Technological Feasibility Report

Git OSS-um

Gary Baker
Van Steinbrenner
Stephen White

November 5, 2018

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

Faculty Mentor: Ana Paula Chaves Steinmacher

Overview

The purpose of this Technological Feasibility Report is to document the technological challenges that need to be overcome to solve the problem presented by the project sponsor and explore the feasibility of possible solutions. This report also documents the research and calculated conclusion on best fit technologies for each technological challenge to be faced in the process of completing this project.

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.

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. There is a need in the open source community for a tool that will allow a novice passing through to quickly and efficiently see how "newcomer friendly" a repository is.

Dr. Steinmacher and his colleagues have created a web application to address this need, called FLOSScoach[1], where newcomers can investigate the skills needed to contribute to open source. Although FLOSScoach has been helpful, it has its shortcomings: FLOSScoach is manually fed, must be constantly maintained to provide updated information, and displays paragraph descriptions instead of visual representations of collected data to users. These

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

limitations restrict use of the project to already available information and does not assist with finding new projects to contribute to.

Due to the shortcomings of the current solution Dr. Steinmacher and his colleagues have come up with, we will be spearheading a new solution. 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, thereby allowing them to decide if the specific OSS project will support their needs. This will allow them to feel at ease when picking an open source project to contribute to.

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.

2. Technological Challenges

Every technical project introduces barriers and challenges to overcome in order to produce a working product. After careful analysis by the team, we have classified five technical challenges that we must be solved. These five challenges are broken down below.

- Data Mining Tool:

We will need to select a programming language that will allow us to mine data from the GitHub API that will then be analyzed.

- Database:

We will need to store the data that we have collected in a way that will be easily accessible and provide us with tools for generating cohesive reports.

- Analysis Tool:

We will need to analyze the data that we have collected and stored in order to produce interactive representations of information.

- Web Hosting Service:

We will need to host our web application through a hosting service that will allow us to easily integrate solutions to the aforementioned challenges.

- Website Framework:

We will need to make use of website framework that will allow us to present the analyzed information in a comprehensive and intuitive way for the end-user.

By understanding the overview of the technological challenges the team will be facing, we will make informed decisions about what tools would best come together to form our product, and create a solid base from which we can successfully develop this software. In the next section, we will break down each technological challenge, introducing possible solutions to each, and justifying our conclusions as to what the best option to overcome said challenge will be.

3. Technological Analysis

In the previous section we have identified five technical challenges for the team to overcome in order to produce a viable product. Within this section, we break down each challenge into possible solutions, and analyze said solutions to determine what is the best plan of action. This will provide sufficient justification as to why a specific technology has been chosen over another.

3.1 Data Mining Tool

The first piece of functionality that our project must have is the ability to mine data from GitHub. GitHub consolidates information about repositories, pull-requests, commits, users, and issues in a format known as JavaScript Object Notation (JSON). This information is readily available through querying the GitHub API, however, our issue is to find a tool that will easily allow us to make requests to the API, and download these JSONs for storage and analysis.

In order to decide which programming language best fits our needs, we must uncover one that will provide us with the best balance of learnability and existing tools. Specifically, tools that will allow us to make requests to the GitHub API, download the JSON file returned by the request, and parse that JSON file if need be. Although there are a great deal of potential options to choose from, we have narrowed down our choices to three possible options, namely Python, Java, and C#. Upon collaborating and researching different potential pathways, we believe that these are the top options that fit our needs as well as match our skillset.

3.1.1 Python

The python programming language comes with the benefit of a large open source community that creates a great deal of packages useful for data mining. The two most beneficial libraries that would assist us in mining data from GitHub would be “requests” and “json.” These two libraries allow us to make outgoing requests to the GitHub API through HTTPS Basic Authentication as well as OAuthentication, and to download/interpret JSON files with ease. Another positive factor for Python is that it is very easy to quickly write code that is easy to understand and that functions properly. Each member of the team has used this language before, and it would only be a matter of understanding these two libraries.

3.1.2 Java

The Java programming language also provides tools for making an HTTPS request, and reading in a JSON response, however, the process does not appear to be as intuitive as Python. Making use of the HttpsURLConnection class, the team would be able to validate ourselves to the GitHub API and successfully create GET requests to return a valid JSON object. We would then make use of the JSONObject class to interpret the results of the request for future storage. While the team has extensive knowledge of the Java programming language and the Object-Oriented Paradigm, making requests, interpreting results, and saving those results to a file is much more difficult in this language, and a greater period of time would be spent simply attempting to replicate (in much more lines of code) what Python can do much more simply.

