Requirements Specification Document

12/11/19

Version 2.0

Project Sponsor:

Dr. Igor Steinmacher

Team Mentor:

Fabio Santos

Team Members:

Adriana Aguilera, Joseph Danciu, Jevin Dement, Ugo Dike, Stavros Triantis

Client Signature:	**Team Signature:**
______________________________	______________________________
Date:_____________	**Date:**_____________

Table Of Contents

1 Introduction

The world of open source software is vast and growing quickly. The term "open source software" refers to software that is free for the public to "inspect, modify, enhance, and distribute [1]." The software is developed collaboratively via a public platform such as GitHub. Open source software (OSS) does not just benefit the tech world, it can benefit non-programmers alike. Some of the most well known open source projects include WordPress, Mozilla Firefox, Audacity, and OpenOffice. OSS gives developers the ability to collaboratively create projects in a free environment from anywhere around the world. It can also help businesses reduce their total investment costs when producing their products and services. Still, in order to produce more of this software, there is a demand for more developers to make contributions. A recent poll found that "71% of people surveyed" were expected to use open source software day-to-day in their developer jobs [2]. Not only are developers able to contribute to projects, but gain benefits such as improving their coding skills, being part of a community, learn new technologies, and bettering their resumes. The issue is, for many newcomers, this can be intimidating and overwhelming.

Match Source is a capstone team consisting of five computer science seniors at Northern Arizona University. Our members include Jevin Dement (Team Lead), Joseph Danciu (Release Manager), Adriana Aguilera (Recorder), Ugo Dike (Architect), and Stavros Triantis (Client Communicator). Our client, Dr. Igor Stienmacher, is a professor and researcher at the School of Informatics, Computing, and Cyber Systems here at Northern Arizona University. His research specializes in "the support of newcomers to Open Source Software development communities" [3]. Dr. Stienmacher has come up with the idea of a simple web application, which is a layer on top of GitHub, that newcomers can use as a resource to find projects that suit their skill set. With over "100 million GitHub repositories" alone, many of which being open source, there is an overload of information for newcomers [4]. Our client hopes that this web application will eliminate the information overload these newcomers are experiencing and encourage them to make more contributions to OSS. Our team is working diligently alongside Dr. Steinmacher to bring his vision to life. This document is a comprehensive outline of our client's problems, our envisioned solution, the requirements, risks associated, and our final plans to develop the final product.

2 Problem Statement

The problem for many developers is finding a project worth contributing to. The average developer will go to GitHub's website and navigate to the *Explore* section to begin their search. There are website tabs that filter the search, such as *Topics* and *Trending*, which is where most developers begin [5]. For example, let's assume a developer would like to choose the *Android* category. This method has not provided much support, as there are "60,000" available OSS projects within this category, with the same problem for around 200 available categories [6]. Once a developer finds a project on GitHub that interests them, other issues ensue. Some projects may have a certain level of complexity that surpass our example developers skill set. Referencing a poll of over 4,300 respondents regarding OSS contributions, "43% [state] they feel like they do not have the right skills to contribute" [4]. This may lead to some developers either reaching too high or low with their current experience level when contributing to a project. OSS projects rely on contributions to fix bugs, implement new features, and push the newest release. Due to the staggering number of projects available, as well as the sheer amount of code within these project files, many are discouraged from the start. Developers must be pointed in the correct project direction, to ensure people of all skill levels have a place to contribute effectively, without being discouraged by doubt. Many developers, because of this, are missing out on an opportunity to gain coding skills, experience, and work in a collaborative environment. As previously mentioned, open source projects are an extremely valuable resource to businesses, developers, and average tech users alike. There must be a way to encourage and increase participation. Our client is currently working with multiple teams, including MatchSource, to solve these OSS contribution issues.

