Brandon Greer

Assignment 9: Requirements Analysis

Outline:

Introduction

1.1 Purpose

An important consideration when adopting open source software is the possibility of that software being abandoned by its creators/maintainers. Individual projects are often abandoned as their creators move on and no one steps up to maintain the software. Additionally, since most software is built on top of open source languages and libraries, the retirement of a language version (e.g. Python 2) may create more work than a maintainer wishes to put into an older project, which in the case of a library may have similar downstream effects on many other projects, even if those maintainers were willing to port their own software to the newer versions.

For this reason it would be beneficial to develop both a set of metrics and an easy to use platform which could be used to monitor a piece of softwares status, both in terms of the responsiveness of its maintainers and the regularity of updates or changes.

1.2 Scope

This document describes the proposed system, and additional investigation would likely be required for actual implementation

1.3 Assumptions and Dependencies

Our platform would be dependant on some type of web server (likely Django), integration with source control for the open source projects being tracked, Github and Gitlab, and a cloud hosted linux VM

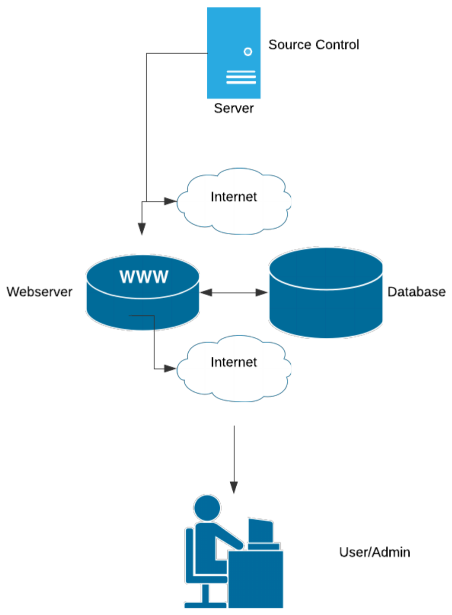
2. Product Overview

2.1 System Scope

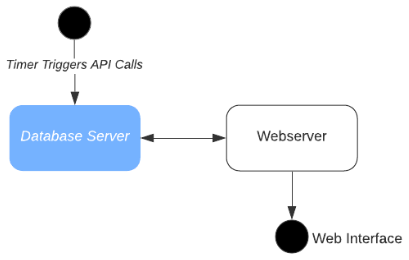
The system would likely be essentially self contained. Data from one or more source control providers could be gathered processed and displayed, all within the system.

2.2.1 External view

The below figure illustrates the external architecture of the system. Essentially data provided from source control will be stored in a database server and compared against preset metrics of project health, all of which can then be viewed by an external user through the web interface.



2.2.2 Internal View



The internal view is very similar to the external view. The database and web server could reside on a single machine or be seperated, depending on the expected load. Placing duplicate web servers and database servers or two servers which serve both purposes behind a load balancer would allow for easier updates and maintenance at an increased cost. All ‘hardware’ would be virtual.

3. System Use

3.1 Actor Survey

System Admins:

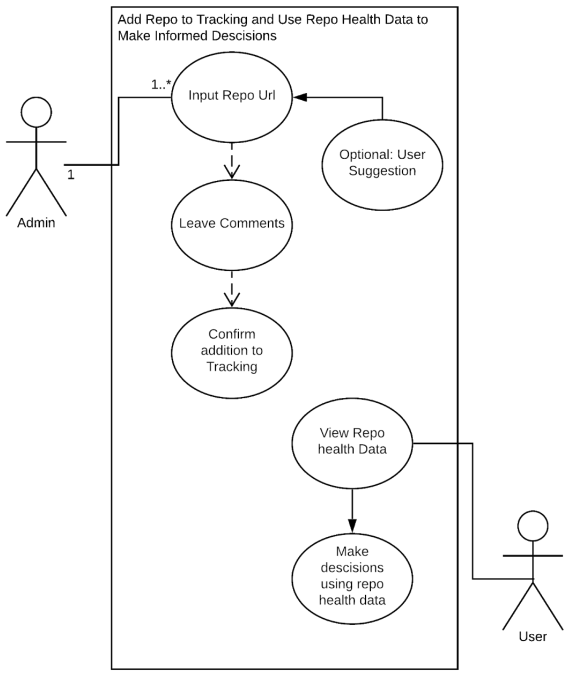
- Add/Remove Repos to Tracking
- Mark Repos as Abandoned
- View List of Requested Repos
- Link Related(Fork/Continuation) Repos Together
- Modify Health Criteria

User:

- View Repo Statistics and System's Evaluation of Repo Health

4 System Requirements

4.1 Use Case Examples Admin Adds Repo to Tracking & User Views Repo Info


Add Repo to Tracking and Use Repo Health Data to Make Informed Descisions
1..*
1
Admin
Input Repo Url
Optional: User Suggestion
Leave Comments
Confirm addition to Tracking
View Repo health Data
Make descisions using repo health data
User

4.2 Feature Overview

System Admin:

F1: Add a repo to tracking

F2: View requested repos which users have requested tracking on - external users will have the ability to request that a repo they use, are considering using, or control to be tracked. This feature would allow admins to view requested repos and approve them

F3: Set/change thresholds of which control the systems expectations regarding repo health

F4: Remove a repo from tracking

F5: Declare a repo abandoned - admins should have the ability to declare a repo abandoned upon announcement of such from the maintainer or based on data from the system

F6: Link a repo as being the fork/continuation of a (possibly abandoned) repo - Declaring a repo as a fork or continuation of another repo may aid external users in deciding which repose to integrate into their projects or find alternatives to repos they have been/ are considering using

External User:

F8: View Repo Data - view data, presentation, and assessment of a given repo

F9: Request Adding of New Repo - Users should be able to request the addition of a new repo either as maintainer, user, or potential user

4.3 Non-Functional Requirements

1. Interface will need to be user friendly and easy to navigate
2. System Administrators may need to make notes regarding specific repos, if the repos themselves contain insufficient information to make it clear what the repo is.

5 Design Constraints

1. The software would only be able to import and report on repos contained within platforms with robust api support
2. The software would face a tradeoff between resource use and timeliness of reporting. For example, several critical open issues remaining open for more than a week would likely indicate poor repo health, but if repos were only polled once a day and several issues were fixed within the same day, it could be up to 24 hours before the systems reporting changed. On the other hand polling once every 5 minutes, even for a modest number of repos (a few thousand as an example) could both consume massive resources and a large number of API calls for little benefit in overall accuracy.
3. The system may be unable to accurately interpret certain scenarios that might logically appear. If a software release included functions which did not behave as expected in some scenarios but were not-security related, and could be worked around, the maintainers may choose to delay resolution until the next minor version, which could leave issues as unresolved for several months
4. Repo maintainers would have no real ability to opt-out if the system were built using publicly available APIs
5. It might be difficult to categorize whether certain projects are retired or not For example Python 2 has been declared end of life by the Python Software Foundation, however Red Hat has committed to maintaining the software until the last version of Red Hat Enterprise Linux that shipped with Python 2 has

reached end of life, and the license of Python 2 would require them to release those changes.

6. Purchased Components

1. 1 or more virtual Linux servers on a cloud platform like AWS or Azure

7. Interfaces

Many of the softwares features could be made externally available relatively easily

1. Re-reporting of repo statistics like commits, line of code changes, issue resolution timeframe, active maintainers/developers etc.
2. The systems report on the status of repo health, based on set criteria