3.1.3 C#

The C# language would allow us to make use of the .NET framework to make outgoing requests to the GitHub API as well. Within C# there exists a class by the name of WebRequests (coming from the System.NET namespace) that will make outgoing requests to the API, and the WebClient class that can download JSON files returned by the request. Within C# however, parsing JSON objects is quite tricky, and would require “deserialization” of a raw JSON object, and this functionality is not native to Visual Studio. While C# would be easier for the team to understand, as it is very similar to Java, it comes with the overhead of Visual Studio for executing code, and it may be more difficult to integrate into a web framework down the line.

3.1.4 Conclusions

The following table rates each of the options on a scale from 1 to 5 of how well it works for the specific feature (one being it does not work at all and five being it works well).

Figure 1: Dating Mining Tool Scores

	Ease of making Requests to GitHub API	Downloading JSON Files	Parsing JSON Files	Learnability	**Total Score**
Python	5	5	5	5	**20**
Java	3	3	2	3	**11**
C#	2	3	1	2	**8**

Figure 1: Visualized by the table above, with the three programming languages being rated one to five with five representing the highest value of acceptance possible, Python won with a perfect score of 20. The other two languages (Java and C#) were too difficult to make use, earning a scores of 11 and 8 respectively.

To decide which programming language would best fit our needs in regard to mining JSON files from the GitHub API, we believe that Python will provide us with the best solution. We plan on developing the data mining portion of the application in Python, making use of the “requests” library to obtain data from GitHub, and interpreting JSON files as needed with the “JSON” library. In the event that the data mining tool also needs to create CSV files, Python will grant us this ability as an added bonus.

With our data successfully being mined, it is only logical that we come up with a plan to properly store it for future manipulation and interpretation. This brings up the need for our team to decide on the proper database to house mined JSON files.

3.2 Databases

In order for our product to be useful, we must be able to store the data that we have mined from the GitHub API. Data will have been collected, and returned in the form of individual JSON files; the database must be capable of storing information in this format. In addition, we must be able to retrieve it using the key:value pairs.

When retrieving the data for analysis, it will be exported to a CSV file in order to be interpreted by the data analysis tool, allowing the analysis tool to return correct representations for the user quickly and easily. We will be researching MongoDB and MySQL as the main technologies for storing the data that we mine as well as Django and Passport.js for storing our user authentication information. These are the technologies we chose to research because we are mostly familiar with them and they should fit easily into our framework that we choose for the web application.

3.2.1 MongoDB

The MongoDB database solution is a document based, noSQL, non-relational database program that uses JSON-like documents with schemata to store information. Some of the main features of MongoDB include ad hoc queries, file storage, and server-side JavaScript execution. Ad hoc queries will be useful to allow our application to make queries on the fly, then store the files received from the GitHub API using MongoDB file storage, and finally execute JavaScript functions on the data to generate visual representations of the data.

Queries can also be configured to return sample results of a given size. Fields in the MongoDB can be indexed for increased ease of use. MongoDB provides replica sets which include two or more copies of the data which can be useful in the event of corruption or loss of data. The replica sets can also potentially be useful for using one set for certain data collection and a secondary set for other collection to increase efficiency. This database can also be used as a file system with load balancing and data replication features for storing files over multiple machines.

3.2.2 MySQL

MySQL is an open source relational database management system owned by Oracle Corporation. MySQL has a wide array of key features that it has to offer, most of the same features as mentioned for MongoDB as well as many others that are specific to the relational model.

The relational model is an approach to managing data using a structure and language consistent with first-order predicate logic, where all data is represented in terms of tuples, grouped into relations. Typically used in the relational model is the SQL language and the data is set up to be stored based upon a system of tables that are connected by the previously mentioned relations.

MySQL is a very powerful database management solution with many different applications, however as the size of the data that needs to be stored starts getting large, the efficiency of retrieving that data begins to decrease. This seems to be an important issue for our project since the user will be requesting data on the fly and our database must be able to react and return the information needed for analysis in a very efficient way in order to give the user as good of an experience as possible.

