Requirements Specification

Git OSS-um

Gary Baker
Van Steinbrenner
Stephen White

December 6, 2018

Project Sponsor: Dr Igor Steinmacher,
Assistant professor, SICCS, Northern Arizona University

Faculty Mentor: Ana Paula Chaves Steinmacher
Version 2.0

Accepted as baseline requirements for this Project:	
Client:	Team Lead:

Table of Contents

1. Introduction

Open-Source Software (OSS) is a type of computer software in which source code is released under a license, where the copyright holder grants users the rights to study, change, and distribute the software to anyone and for any purpose. OSS may be developed in a collaborative public manner using a version control system named Git and hosting the project itself on a site known as GitHub. This version control system tracks changes made to a project and easily allows projects to have multiple contributors. It also allows for a project owner or manager to accept or decline contributions depending on their opinion of the quality of the change through what is called a pull request.

Unfortunately, a substantial amount of developers do not contribute to open-source projects. We have all been there: we don't know where to start, what files are important, or what needs to be changed. The community may be friendly and quick to provide assistance to a newcomer, or the contributor may get berated for submitting a pull request that does not adhere to the project's standards. It is so daunting, that the newcomer gives up, and moves on. This experience shows that there is a need in the open-source community for a tool that will allow a novice passing through to quickly and efficiently determine the newcomer "friendliness" of a repository.

Our client, Dr. Igor Steinmacher, is a researcher and assistant professor at Northern Arizona University. Dr. Steinmacher and his colleagues have created a web application called FLOSScoach[1] to address this need for a tool to help newcomers, where a user can investigate the skills needed to contribute to open-source. Although FLOSScoach has been helpful, it has its shortcomings, in which will be discussed in a later section.

[1] FLOSScoach: http://www.flosscoach.com

Due to the shortcomings of the current solution Dr. Steinmacher and his colleagues have come up with, we will be spearheading our own solution, independent from FLOSScoach. Our job is aimed at giving power back to the newcomer by creating an autonomous process where relevant data of a requested repository is presented to the user. This will allow them to decide if the specific OSS project will support their needs and help them feel at ease when picking their first open-source project.

In order to develop a successful newcomer meter for open-source dynamics, we will create a web application that will take a URL leading to a GitHub project page, which will then be accessed via the GitHub API. Desired data such as newcomer acceptance rates and newcomer retention, among various other data points will be collected. This data will be used to assist users in finding projects, and interactive graphics will demonstrate to the newcomer what OSS projects are best for them.

Within this document, we will break down our envisioned solution into the domain-level, functional, non-functional, and environmental requirements associated with our client's vision. We will be discussing the potential risks involved in building our software, and will describe how these risks will be mitigated. We will end with a brief discussion of the project timeline to outline our plan for development. In the following section, we will describe the problem statement and explain in greater detail the need for a newcomer meter for open-source software.

2. Problem Statement

As open-source grows in popularity, more developers may wish to contribute to projects; however, contributing to OSS presents its own difficulties. As mentioned in the previous section, Dr. Steinmacher and his colleagues have created a web application called FLOSScoach, where newcomers can find the skills needed to contribute to an OSS project. Despite this, FLOSScoach does have its flaws, including:

- Manually fed information
- Must be constantly maintained
- No interactive graphics

These limitations restrict use of the product to already available information and does not assist with finding new projects to contribute to. Perhaps the biggest flaw of FLOSScoach is the manual input of projects to view. This makes the number of projects that newcomers can view very little. Because of the manual input of information, FLOSScoach must be maintained constantly. The absence of interactive graphics from a research area that needs a lot of data is a big fault, as the application will have a less powerful way of letting the user know how the specific project is compared to another one. We aim to fix these three issues into a more dynamic solution.

3. Solution Vision

We will be building a dynamic web application for newcomer characterization that features data mining from GitHub and interactive data visualizations that the user can explore to form their own ideas of what project they want to contribute to. The web application will allow users to store the mined data and compare repositories for further analysis. Below is a list of

