Requirements Analysis

Software Engineering Spring 2020

Collin Pounds

Table of Contents

1. Introduction

1.1. Purpose

This document's purpose is to specify the system requirements of Augur, a software tool developed by CHAOSS. CHAOSS is a community of software developers in the Linux Foundation Projects Group. The aim of this group is to create a system of metrics to define the health of an open source project. The Augur project is the prototype of the CHAOSS community's aim with focus on making sense of data using human based data science strategies. This project has many uses including but not limited to providing open source projects with a way to analyze their own performance, and for contractors to compare different development communities ability to run healthy open source projects.

2. Software Product Overview

2.1. Augur

Augur is a web application, Python library and server with an aim to make sense of the data to assess the health and sustainability of open source projects. It provides different metrics on open source software which makes it easy for companies to find parts of a project that are lacking or striving. It also provides visualizations of the data to make the metrics more user friendly and readable. Data can be obtained with Python or directly from the server. Augur provides a vast array of information on individual repositories and group repositories and puts it into user-friendly, readable format.

3. System Use

3.1. Actor Survey

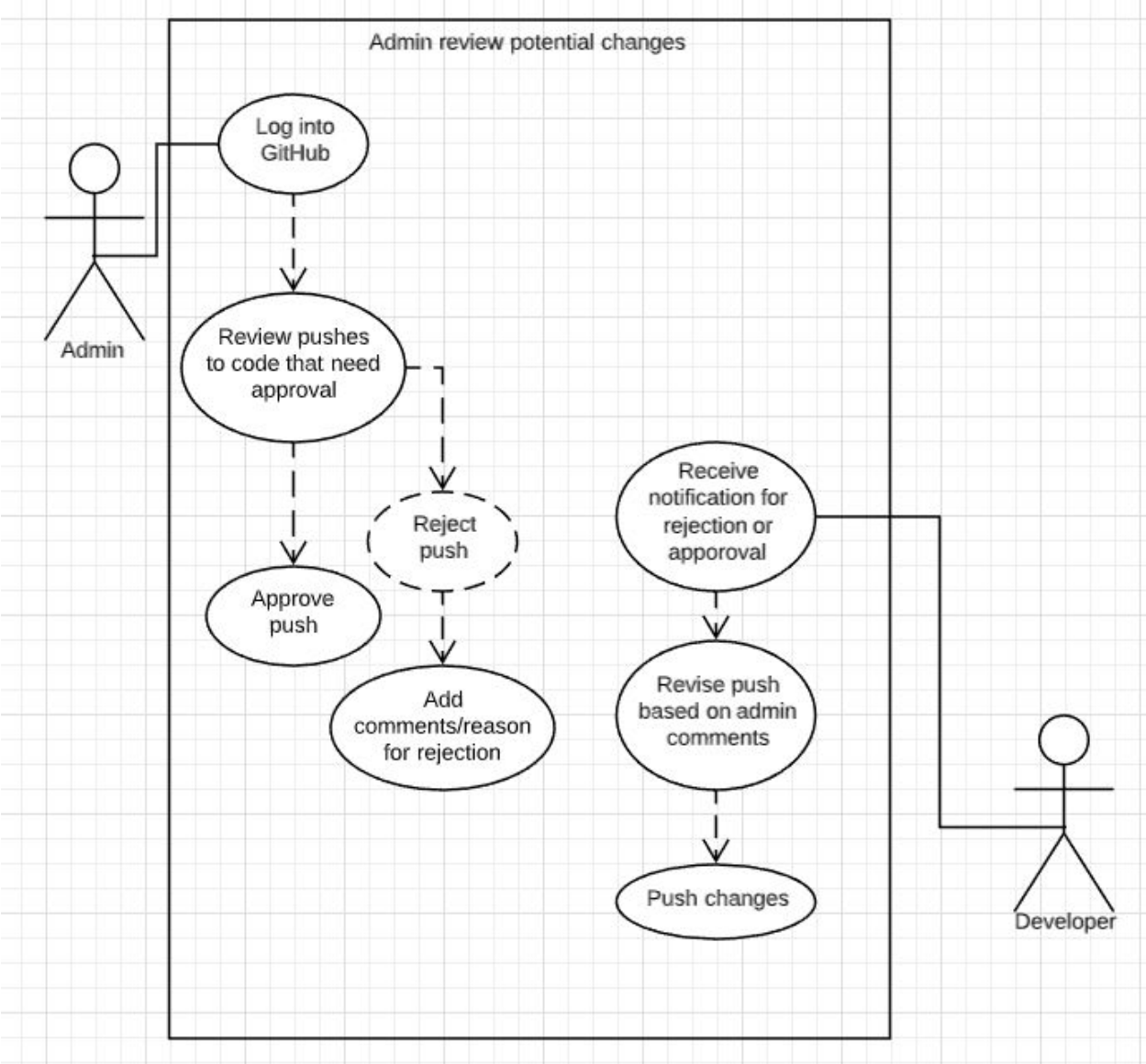
Administrator
An administrator has responsibilities that include reviewing potential changes to the software, finalizing additions, requesting resources from project funders, assigning funds to certain areas of the project, and assigning employees to specific jobs. The administrators can also work on areas of the project that need more man-power when their other responsibilities are complete.

