SOFTWARE DEVELOPMENT MANAGEMENT PLAN

for

Interface for Software Infrastructure Monitor

Release 1.0

Version 1.0 approved

Prepared by Jake Whitehead, Jalen Goodloe, Kayla Walton, Aaron d'Ablaing

Contents

List of Figures

List of Tables

Revision History

Date	Description	Revised by
02/15/22	Initial draft	Interface Team
02/22/22	Revised draft	Interface Team
03/01/22	Revision of Document	Interface Team

Table 0.1: Revision History

1 Introduction

This is the Software Development Plan for the Interface Component of the System Monitoring Software. It defines the management goals for the component, implementation with other components and the deadlines that need to be made.

1.1 Purpose

Ensuring that the customers requirements are met. Defines the scope of the project and provides a reference to verify that all implicit and explicit requirements are met.

1.2 Acronyms

- GUI - Graphical User Interface.
- RD - Requirements Document.
- SDMP - Software Development Management Plan.
- PR - Pull Request.

1.3 References & Standards

- WIP

2 Project Overview

This software allows monitoring of different systems over a network. Provides different metrics such as RAM usage, Network usage, and CPU usage. The server monitor can be setup on a Windows or Linux system. The client can be setup on a Windows or Linux system.

2.1 Software Overview

Providing a GUI that allows easy access to multiple different computers and their metrics. Can view past metrics, single metrics, total system load and allows a user to see if there are connection issues or read issues with the server connection. The GUI can be used on Windows and Linux system.

2.2 Schedule

See Table 2.2 for schedule.

2.3 Budget

N/A

2.4 Project Deliverables

See Table 2.4 for deliverables.

Date	Activity	Artifact
02/16/22	Initial draft	Rough draft of SDMP and RD
02/23/22	Revision	Revision of SDMP and RD
03/09/22	Initial design review	Modified design
03/16/22	Implementation begins	Final design is completed and implementation begins
03/23/22	Planning and review	Test plan and prototype review
04/06/22	Testing and review	Test results and prototype review
04/13/22	Testing and review	Test results and prototype review
04/20/22	Testing, review and integration	Test results, prototype review and integration testing
04/27/22	Integration test, review and acceptance test	Integration and acceptance test results, prototype review
05/03/22	Software package delivery	Software, documentation is delivered during a presentation

Table 2.1: Project Schedule

ITEMS TO BE DELIVERED
Requirements Document
Software Development Management Plan and Traceability Matrix
Design Document
Test Plan Document and Traceability Matrix
Unit Test Results and Traceability Matrix
Integration Test Results and Traceability Matrix
Acceptance Test Results and Traceability Matrix
Team Meeting Minutes
Team Communication
Source code and all files necessary for compiling
Software version control repository
CM Management Data
User-Manual
Prototype
Final Program
PDF of presentation files

Table 2.2: Project Deliverables

3 Management Approach

3.1 Organization and Responsibilities

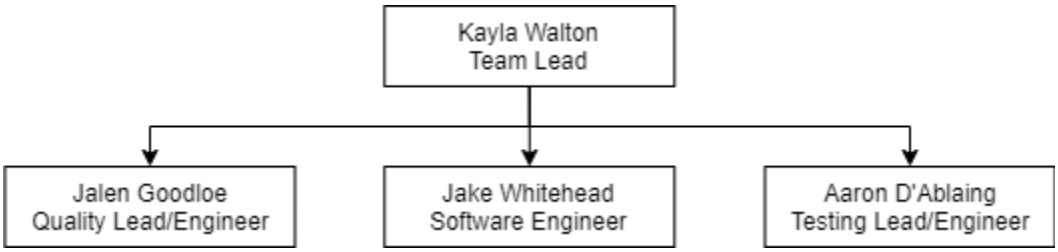

Figure 3.1: Organization Chart

3.1.1 Software Team Lead

- Creates and maintains the SRS
- Advises on coding practices
- Coaches team members on proper coding practice
- Reviews Code

3.1.2 Testing Lead

- Ensures that tests are reachable and in depth.
- Set test objectives.

- Set test policies.
- Estimate the testing to be done.
- Schedule tests for execution.

3.1.3 Quality Lead

- Supervise member(s) of the Quality Assurance Team.
- Manage and organize quality testing.

3.1.4 Software Engineer

Analyze and modify software.

3.1.5 Test Engineer

- Test strategies and test plans.

3.1.6 Software Quality Assurance Engineer

- Conduct quality testing.

