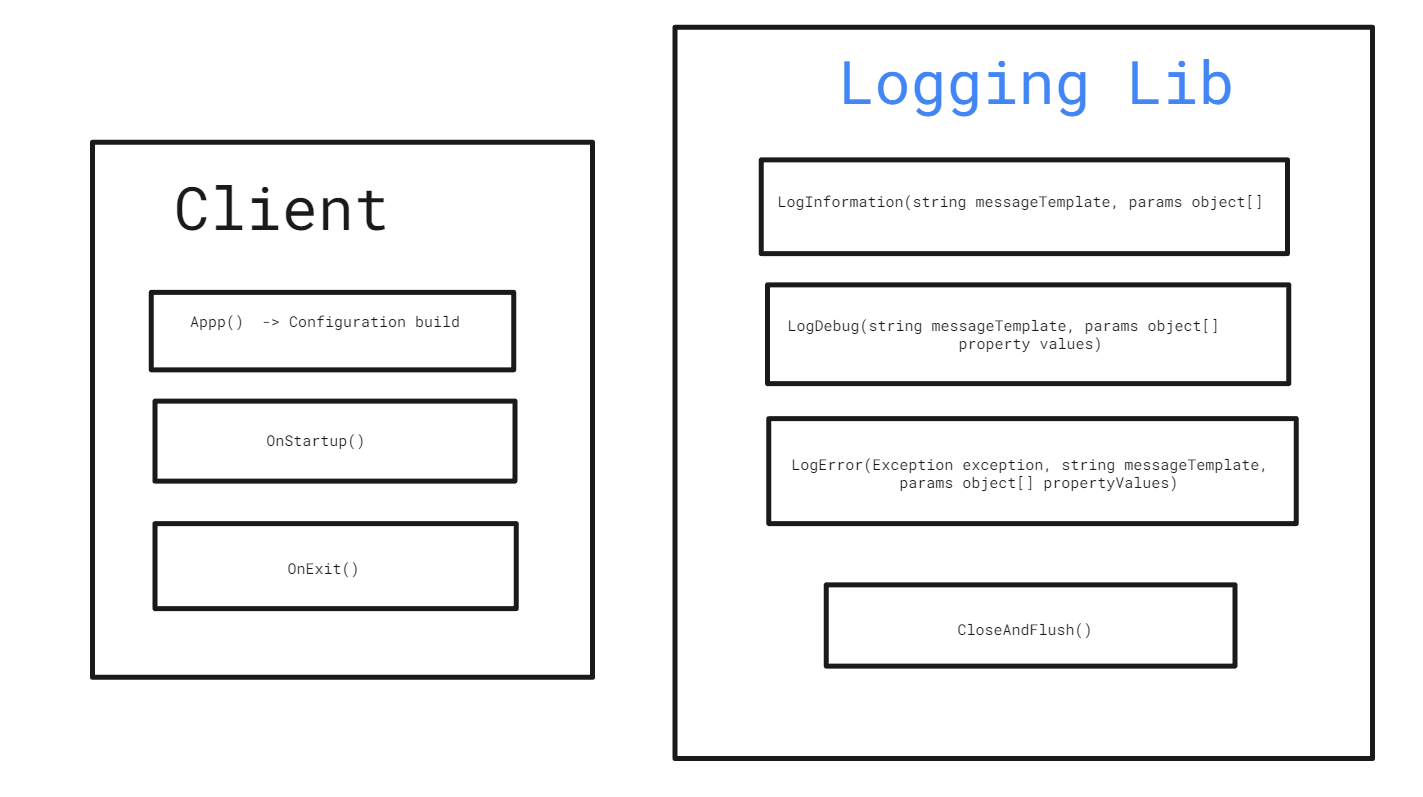
****

**-**

**Fig 2.2:** *Sequence Diagram of Seri Log Component*

**2.1.3 LOW LEVEL DESIGN**

This class diagram represents the attributes and operations/functions made by each class. Here, the client is the application requesting the logging from Seri Log component. Each call is made only to the SeriLog component and not to the original api. This allows the component to manage all the requests of the client as a middle man. It consists of the methods, data and procedures as follows.



**Fig 2.3:** *Class Diagram of SeriLog Component*

**CHAPTER 3**

**TECHNOLOGY & FRAMEWORK**

**3.1 Dot NET**

.NET CORE 6.0

* .NET 6 is the fastest full stack web framework, which lowers compute costs if you're running in the cloud.
* Hot Reload is one of the most striking new features added in .NET 6. You can take advantage of this feature to modify the user interface when your ASP.NET Core 6 application is in execution.
* A suite of developer tools to help you develop, build, test, and update your code

