Technological Feasibility Analysis

October 9, 2020

LOST EXPRES

Sponsor: Joe Llama

Mentor: Volodymyr Saruta

Team Lead: Brooke Caldwell

Architect: Jared Cox

Customer Communicator: Olivia Thoney

Release Manager: Ian McIlrath

Meeting Recorder: Austin Bacon

Table of Contents

1 Introduction

The *Star Wars* Saga, *2001 Space Odyssey*, *Alien*, are all examples of how people's obsession with the existence of familiar life in the universe has made its way into pop culture and affected entire generations. At Lowell Observatory, Dr. Joe Llama and his team are discovering what was once only science fiction. They are gathering millions of data points on the sun every minute of every day to get a better understanding of what makes the sun support life on Earth. Through their research, the team is working towards finding another sun-like star that will lead them to another Earth-like planet. While Earth is the only planet of its kind right now, Dr. Llama is searching for the familiar worlds that have captivated the imaginations of generations.

Dr. Llama's research about the sun relies heavily on gathering radial velocities. The radial velocity of our sun is the speed at which it moves away from the Earth. Astronomers can use this data to find other stars with similar radial velocities which would suggest that those stars are moving in the same way with respect to a nearby planet. Dr. Llama is using the Lowell Observatory Solar Telescope (LOST) along with Yale University's newly developed EXtreme PREcision Spectrograph (EXPRES) to record extremely accurate measurements on the sun.

Led by Dr. Llama, the research team is gathering approximately 40GB of data every day. The LOST, along with the EXPRES, is taking 90-second exposure pictures for 8 hours every day. This data is currently recorded in hard-to-read FITS files that are not easily accessible. This is the problem that Dr. Llama has recruited our team, LOST EXPRES, to solve. We will create a web application for Lowell Observatory that will allow Dr. Llama, his fellow astronomers, and members of the public to view comprehensive and interactive graphs of the data gathered. Our team consists of Brooke Caldwell (Team Lead), Jared Cox (Architect), Ian McIlrath (Release Manager), Olivia Thoney (Customer Contact), and Austin Bacon (Meeting Recorder). Dr. Llama's website will feature role-based permission admission, fast database queries, downloadable files, and dynamic graphs of radial velocity and one dimensional and two-dimensional spectrum. All features will be implemented while maintaining an intuitive user interface that is open to the public.

This document will outline the challenges we expect to face during the span of this project and proposed technological solutions for each challenge. The "Technology Challenges"

section will briefly present an overview of the main requirements of this project and introduce expected challenges in implementing these requirements. The following section, “Technology Analysis,” will go into more detail about each challenge. Several technological solutions will be proposed and analyzed for each challenge. Important pros and cons will be explored for each technology as well as why the technology is relevant to the project. After analyzing each challenge and determining a single solution, the “Technology Integration” section will present how each technology we chose will interact with the others.

2 Technological Challenges

We have identified four main challenges in our application design. First, we need to choose a database management system for interacting with the data from LOST. Once we have a management system, we need some way to communicate with it from our Python backend. This will be the SQL toolkit. Once we can interact with the database from the backend, we need a way to generate the interactive graphs which will be the main content of the web application. We intend to use a plotting library specifically designed for use with Python. The last piece for completing web development will be choosing the front-end framework. The following section will introduce each challenge in more depth.

2.1 Database Management System

For our website, we will need to implement a database management system (DBMS) to store data that is collected from the LOST. This system will need a sorting algorithm to find the correct data, retrieve and store values that are sent via Python/SQLAlchemy, and a way to plot the data onto an appropriate graph type.

The most important task that our database will need to do is have a sorting algorithm to retrieve the data that the astronomers are looking for. The types of values that need to be sorted include the date, radial velocity, and type of star.

Another high priority task that the DBMS needs to accomplish is to take in data from our SQL toolkit and store that data in the database. While this procedure is occurring, we need to

make sure that numeric values are rounded to the correct decimal place so that our graphs more accurately represent their corresponding datasets. We also need to ensure that each piece of data is being stored in the correct spot to prevent data from being lost or misrepresented.

Our DBMS also needs to have a way to plot the data sets onto the appropriate graph type. For this step, the DBMS simply needs to double-check that our values are being stored correctly in the database, and then a specialized algorithm will plot the graph.

The DBMS for our website will need to meet all three criteria above to be considered a viable program. In addition to those criteria, our solution should also aim to optimize usability, run performance, and efficient storage procedures. For our website, usability means that Joe or any of the other astronomers at Lowell Observatory should be able to use our software after one explanation and without difficulty. Website Performance will be compared to similar sites that store and graph solar data. Our DBMS will need to be very storage efficient because the LOST will output about 40 GB of raw data per day. While we are not directly dealing with all of this data, we still need to factor memory into the implementation of the DBMS.

Three types of database management systems were examined to determine which of these three will be the best fit for the website. These are MySQL, Oracle Database, and PostgreSQL.