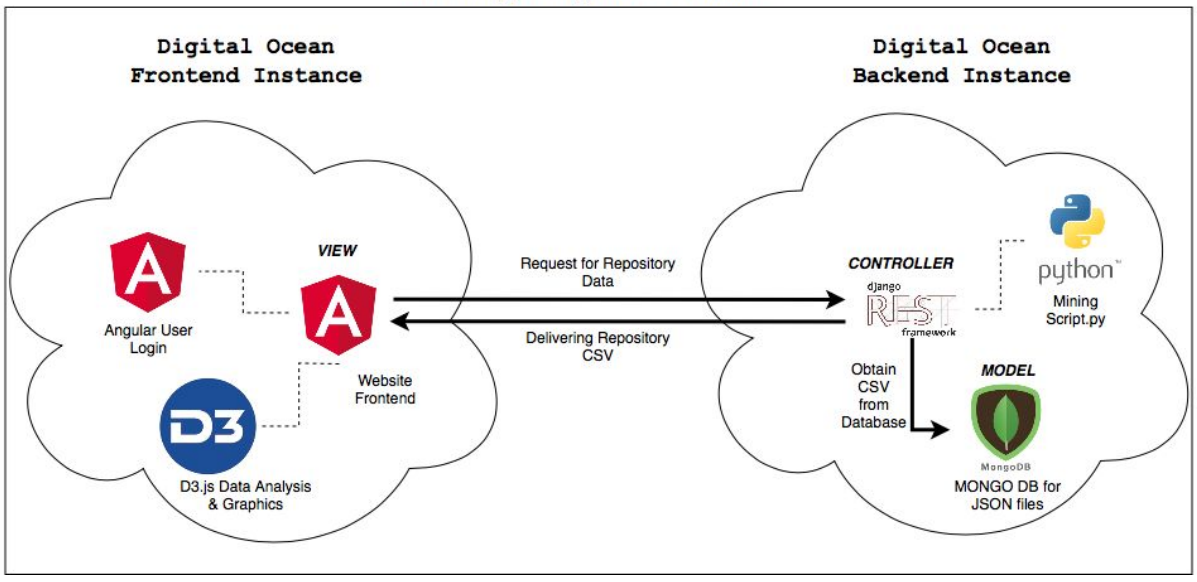
features that our web application will have to assist the newcomer and Dr. Steinmacher's research:

- Search for a specific repository
- Compare stored repositories, up to three at a time
- Explore interactive graphics and raw data
- Admin access to site
- View stored repositories and edit the repo status

The mined data is stored as JSON files, or JavaScript Object Notation, and are obtained from the GitHub API. From the collected JSON files, we will analyze the data and form interactive graphics and visualizations so that users can explore the data and formulate their own decisions for what project they want to contribute to.

We are hoping that our product will change the world of open-source software. With an application such as this, where users can dynamically request repositories to get mined, see interactive graphics, and compare repos, we can change how newcomers can contribute to projects.

Technology Integration Workflow



Above is an architectural diagram of our front and back-end technologies. For our front-end, we mainly use Angular for front-end work with D3.js for our data visualization. For our back-end, we have a Django REST framework that works with our mining script and database to pull data for use at the front-end. Our Technology Integration Workflow figure above highlights our technologies and requirements as well.

4. Project Requirements

In order to obtain the proper domain-level requirements and continuously refine them into much more specific requirements, our team has been meeting with Dr. Steinmacher once per week for the past three months. During the first month of meetings, the main focus was understanding the problem and learning from Dr. Steinmacher about the hole that exists in Open-Source Software that is “Newcomer Support”. The second month of meetings consisted of many idea sharing and brainstorming sessions to come up with the key features as well as “nice-to-have” features, allowing us to formalize them into software requirements.

In our last month of meetings, we proposed a list of use cases to Dr. Steinmacher that included potential requirements and asked him to choose any use cases that he wanted to be included in the final product. After collecting the checked list of requirements, we then asked Dr. Steinmacher to give a priority rating for each requirement. This would allow us to focus our initial effort on the most important features in order to create a useful prototype by the end of the semester as well as know where our focus should be going into the true implementation phase.

The outcome of our requirements acquisition with Dr. Steinmacher can be narrowed down to two main functional requirements with many sub-requirements, three non-functional requirements, and two environmental requirements. The two functional requirements are **FR-1**: An End-User can login and view data for a specified repository on GitHub, and **FR-2**:

Administrative Management Capabilities. The three non-functional requirements are **NFR-1**: Response times within usable range, **NFR-2**: Ease of use, and **NFR-3**: Scalability. Finally, the environmental requirements are **ER-1**: Web application hosted on linux based server, and **ER-2**: Optimized for desktop and laptop use. In the upcoming sections, we will be discussing these in much greater detail in the form of sub-requirements to the higher level of functional, non-functional, and environmental previously mentioned.

4.1. Functional Requirements

The set of functional requirements are the heart of our product. Functional requirements basically outline exactly what the main features the application must have in order to meet the expectations of our client from an application functionality stand-point. We have broken down the many features requested by our client into the following hierarchy of functional requirements:

- **FR-1**: End-User can login and view data for a specified repository on GitHub.
    - **FR-1.1**: Search application database for a repository.
    - **FR-1.2**: Request mining of repository data if not already in database.
        - **Fr-1.2.1**: Decline mining request and notify the End-User who made the request.
        - **FR-1.2.2**: Accept mining request, retrieve data from GitHub, and store it in the database.
        - **FR-1.2.3**: Notify End-User and Admin that mining and storage is complete.
    - **FR-1.3**: Analyze data and display graphics.
        - **FR-1.3.1**: Compare multiple repositories.

- **FR-2**: Administrative Management Capabilities
    - **FR-2.1**: View and edit the application database.
    - **FR-2.2**: Pause and resume mining of unfinished repositories.
    - **FR-2.3**: Configure database updates.
    - **FR-2.4**: Grant other users admin rights.

