SpecMetrix Logging

**Comprehensive Design and Development Plan**

The objective is to design and implement a logging service that will leverage **Serilog** as the core logging solution, integrating it with **MongoDB** for storage and implementing **error deduplication** using custom Serilog sinks or middleware.

The goal is to refactor the existing monolith program to consume this logging system across all critical areas.event logging (e.g., OnClick, errors, and warnings), deduplicates errors, and provides a Blazor-based UI for viewing recent logs. The service will also feature a mechanism for removing logs after 7 days and deduplicating errors if they don’t reoccur within 5 seconds.

Incorporating an additional 2-3 weeks of development effort to refactor the existing monolith program (with ~400 classes and 2 million lines of code) to implement and consume the new logging system, the project will require adjustments to the timeline and scope. \*\*NOTE: This is to refactor the existing system, errors and messages. This is not to add any additional logging or functionality. This will be performed under a different architecture.

The following details outline the expanded development effort.

**1. Functional Requirements**

1. **Event Logging:**
   * Log actions and events like HMI or Integration actions, warnings, and errors using Serilog.
   * Use Serilog to write logs to MongoDB.
   * Logs older than 7 days should be automatically deleted using MongoDB's TTL indexes.
2. **Error Deduplication:**
   * Deduplicate errors to avoid logging the same error repeatedly within a 5-second window.
   * If an error doesn’t reoccur within 5 seconds, it should be cleared from the cache.
3. **User Interface:**
   * Blazor-based UI that displays the last 25 events, refreshing every 5–10 seconds.
   * Support pagination for older log entries.
4. **Monolith Refactor:**
   * Refactor the existing monolith (~400 classes, 2 million lines of code) to consume the new logging system, ensuring all critical areas are covered.

**2. Non-Functional Requirements**

* The logging system must handle high-volume log events without impacting application performance.
* Thread safety is essential to avoid concurrency issues during log writes.
* The Blazor UI must remain responsive under real-time updates.

**3. Revised High-Level System Architecture**

**Components:**

1. **Serilog Logging Service:**
   * Serilog handles all event logging, error deduplication, and MongoDB writes.
   * Custom deduplication logic will be implemented using a middleware or custom sink.
2. **MongoDB:**
   * Logs are stored in MongoDB with a TTL index to remove logs older than 7 days.
   * Serilog's MongoDB sink will be used to simplify the process.
3. **Blazor UI:**
   * A real-time log viewer that refreshes every 5–10 seconds and supports pagination.
4. **Monolith Refactor:**
   * The monolith will be refactored to use Serilog for all logging operations across the codebase.

**4. Revised Sprint Plan with Serilog**

**Sprint 1: Core Logging Service Implementation with Serilog**

**Duration: 1 week**

1. **Tasks:**
   * Set up Serilog in the project.
   * Configure Serilog to write logs to MongoDB using the Serilog.Sinks.MongoDB package.
   * Implement structured logging for events, warnings, and errors using Serilog's key-value pair logging.
   * Ensure Serilog handles asynchronous logging to avoid performance bottlenecks.
2. **Deliverables:**
   * Working Serilog-based logging service that writes logs to MongoDB.
   * Basic test cases for logging functionality.

**Sprint 2: MongoDB Integration**

**Duration: 1 week**

1. **Tasks:**
   * Configure MongoDB TTL Index for the log collection, ensuring logs older than 7 days are automatically deleted.
   * Optimize MongoDB collection schema for structured logging.
   * Perform integration tests to verify logs are correctly stored and retrieved from MongoDB.
2. **Deliverables:**
   * MongoDB TTL Index working to remove logs older than 7 days.
   * Logs being correctly stored and queried from MongoDB.

**Sprint 3: Error Deduplication with Serilog**

**Duration: 1.5 weeks**

1. **Tasks:**
   * Develop custom middleware or sink for Serilog to implement error deduplication.
   * Maintain an in-memory cache that tracks error occurrences and removes them after 5 seconds if they do not reoccur.
   * Integrate the deduplication logic with the MongoDB sink to ensure deduplicated logs are written to the database.
2. **Deliverables:**
   * A working error deduplication system with a 5-second expiration.
   * Unit and stress tests to validate performance under high log volumes.

