Assignment: Build a Simple Metrics Collection and Monitoring System

Objective:

Create a metrics collection and monitoring system in Go that captures system metrics, stores them, and provides a way to query and visualize the metrics.

Requirements:

1. **Metrics Collection:**
    - Implement a metrics collector that captures basic system metrics such as CPU usage, memory usage, disk I/O, and network I/O.
    - The collector should run at a configurable interval (e.g., every 10 seconds).
2. **Data Storage:**
    - Store the collected metrics in a time-series database (e.g., Prometheus, InfluxDB) or a simple file-based solution if a full-fledged database is not feasible.
    - Ensure the storage solution can handle large volumes of data efficiently.
3. **API Endpoint:**
    - Provide a RESTful API to query the stored metrics.
    - Implement endpoints to:
        - Retrieve metrics for a specific time range.
        - Aggregate metrics (e.g., average CPU usage over the last hour).
4. **Alerting Mechanism:**
    - Implement a simple alerting mechanism that triggers alerts based on predefined thresholds (e.g., CPU usage > 80% for 5 minutes).
    - Alerts can be sent via email or logged to a file.
5. **Visualization:**
    - Integrate a basic web interface or use Grafana to visualize the metrics.
    - The dashboard should display real-time data and historical trends for the collected metrics.
6. **Documentation:**
    - Provide clear documentation on how to set up, run, and test the system.
    - Include comments in the code to explain the implementation and design decisions.

Evaluation Criteria:

1. **Code Quality:**
    - Readability, maintainability, and adherence to Go language best practices.
    - Use of idiomatic Go patterns and structures.
2. **Scalability and Performance:**
    - Efficient data collection and storage mechanisms.
    - Ability to handle high-frequency data collection without significant performance degradation.

3. **Reliability and Fault Tolerance:**
    - Robustness of the system under failure conditions (e.g., handling of database outages, network issues).
    - Implementation of error handling and recovery mechanisms.
4. **Completeness:**
    - Fulfillment of all assignment requirements.
    - Additional features or improvements beyond the basic requirements.
5. **Documentation and Testing:**
    - Quality of documentation and ease of setup.
    - Implementation of unit tests and integration tests.

Submission Guidelines:

- Submit the code as a Git repository (e.g., GitHub, GitLab).
- Include a README file with setup instructions, usage examples, and explanations of design choices.
- Provide any necessary configuration files or scripts for setting up the environment.

This assignment will help assess the candidate's ability to design and implement a reliable, scalable system while demonstrating their expertise in Go and SRE principles.