California State University Long Beach

College of Engineering

Computer Engineering Computer Science Department

Tuesday, November 26th, 2019

Sixfold Development Low Level Design Document

Group Members:

Michell Kuang (Team Lead)	**013421094**
Joshua McDaniel	**014542786**
Jingyan Du	**014436615**
Peter Park	**002948398**
Jacen Tan	**012393782**
Daniel Gione	**016513144**

Version History

Version	Date	Author(s)	Comments
V1.0	11/23/2019	All members	-Baseline document
V1.0.1	11/23/2019	Joshua McDanile	-Logging Creation and Writing

1. Introduction

This document serves to outline the low level designs of each individual feature provided by the RoomAid web application. Each diagram will depict the flow of it's specific feature in order to explain the purpose and logic behind each component for developers to understand and implement.

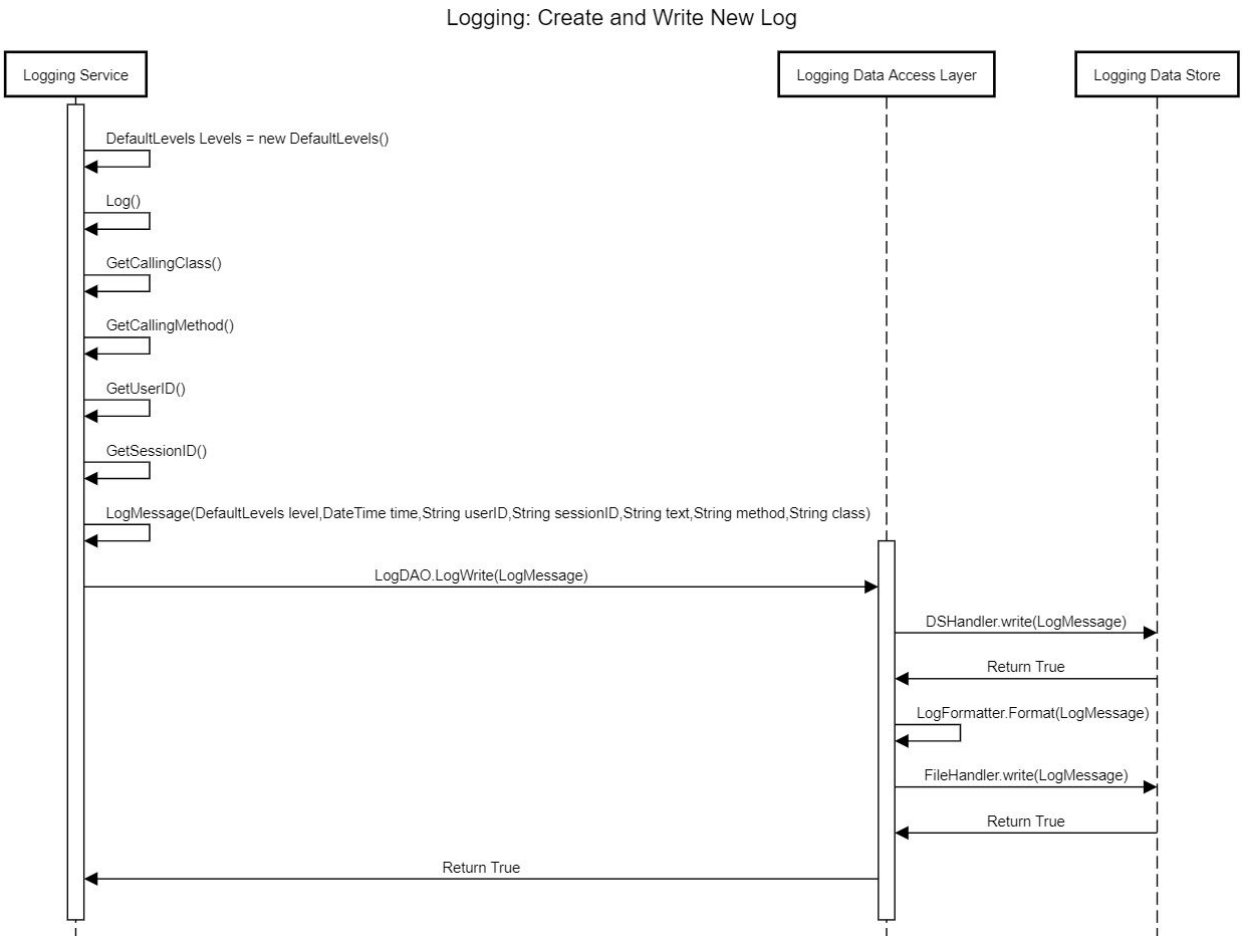
2. Logging

//TODO Consider better wording

Logging will occur throughout every layer of the application and will successfully create logs based upon the function that called it and the user who initialized said function. Logs will be stored within a datastore (MongoDB) and a flat file (.CSV) according to the dates that they are recorded.

