Software Requirements Specification

for

An Aurora microservice tool

GEM: Geodata Event Monitor

Version 0.1 approved

Prepared by: Matthew Jett, Luna Lord, Forrest Wallace, and Jerett Latimer

Northlight Services, LLC

6/11/2020

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Luna Lord	02/23/20	Update Aurora to GEM, add new System Features	
Matthew Jett	3/8/20	Update Title again, removed Eastern Washington University in the title page to correct company name, Northlight Services, LLC.	
Matthew Jett	4/14/20	Cover change, bolding of terms for glossary, product features added.	
Jerett Latimer	6/7/2020	Added changes throughout to reflect current state. Removed all references to Raspberry Pi's, updated User Classes and Characteristics, Purpose, Scope, Interface, Perspective and more. Should be very close to final state for the project.	
Jerett Latimer	6/10/2020	Final revisions	

1. Introduction

1.1 Purpose

This SRS describes both the functional and nonfunctional requirements for the GEM: Geodata Event Monitor, referred to as GEM or the GEM tool from this point on. The features detailed in the document are aimed at release version 1.0. The SRS will be used by the development team as a roadmap for implementing features and functionality as requested by the client. Unless it is explicitly stated otherwise, all features listed are planned to be included in the 1.0 release of the GEM tool as one of the microservices of the Aurora toolkit by Northlight Services, LLC.

Note: For the purposes of the initial developers noted in the title page, these individuals are only developing a non-release, alpha or proof-of-concept of the client's concept and will never reach a 1.0 release. There will be no monetization or registration of intellectual property for this proof-of-concept during the times of 1/30/2020 to 6/8/2020. These dates coincide with the times the students are developing this proof-of-concept for their Computer Science Senior Project/Capstone while under Eastern Washington University's observation/responsibility and as a tuition paying student. After the date of 6/8/2020, the mentioned students will receive a grade and finish their enrollment of the Senior Project/Capstone classes, thereby Eastern Washington University should have no involvement, nor have any rights to the intellectual property in any future iterations of the GEM tool project that was demoed only as a proof-of-concept for the client to the Department of Computer Science at Eastern Washington University.

1.2 Document Conventions

- *Cisco Grid Routers will be abbreviated as CGRs.*
- *Network Operations Center will be abbreviated as NOC.*
- *Field Network Device will be abbreviated as FND.*
- *Minimum Viable Product will be abbreviated as MVP.*
- *Raspberry Pi Mock Routers will be abbreviated as Pi(s).*
- *Home Server will be referred to as the Mock NOC Server.*
- *Northlight Services, LLC will be referred to as the client.*
- *Professor Capaul will be referred to as the project manager.*
- ***Bold Text** indicates a reference to another section in the SRS.*

1.3 Intended Audience and Reading Suggestions

*This SRS is intended for the client, the development team, and our overall project manager. It is our suggestion, no matter what the reader's title is, to read the entirety of this SRS in sequential order. However, certain sections are more technical and aimed more at the developers and project managers, such as the detail sections of **System Features** and the **External Interface Requirements** section. These sections will contain information about the inner-workings of the software that the client may not necessarily need to know.*

1.4 Project Scope

*The initial objective of **GEM** is to implement a proof-of-concept for an event based notification monitoring system pertaining to the health status of Avista's **CGRs**. This notification system will look for changes in the continuously transmitted metadata of a CGR and notify users based on a specific subscription they have registered for to receive notifications dynamically. This will be executed in a proof-of-concept environment that will hope to demonstrate and prove the concept over actual real-world implementation. This will be done by mocking the functionality of the CGRs by altering data in a MongoDB database in the form of **Geodata** (GPS coordinates along with other metadata such as "status = online"). This will mimic a CGR's transmission of its metadata to a **NOC**. CGRs transmit to a cellular tower via a 600MHz microwave signal. The cellular towers then transmit to a NOC via SNMPv3 protocol. The mock data in MongoDB will be changed to replicate a real data change in a CGR. The GEM tool will fetch data from the MongoDB geodata collection instantaneously and notify the subscribed users. The GEM tool will enable the ability for a user to apply a set of rules based on each field in the geodata to monitor the respective field for changes that satisfy the set rule. A group of rules will be known as a **Task**. Once one of the rules of a Task is satisfied, the app will proceed to notify all subscribed users based on the selected user's **subscription group**. This will accommodate prompt action to rectify CGR failures by working around the manual diagnosis process currently employed by a **FND operator**. The goal is to implement the GEM tool generically such that it can be applied no matter what the user wants to track, be it Cisco Grid Routers (CGRs) or any type of point-of-interest location that contains geodata, also known as a **Site**. A collection of Sites will be known as a **Survey**. This way the functionality can be extended to notify users no matter what they are tracking. Making the GEM tool a diverse **GIS** microservice that will aid current NOCs and making the job of a FND operator more automated.*

