MATCH
SOURCE

# Technological Feasibility Analysis

11/8/19

Project Sponsor:

Dr. Igor Steinmacher

Team Mentor:

Fabio Santos

Team Members:

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

Overview:

The feasibility of tools that are an option for developing the solution to our client's addressed problem of OSS projects containing an information overload for a developer. We will be analyzing and determining what tools we can use to overcome those challenges.

# Table of Contents

## 1 Introduction

Team Match Source is a senior capstone group working toward a solution that will change the lives of developers and the software industry as a whole. Our team is comprised of driven Computer Science students, including Jevin Dement, Stavros Triantis, Adriana Aguilera, Ugo Dike, and Joseph Danciu. The problem we are handling is the abundance of developers who are troubled while seeking experience in the open source software (OSS) world. Open source software refers to software that is developed publicly and collaboratively, as well as being free to the public to use and distribute. Throughout the history of OSS, countless developers have been discouraged by attempting to contribute to the overwhelming number of projects available. The mass of potential developers dwindles as these projects become more intricate, or so they seem. Our teams sponsor, Dr. Igor Steinmacher, envisions a solution that will ease the process of finding OSS projects on GitHub, filling gaps in the software industries workforce. Dr. Steinmacher is currently researching "intersections of Computer Supported Cooperative Work (CSCW) and Software [Engineering] (SE), mainly on topics related to Open Source Software development." Alongside Dr. Steinmacher, Team Match Source will create a web application, with an easy to utilize user interface, that will match countless developers with ideal projects based on their skillset.

We begin with section 1, by analyzing the major technological challenges we expect. These include choosing effective Data Mining and Parsing Languages, Databases, Data Comparison Tools, Web Hosting Services, and Web Application Frameworks. The main challenges for these topics will be finding which tools to use for an effective product outcome. In section 2, we will analyze each of these areas carefully in turn, looking at alternatives and rationale for choosing a particular solution. Lastly, for section 3, we will provide a summary explaining our final choices and discuss how the end  product will work.

## 2 Technological Challenges

Every company faces challenging obstacles that must be overcome, Match Source is no different. Our team has analyzed the core technical challenges we will be facing and their corresponding solutions.

- ➢ Data Mining and Parsing: We will need to select the optimum language for gathering and parsing data off of GitHub, from both the users repositories and OSS projects found in order to analyze the data.
- ➢ Database: We will need to select the optimum tool to store the data mined for analysis.
- ➢ Data Comparison Tool: We will need to select the best language to compare the data collected and stored in the database.
- ➢ Web Hosting: We will need to select the optimum hosting platform for our team to ensure reliable access. Each site visitor must come and go with ease, while enjoying their experience.
- ➢ Web Framework: We will need to select the optimum web framework to build our web app on, keeping in mind it should be attractive and easy to use.

Our team has created a system to analyze the core technical challenges that we will be facing. In the technical analysis section, we have gathered information on our top three choices for every challenge. We will discuss the benefits of each language, including the barriers that may restrict us from using them. This process will ensure that Match Source utilizes the best technologies available for this product.

# 3 Technology Analysis

In order for our team to provide the best possible product for both our client and his end users, we want to carefully evaluate all of our choices for each tool before building. In this section we will break down every challenge into subtopics, including carefully selected solutions for each one. To aid in our decisions, we will create a small chart at the end of each section, weighing the pros and cons for each option. This will entail an explanation of why we made these choices.

## 3.1 Data Mining and Parsing Language

The first portion of our product will be gathering information from GitHub, which will include: user commit information, code changes in specific projects, and project information. GitHub stores all of this data and more in JSON format. After this information is gathered, we will parse through all the JSON data and separate it into two sections. The first section will include all user data. We will parse through their commits, contributions, and pull-requests. Next, we will parse through pre-determined GitHub Open Source projects, analyzing and parsing through all the files available. The second section will require us to analyze project data and list a set of skills that are recommended for someone to start contributing to the project.

In order to choose a programming language for mining and parsing, we must breakdown and analyze each option, to see if they are easy to use and learn, can interact with GitHub’s API in an understandable fashion, and have libraries for parsing JSON. These metrics are important to us for a number of reasons. First, we need a language that can be used with ease, and has plenty of online resources for learning. We need a language that is easy to use and learn because our team will be manipulating and analyzing an abundance of data, so we don't want these aspects to cause unnecessary challenges. Secondly, we want a language that can interact with GitHub’s API in a straightforward manner. With all the data collection that will be taking place we want a language that can access that information with ease. Lastly, we need a language that has libraries for parsing JSON. This is an important metric because our team will be parsing plenty of JSON data and analyzing it to assign users skills. With these criteria in mind, we narrowed down our choices to Python, Java, and Ruby based on research and our familiarity with these languages.

