Document Overview

This document contains the official error logging standard for the Art Generator Project for CSCI 2340: Software Engineering, Spring 2024. This document will be broken up into two sections, one section for overall details about error logging levels and another that contains programming language specific guidance on how to log error messages. It is up to each respective team to determine their standard for logging messages.

General Error Logging Guidance

There are four levels of error logging that can be used in designing software for the Art Generator Project. This section contains a table outlining the various levels. The following section, "Programming Language Specific Guidance" will highlight how to use these logging levels while writing software.

Logging Level	Description
ERROR	The "ERROR" logging level is meant to be used for any critical errors that would cause the software to not operate correctly. This includes bad return codes from system calls, out of index accesses, improper allocation of memory, etc.
WARNING	The "WARNING" logging level is meant to track any errors that may appear that are not critical to software functionality. For example, adding an unrecognized field to a JSON file may not cause the system to break, but it is good to track the fact that this unexpected field has been added.
DEBUG	The "DEBUG" logging level is meant to track any calculations, variable assignment, class member creation, etc, that when enabled, can help the programmer see more clearly into how the software is operating.
TRACE	The "TRACE" logging level is meant to track the operation flow of the software. Essentially, the "TRACE" level should be used when entering and exiting function calls, to enable the developer to track how the software is interacting with itself.

Table 1: Error Logging Levels for Art Generator Project

Regardless of programming language, by default both ERROR and WARNING should be enabled. The DEBUG and TRACE options are designed to only be used occasionally when tracking down defects within the system. The latter logging levels tend to fill the log files more quickly, so it is important to not keep them on permanently. Additionally, all error logging should be done through *.log file types when possible. There should be frameworks created for each programming language to handle writing data to a file.

Programming Language Specific Guidance

This section contains the programming language specific guidance on how to log error messages.

Python

To log data using Python, there should be one global class member of the CMN_Logging class that should be used to write data to a file each time the software runs. This class member should be defined in the driver file, and should be accessed by all necessary files. This class is derived so that all data will be

written to one *.log file relative to the timestamp in which the back-end driver was started. Additional information that will be logged as a result of using the class is the date, time, and logging level associated with a message. The *.log file containing the output can be found in <root>/code/Backend/Common/logs.

There should be a standardized text that is used for each logging statement regardless of what level is being logged:

log(LOG_LEVEL_ENUM, “ClassName.MethodName() <LOG MESSAGE>”)

NOTE: If the function being used is not a member of a class, then simply the method name field is adequate.

Below includes a list of what should be logged:

- Use “TRACE” for the beginning and ending of each function call
- Use “ERROR” to check return codes of any API call being used
- Use “DEBUG” to track any creation of new class members or objects