COMP SCI 4ZP6:

HubListener

Requirements Specifications Document

Authored By:
Ahmad, Zed
Jalan, Prakhar
Oliveira, Pedro
Selvathayabaran, Piranaven

Contents

1 Introduction

The following official document is the requirements specification document based on the Volere Template. It contains information regarding project drivers, constraints, functional/non-functional reqiurements and any other critical details that are defined under the templates defintion.The specifications document is subject to change and any modifications will be noted in the Revision History Section.

2 Project Drivers

2.1 The Purpose of the Project

The User Business or Background of the Project Effort
Content: The user of this product would intend on evaluating their GitHub project using our tool, and look for any possible improvements to be made.
Motivation:
The key motivation for this project was our search for the answer to the following questions: What part of a GitHub project is the precursor to success?' What key components do successful projects contain, that unsuccessful projects do not? How do we measure this 'success'? At first, we assumed this measurement to be popularity on GitHub, or a thoroughly and well documented project. We are still in search of what specific metrics lead to success, and hope to discover a solution to this problem as we build and maintain the HubListener.
Considerations The user problem in this case is not serious, as most GitHub projects already use a variety of other tools to evaluate their growth and track their overall progress. However, by the end of the project, the user may clearly notice whether their project was a success or not. This is the moment where our tool can be valued, as it compares popular 'completed' projects to that of the users'. Any significant differences can be highlighted for the user, and thus, the 'success' problem can be solved.
1 Goals of the Project Content
The purpose of HubListener is provide users with relevant metrics, trends, and information regarding their GitHub project, and in some cases, push the user to make appropriate changes (can be organizational) that are intended to lean the project towards a more 'successful' one.
Measurement After analyzing a set amount of projects (i.e. 20), and pointing out any significant/valuable metrics for the user that lead to any changes (minor or major), we can safely say that we have succeeded with the project.

2.2 The Client, The Customer, and Other Stakeholders

The Client: There is no real 'client' for this service. The closest to this would be Dr. Christopher Anand who requires this for completion of the capstone course in exhange for credit towards the developers degree.
The Customer: The consumer of this service is any member of the open-source community. Being that the service is open-source, the consumer definition can range but it is clear from our usability requirements that the service is aimed at adults who have expertise using open-source software and understand not to infringe on copyright.
Other Stakeholders: Other stakeholders include memebers of the open-source community who wish to fork the repository and improve/maintain it after the completion of the project. These stakeholders will be address on a case by case manner.

2.3 Users of the Product:

2. The Hands-On Users of the Product Developers and creators with existing projects on GitHub are our primary user group for this product. Their subject matter experience can range from novice to expert, as we are simply analyzing their repository and comparing it against GitHub's 'best'. Our users can be engineers, students, researchers, companies, and/or any organization looking to evaluate and improve their GitHub project in any way possible.

3 Project Constraints

3.1 Mandated Constraints

This section describes the constrains on the design of the HubListener service. They are the same as other requirements except they are mandatory:

Solutions Constraints:
Describe: The HubListener system shall have a command line interface that has access to the GithubAPI and associated analysis tools.
Rationale: This is central source of the infromation needed.
Fit Criterion: The interface will conform to Github API and javascript standards

Implementaton Environment of the Current System
Describe: The HubListener system shall be written in the NodeJS environment.
Rationale: This is native way of accessing the githubAPI while also setting the system up to become a web based tool.
Fit Criterion: The interface will conform to NodeJs and javascript policies and requirements.

Off-the-Shelf Software
Describe: The HubListener system shall utilize any npm packages that aids in the retrieval and analysis of the repository.
Rationale: This is simplest way of retrieving and analyzing the repository. This will save labour time.
Fit Criterion: The interface will conform to npm and javascript policies and requirements.

Schedule Constraints:
Description: The service shall be available January 31st 2018 as a version 1 release.
Rationale: We want to launch the service by this date so that we can do testing and iterate over the solution so that there is a polished solution for our April presentation.
Fit Criterion: The HubListener serice will be available for testing by January 31st 2018

Budget Constraints: The project has a financial budget at zero dollars.

3.2 Naming Conventions and Definitions

A glossary containing the meanings of all names, acronyms, and abbreviations sed within the requirements specification. The following is a running, ongoing dictionary.

Naming Conventions List	
Naming Convention	Defintion
Github	A web-based version-control and collaboration platform for software developers.
NodeJS	An open source development platform for executing JavaScript code server-side
HubListener	The command-line service that is to be created.
Repository	A digital directory or storage space where you can access your project, its files, and all the versions of its files that Git saves
Command-Line Interface (CLI)	A text-based interface that is used to operate software and operating systems while allowing the user to respond to visual prompts by typing single commands into the interface and receiving a reply in the same way
Cyclomatic Complexity	A software metric, used to indicate the complexity of a program. It is a quantitative measure of the number of linearly independent paths through a program's source code.

3.3 Relevant Facts and Assumptions

3.3 Facts

There are a few factors that influence the product. The first business rule to be addressed is the amount of GitHub repositories we can clone. As we are grabbing several projects for reference directly from GitHub, we are limited to clone at a 'reasonable' pace. Cloning the estimated 100 repositories in parallel is not something GitHub takes lightly, as it can be detected as abusive behavior by their automated measures. Another key factor that affects this product is the specific information we will be extracting from the initial list of projects. This information can vary drastically, and so we will be selective of the types of projects gathered. To elaborate, we may only take projects created in one specific programming language to even the playing field regarding any comparisons to be made.