### 3.1.1 Python

Python is an interpreted, high-level, general-purpose programming language [32] that has many 3rd party libraries, supporting GitHub data mining and JSON parsing. Some of the GitHub API’s 3rd party libraries include: PyGitHub, libsaas, and GitHub3.py [6]. These libraries make using GitHub's API effortless. These are not the only libraries that we can use to access GitHub's data, another option includes the ‘requests’ library. This library is used to make HTTPS requests which could easily be configured to grab a hold of the data we need. When it comes to parsing JSON files Python has a built-in package called ‘json’ used for encoding and decoding JSON data. If the team chooses Python as our language we will be using this package to deserialize the JSON data that we mined so that we can put it into our database for later analysis.

### 3.1.2 Java

Java is a general-purpose programming language that is class-based, object-oriented, and designed to have as few implementation dependencies as possible [31]. Similar to Ruby and Python, Java has libraries to make data mining and parsing JSON much simpler. While this is true, after running the same small test that we created in Python and Ruby we found that the Java test took longer to make and required more online resources for help (and took more lines of code). Although our developers are very familiar with Java, we do not want to be spending a whole lot of time looking at online resources for help.

### 3.1.3 Ruby

Ruby is a pure object-oriented language that focuses on clarity and efficiency. It is an open source language with a thriving community supporting it, providing a mass of resources. GitHub has three official API libraries that they promote, one of which is for Ruby. If we chose Ruby to mine data, we would have the opportunity to use an official ‘sponsored’ library called octokit.rb [6]. This library has a variety of documentation and is very easy to understand. When testing some simple GitHub requests using octokit.rb, we noticed that everything was fairly readable since we had  prior experience with object oriented programming. Though everything in Ruby is an object [28], it still took time and effort to write, while the other languages came more natural. In regard to parsing, Ruby has an import that can handle parsing and manipulating JSON data similar to Python, which is what we were searching for. Being able to make simple manipulation in 1-10 lines of code is more convenient than creating a class as we would do in Java.

### 3.1.4 Conclusion

The table below shows the results of our research on the three languages and rates them from 1-10 based on three factors, Ease-of-use, GitHub compatibility, and JSON parsing.

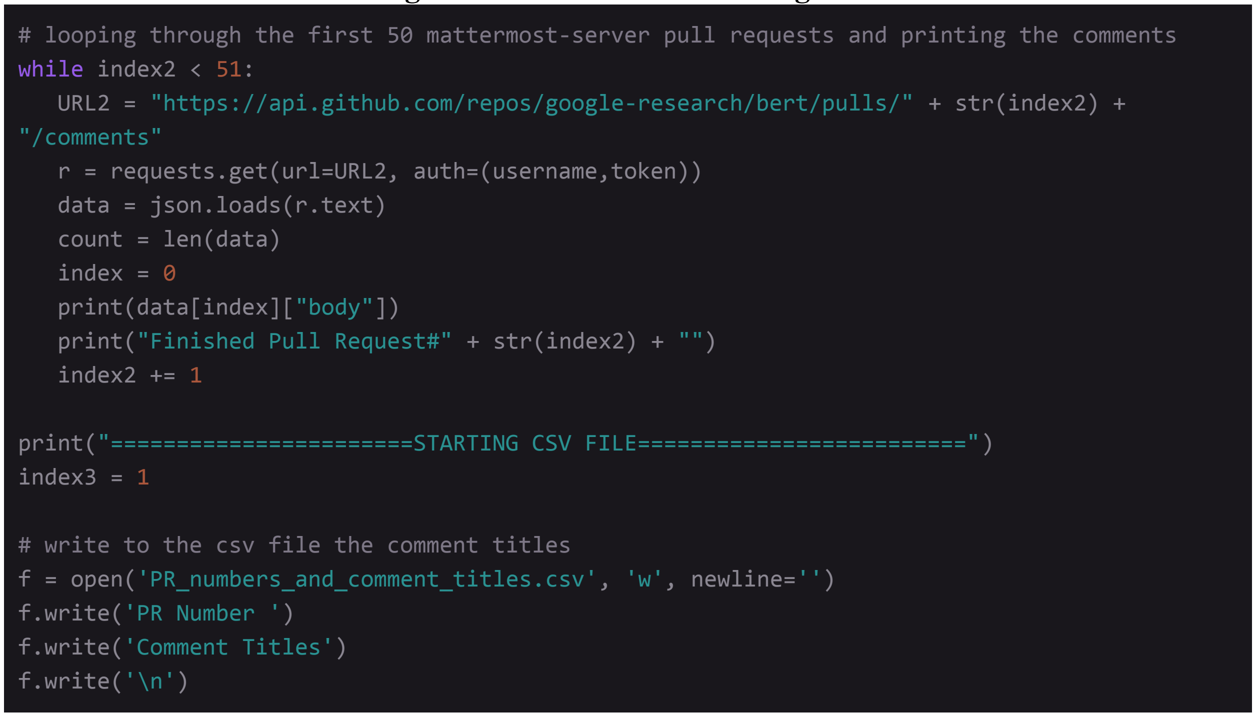
**Figure 1 Data Mining and Parsing Language Scoring**

| | Ease of Use | GitHub Compatibility | JSON Parsing | Total Score |
|---|---|---|---|---|
| **Python** | 10 | 9 | 10 | **29** |
| **Java** | 8 | 9 | 6 | **23** |
| **Ruby** | 5 | 10 | 7 | **22** |

**Figure 1:** Each column above is rated on a scale of 1-10; 1 being “Weakly Correlated” and 10 = “Strongly Correlated”.

As shown in the table above, we believe that Python will be the best overall language for both mining and parsing data. When deciding what language to choose for mining GitHub data, we narrowed it down to either Python or Java. Ruby had an official GitHub library that we could've used while Python only had 3rd party libraries. Python accumulated the highest Ease of Use score because of the result of some simple tests we ran. As shown in Figure 2 below, we found that writing the test in Python was simple to write, and easily understandable by the team.

