Assignment 9: Individual Requirements Analysis

Project Name: CHAOSS Project: Augur

Sarina Gaines

Revision History

Date	Revision	Description	Author
4/1/2020	1.0	Initial Version	Sarina Gaines

1. Introduction

The CHAOSS community is dedicated to developing metrics, methodologies and software meant to evaluate and display the health and sustainability of open source projects. By measuring the health and sustainability for open source projects, transparency and actionability is created and as a result, stakeholders are able to make informed decisions regarding open source project engagement. CHAOSS aims to establish standard implementation-agnostic metrics for measuring community activity, contributions and health CHAOSS also aims to produce integrated open source software for analyzing software community development and building reproducible project health reports/containers. There are five workgroups that are built around four categories of metrics identified by CHAOSS. These working groups are, common metrics, diversity and inclusion, evolution, risk and value.

2. Software Product Overview

Augur is a Flask web application, Python library and REST server. Its purpose is to present metrics on open source software regarding project health and sustainability. Augur is a prototype that is meant to implement the CHAOSS Project on open source software metrics. Augur helps to make sense of data and does so using human centered data science strategies.

3. System Use

Organizations

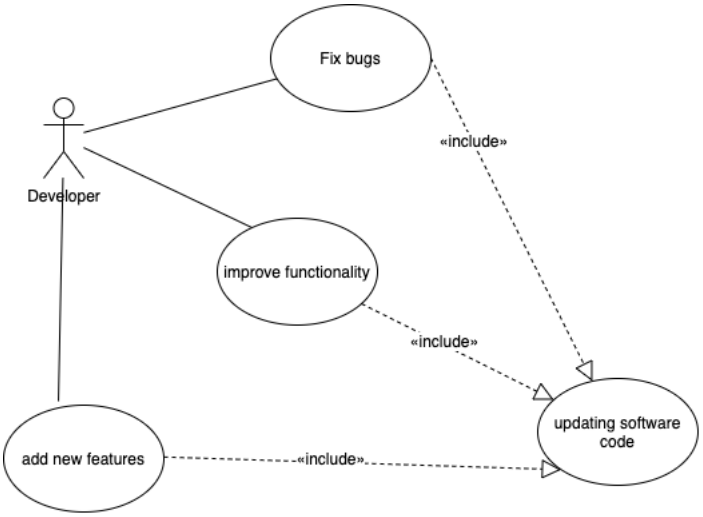
Organizations are the main user of the system. This actor is responsible for setting up the system on their devices. Organizations will need to have a list of repositories and groups they want to be in. Additionally, this actor must obtain access to servers that meet installation requirements, have an http server installed, have databases available and a GitHub API Key. Organizations will interact with the system by creating metric functions, creating metric endpoints and creating visualizations. By interacting with the system in such a way, Organizations will be able to better interpret data.

Developers/Creators

Developers are responsible for maintaining the product and aiding customers. This actor will interact with the system by doing things such as updating software code to fix bugs, improve functionality and add new features. Doing so will allow the Developers to stay on top of the product and improve performance of the system. Developers will also be responsible for providing support to the users of the software regarding any questions users might have.

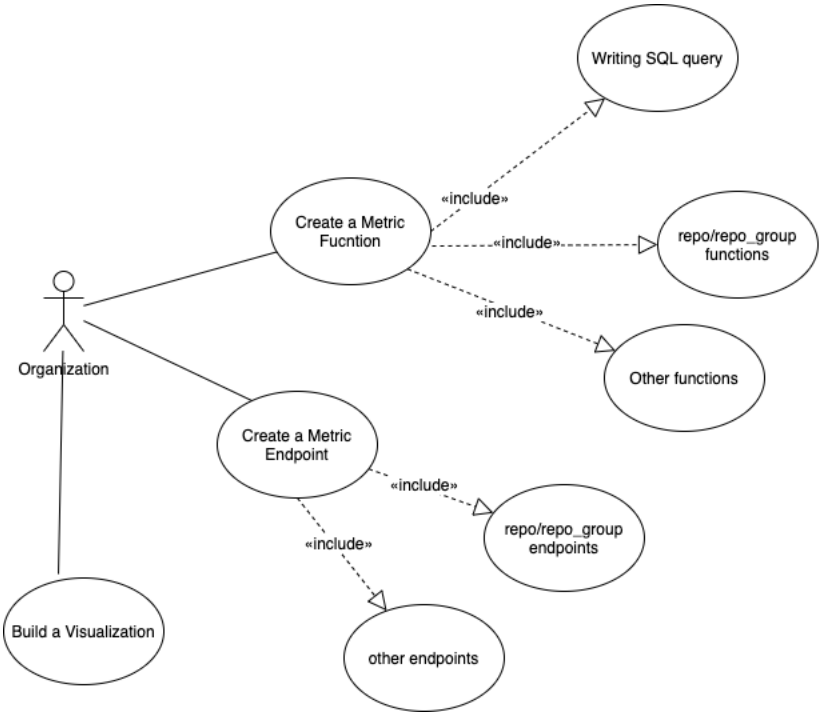
4. System Requirements

Developer Actions



Use-case name	Developer Actions
System	Augur
Actors	Developer
Brief Description	This use case explains what developers can do to update and fix the Augur system.
Basic flow of events	Basic flow begins with a developer deciding what needs to be done. A developer can fix bugs, improve functionality and add new features to the system. To do this the developer will need access to the system code.
Special requirements	Requirements include being a developer, having access to the system and getting approval to make changes.

Organization Actions



Use-case name	Organization Actions
System	Augur
Actors	Organization
Brief Description	This use case explains what Organizations can do when using the Augur system.
Basic flow of events	Basic flow begins with an organization deciding what they would like to do using the Augur system. An organization may either create a metric function, create a metric endpoint or build a visualization. Create a Metric Function 1. Write an SQL query 2. Repo/repo_group 3. Other functions Create a Metric Endpoint 1. Repo/repo_group endpoint 2. Other endpoints

	Build Visualization 1. Further information regarding visualization can be found in the front-end documentation.
Special requirements	Requirements include having Augur appropriately installed and having access to required serves and repositories.

Functional Requirement: Augur supports most UNIX systems. If you do not have access to a UNIX system, an Ubuntu 18.04 VM can be set up to use Augur.

Non-functional Requirement: simple UI

5. Design Constraints

1) Use of Augur requires the repository to be cloned.
2) A virtual environment in your home environment will then need to be created.
3) Python 3.6+
4) A PostgreSQL 10 or higher installation is required.
5) GitHub Access Token is required, however, no write access required
6) For frontend, you will need Vue.js, vue-cli, node and npm.
7) After that, you may begin the installation process. This process will install Augur's backend, install the data collection worker, generate documentation, generate the configuration file, optionally, install the database schema and load sample data, and optionally, install Augur's frontend and its dependencies.

6. Purchased Components

1) Computer, mouse and keyboard
2) Internet

7. Interfaces

Software Interface:
Access to servers and databases in order for Augur to function.