2.2 SQL Toolkit for Python

One large aspect of mapping out and plotting information is being able to interact with a database using Python. The technological challenge here is finding which database tool kit will perform best for our purposes. This will largely depend on the database management system that we are using. We will be analyzing a few different database-linking tools and comparing how well they integrate with the database of our choosing, if we are using a tool kit that does not work well with our DBMS, there is no point in even considering that tool. The second thing to consider is how user-friendly the tool is. We need a tool that we can easily pick up and learn to make it easier to complete the project and perform maintenance on in the future. Lastly, we will be researching the flexibility of each tool, we want to use whichever tool supports multiple DBMS’s or even other tools, this will be especially helpful in the unfortunate case that we need to change our database management system. To solve this issue we are considering using one of

two different object-relational mappers, Django and SQLAlchemy, as well as just a MySQL connector called MySQL Connector. Object-relational mappers link Python to a SQL database, and even more than that, they allow for directly writing Python code to insert, delete, and edit databases directly, while MySQL Connector will simply allow for Python to communicate with our database.

2.3 Python Plotting Libraries

The main function of the LOST web application is to present raw astronomical data graphically. The application needs to render a High-Resolution Spectroscopy graph for the data gathered on the sun. Not only does the application need to render a single graph, but the graph must be interactive. Examples of interacting with the graph include zooming in/out, seeing details about a data point, and customizing the x and y-axis. As developers, we do not intend to create and implement the logic for creating this interactive graph. Instead, we intend to use a high-level front-end plotting/charting library for Python. We will analyze the features of a handful of different libraries available by comparing the big data capabilities, visual aesthetics, usability for developers, usability for researchers, and interactiveness. Many plotting packages exist for visualizing small data sets, such as the results of a customer survey on a commercial site. However, we need our plotting package to be able to handle plotting hundreds of data points for each graph without compromising speed. Visually we are looking for libraries that allow us to create a detailed enough graph to be useful to researchers, but a “pretty” (for lack of a better term) enough graph to interest the general public. As developers, we do not want to waste time on trying to understand an incredibly complicated plotting library when there are easier-to-use options available. Likewise, the final graph on the web application should be easy to use and interact with as well. Finally, we are aiming for as much interactiveness in this graph as possible. While our initial goals only include a few interactive features, we would like to offer our client the ability to expand upon the application in the future. The plotting packages we are analyzing are Bokeh, Plotly, and HoloViews.

2.4 Front-End Framework

Front-end development frameworks are libraries used to quickly and simply write code that is run on the client-side when visiting a web page. In the past, a website would be made with little styling and few responsive features to speak of, written with few lines of pure code, if any. However, with the expansion of the internet, more and more frameworks have emerged to streamline the website construction process, as well as introducing more interactive elements on even basic websites.

As a Web 2.0 application, the telescope's website needs a level of interactivity that a front end framework can provide. This framework will be the basis of how the web app will display our plots, but there is enough variety in the multitude of options to choose a framework that suits our own needs. The framework will be fetching and handling very large files to process into displayed plots. The inclusion of front end libraries should be limited to a lower number of kilobytes (KB). The amount of data displayed on the plot reaches into the tens of millions of floating-point values when opening up a specific file. While opening this data, it would be beneficial to handle it time efficiently and allow the site to free that memory as soon as it is unused. Sensitive data like passwords will be protected on our site to keep track of researchers and admins, versus public users who do not need to log in. Any front end libraries used should be able to support the use of our plotting libraries needed for the functionality of the web app. On top of this, it must be compatible with the back end, where files are stripped of unnecessary details. Overall, we are looking at four main characteristics when analyzing front-end frameworks: size, scalability, security, and usability.

3 Technology Analysis

After identifying each challenge, our team researched different technologies to find a solution for each challenge. Each analysis chose at least three different technologies that would provide a viable solution to the presented challenge. Then, each solution was researched with special attention to key characteristics that are defined for each challenge. After careful consideration of each technology and a thorough comparison of each option, one solution was

selected as the solution to each challenge. The in-depth analyses of each challenge are presented below.

3.1 Database Management System

3.1.1 MySQL

MySQL is a relational database management system that stores its indexes in a B-tree, which is a self-balancing tree data structure. The structure of a B-tree enables searching, inserting, and deleting operations and prioritizes the run-time of these operations. MySQL is also easy to be customized by the programmer of the database and allows for more ways to display and access the indexes of the database. Since MySQL also happens to be one of the most popular DBMS platforms available, there are more third-party tools available to use when compared to other relational databases.

There are typically two approaches for implementing a user interface for MySQL. The most common approach is to create a graphic user interface using MySQL workbench or some other third party front end software. Another approach is to build a command-line interface using MySQL utilities or a separate command-line tool. A command-line interface allows the user to access and modify the DBMS from their command line. For our purposes, a graphic user interface makes much more sense because this type of interface is more typical for website use and is generally easier to use than a command-line interface.