**Figure 2 Code Used For Mining Test**



```python
# looping through the first 50 mattermost-server pull requests and printing the comments
while index2 < 51:
   URL2 = "https://api.github.com/repos/google-research/bert/pulls/" + str(index2) +
"/comments"
   r = requests.get(url=URL2, auth=(username,token))
   data = json.loads(r.text)
   count = len(data)
   index = 0
   print(data[index]["body"])
   print("Finished Pull Request#" + str(index2) + "")
   index2 += 1

print("=======================STARTING CSV FILE=========================")
index3 = 1

# write to the csv file the comment titles
f = open('PR_numbers_and_comment_titles.csv', 'w', newline='')
f.write('PR Number ')
f.write('Comment Titles')
f.write('\n')
```

**Figure 2:** Shows the code used to test mining and writing (something we would eventually need to do as well) in Python.

That being said, we decided Python would let us get more done in less time. This would give us the chance to go above and beyond our MVP. When it came to deciding which parsing language we wanted to use, Python obtained a score of 10, which was higher than Java and Ruby, based on tests that were ran with their most popular JSON parsing libraries. Parsing a simple JSON file was straightforward in Python unlike Java and Ruby. Python will be our primary language used in both mining and parsing data.

## 3.2 Databases

To complete our product, we must be able to store the data we mined from the GitHub API. The database we decided to use must be efficient in storing information in JSON files, which will be data format collected in. When retrieving the data for analysis, it will be exported as a CSV file to be handled by the data comparison tool. The main technologies that we will be researching to fulfill the need for storing data are MongoDB, PostgreSQL, and Cassandra. Subsequently, we decided to research these technologies because we anticipate that they fit seamlessly with the framework chosen for the web application.

### 3.2.1 MongoDB

MongoDB is a document-oriented NoSQL database that uses JSON-like documents that allow varied structures to store any type of data. Since it is schema-free, the database gives users the freedom to create documents without having to first create a structure. MongoDB implements horizontal scalability through sharding, that allows users to add additional instances to increase capacity when required [25]. The advantage of horizontal scalability is that it can provide users the ability to increase capacity on the fly, being only limited by how many machines can be connected successfully [25].  Query performance is one of the strong points of MongoDB. The schema-free structure allows this database to handle unstructured data very fast, allowing users to query in a way that caters there workload. Majority of the workable data is stored in the RAM which gives this querying a boost in performance [23]. MongoDB integrates easily with all of the predominant programming languages and top web frameworks such as Angular, Express, React and Django.

### 3.2.2 PostgreSQL

PostgreSQL is an open source relational database that emphasizing extensibility. It uses tables, constraints, triggers, roles, stored procedures and views as the core components that are worked with. A table consists of rows, and each row contains the same set of columns. PostgreSQL uses primary keys to uniquely identify each row in a table, and foreign keys to assure the integrity between two related tables. This design structure is more strict in comparison to a NoSQL database. It is designed to handle a range of workloads, from single machines to data warehouses or web services with many concurrent users [21]. PostgreSQL provides some features that help build a scalable solution, however, independently it is not scalable. It supports both SQL for relational and JSON for non-relational queries [21]. PostgreSQL is compatible with various platforms using all major languages and web frameworks.

### 3.2.3 Cassandra

Cassandra is a column-oriented NoSQL database that is designed to handle large amounts of data across many servers, providing high availability with no single point of failure. Cassandra supplies linear scalability, meaning that capacity may be easily added simply by adding new nodes online. This database utilizes a mixture of tabular and key-value stores that allow all possible data formats including structured, semi-structured, and unstructured [22]. Cassandra supports the CQL query language, which is very similar to SQL, but still non-relational, so it has different ways of storing and retrieving data. Similar to the other contenders, integration with programming languages is great, but for web frameworks it requires more work to connect everything together smoothly. The CQL query language is only unique to Cassandra which is a reason why integration is not as easy as others databases.

### 3.2.4 Conclusion

The table below shows the results of our research on each language.

**Figure 3 Databases Scoring**

| | Integration | Structure | Scalability | Total Score |
|---|---|---|---|---|
| **MongoDB** | 10 | 10 | 7 ms | **20** |
| **PostgreSQL** | 9 | 7 | 1883 ms | **16** |
| **Cassandra** | 8 | 9 | 8.5 ms | **17** |

**Figure 3:** Each column above is rated on a scale of 1-10; 1 being "Weakly Correlated" and 10 = "Strongly Correlated". Scalability based on a latency benchmark. Latency is a measure of how long it takes to respond to a single request so the lower the time the better scalability is. The test consists of 50% read and 50% updates for 400 million records.