We have these specific requirements in order to establish a strong understanding between our team and Dr. Steinmacher about what will be expected in the final product. In the following pages, we will be discussing each functional requirement in greater detail so that readers can understand how these connect with our domain-requirements.

FR-1: End-User can login and view data for a specified repository from GitHub

User Profile: End-User

Action to Trigger: Login to application and search for a repository

Description:

The very first step for an End-User to make use of our product is to create an account and login to our application. If the login credentials entered are invalid, the End-User will be notified of this and given the opportunity to enter the correct credentials; if they enter valid credentials, they will be taken to the homepage of the application. From that home page, they will be able to navigate to any other part of the application that is desired. In order for the End-User to start looking for the right repository for them to contribute to, they will navigate to the search page using the navigation bar at the top of any page in the application.

FR-1.1: Search application database for a repository

User Profile: End-User

Action to Trigger: Select filters or search for a specific repository

Description:

Once an End-User navigates to the search page of our application, they will be shown a search bar where a repository name can be entered if one is known. Along with the ability to search for a specific repository, they will also be able to select filters if they are unsure about what exact repository they would like to search, but would like to find repos that fit their strengths. The filters that will be available are programming languages, type of license, project size, project age, issue labels, and presence of a code of conduct. The search results will appear below the search bar and high priority filters such as programming languages and type of license will be selectable on the left-hand side of the screen as well as the remaining filters in a menu below the search bar. Below in Figure 1 is a mock-up displaying the intended functionality of search in a visual representation.

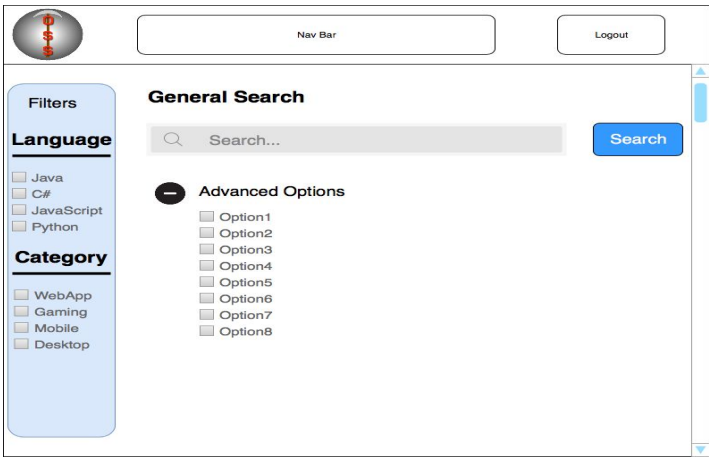


Figure 1: General Search Mock-up.

If an End-User searches for a specific repository and there are no search results within the application, they will be notified via on screen message where the search results normally appear saying that the repository searched is not in the database. This on screen message will also give the End-User the option to request that the data for that repository be mined from GitHub by the admin in order for the desired graphics and information to be generated and displayed for the End-User to view.

FR-1.2: Request the mining of a repository if not in the database

User Profile: End-User

Action to Trigger: No results returned for a searched repository and the End-User chooses to request that an admin mines the data from GitHub.

Description:

In the event that the repository that an End-User searches for is not already in the application database, they will then be able to request that the admin of the application mine the appropriate data from github and store it in the application database. Upon submission of a mining request, there are to potential outcomes for the End-User, their request will either be accepted or declined. These two outcomes are explained further under section **FR-1.2.2 - FR-1.2.3** for accepted requests and **FR-1.2.1** for declined requests. When a request is sent, all admin users will receive a notification within the application that a new mining request has been made. It is up the the admin from that point to decide whether or not that repository should be added to the application database.

FR-1.2.1: Decline mining request and notify the End-User who made the request

User Profile: Admin

Action to Trigger: Repository mining request has been submitted by an End-User.

Description:

In the case of a repository mining request being denied, the End-User who submitted the request will receive a message telling them that the request was denied and that they can reach out to the admin via internal messaging system within the application for an explanation as to why it has been denied. This requirement is more for the prevention of spamming of data mining rather than there being many repositories out there that will be declined but in some cases there may be something requested that contains inappropriate content that the admin may not want to accept to be shown on the application.

FR-1.2.2: Accept mining request, retrieve repository data from GitHub, and store it in the database

User Profile: Admin

Action to Trigger: Repository mining request has been submitted by an End-User.

Description:

In order for our application to mine data from GitHub, End-Users will submit a request to the admin for a specific repository to be mined (see **FR-1.2**). As an admin of our application, upon logging in there will be all the same options as a regular End-User as well as an admin dashboard page (admin details are laid out under section **FR-2**). From their dashboard, the admin will view any requests for mining that have been made and have not been closed yet. Upon viewing the request, the admin will have the option to accept or decline the mining of the requested repository. If the admin accepts the request, the application will then go out and mine data from GitHub. The data mined for any repository will be a fairly large set of points stored by GitHub for all projects created. Some of those data points that will be mined are: repository name, description, date created, language, number of forks, number of watchers, license name, and number of open issues. There will also be data points mined from the individual pull requests within a repository such as: PR number, state, title, user who submitted, body of request, date created, date closed, date merged, labels, comments, commits, additions, deletions, and changed files. Once all the data is collected for the accepted repository, that data will then be stored in the application database for the ability to use later on.

