SOFTWARE DESIGN DOCUMENT

for

Data Storage System for Monitoring System

Release 1.0

Version 1.0 unapproved

Prepared by Baxter Halder, Claire Hall, Kayla Jamar, Jennifer Olszyna

Contents

List of Figures 3

List of Tables 4

1 INTRODUCTION 6

1.1 Purpose 6
1.2 Scope 6
1.3 Overview 7
1.4 References 7
1.5 Definitions and Acronyms 8

2 SYSTEM OVERVIEW 9

3 SYSTEM ARCHITECTURE 12

3.1 Architectural Design 12
3.2 Decomposition Description 12
3.3 Design Rationale 13

4 DATA DESIGN 14

4.1 Data Description 14
4.2 Data Dictionary 14

5 COMPONENT DESIGN 15

6 HUMAN INTERFACE DESIGN 16

6.1 Overview of User Interface 16
6.2 Screen Designs 16

7 REQUIREMENTS MATRIX 17

List of Figures

2.1 Database Design Depicted as an E/R Diagram 9
3.1 Database Design Depicted as an E/R Diagram 13

List of Tables

1.1 References 7
1.2 Acronyms and Abbreviations 8

7.1 Requirements Matrix 18

Revision History

Date	Description	Revised by
03/11/22	Initial draft	Data Store
03/16/22	Notes	Data Store
03/20/22	Additions	Data Store

1 INTRODUCTION

1.1 Purpose

This software design document describes the architecture and system design of CS455 Monitor System. This document will illustrate the purpose and complete declaration for the development of this system as well as explain the system's constraints, interface, and interactions with other external applications. This document is primarily intended to be proposed to a customer for his approval and act as a reference for developing the first version of the system for the development team.

1.2 Scope

The CS455 Monitor System software will monitor both the activity and health of all machines on a network. The system itself will be a combination of four sub-systems; Agents, a Monitoring Engine, a Data Store, and a Dashboard. The Engine and Data Store are hosted on the CS server. However, the Engine and Data Store will be portable to other server machines that run on a Unix-like OS. The Agents, installed on devices on the network, can run on both Windows and Unix-like systems. The Dashboard will display the status of the monitored machines. The Dashboard can be used on Windows or Ubuntu machines. Users will be able to view specific information such as CPU usage, memory usage, disk usage, and network usage of the monitored machines on the network from the Dashboard. This information passes through the Monitoring Engine to be validated, such as checking valid ranges for data, before being stored in the Data Store for the Dashboard to retrieve. Users will be able to view the specific machines the Agents are monitoring that are connected to the network in order to track down the cause of issues. Examples of these issues include a monitored machine being disconnected or unresponsive, a monitored machine's disk is close to or over capacity, a monitored machine's CPU usage is maxed out, which services should be running on the monitored machine that are not currently running, the network usage is less than the user expected for a monitored machine, or the monitored machine's memory usage is close to capacity.

1.3 Overview

The database acts as a place to store collected metrics and logs. The database will be scalable to accommodate up to hundreds of active servers and a dozen active dashboards.The database system will be hosted on the CS server. The data storage will take data from the monitoring system and sort it into one of two fundamental types of data, metrics or logs. Metrics are numerical values that describe some aspect of a system at a particular point in time. They are lightweight and capable of supporting near real-time scenarios. Logs contain different kinds of data organized into records with different sets of properties for each type.

1.4 References

Standard	**Reference**
SEI CERT	SEI CERT Secure Coding Practices
Microsoft SDL	Microsoft SDL Secure Development Practices
SQL and PL/SQL Coding Standards	PL/SQL best practice Standards tips
Standard SQL Naming Conventions	Oracle naming standards tips
JSON	RFC 8259

Table 1.1: References

1.5 Definitions and Acronyms

Acronym or Abbreviation	**Definition**
KPI	Key Performance Indicators are quantifiable measure of performance over time for a specific objective that provides the admin insights into the network.
QA	Quality assurance is a system for evaluating performance.
GUI	Graphical User Interface
SRS	Software Requirements Specifications
UNA	University of North Alabama
CS455	The computer science class at the University of North Alabama in which this project is being made

Table 1.2: Acronyms and Abbreviations

2 SYSTEM OVERVIEW

The CS455 Monitor System is a network monitoring system that will monitor UNA's CS network in order to help users identify and address potential network problems as they occur. The system will automatically collect and store data, as well as provide warnings and error messages when certain events trigger, such as an unresponsive agent or disk is at capacity. However, the system will rely on users to see the messages and act on them; the system is **not** automated to handle the events. (As a specific example, if an error message pertaining to an unresponsive agent appears, it is the user's job to determine why the connection has failed and handle the issue.) The Monitor System contains four sub-systems working in tandem as shown in Figure 2.1; Agents, Monitoring Engine, Data Store, and a Dashboard.