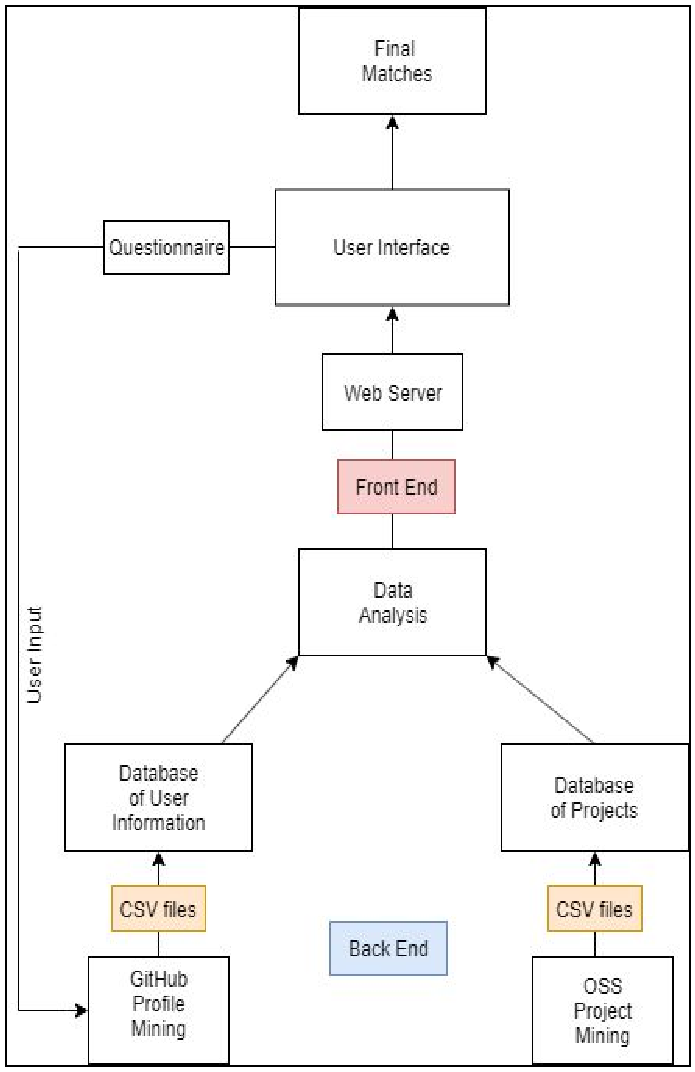
Dr. Steinmacher has a vision of easing the contribution process in OSS projects for developers. Unfortunately, he does not know exactly how this will be done. Currently there are a handful of missing capabilities, which include:

- Which platform to start assisting developers on
- Identifying a developers skills
- Finding available OSS projects
- Matching developers to available projects

3 Solution Vision

Our client, Dr. Steinmacher, wants us to help create a web application that is a layer on top of GitHub, where each user inputs their GitHub account username, allowing us to match them with available open source projects. Unfortunately, as of now, he does not have a way to parse through GitHub repositories nor available GitHub OSS projects. There is also no implementation of code that will match the developer with an available project based on the mined data. At this point, our client needs us to figure out the criteria necessary to make all of this happen.

The following flowchart explains how team MatchSource will solve these problems:



1) Mine through developer repositories
2) Parse through available OSS projects
3) User-Specific Data Storage
4) Available Project-Specific Storage
5) Match User to Projects
6) Display results

MatchSource will implement the parser using the Python3 programming language, which was a decision made based on deep analysis in our Technical Feasibility document. The Python script will gather key information from the GitHub API, which displays information in JSON format, most importantly, the projects that they have contributed to. In this iteration we are specifically searching for Java projects. Once the mining phase has begun, a user will receive an email informing them that the search has begun. We will need to search through our list of previously mined projects, and compare them to the users skills, all of which are stored in a database. The extraction of user skills will be completed with the same modular code used for acquiring the applicable projects. The information stored in the user-specific column of the database will be for those trying to find prior matches, and/or to narrow their next search.

The matches will be made by comparing the user-specific column to the available projects column. If the user does not have any information to extract over the last 90 days, they will be given a questionnaire. This form will ask them to truthfully fill our their skills, which will then be fed back into the process. Once everything has finished, the user will receive another email notification. Finally, the matches will be available on the appropriate web page tab.