The other downfall of MySQL is that the format of the data as it is stored is less straightforward from the perspective of parsing the data and exporting the data for analysis. This is because the format we will be receiving the data from GitHub is JSON and if we were able to just simply store the data as it is presented to us from GitHub, it would cut out a few steps in between storing and exporting the data for analysis.

3.2.3 Django

For our product to be more useful and personalized to each user, we must have a user authentication system. One of the main web frameworks that we have looked into using is Django, Python's add-on for web application development. Django has a User Authentication package which starts as a framework, then execution of a management file creates the needed database that takes care of authentication. User authentication is used for confirming that the user is who they claim to be, and authorization determines what that user is authorized to do within the web application.

Django's Authentication is easily customizable and comes with many different features on top of the typical authentication and authorization, a few of these features include password strength checking, throttling of login attempts, and a pluggable back-end system.

3.2.4 Passport.js

Passport.js is another option of user authentication and storing all the application's user information. This package is an add-on to Node.js which is another web framework that we are considering using for our web application. Passport.js offers all of the same base features of Django and advertises its ease of adding into any express-based web application.

Some nice to have features that Passport.js includes are many different forms of login authentication including username and password, Facebook, and Twitter to name just a few. Passport.js also includes support for persistent sessions and provides many other features with a lightweight codebase.

3.2.5 Conclusions

The following table rates each of the options on a scale from 1 to 5 of how well it works for the specific feature (One being it does not work at all and five being it works well).

Figure 2: Database Tool Scores

	Efficiency	Scalability	Web App Integration	Data Extraction	**Total Score**
MongoDB	5	5	5	5	**20**
MySQL	4	5	3	2	**14**
Django	5	5	4	5	**19**
Passport.js	5	5	5	5	**20**

Figure 2: Shown by the table above, with all of the database solutions being rated one to five with five representing the highest value of acceptance possible, MongoDB won with a perfect score of 20 for the general data storage and Passport.js won with a perfect score of 20 as well for handling user authentication storage. MySQL appeared as though the storage format of the data and data extraction would not easily fit our needs, earning a scores of 14. Django's user authentication would be a useful option but the integration into our web application would be just slightly less of a good fit than Passport.js, Django scored a nearly perfect score of 19.

After further analysis of these database solutions, it has become apparent that MongoDB is going to make the most sense for our data storage system due to its base storage format being JSON-like files which is exactly how the data is being collected from the GitHub

API and being document-based, the efficiency of the database will remain high whereas MySQL will slow down when the scale increases to a fairly large dataset. The last reason that MongoDB makes the most sense is that JSON is easily exported to CSV format, which is the file type that will be passed to data analytics tools for easy parsing and use of the data contained. MySQL is less straightforward in terms of parsing the data and converting that to CSV in comparison due to how simple it is to transition from JSON to CSV.

As far as our user authentication storage, Passport.js looks like the way to go simply because it will be just a bit easier to integrate into our application than Django's user authentication solution. Overall, in order to have efficient access to our data, keep transition from collected data to stored data as simple as possible, and to have as smooth of a conversion from our database to CSV files for analyzing, MongoDB is the way to go. Furthermore, in order to have a robust and easily integrated user authentication system, Passport.js is the best option for our application.

The next challenge in the project workflow is to analyze the data in a way to make it useful for our end users. In order to do this, we must explore options for data analytics tools.

3.3 Data Analysis Tools

Data is everywhere, and given the appropriate knowledge, what can appear to be nothing but a collection of numbers, trues/falses, and categories can in reality, become a wealth of valuable information. In order to draw conclusions from data, the team must access a great deal of it, and with assistance from the chosen database, we will easily be able to access large amounts of repository data. After information has been stored within the database, a (nearly universally accepted) file format known as a Comma Separated Value can be generated and plugged into a separate tool to be analyzed.

In order to decide what tool would best serve our team to analyze packaged data, we must determine what tool will provide the best balance between statistical computing capabilities, ease of integration into a web application, difficulty to develop, and the ability to visualize data in an interactive manner. There are several tools in existence that provide the ability to perform simple analysis on datasets, however, the team has narrowed down the best options to R, Python, and JavaScript as they provide us with the optimal balance described above.

