Open Source Software Health and Sustainability Metrics Tool

Introduction

Our product aims to help companies identify whether their development has chances of surviving in the market and analyze the evolution of the deliverable. The analysis will consist of multiple metrics that will help developers identify the frequency of updates, most active contributors on the team, data about open and closed issues and other info that will draw the picture of the overall quality of the development process.

System Product Overview

Just like some of the existing products on the market, our solution is distributed as a Python library. The set up process will include setting up a database to store the reports locally. In addition to that, our product allows uploading the data visualizations to our Community Reports website to make them accessible from any device.
The visualizers themselves will represent a wide range of data. For example, numbers of commits, open/closed issues per users, changes in development dynamics, and much more. Every one of those may be represented as a graph or a table

System Uses

Title	**Actors**	**Description**
Project Upload	User, Project Contributors	If users are interested in getting to know the metrics related to their development process, they need to go through the default process of uploading their Git repo URL, choosing metrics, and receiving the report. The report may also contain information on the productivity of certain contributors.
Upload Report to Community Repos	User, Community	Users may choose to upload their reports to the Community Repo. This will make it visible for public while also making it more accessible from various devices
Delete Report from the cloud	User	Users should be able to login to their online profile that they used to upload the repo report and request to remove

Project Contributors Commitment	User, Contributors	Data about different contributors can be analyzed to determine the most and the least engaged users
Project Popularity Over Time	User, Git Users	An analysis of how many other GitHub users interacted with the project repo: forks, favorites, open issues, etc
Development Dynamic Change	User, Contributors	Users may research the frequency and the sizes of commits over time
Contributors List	User, Contributors	Users are able to gather the full list of users who have ever contributed to the project
Developer Feedback	Users, Software Developers	We must have the ability to have communication with the users to help us improve the product in future

System Requirements

Use Case: Project Commitment Data per User

Background

Due to the difficult financial state of the company, the employer decides they want to let some of the employees go and keep only the most active and engaged people on the project. They may define their understanding of the “Most active and engaged” and use the metrics provided by our software to visualize data that will help them to make the tough decision.

Description

The person who needs the analysis feeds the URL to their project’s GitHub repo into our program. After choosing the metrics that they find useful (for example, average number of events per month, or the total number of open and closed issues), they may submit the request and the program will output the data in the appropriate form such as graphs or tables.

Triggers

The user seeks data about their contributors’ activities that would help them highlight the most and least active developers

Actors

1. Employer
2. Employees

Preconditions

1. User provides a URL to the project repo that has enough data provide analytics
2. User has installed all the prerequisites that our program might be using
3. Optionally, then names or the work email addresses are provided to identify employees in the report

Main Success Scenario

Employer receives information about who was the most and the least active contributor to the project and makes the decision based on that

Alternative Success Scenario #1

The output was produced however not all analysis info was available due to the limited amount of data on the repo

Alternative Success Scenario #2

The output was produced for a limited number of employees due to lack of data on them

Failed End Condition

1. Not all prerequisites were installed so data upload or data processing failed
2. Not enough data or insufficient data was provided so there is no adequate analysis available