3.1.2 Oracle

Oracle Database is a multi-model database, which means that it supports many data models such as documenting and graphing for the data. Oracle is generally considered to be a secure database because it requires a username, password, and validation, while MySQL only requires the first two and the hostname. Oracle is also capable of using data partitioning to distribute data across multiple data tables. This is important because partitioning data across multiple tables can increase both query runtime and database manageability.

However, some of Oracle's attributes don't benefit the design of our website. It cannot be customized at all because Oracle Database is closed source software. Customization of the

database is important to us because customization can make it easier to plot our datasets onto graphs. Oracle also is better suited for large scale projects that require 5+ different data tables to use. For comparison, we estimate that we will need 1-2 data tables for our website, so it might be a waste to use software that is better designed for a bigger project.

3.1.3 PostgreSQL

PostgreSQL is an object-relational database, which gives it features such as table inheritance and function overloading. Both of these features give this DBMS similar capabilities to most object-oriented programming languages. This means that our data tables can inherit structural design from each other and overwrite sorting methods in different tables. Postgre is also more optimized for concurrency because it implements Multiversion Concurrency Control (MVCC). MVCC allows our database to create queries in parallel, and create partial indexes (for example, if your DBMS needs a soft delete feature, you can make partial indexes that check for a deletion flag). PostgreSQL forks a new process for every single client connection, and this uses a non-trivial amount of memory (10 MB). This is a big deal because our project is already working with large amounts of data, which means that this DBMS might not be optimized to handle the amount of data that we have.

3.1.4 MongoDB

MongoDB is a schema free relational database that stores its data in files similar to JSON files. The documents that MongoDB stores its data in can each have their own structure, and new fields can be added whenever. This is because the database structure is defined within these documents. These features make MongoDB useful to represent things such as hierarchical relationships and data arrays. Compared to most other relational databases, the queries for MongoDB involve object querying, which makes querying more secure in comparison. These quirks offered by MongoDB make it an excellent choice for unstructured data or data with the possibility to grow quickly.

3.1.5 Chosen DBMS

	Size	**Scalability**	**Security**	**Compatibility**
MySQL	8	8	7	10
Oracle	5	9	9	6
PostgreSQL	6	8	6	7
MongoDB	6	8	10	6

Table 1.1: Analysis of different types of DBMS

The 4 relational databases that we examined all have their ups and downs. Some important things to evaluate when we pick a DBMS are the sorting algorithms available and how to take in data from a SQL toolkit and store that data.

When evaluating the viability of MySQL, it is important to consider that Lowell Observatory is currently using MySQL for their database purposes This may influence our decision so we do not introduce complexity to the system by using different technologies. Our client, Mr. Llama, is also using Python code and SQLAlchemy to access the database, and these technologies also work well with MySQL. Both of these factors give us a big incentive to use MySQL for the DBMS.

MySQL also has some advantages beyond compatibility with our client. MySQL is one of the most popular relational databases and has plenty of third party tools available to use as a result. This can be beneficial if we decide to use additional technologies later on. This DBMS is also easily customizable and has an active open source community that you can go to for help.

Oracle Database appears to be the relational database that is most optimized for very large projects, due to the nature of its multi-model schema and data partitioning. Most of these features are not very beneficial to our website because the complexity of our database is not high enough to abuse these features. However, the biggest problem with Oracle is the lack of flexibility and customization. These features are necessary because we will need these tools to plot solar data.

PostgreSQL has a few features that make it appear desirable when first looked at. Since Postgre is an object-relational database, it can allow for a more complex structural design and overwrite methods from other tables. However, there is one critical flaw with Postgre that makes it completely unviable. This DBMS will fork a different process every time a client makes a connection, which uses about 10 MB every time. This is a large amount of data to use for every connection when you factor in the amount of data we already have to account for.

The aspect of MongoDB that might make it an optimal choice as a database is how well it scales with growing datasets. Since our database will need to handle a large amount of data on a daily basis, the method of storing datasets in JSON files can be more efficient with query speeds. One major downside of this design quirk is that it works much better for an unstructured database. Since our database will need to graph over 80 subsets of data, it might not be wise to use a relational database that is intended for datasets with much less structured data.

After looking at 3 different relational databases, we have decided to go with MySQL. MySQL makes the most sense to use for several reasons: It does not conflict with any of our other technologies, It is highly customizable, Lowell Observatory is using it on their backend for the telescope, and it has a multitude of different third-party tools available to use. MySQL is compatible with Python, SQLAlchemy, Javascript, and Plotly, which cannot be said about closed source relational databases like Oracle DB. MySQL is easy to customize and has many different attributes that can be customized, which can be rather useful for how we want to represent the solar data. By comparison, both Oracle and PostgreSQL are much less desirable in terms of customization and follow a more rigid formula when it comes to data representation. Since Lowell is already using MySQL, Python, and SQLAlchemy, using the same technologies will reduce redundancy and complexity. Finally, MySQL has the highest amount of third-party tools available to use out of all relational databases. This can be very useful if we decide to use additional technologies because of the flexibility provided by these tools. Overall, MySQL is the optimal relational database to use for our database management system because of software compatibility and technical prowess.