FR-1.2.3: Notify End-User and Admin that mining and storage is complete

User Profile: Admin and End-User

Action to Trigger: Repository mining request has been accepted and completed.

Description:

Once the data from a requested repository is mined and stored in the database, the admin as well as the End-User who requested this data will be sent a message within the application notifying them that the repository has been mined, stored, and is ready for use. At this point, the End-User who requested the data, and any other End-User, can now search for the specific repository on the application and get the desired information and graphics returned to them since the repository is in now in our database. More information on the information and graphics returned to an End-User explained in section **FR-1.3**.

FR-1.3: Analyze data and display graphics

User Profile: End-User

Action to Trigger: An End-User has selected a specific repository to view.

Description:

The number one way that our client wants this data to be shown to the users is in the form of interactive graphics. These will be charts that display certain information that should be helpful to the user in deciding whether or not this repository is right for them. The graphics will have the ability to be adjusted by performing actions on specific parts of the graph to show a particular section or data point. In order to maintain ease of use, the graphs that will be used will be simple bar charts and line charts that include interactive elements. There will be four graphics generated for a repository when viewed, those graphics will

analyze the number of newcomers per month, the number of pull requests opened, closed and merged per month, retention rate of contributors per month, and acceptance rate of pull requests by newcomers vs. that of return contributors per month. These graphics will be displayed on a page that is reached by selecting a repo in the search results of our application. They will be laid out in rows of two in order to ensure that there is no cluttering of the screen and to keep the graphics large enough to read as they are displayed.

FR-1.3.1: Compare multiple repositories

User Profile: End-User

Action to Trigger: An End-User has selected two to three repos to compare side-by-side.

Description:

A great way for a user to figure out what repository would be a good fit for their first contribution to open-source, they should be able to conveniently view multiple repos side-by-side for quick and easy comparison. In order to keep the comparison page clean and easy to read, there will be a limit of 3 repositories per comparison. The repos will be set up side-by-side with their name at the top followed by the interactive graphics that belong to each repo ordered in columns below, and at the very bottom of the page will be a table where the repositories are compared based off of mined and analyzed data points such as the repo name, repo age, number of stargazers, language, license key, number of open issues, number of watcher, newcomer retention rates, number of contributors, and whether or not the repo has a

CONTRIBUTING.md. Below is an example of how this comparison page may look.

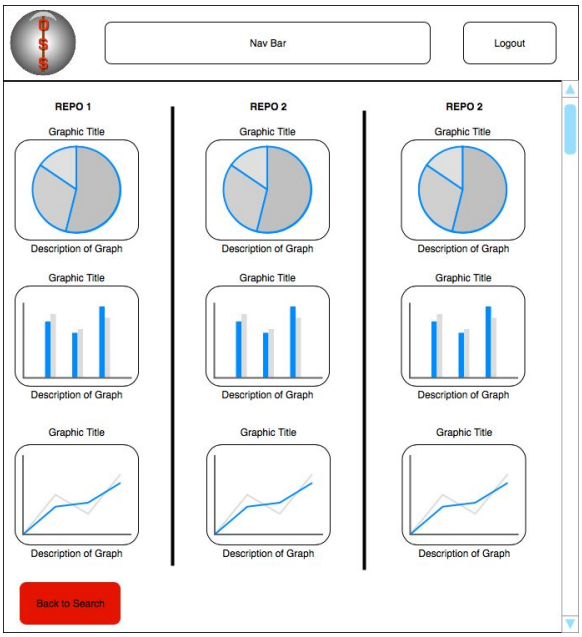


Figure 2: Repo Comparison Top of Page.

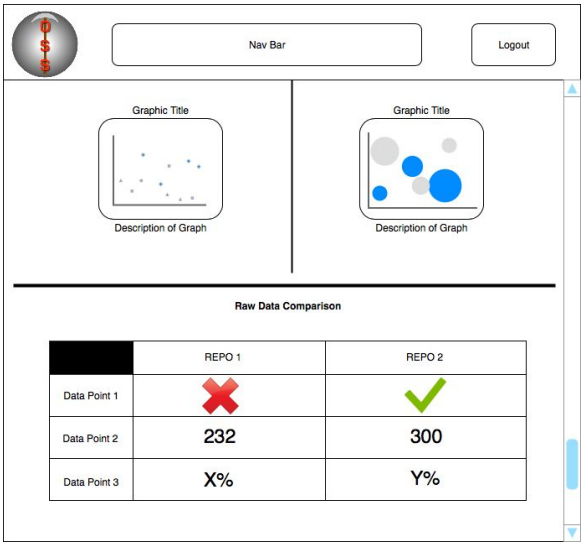


Figure 3: Repo Comparison Bottom of Page.

