Assignment 9:

Individual Requirements Analysis

Created By:

Joshua Hickman

1 - Introduction

1.1 Purpose

The purpose of this document is to describe the software developed to utilize CHAOSS Augur Evolution working group's API data to understand metrics that show the health of software development projects. This document will describe the external behavior and internal workings of the application. The non-functional requirements, design constraints, and interfaces will also be described.

1.2 Scope

Being able to keep track of the lifecycle of an open source software (OOS) application is integral to any company using that application for any operations they may have for it. Thus, the ability to see when a project may be becoming phased out or has lost interest in their upkeep is essential in planning the next step to acquiring replacement software or purchasing an alternative.

The development of the application will be done in phases. First phase, deciding upon which application from Evolution will be tracked. Second phase, development of a web application that will show the user specified data from said software metric. The backend and frontend are to be developed in tandem, using graphs to display the various metrics to be gleaned from the API data.

2 - Software Product Overview

This section will provide an overview of the web application that will serve the API data from Evolution to the user as they choose the different metrics to view

2.1 System Scope

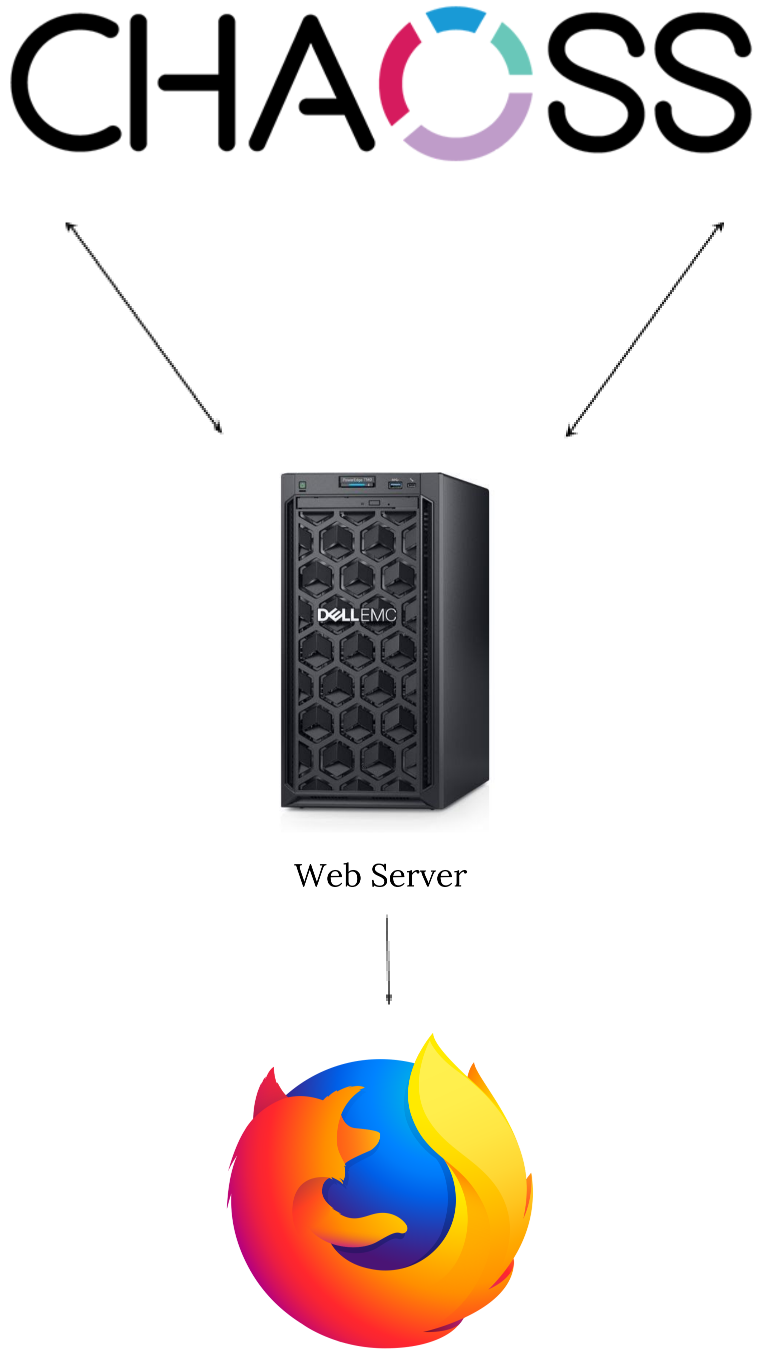
The web application will rely upon API data from CHAOSS Augur to present graphical data on OSS applications to the user. Thus a connection to Augur will be necessary to keep data up to date and accurate. A database of the data will be built, and the application will be made available through browsers.