Developer
A developer has responsibilities that include receiving and completing tasks from administrators, submitting work to be reviewed, and fixing bugs. They receive their tasks from administrators but have a comfortable amount of freedom to get the task done in their own way. Developers can create their own metics or pre-built ones to get information on repositories.

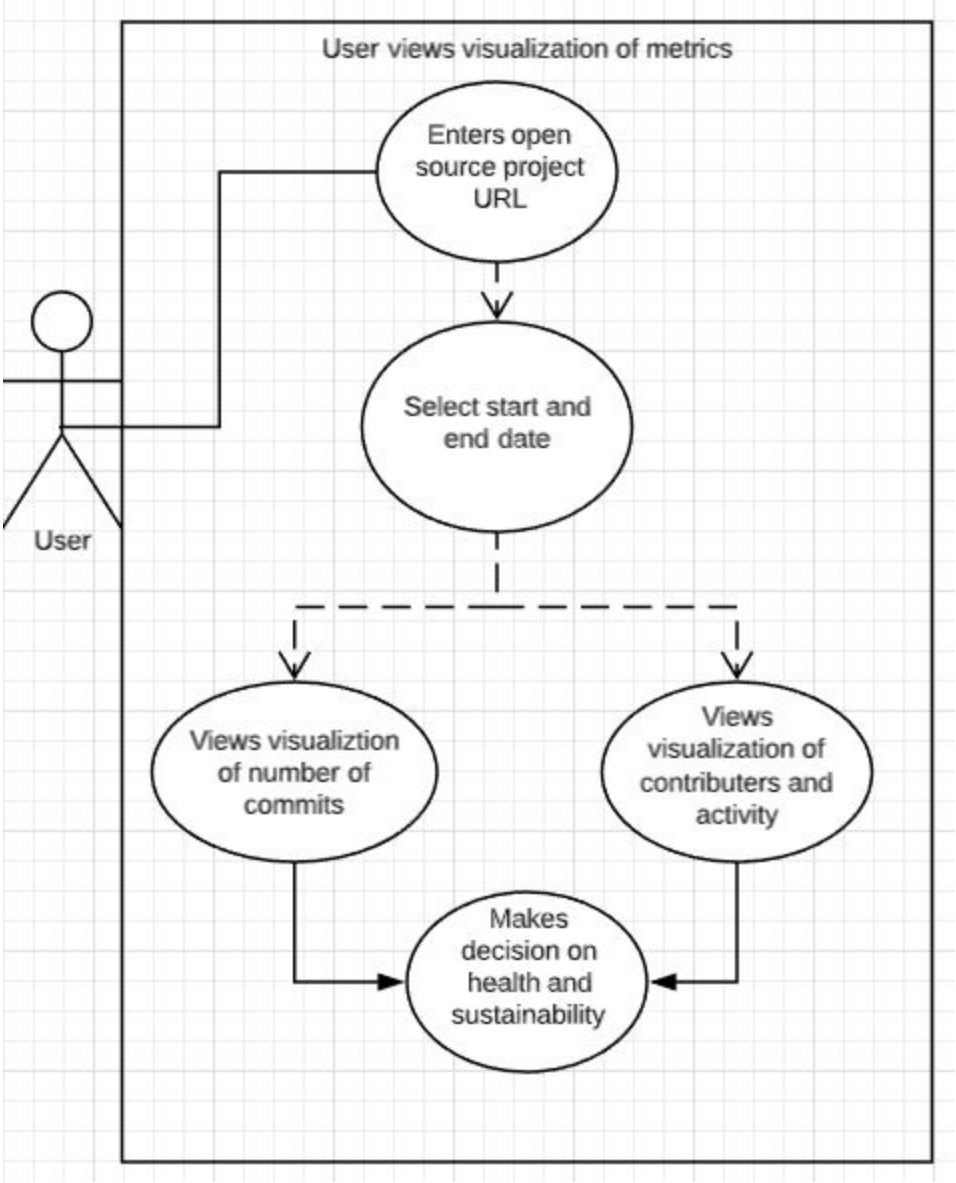
Users
A user of Augur doesn't necessarily have responsibilities because the overall function of Augur does not depend on their work. Users are still required though, as Augur is a prototype, and is not yet a fully developed product and requires user feedback and critique to make the system more user-friendly and useful. Users use the metrics to visualize different repositories or repository groups. They interact with the front end of Augur, and provide feedback for how to make the system better, and stronger. Users can download visualizations or data organized by Augur and interact with the user interface through the web application.