3.3.1 R

R is a statistical language aimed at uncovering the value behind a dataset. The value in utilizing this analysis tool comes with the support of a great number of open source libraries, including but not limited to data.table, HTML Widgets, Dplyr, TidyR, and Shiny. Along with a great deal of tools to analyze data, R also presents options that would allow us to produce interactive graphics, which is something we must take into account. There are a variety of ways of doing this involving connecting JavaScript to R, or making use of the Shiny package.

Considering that R is a statistics-based language we will natively be able to conduct a great variety of analyses on the data that we will be collecting. The team would need to obtain more experience with this language to understand how to analyze the collected data and how this tool connects to the web.

3.3.2 Python

Python's major advantage over other programming languages is that it is, at its core, a general-purpose language. As explained in the *Data Mining Tool* section of this document, Python gives us the opportunity to leverage numerous open source libraries to analyze collected

data, such as Numpy, Matplotlib, Scikit-Learn, Pandas, and Tensorflow. The majority of Python libraries that exist, however are only useful for producing static representations of analyzed data.

As far as meeting our needs goes, Python is more than capable of filtering and analyzing data. Beyond that, the more difficult question to answer would be how we would visualize the data with this language on a web application, given that Python's reputation as a general-purpose language does not lend itself to interactive representations that well.

3.3.3 JavaScript

JavaScript has the ability to conduct statistical operations thanks to the use of Node Package Manager. There exist several options for computing statistics within JavaScript, such as Jstat, Simple Statistics, DataSet.js, and D3.js. It appears that visualizing data in an interactive manner is also very possible due to the availability of open source packages that we can install.

The team has a varying level of knowledge of JavaScript, and therefore there may be a bit of a learning curve to understand the syntax along with what packages we would make use of to analyze data. This is quite possibly the most surprising contender for the data analysis tool, due to our preconceived knowledge of what JavaScript is used for, however, it can definitely stand up for itself when competing against R and Python.

3.3.4 Conclusions

The following table rates each of the options on a scale from 1 to 5 of how well it works for the specific feature (one being "it does not work at all" and five being "it works well").

Figure 3: Data Analysis Tool Scores

	Statistical	Learnability	Web App	Interactive	**Total**

	Computing		Integration	Graphics	**Score**
R	5	3	3	4	**15**
Python	5	5	3	2	**15**
JavaScript	4	4	5	5	**18**

Figure 3: As you see by the table above, after categorizing the necessary components of a statistical tool and ranking them from one to five, with five being the best, JavaScript won with a score of 18, while R and Python tied with a score of 15.

With our original challenge of needing to decide on what tool would best serve the team in regard to performing statistical analysis on datasets in the form of a CSV file, and producing interactive graphics that would be intuitive to the end user, we believe that JavaScript would be the optimal solution. We plan on making use of the immense amount of open source libraries available to JavaScript through Node Package Manager to parse the data that we have collected, and come up with a sleek graphical user interface that can be manipulated by our users.

Mining and analyzing data is only useful as long as the team is able to share that analyzed data with the world. This presents our next technological challenge: selecting a web hosting service that will allow us to build our web application.

3.4 Web Hosting Services

A large component of our web application is the choice of web hosting. This can be a complicated issue for us as students, however, our sponsor will be able to take care of any payments of our project, this includes our web hosting service for our web application. Our goal is to find a service that can handle large scale data mining and have data visualization as an output from a user’s request.

In order to make the best selection, we have four available options of services that could give us the best performance, pricing, scalability, storage, and security; Along with unique features that each service is capable of to further compare our options.

3.4.1 Digital Ocean

Digital Ocean emphasizes their “Droplets” of “scalable platform of compute instances” for their web hosting service. These Droplets have add-on storage, security, and monitoring capabilities for our web application. Besides the Droplets, Digital Ocean has 8 data centers across the globe, giving us reliable connections wherever our users are. To assist administrators, Digital Ocean comes with many features such as monitoring droplet performance visibility, team management system to scale our app, storage that can be upgraded at any time, and cluster deployment of Droplets to grab more data. Arguably, the best feature of the Digital Ocean service is the Elegant API, available in the Ruby, Curl, Go, and Doctl languages. The Elegant API enables administrators to deploy and manage Droplets programmatically. The API also has conventional HTTP requests and allows OAuth support. Sadly, Digital Ocean’s hosting service is not free, however, it does have a pay-as-you-go pricing with explicit pricing of features per measurement.