3.2 SQL Toolkit for Python

3.2.1 MySQL Connector

MySQL Connector is a simple Python-database connector that will allow for Python to communicate with the database. The MySQL website explains the connector as " a self-contained Python driver for communicating with MySQL servers." The connector was written by and is maintained by MySQL's parent organization, Oracle. MySQL Connector does not seem to have any kind of website dedicated to it, but there is plenty of information on it elsewhere. MySQL Connector does not have an implementation on any other relational database besides MySQL but can be used within different languages. MySQL Connector seems easy enough to use and there are even tutorials online to help get started.

3.2.2 SQLAlchemy

SQLAlchemy lists key features of their product on their website, stating that you do not need to download any other object-relational mappers (ORMs) to use theirs and that SQLAlchemy gives you the ability to write SQL code directly within your Python scripts. Interestingly, SQLAlchemy supports a large range of database management systems, these include but are not limited to SQLite, PostgreSQL, MySQL, Oracle, MS-SQL, Firebird, Sybase, and others. As I was looking further into the flexibility of SQLAlchemy, I noticed that it is available on Python versions 2.5 through current versions, so it is nice to see that it continues to be refined and maintained and that it is flexible in terms of Python versions as well as its support for multiple DBMS's. While looking into SQLAlchemy's ease of use I noticed that there were tutorials just about every place that I looked, so I do not think that it will be hard to learn or get comfortable with.

3.2.3 Django

Django states that they include a default ORM within their framework, just like SQLAlchemy, and similarly, there is no need to code anything in the databases themselves because Django allows you to do that within Python itself. Just as SQLAlchemy listed, Django supports several database management systems such as SQLite, PostgreSQL, and MySQL.

Django features tutorials on their website, https://www.djangoproject.com, and seems like it is pretty simple and straightforward to use. Along with the website tutorial, there are numerous tutorials online elsewhere.

3.2.4 Chosen SQL Toolkit

All three alternatives have their pros and cons, while SQLAlchemy is the oldest of the three, it has continuously been refined and maintained to make the current version the best of its kind. Django allows for more than one relational database but does not seem to have the community or development that is shown in SQLAlchemy. Finally, MySQL Connector lacks the support and implementation of different relational databases, and only accepts MySQL as its database. Below is a table summarizing the analysis of these three alternatives:

	Usability	**Flexibility**
MySQL Connector	6	2
SQLAlchemy	10	10
Django	8	8

Table 1.2: Analysis of SQL Toolkits

As seen in the table above, I gave a perfect score to SQLAlchemy, this is because it has a large community behind it and tons of tutorials all over. It beat Django in ease of use firstly because our client is already familiar with SQLAlchemy, but also because of the vast amount of help and information about SQLAlchemy. It beat Django in terms of flexibility simply because it offered more support for different database management systems. MySQL Connector didn't stand a chance, simply because ORM's are in a different field completely. ORM's have more uses and can simply do more in general.

To prove the feasibility of SQLAlchemist we plan to use sample data to make a simulation of the project, then test to make sure that SQLAlchemist can handle the data we are giving it with no issues. We are already aware of the fact that our client has been using this ORM and are hopeful and confident that everything will run smoothly.

3.3 Python Plotting Libraries

3.3.1 Bokeh

Bokeh is a Python library for interactive visualizations specifically for the browser. Bokeh has multiple language bindings, including Python and R, which produce a JSON file. This JSON file then works as an input for BokehJS, a Javascript library. This is what presents the data to web browsers.

As a data-science tool, the library is developed for processing and plotting several hundred data points. The Bokeh User Guide does divulge some issues when working with extremely large datasets. Because Bokeh copies data from Python directly into the browser, plotting millions or billions of points becomes difficult as web browsers are restricted in how much data they can work with. Also, the plotting of such a large number of data points is "often misleading due to overplotting." developers for Bokeh overcame this problem by using Datashader. This is another Python library that pre-renders large datasets into a fixed-size raster image. Bokeh can then make plots that re-render these images when zooming in, making Bokeh's interactive capabilities work with any size of data.

After viewing a couple of examples of Bokeh in action from the gallery, applications created with Bokeh are user friendly and professional looking. Bokeh offers a variety of default themes and visualizations while still allowing developers to customize their lots through JavaScript. Overall, the aesthetics of Bokeh meet or exceed the requirements of the LOST web application.

Bokeh is a very complete library with detailed documentation and a very organized User Guide. While it is relatively new and some features are still under development and need improvement, the basic Bokeh commands are fairly straightforward and simple to understand. Bokeh is created for data scientists who want to create beautiful visualizations. Because of the target audience, Bokeh is not a code-heavy library. That being said, for users who have more of a computer science background, Bokeh can be extended with BokehJS. BokehJS allows the developer to create custom models not included in the standard library. The BokehJS API is still experimental and future changes in upcoming releases may cause issues in development.