4 Project Requirements

After extensive discussion with our client, and various team meetings, we have acquired the following requirements that are essential to building an effective product. In order to get a better understanding of low-level requirements, we start by listing our high-level requirements.

High Level Requirements:

- Find User Skills
- Find OSS Projects
- Match Skills to Projects
- Display Matches

The following requirements have been broken down into three separate sections including functional, performance, and environmental.

4.1 Functional Requirements

The first set of requirements are the foundation of our product. Functional requirements essentially outline the main features/functions that must be implemented in our application to create a viable product. We have broken down the features requested by our client into the following six functional requirements:

Functional Requirements:

- FR 1: Acquire GitHub username and authenticate
- FR 2: Check if project is in Java programming language
- FR 3: Clone repository
- FR 4: Run miner/parser on projects
- FR 5: Insert into database
- FR 6: Match user skills with available projects requirements
- FR 7: Display matches
- FR 8: Sort Matches
- FR 9: Read and Write Questionnaire Sheet
- FR 10: User ability to add projects
- FR 11: User ability to update skills
- FR 12: User ability to disable projects
- FR 13: Admin may set OSS projects to be updated every 7 days
- FR 14: Notify user on completion

We have come up with these fourteen requirements specifically to establish a strong understanding between our team and Dr. Steinmacher regarding what will be expected in the final product.

FR 1: Acquire GitHub username and authenticate

For our application to get started, we must acquire a Github username from all users. We plan on implementing a Github authentication that has users enter their Github username and password straight into the Github website, which will allow us to access their information securely.

FR 2: Check if project is in Java programming language

After the Github username is obtained, the mining script will begin to parse through their repository information on the GitHub API. It will check for projects that have accepted the users contributions. Once these projects have been identified the parser will check if the projects are written in the Java language. If they are written in Java, the parser will add them to a list, if not, the parser will skip.

FR 3: Clone repository

After the parser has a list of Java projects that the user has contributed to, it will then clone all of those repositories. Saving all of the repositories will allow us to run the project parser and acquire the necessary skill requirements to contribute to that specific project.

FR 4: Run parser on projects

Once all of the repositories have been cloned, the mining phase concludes. Next, the script will take the cloned repositories and run them through the project parser provided to us by our client. This project parser will provide us with a list of skills that are required to contribute to the project. As of now, the parser provides this requirements list from the users contributions because we are using the project parser for this section. Later, we will have a user repository parser that simply collects skills, without this extra step. Since the user has successfully contributed to that project, we will assign them all the skills needed to contribute to that project.

FR 5: Insert into database

Once the parser has finished running through all the projects that were contributed to, it will then take all the skills it has assigned to the user and store them in a database. This information will later be used in the matching phase of the process.

FR 6: Match user skills with available projects requirements

One of our biggest functional requirements is being able to match user skills to projects. The way that we plan to do this is with a comparison tool. This tool will take in two pieces of data. It will take in the users skills that were just mined, and all of the available projects requirements. Once the comparison tool has all the data, it will start matching skills to project requirements. After it finds matches, it will send them to our database to later be displayed to the user.

FR 7: Display matches

Once the user skills are matched to projects, we will display the matches on our web application. We plan on displaying up to 20 projects according to which best fit the users skill set. Additionally, if possible, we want this list of projects to be downloadable, so the user won’t have to wait for the mining and comparison tools to run every time they want to see their list again.

FR 8: Sort Matches

For organizational purposes, we will allow users to organize their recommend projects in a way that is most convenient for them. By default, the projects will be sorted according to their similarity to the users skill set. However, we want to include ordering alphabetically as well.

FR 9: Read and Write Questionnaire Sheet

If a user does not have a Github account, we will have them fill out a questionnaire. The questionnaire will consist of skills and languages that the user may choose, which will replace users skills that would have been mined from their Github repositories. The questionnaire will have the feature of a drop down list, which will allow them to pick multiple skills. This will ensure that the skills picked are spelled correctly and eligible to be matched to available projects.