The web application will be developed for desktop web browsers with the potential of mobile browsers depending on the capabilities of the development platform chosen, **(and the familiarity of said platform from my standpoint).

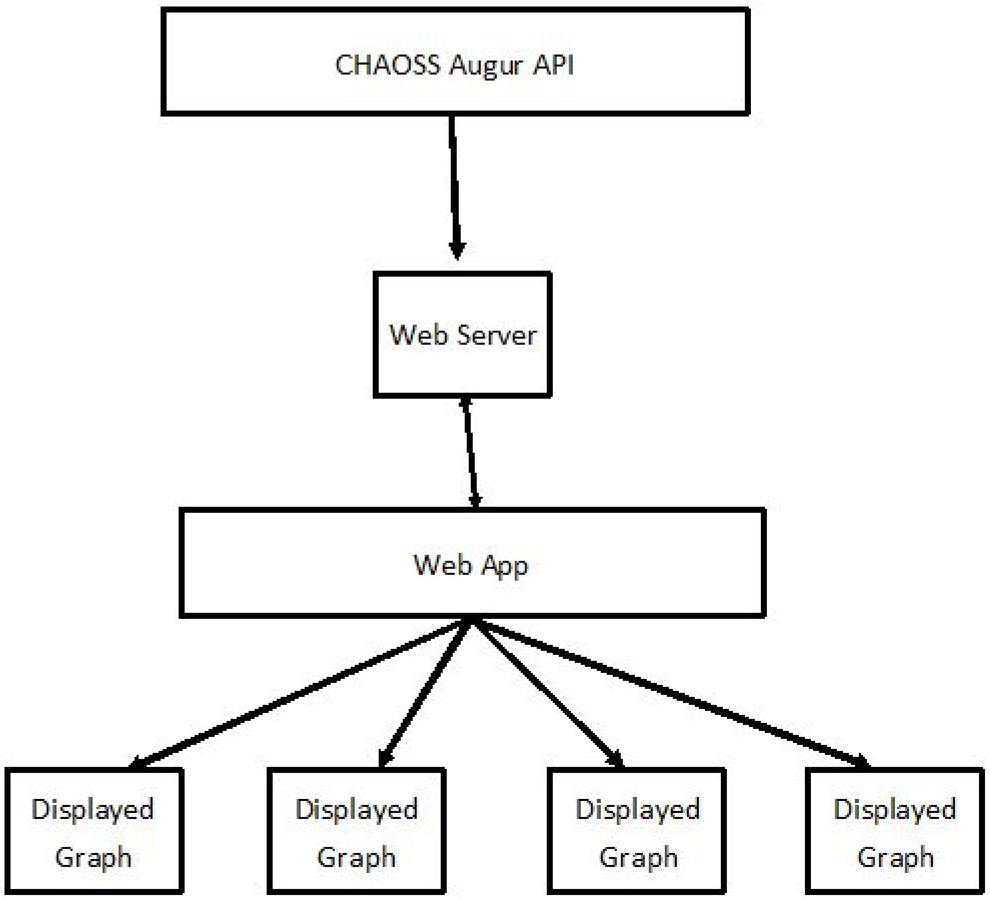
2.2 System Architecture

This section will describe the external and internal architecture of the web app

2.2.1 External



2.2.2 Internal



3 - System Use

3.1 Actor Survey

Owner / Administrator

The owner/administrator is responsible for designing and maintaining the web server and web application. This involves ensuring API link stays connected and services running 24/7 to ensure anyone who wants the data would be able to access it at their convenience. They are responsible for fixing any bugs, or failures in the hardware or software of the web application system or web server. If CHAOSS Augur is offline then the user should be able to see clear notification of said fact.

System Features:

- Easily navigable interface that explains what the user is seeing
- Graphical representations of metric data from CHAOSS Augur
- The ability to refresh the data

User

They are the ones navigating to the web application to view the data.

