Skyward

Technological Feasibility Report

10/26/2016

Team Name:

Skyward

Team Members:

Gage Cottrell
Kaina Crow
Justin Kincaid
Chris French
Alexander Sears

Sponsors:

Dr. Michael Mommert and Dr. David Trilling

Mentor:

Dr. Otte

Table of contents

Introduction

Team

We are team Skyward. Our team members are Justin Kincaid, Gage Cottrell, Chris French, Kaina Crow, and Alexander Sears.

Dr. Michael Mommert and Dr. David Trilling are our project sponsors and client, and are heading the FRoST monitor project.

Project Description

The FRoST monitor project uses the Flagstaff Robotic Survey Telescope (FRoST) to track Near Earth Objects (NEO's) with the interest of identifying and collecting data on objects that could potentially impact the Earth.

Most modern observatories are outfitted with an online status page that showcases important observatory information such as an all-sky view image, weather conditions, current telescope position, a list of target objects, and other information. The primary objective of our project is to provide our client with a dynamic web application that is able to display observational and telescope data - that is deemed important by our client - to anyone interested.

In addition to displaying the telescope and observatory data, a secondary portion of our project will be aimed at adding additional features such as the ability to change the telescopes target list, as well as the ability to manually shut down the telescope all together.

We will be working very closely with our sponsors to ensure that we are successful in providing the basic requirements as well as deliver on all of the requested above and beyond features.

Sponsors/Project Problem

Networking

Our client will be providing our team with a load out of information (???) gathered from the observatory. We need to find a way to network into their database to gather and parse all of the information for us to use.

As it stands right now, the telescope and weather data will be fed and formatted through a "crunch machine" (a local machine on the NAU network that is maintained by our client) that is still in development by our clients. Once the crunch machine is completed, we will need to be able to connect our web application to the data that is outputted by the "crunch machine", so that we can display it within our web application.

If our clients fail to complete the crunch machine before the designated completion date of this course, dummy data will be provided from our client that adheres to the planned format of the output from the crunch machine. Using this dummy data we will still be able to create a dynamic web application that is able to display the necessary data in which the completed crunch machine will be able to connect.

Dynamic Web Application

A dynamic web application must be designed and implemented to display all of the FRoST telescope and observatory information in a clear and meaningful way.

As mentioned above, most observatories are outfitted with a web page that displays various diagnostic data such as all-sky view images, current weather conditions, current telescope position, what object is currently being observed, ~~and who is observing.~~ (Note: telescope is robotic)

To consider our web application a success, we must display all of this data in a clear meaningful way to end users. For example, rather than display an unformatted list of NEOs in the vicinity of the FRoST telescope, the data may be converted into a graph that allows the user to easily understand the data.

Additional pieces of data to display within the web application may include things like a queue of what the telescope will look at next, current weather information, an all-sky view overlay that shows the telescopes path throughout the evening, and the last known command given to the telescope (in case of network failure).

Database

A database must be made to store all of the gathered information from the telescope so that it can be displayed within our dynamic web application.

The information gathered from our client's crunch machine will contain certain things like raw images and image metadata (such as timestamps), and certain other text and numerical details about observed NEOs. It is not required that this information be queryable, but this feature would be good to have.

We will be creating a database to link pieces of the data gathered in order to properly display it ~~in~~ on the website in a clear and understandable way.

FRoST Monitor Controls

Additional features include altering the queue manager, which tells the telescope what to look at next, and an emergency shutdown button in the event of extreme weather. We need to find a way to network with the FRoST observatory to introduce these commands.

How we will issue queue and emergency shutdown commands to the observatory is unclear at the moment (Note: just mention that you will send a signal of some kind, e.g., on a socket; we will take care of the rest). The data flow at the moment seems to be traveling in one direction,

which is our sponsor's crunch machine pulling data from the observatory, and our website pulling data from the crunch machine.

As we get further into development, we will better understand how to approach sending actual commands back to the observatory.

Login and Security

Certain features of the website should not be useable by the general public, which means we must provide a layer of security to prevent unauthorized access. Implementing user accounts will allow us to keep exclusive administrator actions hidden from end users of the web application.

Document Description

What is This Document

This technology feasibility document, briefly introduces each of the problems that our client would like to have solved. Each problem is then organized into a structure that would more easily showcase each type of solution. Each solution is then explained in greater detail, to get a better idea of how the solution will solve the sponsor's specific problem, and how the solution will tie into the entire system as a whole.

Document Goals

The goal of this document is to set up a rough guide for the direction we would like to take during the implementation phase of this project. On the surface, we do not yet have a tight grasp of all of the smaller details and requirements of the project. We are working from the top down to understand all of the details, and this document will give us a nice solid starting point to do so.