FR-2: Administrative management capabilities

User Profile: Admin

Action to Trigger: An admin user logs in with correct credentials and accesses the admin dashboard.

Description:

In order for our application to have a strong ability to be managed properly, there must be a level of control higher than that of the End-User who is looking to use the application for gaining information about a repository. With that being the case, there will be two types of accounts, End-Users and Admin Users. When an admin user logs into the application, they will have access to the admin dashboard from the regular homepage which regular End-Users will not. Within the admin dashboard, there will be a page to view and edit the database (**FR-2.1**), pause/resume repository mining (**FR-2.2**), configure database updates (**FR-2.3**), and grant other users admin rights (**FR-2.4**).

FR-2.1: View and edit application database

User Profile: Admin

Action to Trigger: An admin user navigates to the manage database page from the admin dashboard.

Description:

From the admin dashboard, the admin user will be able to navigate to the database manager page. Here the admin will have a couple options. First is the ability to view the entire contents of the database. The other main function of this page is to edit the contents of the database. With editing rights, the admin will have the ability to select one or more repositories in the database and remove them completely. This action is permanent and can not be undone without re-mining the data that has been deleted, this being the case, there will

be a confirmation window that pops up before the deletion takes place to prevent any accidental actions. In order to delete a repository from the database, the admin will need to select the unwanted database and select the delete option located on the bottom of the page, at this time the confirmation window will appear and the admin will then select the confirm option within the window and the deletion will take place. Upon deletion of anything from the database, the manage database page will refresh and display the updated contents where the deleted repositories are no longer present.

FR-2.2: Pause and resume unfinished repositories

User Profile: Admin

Action to Trigger: An admin user navigates to the manage requests page from the admin dashboard.

Description:

In some cases, the mining of a repository may need to be stopped for a certain amount of time and resumed later on. This is a functionality that can be accessed from the manage requests page of the admin dashboard in which the full list of requests are displayed showing the repository name, the link to the repo, the End-User who requested it, the status of the request, and the acceptance tag for the request. The admin user while on this page will have the ability to select a repository request with any of the four status tags: open, mining, paused, or closed where open means the request has been received but not accepted or declined yet, mining means that the specified repository is currently being retrieved from GitHub, paused meaning that the mining has been paused for that repository request, and closed means that the repository request

was either accepted and stored in the database or declined and not mined at all. These requests on this page will also have a flag called the acceptance flag that tells the admin that if the request has been closed, whether it was accepted or declined that way it is clear at a glance what the outcome of the request was. The requests will be laid out in somewhat of a tabular format with the ability to search as well as set filters to narrow down the displayed requests to a certain criteria. The filters will include their status tag (open, mining, paused, closed) as well as their acceptance tag (accepted, declined). These tags will be set for each request at the time of a status change and will be set as open by default when the End-User sends the initial request. If a request is accepted and mining begins, the status will update on its own to be mining and pausing the mining of a request will automatically switch its status to paused. The closed tag will be applied in the event of either a request being declined and having the acceptance tag be set to declined as well or when the mining and storage of the requested repository is completed and setting the acceptance tag to accepted.

FR-2.3: Configure database updates

User Profile: Admin

Action to Trigger: An admin user navigates to the configure database page from the admin dashboard.

Description:

Since the data for a single repository will only be mined once when an initial request is accepted and repositories are always changing on GitHub, there needs to be a way to keep the data in our database current and accurate. In order to do this, the admin of our application will have the option to navigate

to the configure database page from the admin dashboard. From this page, the admin user will be able to adjust the frequency of updates that take place on the database, whether it be once per day or once per year, the admin will have the flexibility to configure the updates to the frequency they feel necessary. When these updates occur, the application will go out and compare the data in our database to the repositories GitHub data and update anything that is different. This process will basically be re-mining the data and with the file that is returned by mining, the application will be able to compare that file side by side to what is in the database and automatically update the database as needed.

FR-2.4: Grant other users admin rights

User Profile: Admin

Action to Trigger: An admin user navigates to the grant access page from the admin dashboard.

Description:

As the size of the application itself as well as the user-base of the application increases, there will surely need to be more than the initial one admin that there will be at launch. Dr. Steinmacher being the only initial administrator, he would like to grant admin access to other trusted users to help in the management of the application. This will not only free up Dr. Steinmacher to handle different activities but it will also allow End-Users to get better response times from admin on mining requests and any other contact with the admin since there will be more of them to handle requests and messages in order to issue responses. The admin users of the application will be able to grant these admin rights to other users by navigating to the grant access page

from the admin dashboard where there will be the option to enter a user's email that was used for creating the account and a confirmation that the specified user is to be upgraded to an admin of the application. Dr. Steinmacher will have a slightly different experience in this page. Being the product owner, he will have the option to choose what type of account that the user is to be set as, Admin or End-User. This will allow him to remove admin rights if necessary as well as granting them. The ability to remove admin rights is exclusive to Dr. Steinmacher to prevent any other admin from being able to remove the admin access of others.

