Assignment 8: Individual Requirements Analysis for Semester Project

Tim Kuehner

Introduction

The purpose of this document is to give a detailed understanding of the requirements for the Augur software that we are using for our semester project. Augur is a project built to help catalog the number of commits and other processes along with details on those processes done on a specific repo. This will help businesses and others have a better understanding of the health of a specific project as well as have a visualization of who is contributing to each repo and how much. My specific group has been put on the task of working on the API for the Augur system so this document will be an overview of the Augur software but will look more closely at the API part than other parts of the software.

Software Product Overview

Augur is a system comprised of a backend, an API service, and a frontend visualization. The backend is comprised of servers that are created using SQL and other implementations such as Linux. SQL is the base code for the databases that store the information used by the frontend and API later. The information, that includes pull requests, commits, ID's and many more, are stored in respective tables that can be easily quarried using SQL architecture. The extensive system of connected tables is subject to queries to pull the data from the database to be used by the users. The API service is the service that connects the frontend visualization to the backend servers. API endpoints are created that allow frontend developers to make calls that return JSON objects that can parsed and turned into usable data to be visualized. The frontend visualization portion of Augur as described is responsible for displaying and requesting the data held in the databases. Whether it is plain text or a timeline, users can call an endpoint that completes the task that they are wishing to complete and are returned the data.

System Use

This system can be used by a multitude of people as seen in the actor survey below

Actor Survey

Project Manager –

A project manager is responsible for determining the health of project and repositories being worked on by developers in their domain. This actor will interact with the system by searching specific repositories to see who is contributing to what repositories and when. They are also able to see the interactions with various points of a project to deem what the focus should be for their team in the immediate future.

System Features:

- View commits

- View committers
- View pull requests

System Requirements

Use Cases and Requirements

Repo contributor:

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.

- Data selected from key **cntrb_id** from **Contributors** table. Each cntrb_id has a one-to-many commits that have the relevant info needed: cmt_committer_date is the only required data needed and the cntrb_id to query the results.

Repo manager:

A manager of a repo identifies points of interest in the life of the repo to identify peak productivity. They want to see times of peak productivity in workers, so they use the endpoint to gather data to visualize the commits over time

- From **repo** table we need the **repo_id** key, from there we can select a count of unique cmt_id keys from commits. If we want to organize by date we need the cmt_author_date as well.

Required:

User terminals

Data entry

Data management

System management/Config

Understanding of endpoints

Access permissions

Non-Functional Requirements:

- User-friendly interface
- Network security and reliability
- Thorough and detailed documentation

- Consistent error checking
- Ability to use on any system from Windows to Mac

Design Constraints

1. The software must be able to run on Chrome, Firefox, and Safari at a minimum
2. The interface must be user friendly with documentation to aid the user
3. The databases tables must be updated every 10 minutes to allow for real time use
4. The interface must notify the user if the repository they have searched is empty or restricted
5. The interface must notify the user if the repository they have searched does not exist or was removed

Purchased Components

Web server (AWS or physical server)

Database server (Physical server)

Interfaces

A website domain must be hosted and be bound to a public DNS. This interface must be able to communicate with the API service and parse the returned JSON object to be displayed.