4. System Requirements

4.1. Use Case - Admin review of potential changes



Use Case - User views visualization of metrics



4.2. System Functional Specification

User Functions

Search:

UIF1: SetFilter

- UIF2: GetUserInput
- UIF3: SearchUserInput

Metrics Visualization:

- MVF1: ChangeYAxisMetric
- MVF2: ChangeXAxisMetric
- MVF3: ChangeGraph
- MVF4: ChangeColorScheme
- MVF5: DownloadGraph
- MVF6: SaveGraph
- MVF7: ResetToDefault
- MVF8: SetDates

Account:

- AF1: ChangePassword
- AF2: SearchSavedGraphs
- AF3: DeleteSavedGraph
- AF4: Login
- AF5: Logout

Administrator Functions

System:

- ASF1: Login
- ASF2: CreateDeveloper
- ASF3: EditDeveloper
- ASF4: EditDeveloperPermissions
- ASF5: DeleteDeveloper
- ASF6: ConfigureSettings
- ASF7: Logout

Tasks:

- ATF1: ViewTask
- ATF2: Edit Task
- ATF3: AssignTaskDeveloper
- ATF4: DeleteTask
- ATF5: ReviewChanges

ATF6: ApproveChanges
ATF7: RejectChanges
ATF8: AddComment

Developer Functions
System:
DSF1: Login
DSF2: ViewAssignedTasks
DSF2: MarkTaskComplete
DSF3: SubmitChanges
DSF4: AddComment
DSF5: Logout

4.3. Non-Functional Requirements

1. User should be able to read graph in default mode
2. User interface is user friendly for all functionality
3. Changes must be approved by administrator
4. Cannot set dates past current date
5. Visualization customization should be easy to configure

5. Design Constraints

1. Repository to pull data from must exist
2. Web application can run on machines with internet and Windows, MacOS, and Linux operating systems
3. System communicates from Flask Web App to Python library and REST server seamlessly
4. System can handle requests for multiple repositories at a time
5. Requests for repository metrics will not take more than 5 seconds

6. Purchased Components

1. Licenses for software
2. REST Server

7. Interfaces

The Augur system is a Flask web application communicating with a python library and a REST server. The main interface for program functionality is the web application user interface. The user interface will allow users to search user entered repositories and repository groups and show visualization of the metrics pulled about their selection. The Python library will act as the main functionality to obtain the data from the repositories.