4.2. Non-Functional Requirements

As you can see from the section above, functional requirements are all about "what" the application will be doing. Non-Functional requirements on the other hand are all about "how well" the application does those things. As far as our client is concerned, there are three main non-functional requirements to be met which are laid out as follows:

- **NFR-1**: Response times with all data within one minute.
    - **NFR-1.1**: Displaying table of raw data within 5-10 seconds followed by graphics as they are generated.
- **NFR-2**: Easy to use.
    - **NFR-2.1**: Layout of application is easy to follow and quick to access desired pages.
- **NFR-3**: Scalability.
    - **NFR-3.1:** The ability to handle a growing amount of tasks from users and other requirements.

In the following sections, these non-functional requirements will be explained in detail to provide a more in-depth look at what the non-functional requirements will cover and how they will be handled.

NFR-1: Response times with all data within one minute

NFR-1.1: Displaying static table of data within 5-10 seconds followed by graphics as they are generated

Our web application will be generating interactive graphics for the user to explore, however, an initial table that displays data regarding the repository will be shown. We have chosen this form of representing data because the initial creation of our graphics may be rendered slowly and the table will show the user the overall basics of the data represented. The table, overall, will be more of a reference to the user before the graphics are shown.

NFR-2: Easy to use

NFR-2.1: Layout of application is easy to follow and quick to access desired pages

Aas this application assists developers and newcomers in finding the skills needed to contribute to a project, this application must be easy to use and understand. In particular, navigation and layout of the pages must be easily interpreted by users. A confusing layout will limit the amount of people using our application and a disorganized navigation system will be a strenuous task for users to get where they need to be. An organized navigation system and page

layout will make a user friendly experience and allow them to access the information and services that the users need.

NFR-3: Scalability

NFR-3.1: The ability to handle a growing amount of tasks from users and other requirements

As our web application grows and more users are joining the app, we must be prepared for taking on more work. This is where the abilities of our web hosting service, Digital Ocean, come in. Digital Ocean offers scalable and management functionality so that admins can adjust the storage of mined data and the process of mining that data. It is important for us to have this feature for the application so that it can grow and adjust to a growing amount of users, and, in turn, have more storage for mined data. Administrators of the application can manage the the scalability as well so that they can adjust the app to the demand of users. Scalability is a high priority as our application begins to grow with both users and increased tasks.

Our non-functional requirements provide the backbone to the rest of our requirements. From an easy to use interface to a scalable server that can adjust the needs of our web application's users and needs of further tasks. Now, we will move forward to our environmental requirements that will affect our web application.

4.3. Environmental Requirements

Our environmental requirements are a similarly short list to the non-functional requirements. According to Dr. Steinmacher, this application has two environmental requirements in which are broken down as follows:

- **ER-1**: Desktop and laptop optimization.
    - **ER-1.1**: Usable on mobile, not optimized.
- **ER-2**: Hosted on Linux-based web application.
    - **ER-2.1**: Digital Ocean web hosting service.

Below are the more detailed looks at our environmental requirements and how they fit into our product.

ER-1: Desktop and laptop optimization

ER-1.1: Usable on mobile, not optimized

Our client is not particularly concerned with mobile use of the application at this time, therefore, mobile implementation is not a requirement but it has been stated that the application should function on mobile, just not specifically optimized for mobile.

ER-2: Hosted on Linux-based web application

ER-2.1: Digital Ocean web hosting service

As described above in our Non-Functional Requirements, Digital Ocean will be handling scalable solutions to users of our web applications, however, our web hosting service is capable of so much more. From scalability and management, Digital Ocean offers an interface to monitor data and a secure service where data will be safe. Digital Ocean is a hosting service with many

solutions and has the documentation to assist us and administrators manage and monitor the data that users mine.

Now that we have our requirements sorted out, it's time to talk about some potential risks that may evolve into actual issues with our application in the future if neglected.

5. Potential Risks

Throughout the requirements acquisition phase, the team has identified and considered four main risks. The first risk involves displaying information in a way that does not present an opinion of a repository, while the remaining three potential risks related to GitHub itself.

5.1 Presenting Facts, Not Opinions

From the very beginning of this project, the team spoke to our client about how information will be presented to the end user, and what this information would look like. After an iterative process, we have determined that it is in the team's best interest to provide soley facts about a repository, and to stay away from presenting opinions. The reason for this is quite simple, as rating a repository as good or bad, five stars or 1 star, or any other variation of this kind could actually prevent newcomers from attempting to contribute to a repository that would be a great fit for them. It should also be known that what one user might find newcomer friendly is not necessarily going to be the same for another, therefore it is best for the team to avoid providing that type of analysis altogether.

