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 requirements and any other critical details that are defined under the templates definition. 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.

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 exchange for credit towards the developer's 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 members 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:

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:

Schedule Constraints:
Description: The service shall be available April 1, 2018.
Rationale: We want to launch the service by this date as it is the date that the project is due for assessment.
Fit Criterion: The HubListener service will be available for evaluation April, 1 2018

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

3.2 Naming Conventions and Definitions

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

Naming Conventions List	
Naming Convention	Definition
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.
Node Package Manager (NPM)	is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry.
Metric	A software metric is a standard of measure of a degree to which a software system or process possesses some property.

3.3 Relevant Facts and Assumptions

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.

Assumptions

There are a list of assumptions made right off the bat regarding this project:

1. The initial database of projects we will use for comparison, will be that of the most popular and 'successful' GitHub projects
2. The most popular projects on GitHub contain quality code and documentation
3. NPM packages used as dependencies are accurate, complete and correct.
4. Each user will have their own GitHub API Authentication token.

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 analyze 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 gauge how they are trending.

4.3 Functional Requirements

Functional Requirements			
Reqt No.	**Reqt Type**	**Description**	**Fit-Criteria**
1	Functional	HubListener must provide user selected metrics to the end user	HubListener logs shall match the user input and the provided metrics.
2	Functional	HubListener must take in a GitHub checkout link as input.	HubListener logs should validate that the input link is a GitHub checkout link.
3	Functional	HubListener must have a help section as part of command line interface.	HubListener logs should validate that the main output includes a help section.
4	Functional	Users must be able to customize which metrics to analyze	HubListener will run with one metric selected and all metrics selected
5	Functional	User must be able to install, update or uninstall HubListener through the node package manager interface.	HubListener logs should validate a npm install, update and uninstall have succeeded when called upon.
6	Functional	Metrics gathered from the analysis of a repository must be added into a database	After HubListener has done an analysis, the results are view able in the database. Logs validate an addition to the database.
7	Functional	HubListener must analyze the metrics and display one or more trends.	HubListener logs at least one trend.

Functional 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 - Number of Variables - Number of Issues - Number of Bugs - Number of Stars - Functional Coverage Score - Condition Coverage Score

5 Non-functional Requirements

5.1 Look and Feel Requirements

1. The application should provide an easy and clear command line interface for user to use the service.
   (a) **Fit-Criteria:** 80% of the users from our survey shall be able to navigate through the interface and utilize the service. The list of available options should be available on the main screen.
2. The application shall comply with Open standards.
   (a) **Fit-Criteria:** 100% of users in our survey can verify that the application is easy to accessadopt and open for public review and debate.
3. Useful information (such as help, report issues, training) should be easily accessible.
   (a) **Fit-Criteria:** 90% of users in our survey with knowledge of GitHub will be able to access areas for help, reporting issues and training.
4. When doing calculations or data handling like repository retrievals, the application should display an animated progress bar.
   (a) **Fit-Criteria:** 100% of users in our survey can successfully verify that the application displays an animated progress bar

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.
   (a) **Fit-Criteria:** 90 % of the users in our survey are able to use the application and all its features and deem the application understandable.
2. The application shall make it easy for the average user to find user-use guidelines.
   (a) **Fit-Criteria:** 80 % of user in our survey are able to successfully find the user-use guidelines within one minute.

5.3 Performance Requirements

1. After the user provides their repository link, the application shall generate charts and metrics in a timely manner.
   (a) **Fit-Criteria:** 80 % of the users in our survey agree that the charts and metrics were delivered in a timely manner.

2. The application shall save the users last request and results.

    (a) **Fit-Criteria:** 100 % of users in our survey, after their first request, are able to navigate to the history section where they can view their last request and results

3. The application shall analyze the results and provide metrics and trends.

    (a) **Fit-Criteria:** 100 % of the user in our survey are able to successfully identify at least one metric and one trend after running the application.