FR 10: User ability to add projects

As stated in the problem statement, many new developers do not know where or what to start contributing to. One result of this is the many smaller OSS projects needing contributions that do not get attention. To solve problems like this, we will allow users to add to the list of projects that we recommend to users. This will allow developers to have more options after they see where their skill set applies.

FR 11: User ability to update skills

Users will be able to update their current skill set by inserting their GitHub profile username back into the matching process. Alternatively, they will be able to fill out a questionnaire at any point. This will ensure the developer always has up-to-date projects available to access.

FR 12: User ability to disable projects

If a user wishes to disable, or remove projects that the matching algorithm accumulated, they will have the option to further narrow their search. This will be used in case the user does not want to contribute to particular projects. Disabling of projects will remove the specified project(s) from their visible recommendation list.

FR 13: Admin may set OSS projects to be updated every 7 days

In order to keep the application users motivated to return, we must perform weekly suitable project updates for each developer. This will entail another search made, finding available projects that match the developers current skill set. This will also ensure that changes in developer experience are accounted for.

FR 14: Notify user on completion

One issue that we face is the amount of time our mining and comparison tools will take. To resolve this issue, we plan on notifying the user upon completion. Notifying the user will prevent them from thinking our site is frozen or that they need to refresh the page. As a group we have come up with two possible solutions for notifying the user upon completion. The first solution is an email notification letting the user know that their list of projects is ready to be viewed. The second solution is a browser notification that would include the same message as sent via email.

4.2 Performance Requirements

The second set of requirements are the performance requirements. These requirements outline how our system is expected to perform when in use., they specify goals for learnability, response time, and usability.

Performance Requirements:

- PR 1: Each Available GitHub Project will be mined in less than 2 hours
- PR 2: Each user project will be mined in less than 2 hours
- PR 3: Information will be readily available upon return

Our team and Dr. Steinmacher have come with these three requirements based on how some of the functional requirements, mentioned above, should run during execution.

PR 1: Each available GitHub project will be mined and parsed in less than 2 hours

Since we will be allowing users to add projects to our database we will need to be able to clone and parse that project fairly quickly. So, we believe that cloning and parsing through a project should be accomplished in a max of 2 hours. We do not want to make a promise for anything less, since there are various sizes of projects that can be added. Alternatively, we have decided anything over two hours may cause question regarding our products efficiency.

PR 2: Each user project will be mined in less than 2 hours

Since our product will be mining user data in real time, one issue that we face is the speed of user project mining. Our product will be used by a wide variety of developers ranging from those who contribute consistently to those who do not. Since the common contributors will have so much data to parse, our mining tool should mine a list of projects the user has contributed to and acquire their skills in less than 2 hours, multiplied by the number of unique projects that they have contributed to in the past 90 days.

PR 3: Information will be readily available upon return

As mentioned above, mining and parsing through user skills and projects may take some time. To combat this issue two things should be done. The first thing is that the users results should be saved so that if they come back they will not have to wait again. Additionally their recommendation list should be updated if any new projects are added or the project list is updated by the admin.

4.3 Environmental Requirements

Our last category is environmental requirements. These requirements are usually given to us by our client, however, they may also arise from the technologies we use.

Environmental Requirements:

- ER 1: 90-Day Limit on Mining User Data
- ER 2: Recommendations only apply to Java Projects

These two environmental requirements were created through discussion with Dr. Steinmacher, as well as well as through technological constraints our team has faced.

ER 1: 90-Day Limit on Mining User Data

After much deliberation with our client, we came up with a 90-day limit on mining user profiles. This requirement would help solve a couple of problems. The first problem it addresses is the speed of our profile mining tool. With less data to mine, our mining tool wouldn’t take as long, giving the user a better experience. Additionally, our client wants us to mine data that is relevant to the user. If we mine skills from 4 years ago there is no way of guaranteeing that the user still has those skills.