3.3 Assumptions

There are a list of assumptions made right off the bat regarding this project. To start off, the initial database of projects we will use for comparison, will be that of the most popular and 'successful' GitHub projects. It is difficult to measure success of any given project, without making a set of assumptions first. We would have to assume that the most popular projects on GitHub contain quality code and documentation, as well as a progressive and steady incline in contributions and overall growth. As these projects will be our primary models for reference, it is critical that we address these assumptions. Considering these assumptions, we can also precisely state what our product will not do. As we are aiming to provide quality metrics, we are not intending on 'fixing' project flaws. We are merely laying out relevant information for the user to make their own adjustments; whether they choose to do so, or not. Also, the information we provide may or may not be useful depending on which stage an entered project is in. We are simply comparing and analyzing projects to possibly highlight any significant differences, or none at all. It is up to the user to determine what needs change.

4 Functional Requirements

4.1 The Scope of the Work

The Current Situation: Currently, there is no way to compare github repositories against similar repositories within the ecosystem. There is no way to anaylze your code against similar project or see how your repository is trending in comparison to similar projects within the ecosystem.

4.2 The Scope of the Service

HubListener aims to provide a service which allows the user to compare his/her repository or any open-source repository against similar repositories in the ecosystem. The end user will be able to attain meaningful information that they can use to improve their current repository and guage how they are trending.

Input And Output List	
Input	Output
Github Checkout Link	- Cyclomatic Complexity - Essential Complexity - Integration Complexity - Cyclomatic Density - Lines of Code - Lines of Comments - Maintainability Index - Coupling Metric - Number of Methods - Numbe rof Variables - Number of Issues - Number of Bugs - Number of Stars - Functional Coverage Score - Condition Coverage Score

4.3 Functional and Data Requirements

Functional Requirementts		
Reqt No.	**Reqt Type**	**Description**
1	Functional	HubListener must provide user selected metrics to the end user
2	Functional	HubListener must take in a Gihubt checkout link as input.
3	Functional	HubListener must have a help section as part of command line interface.
4	Functional	Users must be able to customize which metrics to analyze
5	Functional	User must be able to install, update or uninstall HubListener through the node package manager interface.
6	Functional	Metrics gathered from the analysis of a reposiotry must be added into a database
7	Functional	HubListener must analyze the metrics and display one or more trends.

5 Nonfunctional Requirements

5.1 Look and Feel Requirements

1. The application should provide an easy and clear command line interace for user to use the service.
2. The application shall comply with Open-source standards.
3. Useful information(such as help, report issues, training) should be easily accessible.
4. When doing calculations or data handling like repository retrievals, the application should display an animated progress bar.

More to be determined at a later date.

5.2 Usability and Humanity Requirements

1. The software must be simple for a person aged above 18 years, with knowledge of Github/open-source technology, in able condition to understand and use all its features.
2. The application shall make it easy for the average user to find user-use guidelines.
3. The system must meet all open-source software accessibility standards enforced by the gouvernment of Canada.

5.3 Performance Requirements

1. After the user provides their repo link, the applicatoin shall generate charts and statisitcs in a timely manner.
2. The applicaiton shall save the users last request and results.
3. The application shall analyze the results and provide metrics and trends.

5.4 Operational Requirements

Expected physical environemnt

1. Users will use the application on their internet-connected computer

Expected Technological Environemnt

1. The application should work on devices that have Node.JS installed on their machine

5.5 Maintainability and Support Requirements

Maintainability

1. The sotware application is to be easily modifiable.
2. The application should notify user's to check for an npm update every 6 months.
3. The application shall be ready to be deployed on any OS.

Portability

1. The application shall be availble on any computer operating system such as MAC, Windows and Linux

5.6 Security Requirements

5.7 Cultural and Political Requirements

1. The application should not display any offensive text or information
2. The application should be available in English.

5.8 Legal Requirements

1. The application shall comply with all relevant information privacy acts.
2. The application shall comply with all relevant open-source laws.
3. The application will abide by all developr guidelines as denoted by Windows, Apple, etc.

6 Project Issues

6.1 Open Issues

All open issues can be see on our Issue tracking board. (HubListener on ZenHub)

6.2 Off-the-Shelf Solutions

Attempting to emulate similar applicaiton could greatly reduce the time needed to design and implemnt HubListener. If an off the shelf solution is available and fit our reqiurements, we will use it as part of the software, providing the necessary credit as needd.

6.3 New Probelms

6.4 Tasks

All open taks can be seen our Issue tracking board (HubListener on ZenHub)

6.5 Migration to the New Product

No new product to migrate to. This section may become obsolete.

6.6 Risks

6.7 Costs

There are no costs associated with the development of this project other than the time dedicated to development/documentation. It is developed under the open-source environemnet and therefore is useable for not-for profit purposes. In future, there may be a cost associated with maintaining the project, hosting the project or use by a professional company.

6.8 User Documentation and Training

User documentation will be available on the Github Wiki as well as on the npm description section.
Training will stem from this wiki and any additional information, changes made will be reflected in this document at a later point

7 Revision History

Table 1: Revision History

Date	**Developer(s)**	**Change**
November 1st, 2018	Piranaven Selva	Make Foundation for Specifications Document as per Issue #6