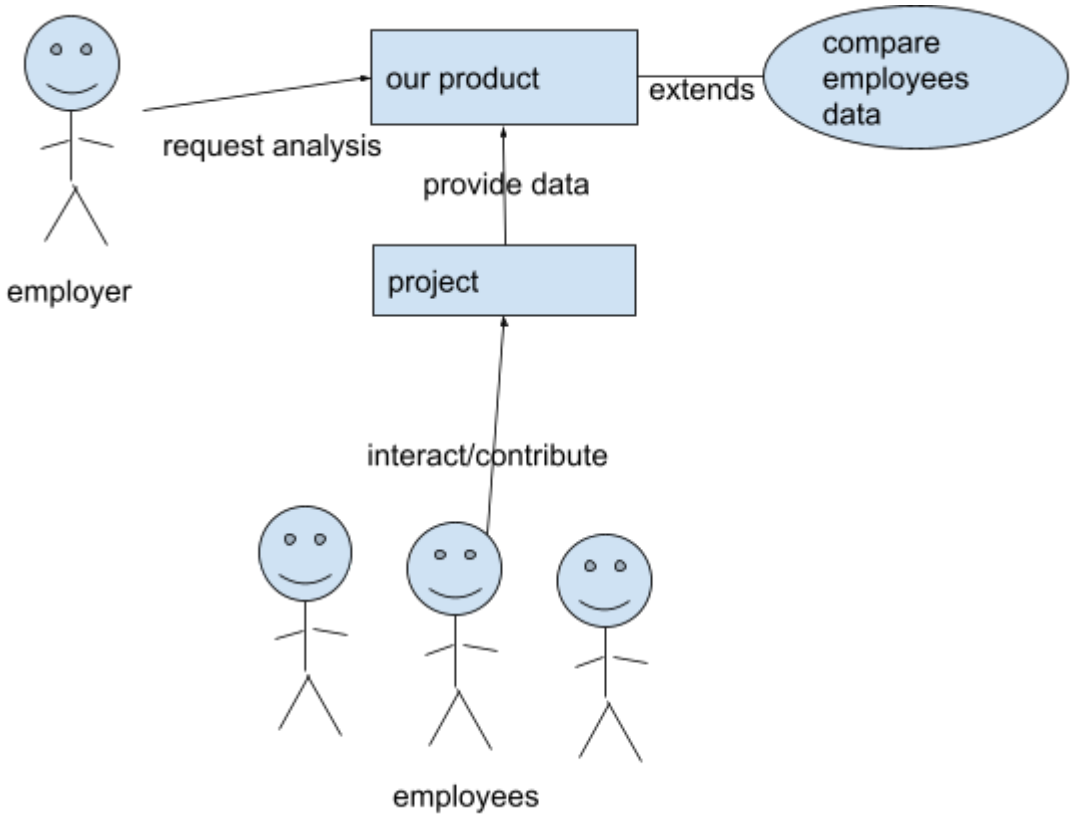
Extensions

Compare data between employees

Steps of Execution

1. Install all required prerequisites on the machine
2. Provide the URL to the Git repo to our program
3. Choose the data and employees whose metrics needs to be analyzed
4. Submit

5. Review the report



Functional Requirements

Priority	Component	Title	Description
1	Report Management	Local Storage	Be able to store the reports locally.
1	Report Generation	Render Data	Be able to receive data reports in the form of tables/graphs
1	UI	Metrics Filters	Ability to choose which metrics are going to be considered in the report
2	UI	Interactive Graphs	Interaction with the graphs in the reports depending on the context. For example, users may wish to change the date range on a graph where X-axis represents dates

2	Feedback	Reaching out to Developers	Have a way to communicate with developers and provide feedback/bug reports
2	Feedback	Updates	Ability to receive or automatically update the library
2	Community Reports	Community Reports Login	Loginning into the online repo with Community Reports
2	UI	Error Output	Show user any errors/warning encountered during the report generation
3	Report Management	Cloud Storage	Uploading the reports to the cloud. Community Repo reports can be viewed by anyone and also allow users to access their reports from any device with internet connection
3	Report Storage	Export Data	Being able to export graphs/tables in the reports as pngs, pdfs, or other formats
3	UI	Recent Reports	A separate menu with a quick access to all previously created reports
4	UI	Themes/Colors	Adjust the appearance of the generated graphs/tables

Non-Functional Requirements

Priority	**Component**	**Title**	**Description**
1	Data Collection	Git API	Access to the Git API that will allow us to read and analyze the repo data
1	Report Render	Graphics Library	A graphics library that will allow us to render graphs and tables
1	Data Collection	Data Anomaly Handling	Ability to handle or tell the user that the data received from the source control was corrupted/incomplete

2	Foundation	System Architecture	An architecture that will allow the use of our library on majority of the biggest OSs
2	Maintenance	Updates	A system that will notify all users of the library about new versions
2	Report Generation	Compression	Compression algorithm for the reports to not take up too much storage space
3	Report Storage	Cloud Storage	A server and database system that will allow us to store community reports in the cloud
3	Maintenance	Logs	An execution/crash log that's running alongside the data analysis algorithm (to simplify future maintenance/bug fixing)

Design Constraints

- Amount of data that can be collected is constrained by Git's API. The development team may find workarounds to collect certain types of data but it is all still very limited to what the API provides
- User privacy question arises when we are collecting data about project contributors (such as the Use Case above). We need to limit the amount of data we collect to remain ethical.
- Depending on which 3rd party library/solutions we choose to build this product, the final library may not be cross platform
- Being an open source project, we have to mostly stick to open source libraries and other dependencies when developing
- The repos we are able to analyze are not only limited to open source projects but to projects uploaded on platforms that provide APIs to access repo data

Possible Components/Hardware that needs to be purchased

- Server to host the Community Reports repo

Interfaces

- Data/UI Interface that will connect the data analysis algorithm results with our graph/table renderer
- Feedback Interface that will be able to connect multiple components of the system that can report how they work to provide better maintenance in future
- Data/Analysis Interface that will connect the databases with raw data with the analysis algorithms