ER 2: Recommendations only apply to Java Projects

As mentioned in the environmental requirements description, these requirements may come up as constraints that arise from the technologies we use. The project parser given to us by our client can currently only parse Java projects. This is the reason that our mining tool only looks for Java projects, as well as why we will only be able to analyze and recommend Java projects to users. This constraint exists because of the amount of time it would take to gear the parser toward multiple programming languages.

5 Potential Risks

Analogous to all projects, our solution comes with potential risks. Before we begin to develop our product, we want to negate as many of these risks as possible. Identifying these risks is important to make decisions that improve the product outcome. After many discussions with the team, including collaboration with our client, we have come to the conclusion that risks associated with our project are minor. The main risks associated with our project are predominantly out of our control, or limited. These include changes to the GitHub API, the over or under qualification of users, and concerns with user privacy. In the following three sections, we will break down the severity and likelihood of each risk, and discuss how we will accommodate each problem, in the event any occur.

5.1 Changes to the GitHub V3 API

Figure 1

	Severity	**Likelihood**	**Overall Risk**
GitHub API V3 becomes unsupported or is modified	HIGH	LOW	LOW

Figure 1: A table depicting the overall risk of GitHub's API changing

A substantial risk to our project includes potential changes, modifications, or complete removal of the GitHub API V3. Since our project is heavily dependant on this version of the GitHub API, our product would be rendered useless. The severity of this happening is high, but the overall likelihood is low. We do not anticipate this being a problem, as GitHub has released a new, stable version version of their API (V4) in the past, while making no alteration to V3. This is a risk we have decided to accept.

5.2 Over or Under Qualification of Users

Figure 2

	Severity	**Likelihood**	**Overall Risk**
Over or Under Qualification of Users	LOW	HIGH	LOW

Figure 2: A table showing the overall risk of our product over or under qualifying users

Another risk associated with our project is the over or under qualification of users by our matching algorithm. As seen above in Figure 2, the severity of this risk is low, but the likelihood is high. Currently our system assumes that users have all the skills associated with a repository that they have contributed to in the past. This is a problem as users may have only contributed to a very small portion of the project, or documentation, but still receive the skills that the project requires. Unfortunately, our team is limited to one semester, and the technology required for a more specific breakdown would require all of our time. It would likely require us to analyze individual snippets of code and extract skills from there. We have accepted this risk, as our client has decided that he is currently content with our current system. He will likely expand on this project in the future, and focus on skill mining specifically. Refinement of our match algorithm will be a stretch goal.

5.3 Privacy Concerns

Figure 3

	Severity	**Likelihood**	**Overall Risk**
Privacy Issues	LOW	MEDIUM	LOW

Figure 3: A table showing the overall risk of privacy issues

Lastly, our final risk involves the users being able to access data that is not their own. As seen above in Figure 3, we portray that the severity is low and the likelihood is medium. Overall we have decided that the risk of this happening is low, as the information we store is already public data, and not useful for malicious purposes. Still, we want our users to have a sense of security when using our web application by keeping their information private. To deal with this risk, the proposed solution is to authenticate via GitHub. Our backup plan is to make our own authentication system. We believe this is the most optimal solution that will ensure user privacy. Additionally users will also be asked to check off a box that acknowledges that we will be collecting and storing their data.

6 Project Plan

The current project plan is depicted by Figure 4 below.