2.1 Log Create and Write

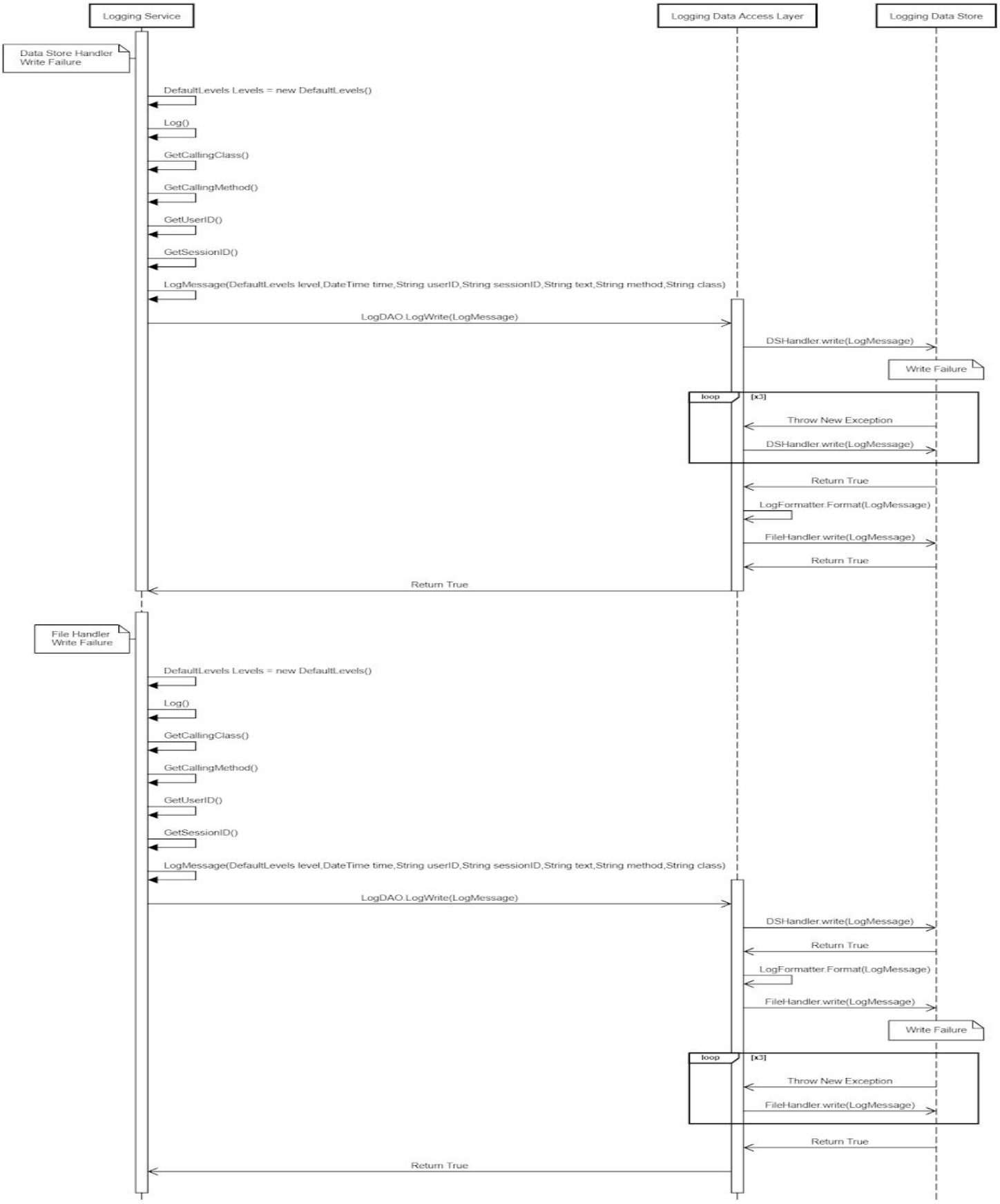
When a logging function is called, a log will be generated using values that are gathered from the calling function. These include the time of calling, the class of the calling function, the name of the calling function, the user identification of the user that initialized the function, the session identification of said user, the plaintext description of the log, and the level of the log itself(i.e. None, Debug, Info, Error, Warning, Fatal). Upon generation of each of the above criteria, the information is passed into LogMessage() to create an instance of that log. Using that instance, the handlers, DSHandler(...) and FileHandler(...), will be able to write the appropriate data to their specified locations. Since the LogMessage object is passed into each handler, the FileHandler(...) will require the data to be formatted into a string before writing.



2.1.1 Log Create and Write: Failure to Write: Successful Rewrite

Provided that the writing to either location fails, the system will throw an exception identifying the failure and will attempt to rewrite the log a total of three times. If any attempted rewrite is successful, the function will return and the process will continue as expected.

Logging: Create and Write New Log: Write Failures



2.1.1 Log Create and Write: Failure to Write: Failure to Rewrite

If the system fails to write to either location, and the rewrite process also fails, the system will automatically notify the system administrator, Since the writing process is asynchronous, it is possible for a write to be successful prior to the other failing. If this occurs, the system must delete the successful log from the other location to ensure that both locations have the same data.

//TODO: Michell’s deletion diagram