3.4.2 Amazon Web Services

Amazon Web Services promotes free product offers and services to assist in building web applications, however, we do not know how long the “free” offers and services will last for, especially after we finish the product in late Spring 2019. There are specific services that AWS offers that could prove useful to us, this includes Amazon EC2, AWS Lambda, and Amazon S3. Amazon EC2 offers 750 hours per month Linux and other server instance usage with resizable computing capacity in the Cloud. AWS Lambda would run our code in response to certain events and manage resources with one million free requests per month and 3.2 million seconds

of compute time per month. Amazon S3 is a scalable storage infrastructure that offers five gigabytes of standard storage, 20,000 get requests, and 2,000 put requests. AWS also offers free tutorials on how to use their services. Unlike Digital Ocean, most of AWS offers free services with opportunities to expand our storage and other functionality that can be relevant to our web application.

3.4.3 Microsoft Azure

Microsoft Azure offers many features and offers, and a free trial for 30 days. These features that we can take advantage of and Azure has the most data centers in the world, offering connectivity for all of our users. Features that we can take advantage of is the Linux virtual machines, a cross-platform file storage system, and access to Azure Cosmos DB, a noSQL database that can assist us with data that is well documented and in large volume. After our free trial, Azure offers a plan of pay-as-you-go payment similar to Digital Ocean.

3.4.4 Google Cloud

Google Cloud promotes a cheap, fast, and scalable platform where web apps can be modified with frameworks such as Django and Flask. The platform also offers cloud computing with storage, data transfers and migration, noSQL in the form of Cloud Bigtable, and big data analytics. Google Cloud offers pay-as-you-go payment plans similar to our other choices. Google Cloud allows developers to implement AI to applications so that data can be interpreted easier, therefore, make visualization and other functions of our web application easier. Serverless computing is also an opportunity where we do not have to manage server infrastructure, the less we intervene, the more functionality we can complete with serverless computing. Google cloud also offers large private regions and networks where connectivity to users will be strong without any obstructions.

3.4.5 Conclusions

The table below is a representation of features and services that a web host offers and compares these services to one another to find the best solution.

Figure 4: Web Hosting Service Scores

	Efficiency	Scalability	Storage and Security	Pricing	**Total Score**
Digital Ocean	5	5	5	5	**20**
AWS	4	4	4	3	**15**
Azure	3	3	3	4	**13**
Google Cloud	4	4	4	2	**14**

Figure 4: From the results above, our best option is the Digital Ocean platform for web hosting. Our other options, in order from best to worst options include: AWS at 15, Google Cloud at 14, and Microsoft Azure at 12.

Choosing an efficient, affordable, and secure web hosting service is the key to our web application performing at its best. The storage, security, and scalability of our web hosting service will greatly approve.

Although having a strong web hosting service is key to a fast and efficient web app, it is not the only part of having a great web app, a valuable framework for the website and data visualization is essential for our web app to perform at its best.

3.5 Web Application Frameworks

Creating a beautiful and bold website involves choosing the best framework is not an easy fit. We must choose a framework that will assist in making an efficient UI and work well with our data visualization framework of our choosing. The connections between our web and

data visualization frameworks is the key to create a user-friendly web app with great performance.

Our options for web frameworks include easy to use frameworks like Django, Node.js and Mean.js for compatibility across multiple frameworks and databases such as MongoDB, Angular 6 for more JavaScript compatibility, and finally Bootstrap for creating an efficient UI.

3.5.1 Django

Django is a Python based web framework built for developers with tight deadlines and perfectionist mentalities. Django is quick, secure, and scalable, which is exactly what we need for our application. To assist us, Django offers a highly detailed documentation page on their website; The documentation extends from tutorials to documented functionality from security to data validation. This open-source web framework has other built in functionality, as stated earlier, with databases, and can store information for user authentication. Conveniently, Django has multi-database functionality, so we can pull user authentication data from Django and get data that we mined from another database, if need be. Django also has the ability to run other Python or JavaScript data libraries into the web app.