1.5 References

Aurora suite repo can be found here: https://dev.azure.com/northlightservices/_git/Aurora

2. Overall Description

2.1 Product Perspective

The GEM tool is a proof-of-concept presented as a standalone application with dependencies on MongoDB for manual entry of mock geodata sites and for data storage and Microsoft Azure Cloud App Services to host the application ubiquitously. It will be built using the ASP.Net framework, Razor Pages, and MongoDB with an API layer in between. It will be tested using mock CGR Geodata that will be altered in the database to replicate changes in the real world. The Azure hosted GEM app will then fetch this geodata from the collection using MongoDB's Change Stream feature and attach a series of user selected events to the geodata's fields. A website that will serve as the frontend GUI for a user to interact with will allow them to create a Task (a grouping of field monitored events or known as rules) and listen for any changes in the selected data. If changes *occur an email will be sent to the inputted email a user entered or subscribed to. If a router goes down, it's field, "status" will switch from the value of "online" to "offline", firing off a sequence of events to build an email out and send the email to the user instantaneously. This will serve as an MVP for the client and the entirety of our senior project/capstone for the student developers at Eastern Washington University.*

2.2 Product Features

The GEM tool offers multiple features that aid businesses in alerting relevant parties on what they only need to know live, without delay. The tool's features can be broken down into three major feature points.

- ***Tasker** - This feature allows for a user on the frontend web client to select the fields of geodata they want to be notified on by creating a series of conditional values that will trigger an event. There are several terms used here.*
  - ***Field** - A single JSON key found in the geodata blob read from the database.*
  - ***Condition** - Either a logical operator or a value entered to serve as a query on a field's incoming JSON value read from the database.*
  - ***Rule** - A row that represents a field's set of conditions. A rule will be tied to an event that when all conditions are true, it will send a notification to the selected notification group. A rule will be formatted in this way:* `If {field}[checkbox] value {dropdown and radio: {is}, {isnot}, {isgreaterthan}, {islessthan}} {textbox: expected value}, then notify all in {dropdown and checkbox: {all subscription groups...}}`*. A grouping of conditions.*
  - ***Task** - All of the user selected geodata rules. Only the rules in a task are used to attach event monitoring to, ignoring all other incoming fields of data. A saved Task will only display the selected rules, but selecting 'edit' when the Task is opened, all the previously unselected rules are shown, allowing a user to select or deselect rules. Buttons in the Task page when a task is opened will be: Edit, Save, Delete. A grouping of rules.*

  The major benefit of this feature is that instead of everyone in the company being notified whenever anything changes in the geodata, only specific fields in the data are selected to be monitored based on the expected conditions set for each field. Only then when all rules are satisfied, the relevant parties subscribed to this task will be notified. Multiple tasks can be created in the web client and the tasks are saved into a specific collection in the GEM tool database. This will allow for concurrent event based notifications that each subscription group is monitoring.

- ***Notifier** - This feature is only triggered when a task is created and a rule in a task is satisfied. The satisfied rule will look at which subscription groups are stored within the GEM tool database's subscription collection and send notifications to all the subscribers registered into the subscription. The MVP of this feature will be an email notification system that will build out an email to send to all subscribed addresses found within a subscription.*

- ***Live Map** - This feature is a live map displaying all **sites** found within a **survey**. Each site is interactive and will display live values for each field in the site's geodata.*

- ***Subscription Groups*** *- Allows users to create groups and subscribe to them. Letting them create a specific Task for that group that, when satisfied, will notify all subscribers of the triggered Task change.*

2.3 User Classes and Characteristics

Technician:

A technician is the individual who interacts with CGRs on a physical level, performing repairs and maintenance who wishes to receive both emergency and summary notifications via email based on the data they are tracking. There could be any number of technicians as anyone will be able to access GEM. The technician is the most important class as they will be the ones ultimately interacting with the application each and every day. They will likely depend on GEM to do their daily job, as well as respond quickly in the event of an emergency.