Concluding our research on databases options, we decided that MongoDB is going to best fit our needs for storing data. MongoDB's structure received the highest score at 10, because it is document-oriented NoSQL database that is schema free, giving users the freedom to create documents without specifying the structure, which allows for storing various types of data easily. Being that Cassandra is a column oriented database, which shares similarities with relational databases like PostgreSQL, storing JSON data from the GitHub API wouldn't be as efficient due to the structure. To determine the scalability of each database, we used a benchmark that tested its latency. Latency measures the time it takes to respond to a single request, meaning the lower the time, the more scalable the database is. MongoDB came out on top against the others, with a result of 7 milliseconds, with Cassandra coming in at a close second with 8.5 milliseconds. Relational Databases such as PostgreSQL have a hard time with scalability as seen with the result of 1883 ms. MongoDB's integration received the highest score of 10 due to how easily it fit with other languages and frameworks. Its documentation on how to connect the database to these frameworks and languages makes the integration process simple.

## 3.3 Data Comparison Tool

At this point, the data is collected and stored in a database. The technical challenge is that data means nothing until it is interpreted into useful information. The solution to this challenge requires a connection between the user's skills data and the product requirements data. Match Source needs a language that will be suited for this type of interaction.

### 3.3.1 Python

Python is a versatile language that offers a simple and readable syntax. Python offers a lot of support for open source libraries with its large community, which will reduce the amount of utility functions we may need. Mongo DB has the option to output as a .csv or json file and Python has an easy time converting CSV and JSON files to dictionaries for ease of access. The language is very easy to pick up and can be easily integrated with C or Java code. This simplifies conversions of any type of mined data because of its ability to be interpreted by the Python language. Most data mining and big data platforms use Python[1] and we will be doing the same type of analysis.

### 3.3.2 Java

Java has been around for over 20 years and has proven time and time again to be a very powerful language. Java offers numerous amounts of resources to use libraries. The community is also very large and provides great support. Java promotes the philosophy of giving back to the society, which is the reason why the community support is big. The Java language is omnipresent because of its stability and scalability[2]. The language is also easy to learn and all of our developers have a lot of experience in Java.

### 3.3.3 C

C is a mid level language that combines the features of high level languages and low level languages. It has a difficult learning curve but it is heavily used as a strong base language for parsing with pointers and creating personalized structures for holding data; which may be convenient when it comes to making comparisons. The community is also very big but support can be difficult to understand with a lot of solutions with compile errors. Integrating json files is a little more difficult in C and data analysis with C would contain much more functions for analyzing strings.

### 3.3.4 Conclusion

Our Match Source developers chose the tool based on ease-of-use, community support, library availability for data analysis and integration capabilities.

**Figure 4 Data Comparison Scoring**

| | Ease of Use | Community Support | Libraries | Integration | Total Score |
|---|---|---|---|---|---|
| **Python** | 10 | 10 | 10 | 10 | **40** |
| **Java** | 8 | 10 | 10 | 8 | **36** |
| **C** | 6 | 7 | 5 | 7 | **27** |

**Figure 4:** Each column above is rated on a scale of 1-10; 1 being "Weakly Correlated" and 10 = "Strongly Correlated".

Python has the most friendly readable syntax and has a lot of built in functions for ease of use. Java came behind Python and was rated lower because of its syntax being less readable than Python. The community support for Python and Java are extensive, however, Python is a more popular choice in the community for data mining and analysis. Being able to convert JSON files to dictionaries in Python with no hassle makes it worthwhile to use for compatibility. The libraries are vast and parsing through anything can be done with these libraries. Python also integrates really well with other languages. C is a great language to build off of because of its ability low level language abilities but it comes last in every category measured.

## 3.4 Web Hosting Services

One of the most important components of our web application is web hosting. When utilizing a web hosting service, Match Source will ensure an enjoyable and reliable experience for website visitors. There are a large number of hosts, providing platforms which require payment for their service packages. However, many offer pay-as-you go services, minimizing unnecessary software applications. Since this application will be hosted for an unforeseeable amount of time, we will make this decision based on a balance between reliability, scalability, and affordability. Taking these factors into consideration, we believe we have found the best solution that will ensure our web application will be sustainably hosted for the longest time possible.

### 3.4.1 Digital Ocean

Digital Ocean offers an affordable, reliable, and customizable service, with a variety of features unique to most web hosting platforms. They have created the concept of "droplets", where each droplet is a customizable Virtual Machine, hosting a server. Each server provides scalability with customizable hardware upgrades at varying, affordable price points. Their base service starts at just five dollars a month, however, this will not be necessary. Our client has offered the team their prepaid Digital Ocean server. This platform has an average uptime of 99.99%, according to Brad Smith, a researcher and developer focussing on Digital Oceans performance since early 2018 [5]. This number represents about four hours of potential downtime in a 365 day period. With such a reliable service, developers will be able to access our web application at any time, providing a more enticing product.