3.5.2 Node.js

Node.js is an open-source and scalable web development framework that treats HTTP requests first other than anything else. Node.js is compatible with noSQL databases such as MongoDB and relational databases like mySQL. Despite some research, integrating multiple databases can be a pain for developers. However, JavaScript offers many frameworks for data visualization including plotly for Node.js. Despite this, the overall documentation of the Node.js project is decently organized but it does not offer too much information regarding data

visualization or database integration. However, HTTP requests are a decent part of the documentation.

3.5.3 Mean.js

Mean.js is another alternative to Node.js where Mean has access to MongoDB, AngularJS, Node.js, and Express. With the implementation of these other frameworks, there will be less detours in development and more opportunity to increase the functionality of our project. Mean is an open-source project that is still under development, which, under further development of the Mean project, could make our web application malfunction when the Mean software is upgraded to new versions. Despite this drawback, Mean appears to be a great tool that draws power from other frameworks, combined into one convenient package.

3.5.4 Angular 6

Angular is another JavaScript framework that optimizes speed, scalability, and performance for any application. Angular also assists developers in creating data models with libraries such as Immutable.js or other libraries. The main aspect of Angular includes incorporating other libraries using observables for uses like data visualization and multiple database integrations. The documentation for Angular is decent and has tutorials and beginner articles, but does not go too far in depth on certain topics like database integrations. Overall, Angular is a solid framework for incorporating as much functionality in the back-end as possible.

3.5.5 Bootstrap

Bootstrap is one of the most popular web development frameworks in the world, and is used to create beautiful websites using HTML, CSS, and JavaScript. Because Bootstrap is exclusively a front-end framework, database integration and interactive graphics will have to be done with back-end technologies such as the frameworks described above. Bootstrap can be

used to effectively create a beautiful UI for users to explore, create a responsive web app on any screen, and separately implement a back-end framework to deploy interactive graphics from collected data, the backbone of our project. Bootstrap does, however, implement responsive charts, but are not as scalable nor as fast as the frameworks described above.

3.5.6 Conclusions

Below is a table representing our top choices of website frameworks that we should use to create our project.

Figure 5: Web Frameworks Scores

	Efficiency	Scalability	Overall Integration	Documentation	**Total Score**
Django	5	5	5	4	**19**
Node.js	4	4	4	4	**16**
Mean.js	4	5	5	3	**17**
Angular	5	5	5	4	**19**
Bootstrap	3	1	3	4	**11**

Figure 5: From the table above, the best options for us include the Django and Angular frameworks at a tie of 19 out of 20. Other frameworks follow closely behind in the points system, however, there is a chance that we will be using more than one framework for our web application.

After reviewing our options, we have found that our best web frameworks are Django and Angular. Building our web application will have more than one standalone framework, as we may need the extra functionality to import certain functionalities in our app.

Within our technological analysis, we have discussed the five technological challenges (data mining tools, appropriate databases, available analysis tools, web hosting services, web frameworks), providing possible solutions to each, determining the optimal solution, and providing sufficient justification for our choice. The individual components of the solutions we have provided would prove useless unless there was a valid plan to integrate all technologies

into one cohesive workflow. Within the next section, we will cover how each technology will interact with one another and provide a graphical representation of what we plan on developing.