Analyst:

An analyst is responsible for analyzing the geodata that the technician collects. This group will likely want notifications about the day to day operations of the CGR allowing them to make necessary changes and deploy technicians to needed areas.

Management:

Management is the individual responsible for the surveyor. This class of user may not want to receive emergency notifications as they will most likely not be the one responding. However, they will want to receive a daily/weekly summary of the data that their surveyors are tracking, as well as data on the surveyors themselves.

2.4 Operating Environment

The GEM will be a cross-platform web application. It is the utmost priority of the development team that as many people across as many devices as possible have access to our application. Anyone with access to a browser should be able to run our application. It has been optimized for mobile through use of Google Chrome.

2.5 Design and Implementation Constraints

Due to the nature of this project, in that it is a proof of concept, we are currently limited by the ability to mock out the already existing framework employed by Avista. This includes, but is not limited to, the mocking out of CGRs, Avista's geodata database, and their diagnostic process. We do not expect these constraints to add more than a low risk. The implementation of the GEM tool is intended to be abstracted from this framework in a way that will allow the development of the MVP even in the case that one piece of the framework is not able to be imitated.

2.6 User Documentation

A readme.txt will be provided for the user to explain the functionality and features as well as walk them through usage of the app.

2.7 Assumptions and Dependencies

The GEM tool is dependent on MongoDB for providing data storage, access to a live map, and the Change Stream feature, allowing the GEM tool to watch for changes in the data. It is also dependent on Azure Web Services as that is where the app is hosted live.

3. System Features

3.1 Email Alert System

3.1.1 Description and Priority

System will alert members of a task group through an email or SMS notification when a task they are assigned meets the rules that make up the task. For example, if a rule is satisfied by a CGR producing behavior out of the range of average operation and a task is built for that rule, when that rule is met the members of the group for that task will be notified. This is integral to the function of the MVP as specified by the client, therefore it is a high priority.

3.1.2 Stimulus/Response Sequences

Stimulus: CGR is non-responsive when queried and this event satisfies a task. Response: An alert is sent to the appropriate task group.
Stimulus: Technician enters survey data that is out of acceptable range and this event satisfies a task. Response: An alert is sent to the appropriate task group.
Stimulus: CGR appears offline after it appeared online in the previous poll and this event satisfies a task. Response: An alert is sent to the assigned task group.

3.1.3 Functional Requirements

REQ-1: All members of a task group should receive an email when a task is satisfied.
REQ-2: Emailer should build an email with a list of (1) the CGR ID that has satisfied the task, (2) the priority code, (3) a summary of the error.
REQ-3: Emailer will not build an email for tasks satisfied under a designated priority.
REQ-4: Management will be able to set a priority threshold for the emailer.

3.2 Task Builder and Manager

3.2.1 Description and Priority

A rule is a condition to be checked by the tasker. A task is a group of rules that must all be met to trigger an email event. The task builder will allow an administrator to create rules using geodata fields returned from the web server maintained by the client. The administrator will then be able to group these rules under a task. The task manager will poll the web server for geodata from the CGRs. When the data changes and satisfies all rules of a task, the task manager will trigger the emailer. This is high priority because the email alert system is entirely dependent on the task builder.

3.2.2 Stimulus/Response Sequences

Stimulus: CGR is non-responsive when queried and this event satisfies a task. Response: Task manager triggers the emailer.
Stimulus: An administrator selects geodata fields in the web app and selects "Create Rule". Response: A rule is created.
Stimulus: An administrator selects existing rules and selects "Create Task". Response: A task is created.
Stimulus: An administrator adds registered users to the group for a task. Response: Added users are alerted when that task is satisfied.