The user interface of the final plot is largely up to the developer. However, Bokeh provides really useful tools for zooming and panning, cropping data, changing the x and y-axis, and more. These interactive features can be presented to the user in various ways including drop-down menus, slider options, and simple toggle buttons that create a clean and intuitive user interface. Overall, Bokeh visualizations are very easy to use from a user perspective.

The interactive features of Bokeh visualizations include everything that has been requested by our client, Mr. Llama. The library is specifically designed and marketed as an interactive data visualization tool, not just a plotting package. The default Bokeh plot includes a toolbar that includes interactive features for panning, box zoom, wheel zoom, save and reset. Features that allow the user to select a group of points, hover over a point to see more information, or link sections across different plots are all easy to implement as well.

3.3.2 Plotly.py

Plotly is perhaps the most well known and widely used data visualization package for making plots in Python. The library allows developers to create plots in Python, R, and JavaScript. Plotly has three different Python APIs: an object-oriented API, a data-driven API that constructs JSON-like data, and a Plotly Express API.

Similarly to Bokeh, Plotly works well with the majority of datasets and will work fine with the size of data being used in the LOST web application. However, the Plotly documentation does provide some direction when working with millions of rows of data. The example solution used pandas, IPython notebook, SQLite and Plotly together to perform out-of-memory aggregations that get directly loaded into a data frame and can then be rendered with Plotly. Much like Bokeh, extending Plotly's big data capabilities is some extra work, but doable.

The visual aspects of Plotly are professional and complete looking. While appearance is objective, Plotly provides clean visualizations and a nice variety of themes.

The Plotly library's size is somewhat daunting and complicated; however, it offers more uses and features that make the added complexity worth it. The library supports over 40 unique chart types that cover a "wide range of statistical, financial, geographic, scientific and

3-dimensional use-cases," as stated on Plotly's site. The Plotly library is well developed with extensive documentation and community forums. Also, Plotly provides templates for easy formatting, as well as Plotly Express, the high-level API that includes fewer customization options but does contain functions that can create entire figures at once. It is considered a common starting point for creating visualizations and would be a good option for using in initial prototypes.

The usability is almost identical to that of Bokeh. Plotly ranks above Bokeh in usability due to its cleaner aesthetics and dashboard themes, as discussed in previous sections. In addition, Plotly enables Python users to create not only web-based visualizations that can be displayed on the browser, but visualizations that can also be displayed in Jupyter notebooks, saved to standalone HTML files, or served as part of pure Python-built web applications using Dash. This is a beneficial feature for researchers as one of the goals of the web application is to export data. The scientists using the application are familiar with Python and Jupyter notebooks so this is a great feature to have. In addition, the flex goals of the project include some type of personal notebook for researchers, which could take advantage of Plotly's rendering abilities in Jupyter notebooks.

Again, Plotly's interactive features are comparable to those of Bokeh, but slightly more reformed and cleaner. Plotly also has better dashboard support for showing several graphs at once. Users can then select certain settings that apply to all the graphs in the dashboard. While this type of customization could probably be achieved in Bokeh, it appears easier to implement with Plotly since it is already included in the package, instead of needing to be implemented through custom JS.

3.3.3 HoloViews

HoloViews is another Python library for data analysis and visualization. However, the company claims to differ from other plotting packages by helping the developer and user better understand the data, and work seamlessly with both the data and the graphical representation. Instead of building a plot using calls to a plotting library, like Plotly and Bokeh, HoloViews requires the developer to "first describe [the] data with a small amount of crucial semantic information required to make it visualizable, then...specify additional metadata as needed to

determine more detailed aspects of [the] visualization." Through this approach, visualization can be requested at any time while the data evolve and grow, and can be rendered automatically by a supporting plotting library (such as Bokeh). Ultimately, to take full advantage of HoloViews and create a fully interactive visualization, another Python plotting library would need to be used.

HoloViews elements are less like data visualizations and more like data containers. Therefore, elements can be declared quickly that contain datasets as large as the memory of the machine. However, once the plotting package goes to render to the visualization of the data, the same problems as discussed with Bokeh and Plotly will arise. The HoloViews documentation advises using the same solution as Bokeh's documentation: create a dynamic Datashader-based Bokeh plot. So, while HoloViews itself is capable of storing extremely large data, it does not do a better job than Bokeh or Plotly at actually rendering visualization of the data, which is the focus of the project. It arguably does a worse job since it renders the final visualization with HoloViews, Bokeh, and Datashader when in reality, the web application only needs Bokeh and Datashader for interactive visualizations.