As far as the facts that we will be displaying to our users is concerned, the team has met with the sponsor on numerous occasions to determine what facts will be summarized in our graphical representations of the data we have collected. Examples of this data include newcomer retention rates, and the length of time that a newcomer typically waits for their pull request to get accepted. In doing this, we hope to avoid labeling any repository in a negative

away, and allow users to make an informed decision for themselves as to whether they would like to contribute to a given repository.

5.2 Maintaining Correct Information

Considering GitHub contains an ever-growing number of projects and innovators, there is always the possibility that a project owner may wish to change the name of their repository to better fit what it is that they are working on. When this happens, GitHub updates their API to reflect the new project name, and if the team were to search for it, we would be returned an error. This represents the risk of having out-of-date information being stored in our repository database, regardless of how minimally people do change their project names. The severity of this error would be handled and the user will be given an error message to tell them what has gone wrong. In order to mitigate the risk of having incorrect information in our database, we aim to produce a robust administrator portal that will allow an admin to remove all instances of the incorrect repository name throughout our database so the new, correct information may be collected and stored. By having an up-to-date database, we reduce the possibility of presenting incorrect information to the end-user, and build trust with the community we hope will use our tool.

5.3 Changes to the API

Very recently, the Microsoft Corporation acquired GitHub, and considering the transition is still occurring, there is always a possibility that Microsoft may make changes to the GitHub API to better suit their business needs. This API change will be destructive to our application to the point of where we would have to rely on old information, if we don’t change our mining scripts. In the event that this were to happen, the team would have no other choice than to read up on what is different, and adjusting the way that we collect information from the API appropriately. By being proactive and keeping up to date with what Microsoft plans to do with

GitHub, the team may avoid potentially severe problems when it comes to mining information, and this will keep us on track to produce a successful product.

5.4 Errors During Data Collection

The final risk that we have assessed is the potential for anything to go wrong with the collection of the data itself as it is occurring. There are so many variables to account for when mining information from an API, that something such as maxing out the 5,000 requests per hour we are granted, or an internal server error on GitHub's end may happen at any time. In order to mitigate this risk, the team will implement a try-catch system that will ensure that the mining tool does not crash with an absurd amount of errors, and may continue as scheduled if possible. We also aim to notify an administrator via some kind of log so manual investigation may take place if needed, and provide the admin with the ability to pause or cancel the collection of a repository as they see fit.

With potential risks notified and assessed, the team needs to have a solid development plan in place in order to understand the timeline of the project through its entirety.

6. Project Plan

Our overall plan for implementation is iterative development for features over the span of the Spring 2019 semester. Below is a Gantt chart of our five milestones that we plan to complete for our project.

6.1 Create Mining Scripts

Our mining scripts are the backbone of our project. If these scripts are not finished and working as intended, than our whole project will fail. The mining scripts are essential for the whole project as the scripts are responsible for pulling data, in turn, must be stored in a

database and visualized as interactive graphics for the user to explore. Over our plan for development, our scripts will be constantly tested to see if they can be improved upon.

6.2 Repository Database

When data from GitHub is mined, the data will be stored in a database where users can access the information. The database will be managed by an administrator, whom will authorize the actions of repositories stored, will have commands to interact with how a repository will be mined. The database functionality is our second milestone, so that we can prepare an adequate environment for the data to be mined.

6.3 Hosting The Website

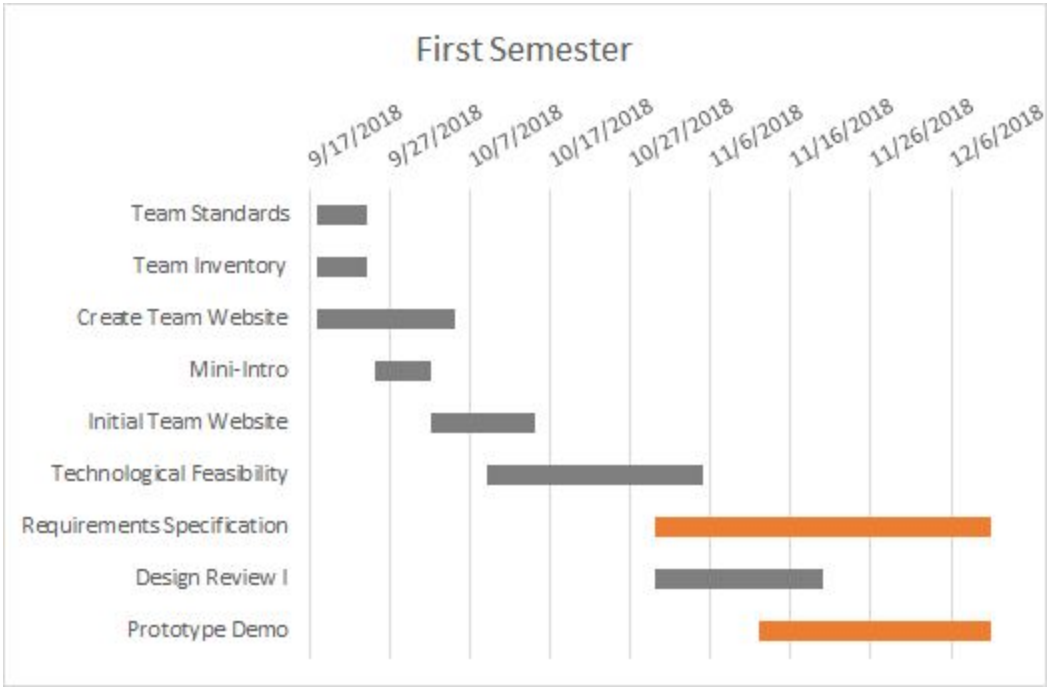
Because our web application will be performing a load of functionality from the mining scripts, it is important for us to have a powerful hosting service to perform all necessary functions. We believe that our web app will take on substantial traffic, making the overall process of the web app quite slow. With the use of the mining scripts, the web app will also take a hit on speed as well. These two factors are why we need a powerful hosting service to perform at the web app's best. We will begin hosting the web app once our mining scripts are finished and our database setup.