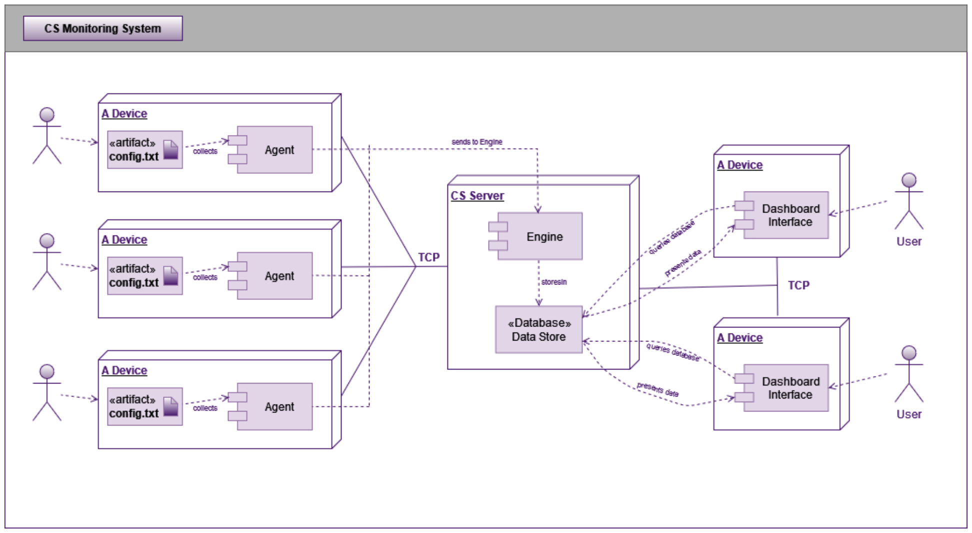

Figure 2.1: Database Design Depicted as an E/R Diagram

The monitoring system is relatively simple conceptually. Agents, installed on devices connected to the network, will collect basic usage statistics such as CPU usage, memory usage, disk usage, network usage, and service applications that are currently running on the device. The frequency at which to collect data will be specified in the configuration text file (though in cases where a time is not specified there is a default time). The configuration file will also associate each agent with a unique ID and name corresponding to the device it is installed on in order to keep track of the data associated as it transfers through the system as a whole. This configuration file also contains the host name of the Engine should the user decide to port the Engine to another device. Each agent will automatically send its collected data to the Monitoring Engine over the network to be stored in the Data Store.

The Monitoring Engine (or simply the Engine), hosted on the CS server, will take in data from the Agents, perform sanity checks on the data, and then pass it on to the Data Store for storage. A ”sanity check” is ensuring that incoming data is in a logical range. For instance, it is impossible for CPU usage to be above 100%. It is impossible for the disk usage to be above 120%. It is impossible for memory usage to be a negative percent. Should the Engine receive invalid data such as this, it will log an error to the Data Store. The Engine also contains a configuration file that details the host name of the Data Store that can be changed should the user want to port the Data Store to another device. This configuration file also knows the frequency it should be getting updates from each Agent. If the Engine receives no update from an Agent after that configured time frame, it logs an error message to the Data Store. The user will never interact directly with the Engine.

The Data Store, also hosted on the CS server, will take data from the Monitoring Engine and sort it into one of two fundamental types of data, metrics or logs. Metrics are numerical values that describe some aspect of a system at a particular point in time. Logs contain different kinds of data organized into records with different sets of properties for each type. The user will never directly interact with the Data Store. The Data Store acts solely as a place to store collected metrics and logs. The Data Store will be scalable to store information of up to fifty active Agents, process queries from up to a dozen active Dashboards, and be able retain data for up to a year. Users are able to remove or archive specific data in the Data Store.

The Dashboard will provide easy access to viewing the metrics of multiple different devices. Users will be able to view both current and past metrics for a device as well as see if there are connection issues with an Agent's monitored device.

3 SYSTEM ARCHITECTURE

3.1 Architectural Design

The Data Store, hosted on the CS server, will accept data sent from the Monitoring Engine and then store it for the user to access through the dashboard interface. The Date Store is composed of three majors components, a part to listen to the Engine, the database, and a part to listen to the Dashboard. As these components flow together and overlap heavily, they were not given individual identifying names.

3.2 Decomposition Description

The first component listens to the Engine and then receives either error messages or JSON files containing information from the Agents.

The second component is the database itself that will implemented as a MariaDB database hosted on the CS server. The database's schema is depicted in Figure 3.1 as an E/R diagram. It is assumed by the Data Store that the data is valid, so once the file is received the Data Store will parse out the information contained in the file and insert it into the *devices* entity or parse and insert the error message details into the *error_log* entity. A database entity is a thing, person, place, unit, object or any element data should be stored in the form of properties. The Data Store's database will be comprised of two entities. *devices*, and the *error_log*. Each entry in the *devices* entity represents a device with an Agent installed on it.