4 - System Requirements

4.1 System Use Cases

Use case name	Display the Data
System or subsystem	Web application, Augur API
Actors	Owners/Administrators
Brief description	This use case explains how the data from the API will be displayed to the user
Basic flow of events	1. Client side makes request to web application 2. The django web application calls Augur API and pulls data into a JSON object 3. The JSON object is used to run queries on the data pulled in order to grab specific information 4. That information is passed to a Canvas.js module to display that data as a graph on the web page
Alternative flow of events	1. Client side makes request to web application 2. The django web application calls the Augur API only for it to

	receive a connection error. 3. The connection error information will be displayed to the user 4. An automatic refresh will be called every 30 seconds or until the user leaves the web application. This is displayed to the user

Use case name	Refresh data
System or subsystem	Web Application, Augur API
Actors	User, Owners/Administrator
Brief description	By clicking on a refresh data button, another call will be made to the Augur API to pull in the latest data, without the need of refreshing the page
Basic flow of events	1. The user clicks on the refresh data button 2. The django web application calls the Augur API and puts the current data into the same JSON object 3. The new JSON object is used by the Canvas.js module to update the data of the graphs on the web page
Alternative flow of events	1. The user clicks on the refresh data button 2. The django web application calls the Augur API, however there is no response 3. The connection error information will be displayed to the user 4. An automatic refresh will be called every 30 seconds or until

	the user leaves the web application. This is displayed to the user

4.2 System Functional Specification

Client Terminals

As this is a web application, the system architecture will be client-server. All requests will come from the client computer to the web server. Mobile browsers are not supported at this time

Server Application Modules

An apache web server deployed on AWS will host the web application and receive requests from client computers that want to connect to the web application.

4.3 Non-Functional Requirements

4.3.1 Usability

The user interface should be clear, fonts and sizes to ensure that the data and text is readable for the user.

The descriptions of each graph should be clear and definitively placed as to understand which explanation goes with which graph

4.3.2 Reliability

The web application should be available 24/7 every day of the year, so the web server should be maintained and kept in an ideal environment. Any disruptions in service should be swiftly resolved.

4.3.3 Performance

The site and its graphs should load quickly and there should not be long load times. Only a small amount of data will be displayed, if expanded in the future, the consideration of adding a database for stored data will be in place.

5 - Design Constraints

The system will run as a web application with open access to the public. The supported browsers will be Chrome, Firefox, and Safari. Mobile browsers will not be fully supported at this time. The user will not be able to see any open source software metric, only a specific one from the Evolution section of Augur. Only the graphs displayed will be the metrics viewable. It will not cover every scenario of the API data, just the most relevant.

6 - Purchased Components

Server needed: Dell PowerEdge T140 Tower server

Intel® Celeron® G4900, Optional Operating System, 8GB Memory, 1TB Hard Drive

$499.00 on sale, reg. $782.00

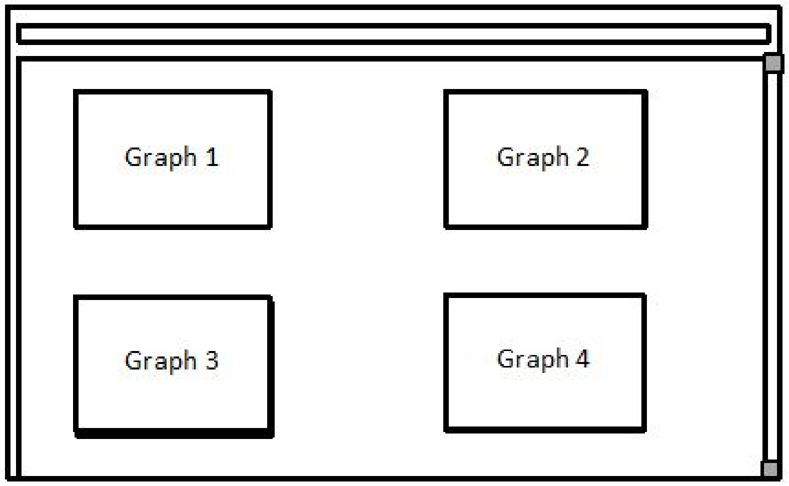
Will function as a web server

Software: services needed are OSS or have already been obtained

7 - Interface

7.1 User Interface

This will be the landing screen for the user when they arrive to the web application



7.2 Hardware Interfaces

The web application server will bind to the IP address of:

{Address TBD}

If different metrics are supported or expanded in the future or increased web traffic occurs, a database may be used to store previously obtained API data to stop as many API calls to Augur