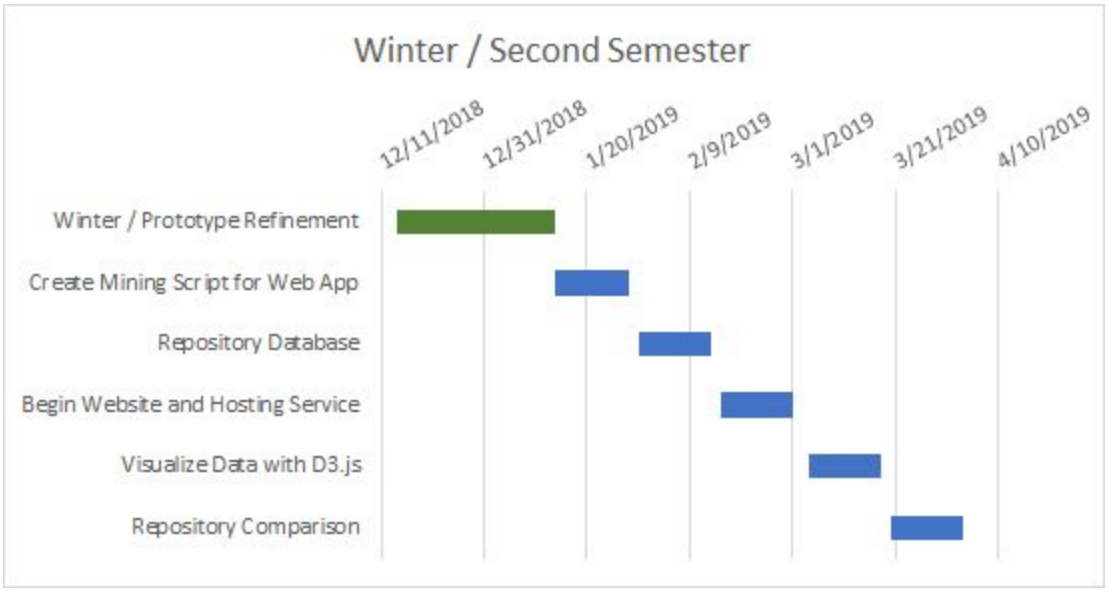
6.4 Data Visualization

One of the main components of our web app is data visualization. We will be using the pulled data from the scripts and database to provide the user with interactive graphics for them to explore. Data visualization will assist the user in finding a project to contribute to, as the user can interact with the visuals to see data points at a specific time. We plan to create our data visualizations after our web is hosted, this is because our framework for visualizing data must be done on a website.

6.5 Repository Comparison

Our last milestone is a repository comparison, where users can compare and contrast up to three different repositories to see the statistics of each selection. This is our last milestone because we need all the previous functionality to work properly before we can implement this milestone. After this milestone, we are open to further development and refine past milestones.





Above are two Gantt charts, one for the first semester and one for the second. Tasks that are complete are gray, orange is in progress, green is Winter, and blue are our five milestones described above. As you can see from the chart, the milestones are evenly separated, this is because we want to complete our first two milestones, data mining scripts and database management, so that our back-end technologies are working efficiently. After these two milestones, we will begin building our web application starting with hosting the product and the visualizations for data. We will also be testing each milestone accordingly. Keep in mind that our schedule is subject to change as time goes on.

7. Conclusion

Open-source software is not the easiest field to be involved in, with our product, we wish to make the contribution process easier to newcomers. Our web application will be able to mine data straight from GitHub, store the data in a database for future use, visualize that data in the form of interactive graphics, and compare repositories so that the user can filter out other options. In this document, we have narrowed down our requirements so that we can begin working on our minimum viable product with our five milestones stated in our “Project Plans” section. Through gathering requirements from our sponsor, Dr. Igor Steinmacher, we are confident that we can deliver a product that not only assists our client with his research, but to bring the power back to the users so that they can learn and find open-source projects to contribute to.