Overall HoloViews graphs do not appear any better than Bokeh or Plotly. The visualizations appear to be more directed at researchers who are proficient in the field and appear less interested and more overwhelming to the general public. The LOST web application needs to be built for both researchers and curious members of the public. It should not intimidate or turn people off of the data, which is what HoloViews gallery examples do.

Before more in-depth research, the team did not realize that HoloViews on its own was not really a data visualization library. While it does have useful features for data analytics, this adds a layer of complexity to the framework since it would still require a plotting package. From a developer perspective, this seems like added work that is ultimately unnecessary since the web application will be solely used for visualizing and interacting with data, not analyzing it.

3.3.4 Chosen Python Plotting Library

The analysis of all Python plotting libraries is summarized in the table below:

	Big Data Capabilities	**Visual Aesthetics**	**Developer Usability**	**Researcher Usability**	**Interactiveness**
Bokeh	8	9	8	9	8
Plotly	8	10	10	10	10
HoloViews	7	8	5	8	6

Table 1.3: Analysis of plotting libraries

After researching and analyzing the three plotting packages Bokeh, Plotly, and HoloViews, there appeared a clear "winner" in the correct package to use for this project. Immediately, HoloViews was ruled out because it is more of a data analysis package that can render visualizations with the help of an actual Python plotting package. Since the web application is designed for viewing data and exporting data, which is then processed by astronomers using their tools, there is no use or need for the data processing capabilities that make HoloViews unique.

Bokeh and Plotly are fairly similar in terms of fundamental features and functionality. They both create clean data visualizations from built-in functions and allow for additional customization through JavaScript programming. A couple of small differences between the two, however, makes Plotly the best choice for this project. First, Plotly has been around longer and has a larger community behind it. While this does not always speak to the quality of a product, the developmental stage of BokehJS turned our team off of Bokeh. In addition, the wide-ranging support for Plotly means our client and other researchers who use our site are already familiar with it. Essentially, Plotly appears to be the "industry standard," and we want to stick with it. Along with familiarity, Plotly's Express API is a simple and fast solution for getting familiar with the package. While we minted to utilize more features and customize the experience on our application, Plotly Express provides a nice solution for creating an initial prototype and exploring data visualization. Plotly's integration with Jupyter notebooks also plays a large role in

the decision to proceed with Plotly. While we do not anticipate any early integration with Jupyter notebooks, it is a future goal that we took into consideration. Finally, as a team, we ultimately like the aesthetics of Plotly more than Bokeh.

3.4 Front-End Framework

3.4.1 Raw JavaScript

Other front-end frameworks perform by running a set of defined Javascript functions that extend and abstract Javascript's functionality. While front-end frameworks are useful for repetitive and responsive websites, it is not uncommon to build websites without using Javascript libraries. Javascript is perfectly usable to implement simple to complex websites and functionalities, as it is a complete programming language. Theoretically, the most compact code would probably be made with Javascript and a program to condense to file size, as you do not have to import frameworks that could be hundreds of kilobytes in size. The main flaw that Javascript has, however, is that it is a terribly implemented, and often contradictory programming language that we cannot change because of how popular it is. Its initial development was rushed and incomplete, and bugs plagued type conversions, equalities, arrays, strings, and is generally infamous for its inconsistencies. In a required act of backward compatibility, Javascript still has these bugs today as when it became popular, any change to its previous functionality will inevitably break some websites' functionality.

More concisely, JavaScript has several benefits including that it is built into all browsers, and therefore has no extra overhead, is a client-side language, and runs on the client browser instead of waiting for the server, and can work very well with other programming languages, giving it cross-functionality. However, JavaScript’s downfalls are that it contains many bugs and non-standard behaviors, and security can be compromised since source code is run on the client-side.

3.4.2 jQuery

jQuery is an industry standard when it comes to extending Javascript's functionality. For example, an integral function of Javascript is to access an element from the DOM, which is how

HTML organizes a webpage, is highly simplified in jQuery compared to its Javascript counterpart. Apart from selection, jQuery extends many functionalities from Javascript, such as fading, hiding, and sliding DOM elements, as well as delays, toggles, and animations. Further extension can be achieved through external plugins and libraries.

jQuery is simple and easy to learn. This simplistic approach leads to faster solutions and saved time in development. However, while it is easy to learn, jQuery is slower than CSS and can be severely slow if not used correctly.

3.4.3 React

Developed and maintained by Facebook, React is an open-source framework implementing reactive Javascript and HTML. In the past, changing text and elements in the DOM required the DOM to keep track of every change that had been made up until that point. React circumvents this by creating a "Virtual DOM" which acts as a middle man to update only what needs to be updated instead of re-rendering the whole DOM. Since React is only concerned with the DOM, a router like Redux is needed, expanding total overhead.

React’s virtual DOM improves user experience while its extensive documentation improves the developer’s experience. In addition, components of React can be reused on the web page which is both convenient and saves time. On the other hand, React has constant updates, which make the framework somewhat unstable and not resistant to time. Not to mention that React cannot work on its own and needs to be supplemented with other technologies to complete web development.