5.4 Operational Requirements

Expected physical Environment

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

    (a) **Fit-Criteria:** 100 % of the user in our survey are able to run the application on their internet-connected computer.

Expected Technological Environment

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

    (a) **Fit-Criteria:** 100% of users in our survey are able to run the application when they have Node.JS installed.

2. The application shall work on devices that have an internet connection.

    (a) **Fit-Criteria:** 100% of users in our survey are able to run the application when they have an internet connection.

5.5 Maintainability and Support Requirements

Maintainability

1. The software application is to be easily modifiable.

    (a) **Fit-Criteria:** 80 % of the users in our survey are able to clone the repository and make modifications to the source code.

2. The application should notify user's to check for an npm update every 6 months.

    (a) **Fit-Criteria:** 80% of the user's in our survey will be notified in six months time, to update their npm packages.

Portability

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

    (a) **Fit-Criteria:** 100 % of users in our survey are able to run the application on their device irrespective of their operating system. The survey ensures that there is at least one user on all three operating systems defined.

5.6 Security Requirements

5.7 Cultural and Political Requirements

1. The application should not display any offensive text or information

    (a) **Fit-Criteria:** 100% of the users in our survey do not report any offensive text or information.

2. The application should be available in English.

    (a) **Fit-Criteria:** 100% of the users in our survey are able to read the application main page and successfully determine that the language is English.

5.8 Legal Requirements

1. The application shall comply with the PIPEDA privacy act.

    (a) **Fit-Criteria:** The application follows all PIPEDA rules as defined in the link.

2. The application shall comply with all relevant open-source laws.

    (a) **Fit-Criteria:** The repository is Licensed under GPL-3.0 which abides by the regulations set by the Free Software Foundation

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 application could greatly reduce the time needed to design and implemnt HubListener. If an off the shelf solution is available and fit our requirements, we will use it as part of the software, providing the necessary credit as needed. Below are the off-the-shelf solutions we are looking at right now:

1. NodeDir
2. NodeGit
3. tmp
4. Simple GraphQL Client
5. Complexity Report
6. JSLint

6.3 New Problems

N/A

6.4 Tasks

All open tasks 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 is currently not applicable, but included for completeness.

6.6 Risks

All projects have risk. HubListener is no exception. Below are the risks identified and strategies as to how we will mitigating these risks:

Risk Table		
Risk No.	**Risk Description**	**Mitigation Strategy**
1	Inaccurate Metrics	As stated in the assumptions, we believe that the npm packages provided will be accurate. To ensure this, we will have test cases that verify the correctness of the npm package. Furthermore, any metrics coded by our team will be available to the open source community, such that if there are any problems in the calculations, issues on GitHub will be created and dealt with by the project team or the community.
2	Inadequate Metrics	HubListener strives to provide as many metrics as possible such that the stakeholders have as much information as possible. If there are any missing metrics which are required by the community, feature requests can be made on our GitHub. The requests will be resolved as per open-source policies.
3	Excessive Schedule Pressure	Although, HubListener is a time-constrained project, the HubListener team is governed by a supervisor and course instructor which will enforce that the project be completed on time. These quality gates ensure the milestones are delivered in a timely manner. Any changes in scope will be documented and relayed to the appropriate stakeholders.
4	Performance	HubListener performance is based on many variables. Benchmark and threshold testing results will be provided to the end User such that they can understand how this variation can occur.
5	Unproven Technologies	Although, the software is new in nature , it encompasses many proven technologies. Any changes or updates in the technologies will be relayed to the stakeholders.

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 environment and therefore is usable 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
November 23rd, 2018	Piranaven Selva	Add Fit-Criteria to Functional/ Non-functional Requirements as per Issue #30
November 24th, 2018	Piranaven Selva	Document: Constraints And Design Choice And Off-The-Shelf Solutions as per Issue #33
November 26th, 2018	Piranaven Selva	Maintenance