**Sprint 4: Blazor UI for Real-Time Log Viewing**

**Duration: 1.5 weeks**

1. **Tasks:**
   * Develop a Blazor component to display the last 25 log entries.
   * Implement real-time log updates, refreshing the UI every 5–10 seconds.
   * Add pagination for viewing older logs.
   * Ensure the Blazor UI remains responsive and performs well under frequent updates.
2. **Deliverables:**
   * Blazor-based real-time log viewer with pagination.
   * Integration tests for the UI and backend logging service.

**Sprint 5: Monolith Refactor (Phase 1)**

**Duration: 2 weeks**

1. **Tasks:**
   * Begin refactoring critical areas of the monolith to integrate new Logging service.
   * Identify and prioritize key classes and workflows for logging (e.g., major workflows, critical paths).
   * Replace existing logging mechanisms (if any) with new Logging service across these critical areas.
2. **Deliverables:**
   * Critical areas of the monolith refactored to use new Logging service.
   * Initial testing to ensure logs are captured and performance remains stable.

**Sprint 6: Monolith Refactor (Phase 2)**

**Duration: 1.5 weeks**

1. **Tasks:**
   * Continue refactoring the rest of the monolith to consume new Logging service.
   * Ensure that all areas of the monolith, including event handlers and error/warning paths, are logging to MongoDB.
   * Conduct integration testing to ensure smooth operation across the entire application.
2. **Deliverables:**
   * The entire monolith refactored and integrated with the Serilog logging system.
   * Logs from all parts of the system flowing into MongoDB as expected.

**Sprint 7: Performance Tuning, Testing, and Documentation**

**Duration: 2 weeks**

1. **Tasks:**
   * Perform full-scale performance testing to validate that Serilog and MongoDB handle the expected volume of logs.
   * Identify any bottlenecks or issues related to log writes or UI updates and optimize the system accordingly.
   * Document the logging service, the integration with the monolith, and guidelines for future log extensions.
2. **Deliverables:**
   * Fully optimized logging system with performance reports.
   * Comprehensive documentation and final testing results.

**5. Development Timeline**

| **Sprint** | **Duration** | **Tasks** | **Deliverables** |
| --- | --- | --- | --- |
| Sprint 1 | 1 week | Core logging service implementation with Serilog | Working Serilog service with MongoDB integration |
| Sprint 2 | 1 week | MongoDB integration, TTL Index setup | MongoDB logging with 7-day expiry |
| Sprint 3 | 1.5 weeks | Error deduplication with custom Serilog middleware | Deduplicated error logs with expiration logic |
| Sprint 4 | 1.5 weeks | Blazor UI for real-time log viewing | Responsive Blazor log viewer with pagination |
| Sprint 5 | 2 weeks | Refactor monolith (Phase 1) | Critical areas of monolith logging to new Logging service |
| Sprint 6 | 1.5 weeks | Refactor monolith (Phase 2) | Full monolith integration with new Logging service |
| Sprint 7 | 2 weeks | Performance tuning, testing, and documentation | Optimized logging system and final testing |

**6. Final Deliverables**

**At the end of this development effort, the following will be delivered:**

* A Serilog-based logging system that handles high log volumes, provides error deduplication, and writes logs to MongoDB.
* A Blazor-based real-time UI that displays and paginates through logs.
* A fully refactored monolith, consuming the logging system across all its critical areas.
* Optimized performance and full documentation for the logging system.

**Final Deliverable**

At the conclusion of this development effort, the entire monolith system will be refactored to consume the new logging service, allowing for efficient and deduplicated logging of events, warnings, and errors. The system will have a real-time UI for monitoring logs and will be optimized to handle high log volumes while preserving application performance.