Figure 4

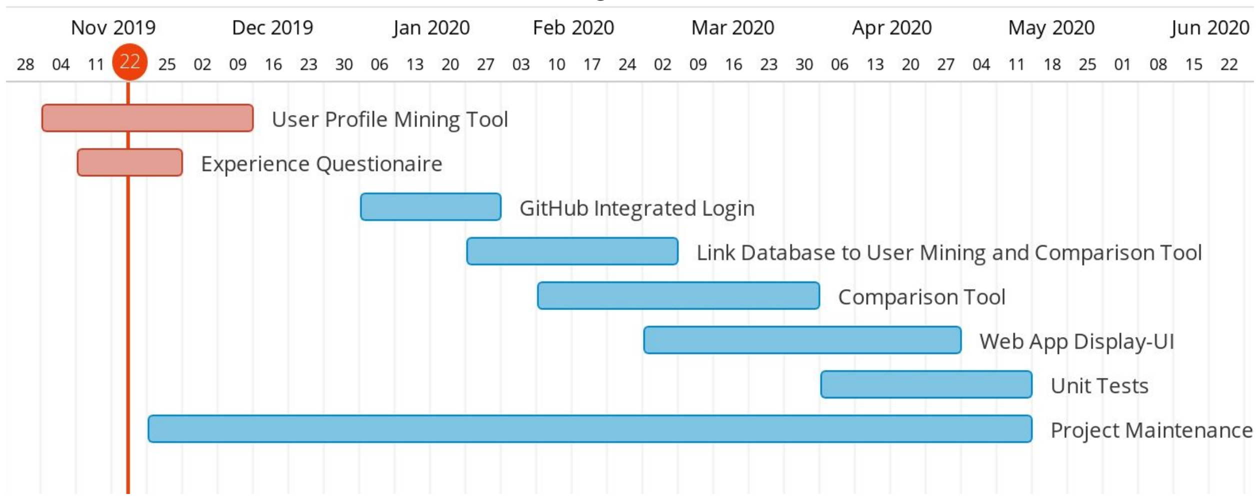


Figure 4: A Gnatt chart depicting Match Source's projected timeline

Our milestones include:

- User profile mining tool
- Experience questionnaire
- GitHub integrated login
- Comparison tool
- Linking our database to the user mining and comparison tool
- Web app display/user interface
- Unit tests
- Project maintenance

Our main functional requirements include the user profile mining tool, comparison tool, and the web app display. Our team is currently working on completing the mining tool and the experience questionnaire by December 9th 2019, in time for the tech demo. As a group we reasoned that getting one of the main functional requirements done for the demo would set us up for success next semester as we would only have two more to complete. The remaining tasks will be completed during the Spring 2020 semester. The first thing we want to get done is the GitHub integrated login so that we can connect it to our mining tool. Next we want to link our database to both the comparison tool and the mining tool so that we can work with real, changing data rather than hard coded data. After the database is finished we want to create a clean and polished user interface that displays all of our data to the user. Lastly, once this is done we want to run unit tests to make sure our product is functional and stable. Additionally, as you can see in Figure 4, our team has decided to align our tasks in parallel so that way if one task gets behind it doesn't push back every other task and we will always have something to work on. This will hopefully prevent us from falling behind from our scheduled due date.

7 Conclusion

The current predicament Open-source software finds itself in is that many developers do not know what to start contributing to. This is a shame since OSS contributions benefit the community and developers far more than we think; from career advancing to technological freedom. The Match Source team is working alongside Dr. Igor Steinmacher to create a web app that will motivate new users struggling to contribute to open source projects on GitHub by curating projects for them based on their skills and experiences. We have gathered enough requirements through interviews with our client to build the documentation necessary to meet our clients needs. Match Source is confident that we can produce a product that will be the basis for allowing struggling developers to be able to find projects that fit their skillset.

8 References

[1] OpenSource. What is open source?. 2019. Retrieved December 10, 2019, from: https://opensource.com/resources/what-open-source

[2] Gousios, Georgios; Vasilescu, Bogdan; Serebrenik, Alexander; Zaidman, Andy. "Lean GHTorrent: GitHub Data on Demand" (PDF). The Netherlands: Delft University of Technology & †Eindhoven University of Technology: 1. Retrieved November 12, 2019.

[3] Igor Steinmacher, PhD. Retrieved December 1, 2019 from https://www.igor.pro.br/

[4] Digital Ocean. 2018. Currents. (October 2018). Retrieved November 26, 2019 from https://www.digitalocean.com/currents/october-2018/

[5] GitHub. Retrieved December 1, 2019 from https://github.com/explore

[6] Daniel Guajardo Kushner. 2019. Instagrantt. (December 2019). Retrieved November 28, 2019 from https://app.instagantt.com/r