Develop a basic logging library that can be used by applications to log messages. The library should handle message logging efficiently and reliably, offering basic configuration options.

Key Requirements:

- Driver Application should be able to Initialize the Library and log messages to the desired sink.
- **Logger has the following capabilities-**
    - Accepts messages from client(s)
    - A logger would have one or more sinks associated with it.
    - Supports defined message levels.
    - enriches message with current timestamp while directing message to a sink
    - Logger is initialized with a configuration eg:logger name, sink(s), buffer size.
    - Logger should support **both sync and async** logging.
    - For Async logger buffer size would determine the maximum inflight messages.
    - Messages must be ordered. Messages should reach the sink in the order they were sent.
    - Should **support writes from multiple-threads**.
    - There shouldn't be any data loss.
- **Sink**:
    - There can be various types of sink (file, stdout, database).
    - Sink has a destination.
    - For this round you may create STDOUT sink, which would print the message to the console.
    - Sink has an associated log level. **Any message with the level lower than the sink level should be discarded.**
- **Message**
    - has **content** which is of type string
    - has a **level** associated with it
- **Log Level**
    - DEBUG, INFO, WARN, ERROR, FATAL ; in order of priority. ERROR has higher priority than INFO.
- Add test cases to demonstrate sync logging, async logging and concurrent logging requests

Sending messages

- Sink need **not** be mentioned while sending a message to the logger library.
- You specify message content and level while sending a message

Logger configuration (see sample below)

- Specifies all the details required to use the logger library.
- Library can accept one or more configuration for an application
- Example:
- time format
- logging level
- sink type

- Logger type sync/async
- details required for sink (eg file location)); this depends on sink type.

Sample Config:

Ts_format: any format

log_level:INFO

logger_type:ASYNC

buffer_size:25

sink_type:STDOUT

Sample Output Log Entry

03-01-2024-09-30-00 [INFO] This is a sample log message.

Sample test case:

Input:

Configuration of the logger is console logging with Info level.

log.info(“Info message”)

log.warn(“Warn message”)

log.debug(“Debug message”)

log.error(“Error message”)

Output:

03-01-2024-09-30-00 [INFO] Info message

03-01-2024-09-30-01 [WARN] Warn message

03-01-2024-09-30-02 [ERROR] Error message

Expectations and Guidelines

1. Create the sample data yourself. You can put it into a file, test case or main driver program itself.
2. The code should be demo-able. Either by using the main driver program or test cases.
3. The code should be modular. The code should have the basic OO design. Please do not jam in the responsibilities of one class into another.

4. The code should be extensible. Wherever applicable, use interfaces and contracts between different methods. It should be easy to add/remove functionality without rewriting the entire codebase.
5. The code should handle edge cases properly and fail gracefully.
6. The code should be legible, readable and DRY.
7. Database integration is not required.
8. Please do not access the internet for anything EXCEPT syntax.
9. You are free to use the language and IDE of your choice.
10. The entire code should be your own.