4. Technology Integration

Figure 6: Technology Integration Workflow

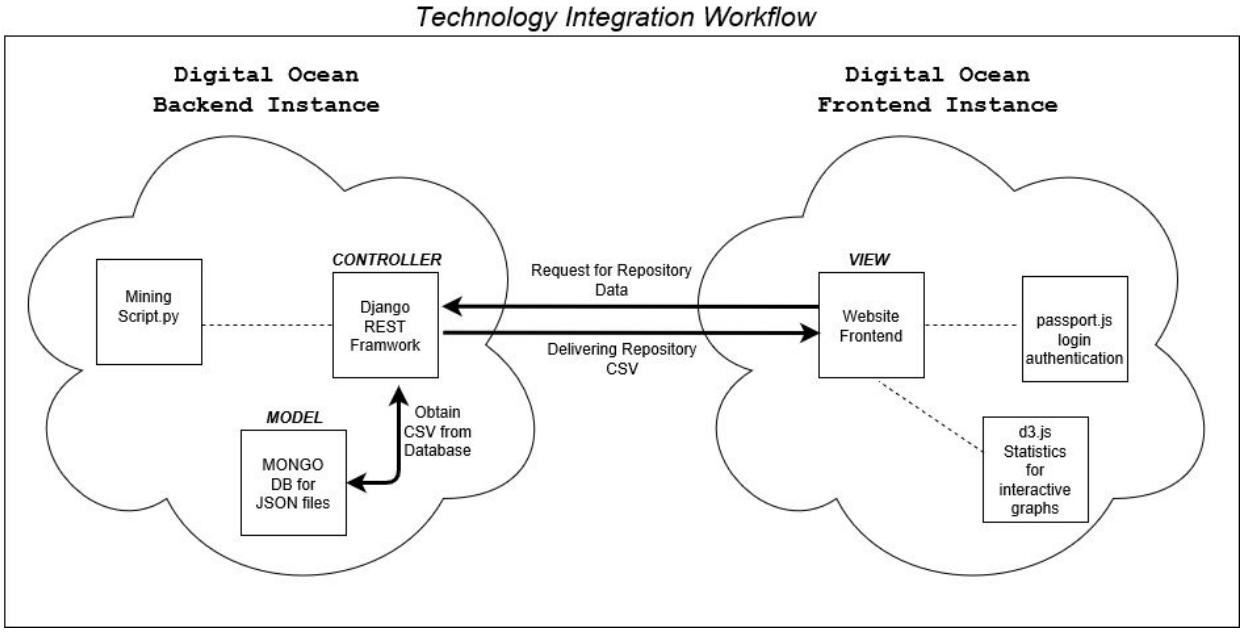

Figure 6 Description: The image above is a visual representation of the workflow of our product. The back-end instance consists of mining scripts, databases, and REST framework. The front-end instance consists of the user interface, a passport.js login authentication, and a D3.js framework for statistics and interactive graphics.

Above is a diagram of our planned prototype system. We envision having two separate Digital Ocean instances that will hold the backend and frontend portions of our product respectively. Within the backend portion of the product, we will make use of the Django REST Framework to be used as an overall controller for every other piece to go through. Through REST API calls, this controller will be able to interact with the data mining script to collect information, and place the collected JSON files within our model: the Mongo DB database system. Mongo DB is simply the container that holds all collected information, and can be queried to analyze information.

The second Digital Ocean instance that houses the frontend of the application will contain all information relevant to displaying information to the end user and allowing them to

interact with the website. This in essence, will contain the view of the project. When someone types information into a field to mine a repository, the view will pass that repository name to the controller, which will then begin collecting JSON data. The controller then places the JSON information into the model, generates a csv file for parsing, and passes that back to the view where JavaScript will take care of analyzing the data, and presenting interactive graphics to the user!

5. Conclusion

In summary, with open source projects becoming a massive market and increasingly more important to the computer science industry, we feel that not only is it critical for people to continue to contribute, but also encourage newcomers to feel welcome to these projects. Our web application will:

- Display relevant data to GitHub newcomers about a requested repository.
- Assist in choosing a good first repository to contribute to by helping the user determine what the community is like.
- Show newcomers that they are not alone and encourage them to contribute to the world of Open Source Software.

Our goal is to assist Dr. Steinmacher with overcoming the challenges newcomers face whilst contributing to open source projects, and keep them from being discouraged. In doing so, we will be giving power back to the newcomer and providing a valuable resource for the world to share.

The following table gives a summary of the technologies that we have chosen and our confidence level that each of these technologies can be used to successfully complete our project.

Figure 7: Chosen Technology Confidence Table

	Proposed Solution	**Confidence Level**
Data Mining Tool	Python	**5**
Database	MongDB/Passport.js	**5**
Analysis Tool	JavaScript	**5**
Web Hosting Service	Digital Ocean	**5**
Web Application	Django	**5**
	Total Confidence	**25**

Figure 7 Description: We have picked these technologies with the end user's experience and implementation efficiency, and task effectiveness in mind. Using Python for our data mining tool, MongoDB for our database of mined data, Passport.js for our user authentication services, JavaScript for our data analytics tool, Digital Ocean for our web hosting service, and Django for our web application framework.

These decisions made on the technologies to be used will allow us be as efficient as possible while developing in order to meet our deadlines, effectively achieve our goal of displaying useful information to the user as well as do so efficiently to keep the user experience streamlined and lightweight, and aid the open source newcomer in having a positive and successful first experience in open source contributing and encouraging them to continue making contributions to the open source software and computer science community.