Requirements Analysis

Wasay Khan

Assignment 9

Project Name: Augur

Table of Contents:

1. **Introduction**

This document will specify the requirements, features, design constraints and purchased components of Augur.

2. **System Product Overview**

System Scope:

Augur is a tool that is used to help determine the health of different open source projects that are in a system. Health can be determined by a variety of factors such as number of contributors and number of commits. The health of a program will help IT companies which projects will be healthy in the future and worth pursuing. Augur allows users to see data graphically and make comparisons with other data sets. The metrics can be downloaded and then used to assess the sustainability of a project.

System Architecture:

Augur is broken down into 5 parts:

1. Workers who collect the data and insert it into the data model
2. Broker that distributes data collection tasks to the workers
3. Housekeepers keeping the data updated
4. Main class that does caching, registers plugins and reads the configuration file
5. A WSGI server that exposes data sources as a REST API

Features:

The overall features of this application can be divided into 4 different functions. Overall, Group, Insight and Repo.

- Overall Function:
    - New User – Allows the creation of a new user for the Augur application and allows them to attach their GitHub accounts to link their repositories.
    - Delete User – Allows the deletion of current users in the system and remove their information from the system.
    - Repository Search – Allows the user to look up their repositories
    - Store Repo – Allows the user to store a repo that they are currently a part of
- Group Functions:
    - Create/Delete Repo Group – Creates a repository group and allows the addition and removal of repositories to and from the group
    - Add Repo to Group Repo – Adds a repository to a group repository, and allows the managers and the people in the group to view it
    - Currently Stored Groups – Presents information about a user's currently stored repo groups, including the name, description, website and last modification
    - Stored Groups Sort – Allows the user the ability to sort by the previously explained statistics from currently stored group functions

    - User Management – Allows administrators to add or delete users from their groups.
- Repo Functions:
    - Currently Stored Repos – Presents information about a user's currently stored repo, including URL, group name, total commit count and total issue count
    - Stored Repos Sort – Allows the user to sort previously analyzed statistics from currently stored repos function
- Insights:
    - Data Projection – Allows the user to view information about their repos/groups with easy to view things such as graphs
    - Switch Projection Function – Allows the user to swap the projections (bar graph, pie chart etc.)

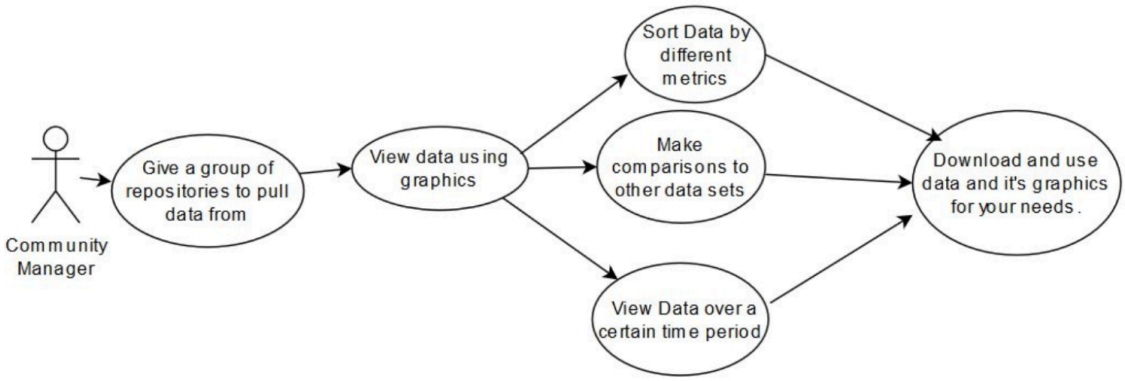
3. System Use

Actor Survey:

Community Manager – Responsible for the creation/deletion of the repository. They use the system for collection of data and determine the health of the communities they are involved with. Augur allows them to determine how a community is doing by providing the ability to manage their repositories. They also have access to an API that makes it easier to access metrics along with the ability to make comparisons with other projects. They can also determine how the metrics will be visualized on the website.

User – The default status of a user for the Augur system. A user can query metrics by using a GET request. The data will then be returned as a JSON. The user status allows them to be able to view metrics from the repositories that they have created or have been associated with. They can see those visualizations as graphs, and they can also be downloaded.

4. System Requirements



System Functional Specification:

- Repository information: Get information about a selected repository. That information can include name, description and number of commits
- Visualization: Get visualization of data from a repository

Non-Functional Requirements:

- Should be user friendly and provide guides
- Should be reliable and accessible

5. **Design Constraints**
    - Will be a web application with access via standard secure login procedures
    - Available on all major web browsers
    - Differences between web browsers should be minimal
    - Desktop and mobile version should be similar, with few changes such as layout and design for ease of access

6. **Purchased Components**
    - Domain name
    - Amazon AWS EC2server space

7. **Interfaces**
    - REST API interface
    - Visual frontend flask application with visualizations