### 3.4.2 Microsoft Azure

Microsoft Azure offers an extensive variety of features in a single package. This includes over 25 services that will always be free, along with a 12 month trial, including their most popular. The most useful services provided include SQL Database & Azure Cosmos DB (a noSQL DB). With data centers spanning 52 regions worldwide, available in 140 countries, it is one of the largest reliable services. A "region is a set of data centers, interconnected via a massive and resilient network [3]." Microsoft provides a comprehensive web page with information on how to adjust the scalability based on the amount of incoming traffic. This guide would be useful for our group when learning how to complete such a task.

### 3.4.3 Google Cloud

Google Cloud offers a globally available, fast, cheap, and scalable platform for web hosting and multi-tiered web application development. Similar to services mentioned prior, Google offers a "pay for what you use" tier system, with each tier containing a variety of services. These services support popular frameworks and architectures necessary for the product, including Django, Flask, Node.js, and MongoDB. Google's "private fiber network spans the globe with over 100 points of presence across 33 countries" (Google). With data centers spread so widely, the website would be running on a solid, reliable foundation. Lastly, Google cloud offers access to "big data" to find answers faster and build a better product. This may be useful for finding information that is otherwise inaccessible to the public.

### 3.4.5 Conclusion

Below is a table comparing Reliability, Scalability, and Affordability for each Hosting Service.

**Figure 5 Web Hosting Services Scoring**

| | Reliability | Scalability | Affordability | Total Score |
|---|---|---|---|---|
| **Digital Ocean** | 10 | 10 | Free | **20** |
| **Microsoft Azure** | 10 | 10 | $10/month | **20** |
| **Google Cloud** | 10 | 9 | $10/month | **19** |

**Figure 5:** Each column above is rated on a scale of 1-10; 1 being "Weakly Correlated" and 10 = "Strongly Correlated".

Overall, Digital Ocean has come out on top for web hosting services regarding reliability and scalability, with a score of 10. The other two options, Microsoft Azure and Google Cloud came close in score, but not price point or ease of integration. Our client has assured us access to their Digital Ocean droplets for this product, making the price point non existent. This also means we will have access to information in person and over the internet, unlike our contenders. With so many resources available, from our client to mentor, and Digital Oceans superior scalability and reliability, the choice was clear.

## 3.5 Web Application Frameworks

A reliable and powerful web framework is a necessity in order for us to present our web app to the final end user. A clean, attractive, and easy to use UI is very important or else the app may be difficult to use or completely unusable. Other considerations are that it should perform well with our back end to provide users with the best experience possible while providing the team with the best tools to accomplish our tasks. We also take into consideration the needs that future teams working on maintaining the app may need. The most important factors that we used to guide our decisions are scalability, database support, integration, and learning curve. Here, our team has narrowed it down to three web frameworks that should best suit the needs of the product and client.

### 3.5.1 Django

Django is an open source web framework that is based on Python. It offers one of the best documentations possible, while being very easy to learn if the developer has prior knowledge of Python[12]. It has been thoroughly tested, ensuring that it is highly durable and dependable. Django is also highly scalable, which is important for our product in the case a user has a large amount of GitHub commits that need to be processed. This framework offers multi-database support, officially supporting PostgreSQL, MySQL, Oracle, and SQLite [11]. Although, 3rd party database backends can be used for other databases [11]. Django is one of the most popular frameworks on the market, used by companies like IBM and Uber [13]. This means there is a greater likelihood of it holding up for future maintenance. Django should also be very easy to integrate into our project.

### 3.5.2 React

React is a frontend Javascript web framework that allows developer to create beautiful user interfaces. React is very flexible and allows for high modularity [16]. React is decently scalable, but really helps to supplement the scalability of other frameworks [15]. It is especially great for collecting rapidly changing data in order for it to be recorded and editing data. It has a simple library that is relatively easy to learn, although it's documentation not great as it is has a greater influx of new features. Unfortunately, React only supports the view component in the MVC, so it's likely we would need to have another framework to fill in the gaps [17]. This also means that it's not ideal for working with databases on its own. React should be extremely easy to integrate into our product, especially if used alongside another framework.

### 3.5.3 Angular

Angular is another popular JavaScript frontend. This framework works best for web apps with dynamic content and complex large scale projects [18]. Angular is very scalable, although it can be made more scalable with the addition of supplementary frameworks [19]. Although

Angular is the youngest framework on the list, it's maintained by Google and used in over 600 of their applications, so it has been thoroughly tested [20]. Since it is a framework and not a library you can get to work more quickly than with React. It also supports multiple databases, such as mySQL, SQL Server, HBASE, Cloud FireStore or CouchDBAngular [19]. A con is that Angular has a notoriously steep learning curve which might make it harder to use [20]. The documentation is good, but also not as strong as our other options [19]. Integration of Angular should not be that difficult, but we must take into account the learning curve as well. While Angular is very powerful, the size of our project is smaller than what a lot of Angular projects are. Therefore taking time to learn such a complex framework may just slow us down in the end.