3.4.4 Vue

Vue is a progressive framework that adopts React's Virtual DOM architecture and JSX markup but builds on top of that by making the library more lightweight, and integrating concepts from Angular and other frameworks. For example, Vue implements two-way data binding inherited from Angular allowing bound data to be updated reactively. Vue also has a set of tooling including Babel and TypeScript support, unit testing, end-to-end testing, and a plugin installation system.

Vue could be beneficial to this project because, like React, it uses a virtual DOM which improves user experience. It also utilizes a two-way binding implemented from Angular. Since Vue relies on JavaScript, it is easy to integrate with other technologies, such as the technologies previously analyzed in this document. Vue's extensive documentation makes it easy to learn and look up any issues that could be encountered during development. While Vue has many pros, it can be relatively complex, such as the two-way data binding discussed before. In addition, unlike jQuery and raw JavaScript, Vue is relatively new which means it lacks large-scale support. Finally, Vue's plugins are much less numerous than plugins for a framework such as React or jQuery.

3.4.5 Chosen Front-End Framework

To re-address each of the frameworks' advantages, raw Javascript is useful for making granular webpages and does not have any added overhead. jQuery extends and simplifies Javascript functionality and includes simplistic functions for transitions and animations. React introduced the Virtual DOM for more stable development of the user experience. Vue includes implementation of React's Virtual DOM in a lightweight package half the size of jQuery. In matters of size, Vue came out on top outside of raw JS, like jQuery, and React required over 100 KBs of data. When addressing scalability, each library has ways of handling large data sets, but jQuery has been reported to not scale well with certain functions and should be performance tested before deployment. Vue apparently can slow down with large data sets but will speed up when the object is set to read-only. In relation to security, Javascript had none, both jQuery and React have known vulnerabilities concerning sanitization, and Vue's vulnerabilities were patched several versions ago. Finally, compatibility is not a problem in this set of circumstances, as all of the libraries are compatible with default Javascript, and each of the frameworks is compatible with our backend as well.

	Size	**Scalability**	**Security**	**Compatibility**
JavaScript	10	10	0	10
jQuery	3	8	8	10
React	5	8	8	10
Vue	8	9	10	10

Table 1.4: Analysis of front-end frameworks

We have chosen Vue as our front end framework for this project. In the decision matrix above, it had received the highest score when compared to other frameworks by our needed metrics. Vue had higher scores in size because of the low amount of script to load compared to React and jQuery. Vue also had a higher score in scalability because of several exponential complexity issues with jQuery, and React's Virtual DOM can lead to slower rendering with large sets; these problems are seemingly absent in Vue. Another mark against jQuery and React is their potential vulnerabilities that have not been fixed in newer versions and must be patched yourself. Vue is a convenient framework that fits better with our criteria than better supported, but larger frameworks.

We plan to use Vue to quickly prototype the web application that will resemble the interface in our final product. While we plan to use Vue, functionality, and planning may change; if we are forced to use a different front end, it would not be much of an issue to convert only a few pages into the new implementation. We will prove feasibility by loading, passing, and displaying the large amount of data compiled by the telescope.

4 Technology Integration

The main consideration in our technology integration is the interaction between the client and the database. The user should be able to view data directly from the telescope in a visual representation and customize and change this data. For example, a user who wants to view the high-resolution spectroscopy of the sun, which queries the database for values in a default range

of dates. The user then zooms in on the graph to get a more detailed view of a specific day, which results in a new visualization being rendered. Or, the user switches the y-axis from displaying radial velocities, to displaying the S index. Again, a new view is rendered. Of course, the largest obstacle in building this application is deciding how to host, interact with, and display the data in an interactive manner that is quick and user friendly. We did not want to re-render the entire page with a new graph created from a new database query every time the user interacts with the graph. Therefore, through our analysis, we have chosen a database management system to contain the data, a Python tool kit for communicating between the Python backend of the application and the database, a Python plotting package for creating data visualizations, and a front-end framework for ensuring an intuitive user experience.