The last component is the part listening to the Dashboard for either queries or error messages. Error messages are processed the same way ones from the Engine are, and quires are transforms into SQL quires and processed. The Data Store then returns the requested data to the Dashboard.

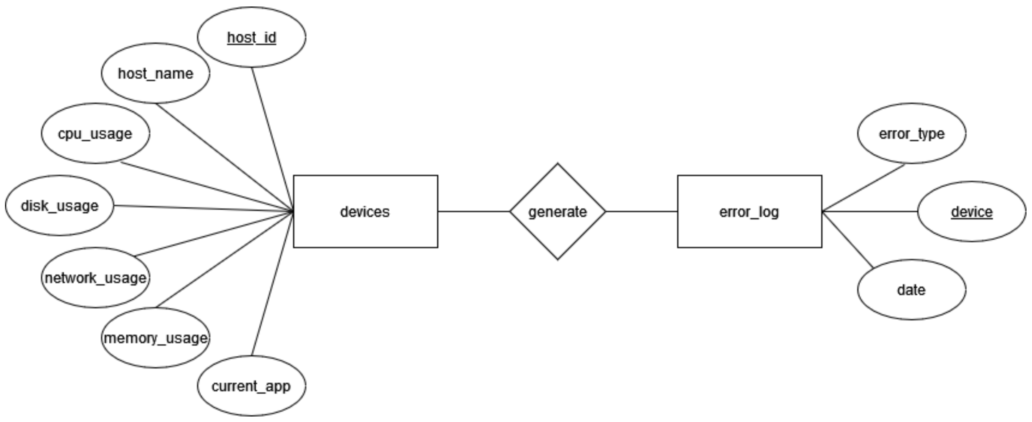

Figure 3.1: Database Design Depicted as an E/R Diagram

3.3 Design Rationale

MariaDB was chosen over SQL or MySQL due to the team's familiarly with the database type as well as the numerous JSON functions built into it.

4 DATA DESIGN

4.1 Data Description

The Data Store receives data from the Engine in the form of a JSON file. This information will be passed to a memory buffer where a function will parse out the information in the file. This information will be inserted into the *devices* entity and continuously updated as the Engine sends updated files. On the - figuratively speaking- other end the Data Store will receive queries from the Dashboard and then send back the results.

4.2 Data Dictionary

The Data Store will receive a JSON object in the form of a text file. The Data Store will transform the file into a SQL insert statement and insert it into the *devices* entity.

5 COMPONENT DESIGN

MariaDB SQL queries

Receive message and store messages

Create device table with entities of host ID, host name, CPU usage, disk usage, network usage, memory usage

JSON_EXTRACT() function used to parse data from the JSON document
Insert host ID
Insert host name
Insert CPU usage
Insert disk usage
Insert network usage
Insert memory usage

Create error log table with entities of error type, device, and date
Insert error type
Insert device
Insert date

Constraint used to validate request
Send data to dashboard

6 HUMAN INTERFACE DESIGN

6.1 Overview of User Interface

The Data Store creates database tables and inserts data into the tables upon receiving information from the Engine. If there are issues with receiving messages from the Engine, an error message will occur. The Data Store will access data that is stored in the database tables upon receiving a query from the Dashboard. If the query fails, an error message will occur.

6.2 Screen Designs

Nothing will appear or be visible to the user if the Data Store successfully runs. If issues occur and the Data Store does not successfully run, a diagnostic message will occur.

7 REQUIREMENTS MATRIX

The requirements matrix provides a cross-reference that traces each component and data structure to the requirements in the Software Requirements Specification document. See table 7.1.

ID	Requirement	System Component
SR01	Engine can create and send a network message to the Data Store	Database Tables Storing Received Messages
SR02	Data store validates requests from Dashboard	Database Table Constraints
SR06	Data Store sends data to the Dashboard	SQL Queries
SR11	Engine sends an error message to the Data Store when upon receiving invalid data	Error Log Table Storing Error messages
SR13	Users can view CPU usage	Device Table Storing CPU Usage, SQL Queries
SR14	Users can view memory usage	Device Table Storing Memory Usage, SQL Queries
SR15	Users can view disk usage	Device Table Storing Disk Usage, SQL Queries

SR16	Users can view network usage	Device Table Storing Memory Network, Queries
SR17	Users can view monitored services	Device Table Storing Monitored Service, SQL Queries
SR23	Data Store runs on Unix-like	CS Server (cs.csis.work)
SR26	Data Store is able to scale in size	Horizontal and Vertical Scaling
SR27	Data store can hold data for at least a year	Analyze Table, Check Table, Optimize Table, Repair Table, and Get Rows Count Queries
SR28	Data can be archived or removed by users	Archive Database Tables, Insert and Delete Queries
QR5	Document and follow a coding style	SEI CERT
QR7	System can be ported to other devices	Configuration File

Table 7.1: Requirements Matrix