**3.3 TOOLS USED**

**3.3.1 VISUAL STUDIO COMMUNITY EDITION 2022**

Visual Studio is an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) from [Microsoft](https://en.wikipedia.org/wiki/Microsoft). It is used to develop [computer programs](https://en.wikipedia.org/wiki/Computer_program) including [websites](https://en.wikipedia.org/wiki/Web_site), [web apps](https://en.wikipedia.org/wiki/Web_app), [web services](https://en.wikipedia.org/wiki/Web_service) and [mobile apps](https://en.wikipedia.org/wiki/Mobile_app). Visual Studio uses Microsoft software development platforms such as [Windows API](https://en.wikipedia.org/wiki/Windows_API), [Windows Forms](https://en.wikipedia.org/wiki/Windows_Forms), [Windows Presentation Foundation](https://en.wikipedia.org/wiki/Windows_Presentation_Foundation), [Windows Store](https://en.wikipedia.org/wiki/Windows_Store) and [Microsoft Silverlight](https://en.wikipedia.org/wiki/Microsoft_Silverlight). It can produce both [native code](https://en.wikipedia.org/wiki/Machine_code) and [managed code](https://en.wikipedia.org/wiki/Managed_code).

* + 1. **MICROSOFT SQL SERVER**

Microsoft SQL Server is a proprietary [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system) developed by [Microsoft](https://en.wikipedia.org/wiki/Microsoft). As a [database server](https://en.wikipedia.org/wiki/Database_server), it is a [software product](https://en.wikipedia.org/wiki/Software_product) with the primary function of storing and retrieving data as requested by other [software applications](https://en.wikipedia.org/wiki/Software_application)—which may run either on the same computer or on another computer across a network (including the Internet). Microsoft markets at least a dozen different editions of Microsoft SQL Server, aimed at different audiences and for workloads ranging from small single-machine applications to large Internet-facing applications with many [concurrent users](https://en.wikipedia.org/wiki/Concurrent_user).

**CHAPTER 4**

**SOLUTION APPROACH**

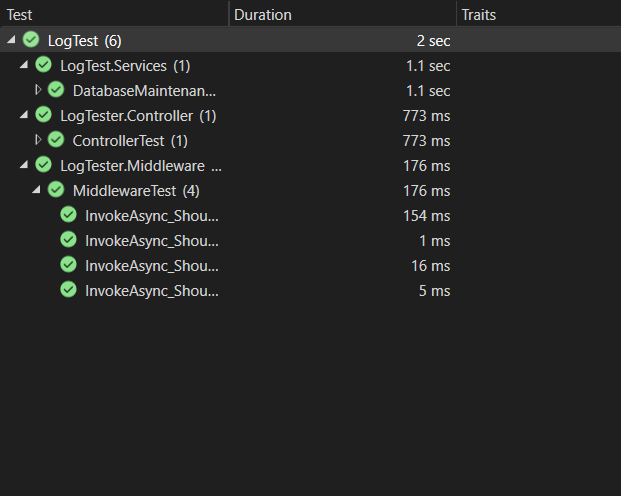
**4.1 APPROACH DESCRIPTION**

This section provides a detailed view of how the components are partitioned and assigned to the functionalities.

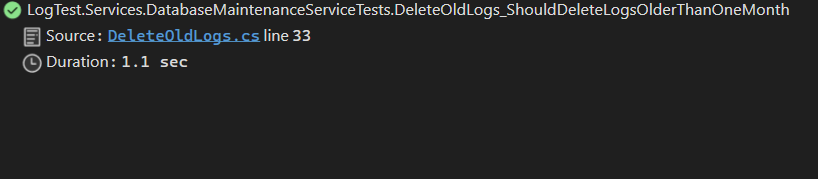
1. Component design diagram briefs about the connectivity of the component.
2. For component creation, we use the .NET CORE 6.0 framework.
3. The component can be called from a client with a request for the Seri Log API.
4. This component further calls the service which is Seri Log API. Seri Log API processes the request and provides the required output to the component.
5. The component receives the result from Seri Log and sends it to the requesting client application.
6. Meanwhile, the result is sent it transforms the same result in a required format including all the details, with time, system details and current variations present.
7. The transformed result is then allocated with a category from the component.
8. The result is stored in the allocated category in the logging files which can be retrieved anytime by the client application for debugging or if issues or problems arise.
9. The test scenarios and expected behaviour of the component are mentioned below.

**TEST CASES AND RESULTS**

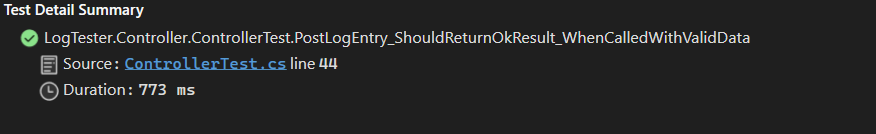
Test cases are a crucial component of software testing. They are specific scenarios or conditions that are designed to verify the correctness, completeness, and quality of a software application or system. Test cases are created based on requirements, specifications, and expected behavior of the software.