### 3.5.4 Conclusion

**Figure 6 Web Application Framework Scoring**

| | **Scalability** | **Database Support** | **Integration** | **Learning Curve** | **Total Score** |
|---|---|---|---|---|---|
| **Django** | 10 | 10 | 10 | 9 | **39** |
| **React** | 7 | 0 | 10 | 7 | **24** |
| **Angular** | 8 | 10 | 10 | 5 | **33** |

**Figure 6:** Each column above is rated on a scale of 1-10; 1 being "Weakly Correlated" and 10 = "Strongly Correlated".

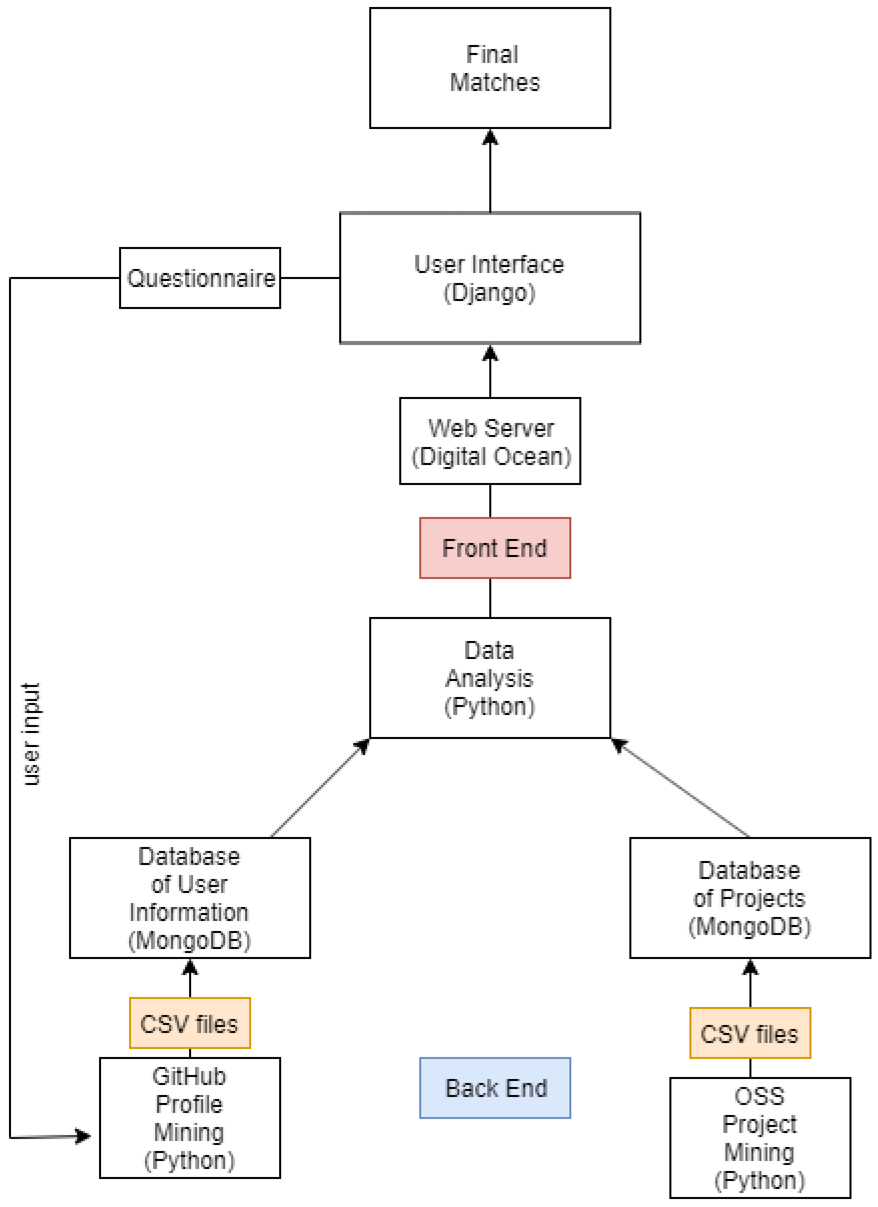
In conclusion, we believe that Django will be the best option for our web framework. It offers the highest quality features that will aid our product. While both React and Angular are highly scalable, they do not compare to Django. Django has proven to handle the worst webpage traffic as it is used by high profile website, hence earning a rating of 10. Another factor is databases, as they are a very important part of our product and we want to pick the best one. While Angular pairs well with MongoDB, Django would not be that difficult to either, giving them both a score of 10. React on the other hand does not offer any support, as it is front end only, making it a score of 0 in this category. All three of these frameworks score equally as high on the integration level, so we would have no problem adding any of them to our product. Lastly, we consider the learning curve of each framework. Django has excellent documentation and is relatively easy to learn if already familiar with Python, which the majority of our product will be written in. Easy it earns 10 in this category. React has a smaller library to learn and is not particularly difficult to understand, hence a small learning curve. However it's documentation is not quite as strong so it earns a score of 8. Angular lastly has the most difficult learning curve, which is worrisome as it may affect our the quality and speed of our work. While there are some useful tutorials, it gets a score of 5 in this category. Given the criteria, we can confidently say Django is our best solution.

