Individual Requirements Analysis for Semester Project

Rebecca Parker

Fall 2019

Contents

Introduction

This document was written to specify the software requirements of the semester project. It will specifically detail the requirements needed to implement 3 new API endpoints. In addition, it will provide an overview of the current software architecture and how the proposed changes will be incorporated. This document will also go over non-functional requirements and other design constraints that will affect development.

Software Product Overview

Augur is a software suite that provides health and sustainability metrics about open-source projects, in collaboration with the Linux Foundation CHAOSS Project. The metrics can be accessed through the backend API or the frontend visualisations. The focus of this project will be on adding to the available backend APIs. The complete data model can be found at https://oss-augur.readthedocs.io/en/master/architecture/data-model.html#complete-data-model. An overview of the three API endpoints to be developed follows.

Each endpoint will follow the same general process. The database of open-source project data will be accessed through an SQL query. After the results of the query are returned, they will be packed into a JSON and then returned at the end of the function.

1. The first endpoint will take a repo_id and return an array of data points that maps dates in time to numbers of commits.
2. The second endpoint works on the same principle as [1], but is for an entire repo group rather than a single repo. This endpoint will return timelines for each repo in the group.
3. The third endpoint will be provided the ID of a specific contributor and will return a list of repos the contributor has worked on, as well as how many commits have been made by that contributor in each repo.

Installation of Augur requires write access to a PostgreSQL 10 or higher database. Since current Augur APIs are written in Python 3, we will utilize Python 3 and a Flask server.

System Use

The primary users of these endpoints will be open-source software developers. These developers will be using the Augur software to view metrics on the health of the open-source project they represent. The developers will use the data from these metrics to advertise their product to investors and/or consumers, or for personal decision-making in regards to management of the open-source project.

System Requirements

Use Case 1: A repo contributor wants a personal “GitHub Report.” They gather data about their personal commits over time to produce a graph to visualize their commits over time.

Use Case 2: A repo manager wants to see the distribution of work for an employee among multiple repos. They are able to gather data on which repos an employee commits most and/or least to, and use this insight to assign work to multiple developers among multiple repos.

Functional Specification:

The API endpoints will need to have the standard function signature used for all Augur metrics, metric_name(repo_group_id, repo_id, period, begin_date, end_date). To create each metric, a PostgreSQL query will be written against the Augur database schema and then wrapped in the metric function. The metric function will return a dataframe, the contents of which can be used in visualizations either as part of the Augur software package or a separate developer’s preferred visualization tool. Finally, the endpoint/route must be defined and documented so that the endpoint can be accessed remotely.

Non-functional Requirements:

The endpoints must fit within the existing Augur framework so they can easily be used by developers already utilizing Augur.
The endpoints must return the JSON data in a time comparable to the return times of existing Augur endpoints.
The endpoint must be accessible from anywhere Augur is also accessible.

Design Constraints

To fit within the existing framework, the function signatures must match the existing standard function signature.
The technologies used are constrained by the technologies currently utilized by Augur. For example, the endpoints will be supported on a Flask server since that is used for existing Augur endpoints.
In addition, the endpoint function will be written in Python 3, as that is the language that all other Augur metrics are written in.
The data returned from the database must be packaged as JSON in order to fit with current Augur frontend development.
Finally, the three endpoints must be completed by the end of the semester, December 5th.

Purchased Components

A server will need to be purchased to support the endpoints.

Interfaces

The JSON results will need to be viewed in a browser or connected to a user interface in order to see the data that is returned. Ideally, the user interface connected to the endpoints will provide some sort of visualization of the data. For example, the first endpoint is designed to be displayed as a line graph.