****

**Services Testing:**

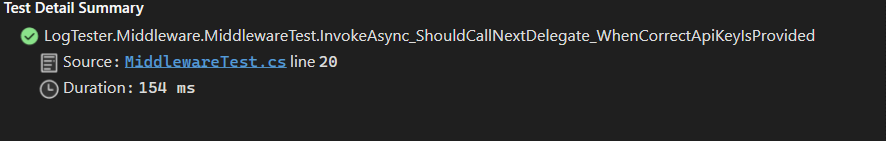
****

**Controller Testing:**

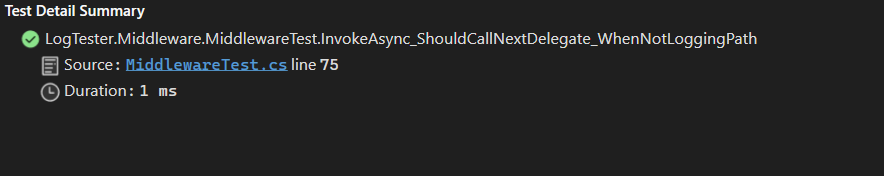
****

**Middleware Testing:**

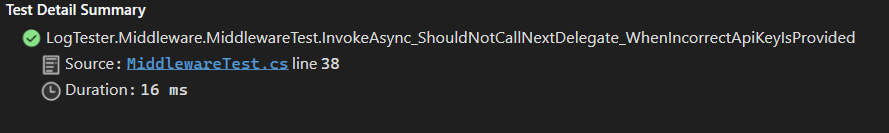
When API-Key is Correct:



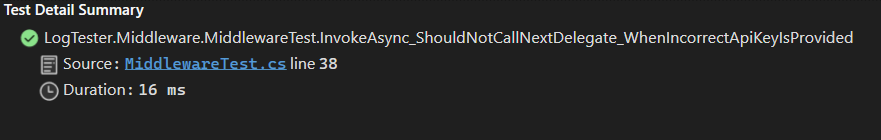
When Logging Path is Incorrect:



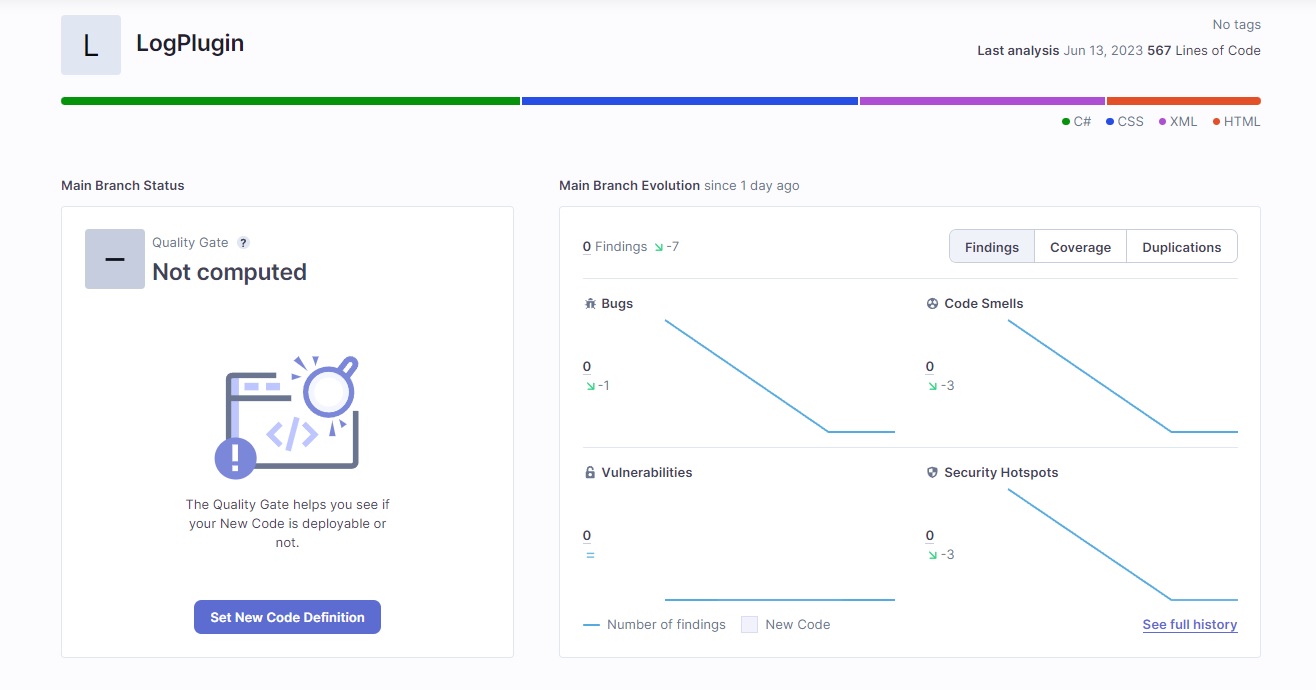
When API-Key is Incorrect:



When API-Key is Not Provided:

****

**SONARQUBE REPORT**



In this project, we have used SonarQube which is an open-source platform for the inspection of the code quality. The project is programmed with zero bugs and nil vulnerabilities.



This graph offers a visual representation of code quality metrics and trends, allows for the easy interpret and analyse the data. The duplication rate is zero and the code quality is adequate.