## 3.6 Conclusion

In compilation, it was a tough process choosing between so many possible tools for this product. Topics mentioned above include Data Mining and Parsing Languages, Databases, Data Comparison Tools, Web Hosting Services, and Web Application Frameworks. Each of these subjects required careful comparison and analysis between each option, to find the best tool for each. In the end we ended up choosing Python for both the Data Mining/Parsing and Data Comparison, MongoDB for Databases, Digital Ocean for Web Hosting, and Django for Frameworks. Each of these came out on top of our scoring system, which was unique to each topic. Every tool had a few things in common, including scalability, reliability, and ease of use. Fortunately, these traits all synergize in a way that will allow the product to thrive, creating an enjoyable and impactful experience for end users. We will be testing our results with a live demo in the near future. Doing this will allow us to prove our results with action, on top of our research and analysis.

## 4 Technological Integration

**Figure 7 Software Architecture Chart**



Final Matches
Questionnaire
User Interface (Django)
Web Server (Digital Ocean)
Front End
Data Analysis (Python)
user input
Database of User Information (MongoDB)
Database of Projects (MongoDB)
CSV files
CSV files
GitHub Profile Mining (Python)
Back End
OSS Project Mining (Python)

**Figure 7:** Data architecture chart demonstrating how the final product will work.

Above we have created a website architecture chart to help further explain our product. Beginning with the back end, we will begin by mining the data of open source projects. This will include information such as programming languages the project is written in. All of this will be imported to our database via CSV files. This database will be updated before the product is available to the public, and updated regularly in the future. Another task on the back end will be mining the end user's GitHub for their experience. This will happen only after the user has provided their profile via authentication on the user interface. A short questionnaire will appear to add additional experience outside of GitHub, this will be optional unless the user has not made any commits. The data will then be exported to CSV files and put into a database for storage. Afterwards, the data will be processed and compared with the open source project database using our matching algorithm. Finally, the user will be given a list of matches available to them via the user interface.

There are 5 major components that we believe will help us produce a great product:

1. Data Mining and Parsing Tool
2. Database
3. Data Comparison Tool
4. Web Hosting
5. Web Framework

The data mining and parsing tool will be built twice, one for the user's skills and another for the open source project requirements. These tools will output the user's personal files and/or save to their database. The data comparison tool will interpret the two files created and output a file of its own or it will access the database and interpret from there. This last file or data in a database will be interpreted and sent to a web page where we will use a framework to design and display. This solution offers a great amount of modularity. The data mining and parsing tools will be completely separated from the data comparison tool which will be completely separated from the display of the webpage.

## 5 Conclusion

The Match Source team, along with our client Dr. Igor Steinmacher, will be working on a web application to pair GitHub users to OSS projects they are qualified to contribute to. The way the app will work is firstly, individuals will log into our system via user interface through their GitHub account. After, the system will begin to mine the data from their account. Lastly, optional questionnaire to add additional experience outside of GitHub will appear. If a user has not made any contributions, the questionnaire will be mandatory. Finally, once all the information is retrieved, the data will be compared to our preexisting database of open source projects and the result will then be displayed back to the end-user. We hope this product will ease the initial information overload GitHub users feel and thus lead to more contributions to open source software.

This document is an overview of the technological challenges we considered before moving forward into the development stage. Below is a table that summarizes our final solutions.

**Figure 8 Final Table of Solutions**

| Challenges | Proposed Solution | Confidence Level |
|---|---|---|
| *Data Mining and Parsing Tool* | Python | 4 |
| *Database* | MongoDB | 5 |
| *Data Comparison Tool* | Python | 5 |
| *Web Hosting* | Digital Ocean | 5 |
| *Web Framework* | Django | 5 |

**Figure 8:** Each solution is rated on a scale of 1-5 with 5 being highly confident, and 0 being no confidence at all.

We believe that with these proposed solutions we will be able to bring our client's vision to life. The Match Source team has thoroughly reviewed the pros and cons for each of our options. We have made sure they all offer the necessary features needed to create the final product. Most importantly they will help us provide the best possible service to future users. As always, adjustments will be made if new issues arise later on. Moving forward we hope that this product will accomplish our client's goals and change the way GitHub users go about finding their next OSS project.