3.2 Software Risk Management

Risk Assessment Table

		Severity of Harm (Impact)		
		Low (L)	Medium (M)	High (H)
Likelihood	High (H)	3	4	5
	Medium (M)	2	3	4
	Low (L)	1	2	3

Risk Assessment Table

Risks will be identified in Inception Phase. Project risk is evaluated at least once per iteration and documented. See Table 3.1 for a list of risks and mitigation strategies.

3.3 Customer Communications

Communication occurs with customer via emails and face to face meetings.

3.4 Team Training

3.4.1 Language Training

Software team needs training in the basics of:

- ReactJS
- Electron

Record of General Training:

- Training will be recorded in a training document stored in the software repository.
- Training document will be maintained by Software Lead.

See Table 3.4.1 for current General training needs and completion.

3.4.2 Secure Coding Practice Training

Software team will be required to train and adhere to the following secure code practices

- SEI Secure Coding

Record of Secure Coding Practice Training:

- Training will be recorded in a training document stored in the software repository.
- Training document will be maintained by Software Lead.

See table Table 3.4.2 for current secure coding practice training needs and completion.

Risk	**Rating**	**Mitigation Strategy and/or Contingency Plan**
Loss of power for extended period of time and unable to update software	3	Purchase a backup generator
Software fails for user	3	Meet with entire team and identify test case and solution
Server used to host software crashes	3	Contact server host and communicate issues to client
Developers unable to resolve errors before deliverable deadline	2	Contact client and reassess realistic timeline with team

Table 3.1: Software Risk Management

Name	ReactJS	Electron
Jalen Goodloe	Complete: 06/01/21	Complete: 06/01/21
Kayla Walton	Complete: 05/14/21	Incomplete: Training Needed
Jake Whitehead	Incomplete: Training needed	Incomplete: Training needed
Aaron d'Ablaing	Incomplete: Training needed	Incomplete: Training needed

Table 3.2: General Training Needs

Name	SEI Secure Coding
Jalen Goodloe	Incomplete: Training Needed
Kayla Walton	Incomplete: Training Needed
Jake Whitehead	Incomplete: Training Needed
Aaron d'Ablaing	Incomplete: Training Needed

Table 3.3: Secure Coding Training Needs

4 Technical Approach

4.1 Development Process

The software will be developed using an agile SCRUM approach and the coding review process will be the peer review. The Google JavaScript Style Guide coding standard will be used.

4.2 Development Tools

IDES USED

- Visual Studio Code v1.64 from Microsoft

TESTING TOOLS USED

- MochaJS v9.2.1 from MochaJS
- ESLint v8.10.0 from ESLint

VERSION CONTROL USED

- GIT v2.35.1 from Git

4.3 COTS & FOSS Tools

- ReactJS v17.0.2 from Meta (Facebook)
- ElectronJS v17.1.0 from GitHub
- C3.JS JavaScript library v0.7.20 from C3js
- Bulma CSS library v0.9.0 from Jeremy Thomas

4.4 Software Reuse

No software is being reused that was created by the team as a whole; however, libraries may be utilized as needed.

4.5 Testing Process

Testing will take place on both Windows and Linux based systems. Tests will consist of:

- Response time to data store
- Correctness of data received from data store
- GUI responsiveness
- CPU usage of the application
- Memory usage of the application

The Testing Lead will maintain and update a test document that is stored in the teams repository. This document will contain a traceability matrix; this allows tests and test cases to be tracked and documented. All testing done will be sent to the Testing Lead for review. Once reviewed the Testing lead will update the traceability matrix accordingly. MochaJS will be used as an automated system to ensure the code is producing the desired outcome, and ESLint will be used to ensure that the coding style guideline is adhered to.

5 Configuration Management

5.1 CM Responsibilities

Each member who wishes to make updates to the software will issue a pull request and assign testers. After initial testing and feedback, the member who initiated the pull request will make the required adjustments and followed up by another review. All pull requests will be approved by the Team Lead before merging into the main code base and the pull request will be documented in a code review report stored in CM.

5.2 CM Resources

- Azure DevOps v2020.0.1 from Microsoft
- Git v2.35.1 from Git

5.3 Change Control Procedures

The team will use Git to make commits and Azure DevOps to monitor changes.

6 Verification & Validation

6.1 V&V Procedures

All testing described will be completed and documented.

Each lead will sign-off on the product then the customer will sign-off on the product.

6.2 Independent V&V

N/A