3.2.3 Functional Requirements

REQ-1: All members of a task group should receive an email when a task is satisfied.
REQ-2: An administrator should be able to assign users to a task group.
REQ-3: An administrator should be able to create rules from geodata fields.
REQ-4: Administrator will be able to create tasks from existing rules.
REQ-5: Rules will not be dependent on tasks and should be reusable.
REQ-6: Task manager will poll web server for changes in geodata.
REQ-7: Task manager will trigger notifier when all rules of a task are satisfied.

3.3 Live Map

3.3.1 Description and Priority

The GEM web application will show an interactive map with all currently "active" CGRs being polled. This will display online CGRs and their geodata. The map will show the CGR as red if it is offline and green if it is online.

3.3.2 Stimulus/Response Sequences

Stimulus: CGRs status field changes to offline from online.
Response: CGR is now shown as a red circle and the geodata field "status" shows offline.
Stimulus: CGRs status field changes to online from offline.
Response: CGR is now shown as a green circle and the geodata field "status" shows online.

3.3.3 Functional Requirements

REQ-1: Live map must display on webpage
REQ-2: Map must populate with CGRs
REQ-3: CGRs must update when their fields change in the database.

3.4 Subscriptions

3.3.1 Description and Priority

The GEM web application allows for users to create subscription groups or add themselves to a subscription group. This will allow them to receive notifications based on Tasks that are assigned to that Subscription Group.

3.3.2 Stimulus/Response Sequences

Stimulus: User enters their personal information, and chooses a subscription group.
Response: User registered for notifications with that group.
Stimulus: Task is satisfied, field chosen changes.
Response: User receives an email notification on the event.

3.3.3 Functional Requirements

REQ-1: User must be able to enter their first name, last name, and email.
REQ-2: New subscription must be created and added to the list of subscriptions.
REQ-3: User must receive a notification if the field changes.

4. External Interface Requirements

4.1 User Interfaces

The user will be able to interact with the app through data entry in text boxes, dropdown menus, and submit buttons to save entered data. The user will be presented with a popup if they do not enter a required field. Also, on each page, the top menu bar will always appear, allowing the user to navigate around the site on each page.

4.2 Hardware Interfaces

Being that GEM is a web app. It requires very low performance from hardware and nearly every computer and phone can run the software. However, on mobile it is preferred that the user use Google Chrome due to styling issues with other browsers.

4.3 Software Interfaces

MongoDB: *Utilized for live map, data storage, and Change Streams.*
Microsoft WebAPI Core: *v5.2.7: Used to implement the API for database communication.*
Microsoft Dependency Injection: *Used to create services and register them with controllers within the API.*
.NET Test Sdk: *Used to implement unit tests.*
MongoDB Drivers: *Used to help with parsing of data and database communication in API.*
Newtonsoft.Json: *Used for working with JSON data.*

4.4 Communications Interfaces

HTTP is used to allow the web app to communicate with the API and the database. Email is also used behind the scenes to notify the user of changed fields in their created task. The user themselves communicate with the application through our interactive web pages which allows them to create tasks, add themselves to a subscription group, and view the live map.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The GEM should have very low performance requirements. Any modern computer and smartphone with GPS capabilities will be able to run GEM without issue. It is inferred at this time that an internet connection will be required in order to utilize the GPS capabilities of GEM.

5.2 Safety Requirements

At this stage of development, there are no safety concerns with GEM.

5.3 Security Requirements

Security requirements will revolve around the email notification system. Ensuring that the user or administrator that is requesting notifications is authorized to do so. Also, checking their input to ensure that they have given a valid email.

5.4 Software Quality Attributes

It is very important that the MVP is a very good foundation that allows for future expansion. ArcGIS has limitless options in terms of what can be tracked/marked on the maps. That being the case, the *development team has the goal of ensuring that the MVP provides a clear route for future developers to expand GEM for their own personal needs.*

6. Other Requirements

There are no other requirements at this time.

Appendix A: Glossary

- *Site - An example can be a router. In the live map , a site will appear as a dot on the map and will be identified by its associated geodata's GPS coordinates.*
- *Survey - A grouping of all Sites found within a database.*
- *CGR - A Cisco Grid Router*

Appendix B: Analysis Models

UML Diagram:
https://www.lucidchart.com/documents/edit/1d92d6be-7c4e-4331-b15e-80b52109ed51/0_0?shared=true

Appendix C: Issues List

Issue: *Task page must restore to the state of the selected task.*