## 6 REFERENCES

[1] PythonForBeginners. Benefits of Learning Python. 2016.  Retrieved October 22, 2019 from https://www.pythonforbeginners.com/learn-python/benefits-of-learning-python/

[2] Fossbytes.10 Reasons Why You Should Learn Java Programming Language. September 2017.  Retrieved October 22, 2019 from https://fossbytes.com/10-reasons-learn-java-programming-language/

[3] Develop your next app with your Azure free account. (n.d.). Retrieved October 22, 2019, from https://azure.microsoft.com/en-us/free/apps/search/

[4] Website & Web Apps Solutions | Google Cloud. (n.d.). Retrieved October 23, 2019, from https://cloud.google.com/solutions/websites/

[5] Smith, B. (2019, October 14). Digital Ocean - Is It Good Hosting & Server Provider? (2019 Review). Retrieved October 23, 2019, from https://hostingfacts.com/hosting-reviews/digitalocean/

[6] Libraries. (n.d.). Retrieved October 23, 2019, from https://developer.GitHub.com/v3/libraries/

[7] Ruby. (n.d.). Retrieved October 23, 2019, from https://www.ruby-lang.org/en/documentation/

[8] 32.1. parser - Access Python parse trees¶. (n.d.). Retrieved October 23, 2019, from https://docs.python.org/2/library/parser.html

[9] Paraschiv, E. (2018, March 1). JSON in Java. Retrieved October 23, 2019, from https://www.baeldung.com/java-json

[10] Jaiswal, K. (2019, February 2). Learn Web Scraping using Python in under 5 minutes. Retrieved October 23, 2019, from https://towardsdatascience.com/web-scraping-using-python-4cb2faade338

[11] Django. (n.d.). Retrieved October 23, 2019, from

https://www.djangoproject.com/.

[12]Advantages and Disadvantages of Django. (n.d.). Retrieved October 23, 2019, from https://hackernoon.com/advantages-and-disadvantages-of-django-499b1e20a2c5.

[13]Top 10 Web Development Frameworks in 2019: (n.d.) Retrieved October 23, 2019, from https://hackr.io/blog/top-10-web-development-frameworks-in-2019

[14]Creating websites using React and Django REST Framework. (n.d.). Retrieved October 23, 2019, from https://hackernoon.com/creating-websites-using-react-and-django-rest-framework-b14c066087c7.

[15] Parthasarathy. (2018, September 6). Building Scalable Applications using React - Part 1: Choosing the right mix of tools. Retrieved October 23, 2019, from https://medium.com/finiteloop-systems/building-scalable-applications-using-react-part-1-choosing-the-right-mix-of-tools-74d5afd9e854.

[16] React vs Angular: An In-depth Comparison. (2019, January 30). Retrieved October 23, 2019, from https://www.sitepoint.com/react-vs-angular/.

[17] The Good and the Bad of ReactJS and React Native. (n.d.). Retrieved October 23, 2019, from https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-reactjs-and-react-native/.

[18] AngularJS. (2019, October 7). Retrieved October 23, 2019, from https://en.wikipedia.org/wiki/AngularJS.

[19] Angular. (n.d.). Retrieved October 23, 2019, from https://angular.io/.

[20] Angular Pros, Cons, Features, and More. (2019, June 12). Retrieved October 23, 2019, from https://www.pluralsight.com/blog/software-development/angular-101.

[21] PostgreSQL vs. MongoDB. (2018, September 27). Retrieved October 23, 2019, from https://blog.panoply.io/postgresql-vs-mongodb.

[22]  Cassandra vs MongoDB - 8 Major Factors of Difference. (2018, November 16). Retrieved October 23, 2019, from
https://data-flair.training/blogs/cassandra-vs-mongodb/#.

[23]  MongoDB: The Good, The Bad, and The Ugly - DZone Database. (2016, October 11).Retrieved October 23, 2019, from
https://dzone.com/articles/mongodb-the-good-the-bad-and-the-ugly.

[24] Postgresql vs. MongoDB for Storing JSON Database. (2019, August 22). Retrieved October 23, 2019, from
https://www.sisense.com/blog/postgres-vs-mongodb-for-storing-json-data/.

[25] Divide and Conquer: High Scalability With MongoDB Sharding - DZone Database. (2016, November 1). Retrieved October 23, 2019, from
https://dzone.com/articles/divide-and-conquer-high-scalability-with-mongodb-t.

[26] Scalability Benchmarking: MongoDB and NoSQL Systems. (n.d.). Retrieved November 4, 2019, from
http://info-mongodb-com.s3.amazonaws.com/Scalability-Benchmarking-MongoDB-and-NoSQL-Systems.pdf

[27] How to Benchmark PostgreSQL Performance. (2019, August 21). Retrieved November 5, 2019, from
https://severalnines.com/blog/benchmarking-postgresql-performance.

[28]Ruby Explained: Objects and Methods. Retrieved November 5, 2019, from
https://www.eriktrautman.com/posts/ruby-explained-objects-and-methods

[29] Pricing - App Service. Retrieved November 6, 2019, from
https://azure.microsoft.com/en-us/pricing/details/app-service/windows/

[30] Managed Cloud Hosting. Retrieved November 6, 2019, from
https://www.cloudways.com/en/pricing.php

[31]Java Examples, Retrieved November 5th, 2019 from
https://press.rebus.community/programmingfundamentals/chapter/java-examples-1/

[32]What is Python? Executive Summary, Retrieved November 5th, 2019 from
https://www.python.org/doc/essays/blurb/

[33]Flexible & Transparent Pricing, Retrieved November 5th, 2019 from https://www.cloudways.com/en/pricing.php