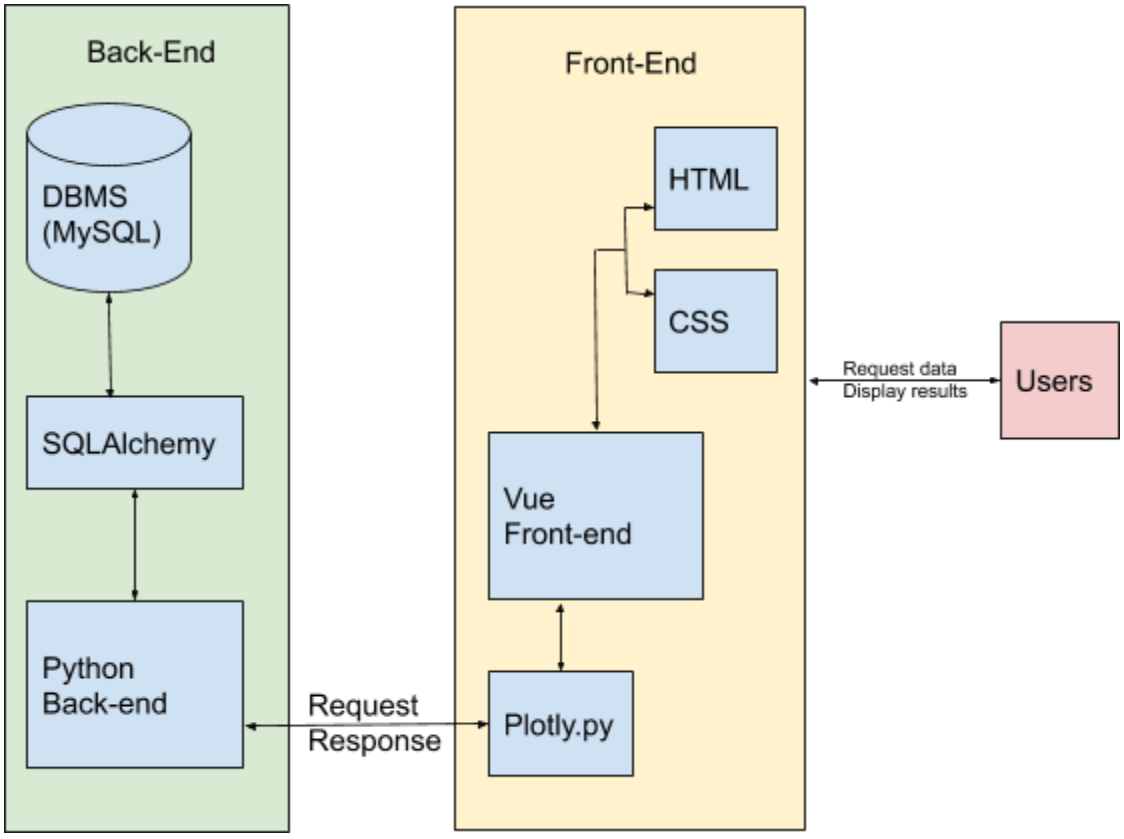

Figure 1.1: Diagram shows how users access data with the front-end framework and describes how the front-end and back-end of the website interact with each other.

The final architecture will use the MySQL database management system. This will allow us to interact with the data which is sent from the LOST then stripped and entered into the

database. Since we are using a Python backend, we will use the library SQLAlchemy to interact with the database. This library is a tool that translates Python classes to tables on relational databases and automatically converts function calls to SQL statements. By querying our database directly using Python, we can then build data visualizations with the Plotly plotting package for Python. Graphs created with Plotly can then be embedded into HTML pages with the simple plotly.tools.get_embed function. These HTML pages along with some nice CSS and a Vue.js front-end framework will create a professional user interface for viewing the LOST data. For these specific technologies to work together, we will use Vue.potly, a thin Vue wrapper for Plotly.js.

5 Conclusion

This web application needs to plot large amounts of data while maintaining an easy-to-use interface and maintaining the responsiveness of the site. By accomplishing this our team will be able to give Mr. Llama and his team a way to fully analyze the massive amounts of data they receive each day from LOST. Also, this allows the public and other astronomers to learn more about the sun and ultimately lead to the discovery of an Earth-like planet surrounding a sun-like star.

Challenge	**Confidence Level**	**Chosen Technology**
Database Management System	8	MySQL
SQL Toolkit	5	SQLAlchemy
Plotting Package	8	Plotly.py
Frontend Framework	7	Vue

Table 1.5: confidence level of each separate technology

We have decided to use MySQL as the database management system to hold all the radial velocity and spectrum data for plotting the graphs that our client needs. Its ability to integrate with the Python code already used, along with our chosen SQL toolkit, and the fast searching algorithms which can be implemented, all led to our selection of MySQL. Our team is very confident that MySQL will be the best database management system for the project.

For our SQL Toolkit, the team has chosen to use SQLAlchemy. This toolkit is flexible, as it can interact with many database management systems and Python. Also, this toolkit seems fairly easy to use and has a large community in case problems arise. Our team is confident that this SQLAlchemy will work on the project.

The plotting package our team will be implementing is Plotly.py. The graphs that can be produced with this plotting package are professional and easy to use. They also have resizing and zooming capabilities as Mr. Llama requested. With this in mind, our team is very confident in Plotly.py to meet all of our graphical requirements.

Lastly, our team's chosen frontend framework will be done using Vue. Vue will have virtually no issues with rendering large amounts of data, and it is compatible with the chosen technologies above. Our team is fairly confident that even though this is a new technology it will be a great addition to our web application.

Overall, our team believes that with the proposed technologies we will be able to create our tech demo by the end of the semester, leading to an excellent web application for our client so that he may continue his research.