**Programming Language:**

**Summary:** In order to achieve the purpose of building a web-based interactive map, we have decided to use JavaScript and its extensive libraries and frameworks.

**Problem:** To develop an interactive map and satisfy client’s requirements, we need to build a web-application to achieve this purpose. Web development can be achieved by multiple languages and different tools, but we would like our code to be written in popular languages so that it is easy to understand and can be easily taken over.

**Constraints:** There are many options when it comes to web development so choosing a language that all of the teammates are familiar with could be challenging due to the differences in experiences.

**Options:** Based on the experiences we have, we targeted languages: C#, JavaScript and Ruby. For C#, one of us is very familiar with C# and two of us have experience in C/C++ so it would be easy for all of use to pick up, but none of us have experience in using C# for website development. Ruby on Rails is an easy to use framework for website development and one of us have a bit of experience in using it for website development, however the other two teammates are extremely unfamiliar with Ruby. For JavaScript, is it very popular and two of the three of us have abundant experience in JavaScript, even though one of us never used it for website development.

**Rationables:** We decided to use JavaScript, because it is widely used in website development with abundant libraries, frameworks for better structuring the code and a large amount of online resources for help. It would also be extremely easy for people to take over the project and maintain in the future.

**Front end:**

**Problem**: We need to build a straightforward, user-friendly map, as a dynamic demonstration of collection usage in different time scopes, to show the specimen record’s access history and trend.

**Constraints**: Our primary constraint is how to make the map straightforward and display the information we need in a user-friendly way. From our prototype, we can see that it is hard to display the information without confusing the users. We are also constrained by the “map” api that we build on. Currently we are considering Mapbox and Google Map API, but we need to make a decision based on further exploration.

**Options:**Since we decided to write code in Javascript, there are tons of frontend libraries/frameworks we can choose. React and Angular become our top two choices. Both appear to be great options, but react has more tutorials available and has a bigger user community.

**Rationale:** We have decided to use React as our frontend. It’s easy to start with, easy to use and comes with a good supply of tutorials and a larger community support, making the learning curve shallow and overall a smoother experience for us.

**Summary**: In order to develop an user friendly frontend, we will create a frontend based on our axure prototype using react to display the dynamic map and dynamic data.

**Backend:**

**Summary:** In order to achieve the purpose of finding data of plants and filtering the data we need, we have decided to use Node.js and Express.

**Problem:** In the backend, we need to find the data of accessed plants in the database and filter the data to fit our requirements, by the time-period options selected from our users. We also need to be able to create login/register forms.

**Constraints:** One important constraint that we have to consider is that we do have the data yet so we do not know how the data is structured, therefore, we are not able to determine how we are going to actually write the functions to achieve the purposes.

**Options:** For a full stack web development, it’s a natural choice to use Node + express + JSON + jest since we decided to use React as frontend.. Other options includes Python Django, Java Spring and C#, PHP. Node.js offers an easy Scalability, easy to learn, large and active community support and consistency with the frontend and other aspects of our project. Other languages would also be helpful because it would be a great opportunity for learning new technologies but it is also the disadvantage that we need to learn them from scratch.

**Rationale:** We decided to use Node.js and Express as our backend technologies, for consistency with other frameworks, libraries and the programming language we use. It is very popular to use MongoDB, Express, React.js and Node.js for website development so it is very easy to get help online and for the project to be taken over

**Database:**

**Summary:** In order to get a fast and flexible database, we choose to use MongoDB.

**Problem:** We need to build an easy and quick database to store the information of the accessed plants and the users who access the plant records. If the database is built inefficiently, the whole user experience will be affected.

**Constraints:** The size of the data we need is expected to be relatively small, but some of the information may be missing for some of the plants, hence, we would need the structure of the database to be relatively flexible. Another big issue is we do not know the structure of the database that the client’s current website uses, so it would be hard for us to determine how we are going to structure our database, if it is needed. Besides, we also need to choose a database that we are all familiar with, so that we can construct a fast and accurate database.

**Options:** We decided to choose from SQL and MongoDB. SQL is severely restricted, hard to tolerate mistakes and non-flexible, even though it has been very popular for a long time and an important tool for developers to know. MongoDB is more flexible, and it has better performance. The data in MongoDB can also be made use of in an easier way. However, MongoDB has some of the data redundancy.

**Rationale:** We finally choose MongoDB. As we have mentioned above, the data we have is relatively small and we are not sure about the structure of the dataset so we need flexibility. We don't care a lot about the redundancy. We care more about the performance of the website.

**Deployment:**

**Summary:** In order to host the website, we chose to use Heroku.

**Problem:** Our website needs to be hosted somewhere in order for our client and other people to see the website.

**Constraints:** None of us have much experience in deploying an application or we don’t really have too much knowledge on the differences between the different options.

**Options:** When it comes to hosts, there are many options. Most of the cloud services are paid services. Heroku is free and very popular, but we have to pay for it if too many people would like to access the website.

**Rationale:** We finally choose Heroku. We are not expecting a large amount of people to access the website and heroku is also free of cost.

**Data Connection:**

**Summary:** In order to get a fast and flexible database, we choose to use MongoDB.

**Problem:** Our data lives on the database of our client’s original website and we need to find a way to connect to that database to get the data into our application.

**Constraints:** We don’t know what database our client’s website is using. We don’t know how the data is structured and we don’t know how to connect to it and use it.

**Options:** For now, we can think of options of how frequently we want to update the data. We have the option of building a real-time data connection, or updating the data every day, few days, weeks, or months. The real-time application will probably need to directly connect to the database, which we don’t have knowledge of, but it would probably display the most updated information the fastest. Other options would not hurt this project very much, as we don’t expect the application to be accessed too often by users other than our client.

**Rationale:** We currently decided to update the data once a day, but that is something we need to talk with our client about.

**Architecture Diagram:**