If nothing else, this document should enable us to ask all of the right questions, and look in all of the right places to further understand the technology that would be required to get our project started in the right direction.

Document Organization

In the Technological Challenges section, we begin by taking a look at the specific challenges we expect to be faced with. Each of the problems will be briefly expanded upon here in a "high-level" requirement fashion, then be broken down much more thoroughly in the technology analysis section.

The Technology Analysis section of the document will introduce each major problem we have identified and expand on the problem in greater detail. First a problem will be introduced, followed by any and all alternative solutions we have researched, the chosen approach for the problem, and finally, a feasibility section. Each alternative will be accompanied by a description as well as any pros or cons if applicable. The chosen approach will also be described and followed by pros and cons, as well as a description as to why the approach was chosen.

In the integration section, details will be given as to how each of the chosen solutions will integrate with each other, and how they will will contribute and fit into the final system design.

We will also mention any possible problems or concerns we may face with each of these solutions.

Technological Challenges

Currently, our main concern is setting up the server environment that this website will be hosted on. We don't know the VPN settings that will be required. The information is currently available through Lowell's VPN, but the website will be hosted on an NAU machine, so it will be a challenge to make sure that our website can access both Lowell's VPN and NAU's VPN at the same time.

Aside from networking solutions, most of the other major problems revolve around creating a website that converting and displaying our sponsors crunch machine output as graphical data, with security and login access.

Challenge 1: Network Accessibility

- There are multiple networks that need to be accessed in order to gain the required data to display within our web application. We need to come up with a way to access these different networks.

Challenge 2: Data Accessibility

- We need a way to send and receive data to and from a client server system.

Challenge 3: Database and Web Application Back-End

- We need a way to have a secure login on our page that will also be mostly hidden from view, since the general public will not be allowed to register or login.
- The telescope data will be pulled from the crunch machine's database, which currently does not exist.
- The weather data will be grabbed through a websocket and be used to populate the correct section on the website.

Challenge 4: Web Application Front-End

- We need a responsive and dynamic web application to display data within. The web application must have a well designed user interface regardless of the screen size of the device accessing the web application.

Technology analysis

The data that we will be using to display within the dashboard is coming from a local machine and database that is currently being built by our client ('crunch machine'). The database will contain all of the data gathered by the FRoST telescope, and we will then access this data and store it in our database. The crunch machine is not yet operational, and the details on how we are supposed to actually network to the crunch machine are not very clear at the moment.

There are several ideas amongst the group on how we would like to proceed with creating the frontend of the website, or how to tackle a backend database for all of the gathered crunch machine data, but several major problems still seem unclear and will require more meetings with our sponsors and more time, in order to get a firm heading on a proper solution.

For instance, the current approach for gathering the required data seems to be networking through a VPN, which is highly ill advised according to the research that has been done thus far.

Because of the ambiguity of some of the requirements at the time, the solutions provided below for each problem are the best guesses we can achieve at this current time.

Network Accessibility

One technological issue we are facing is getting access to the NAU and Lowell servers simultaneously. We need to figure out a way to get access to both of these at the same time from the same computer in order to create the website that is pulling information from both servers. In order to do this, we will need to talk to the IT manager in the astronomy and physics department about getting access to this.

Alternatives

Multiple Machines:
An alternative to not being able to get access to both of the servers on one machine is using two machines. One that has access to the NAU server and one that has access to the Lowell server. Although, the downside of using two different machines is that is doubles the chances of error. If one server fails and the other is still working, the project and connection is still going to fail.

Chosen Approach

Request Assistance from Network Admin:
Our chosen approach to this task is getting in touch with both our Sponsor and the person in charge of server access in the astronomy and physics department to see what can be done.

Proving Feasibility

Proving feasibility for this section will be a bit of a challenge. We have to make sure that we are connected to both networks, either through the same computer or with both. We will need to set up test trials to ensure we are pulling the necessary information from one network to display on the web application that will be hosted on the other network. We can do this by attempting to pull data that is located at on one network to the other network, until we are successful in dual server connectivity.

Data Accessibility

There are two secure networks (VPN) that will need to communicate to provide observational data to the public database and then onto web interface. The networks are at two different locations, Lowell Observatory, ~~FRoST Observatory~~, and Northern Arizona University. Each of these locations has their own Virtual Private Network or will have their own in the future. There is a central data gathering and analysis system(Crunch Machine) that will securely connect to Lowell Observatory and the FRoST observatory and process observational data gathered from these systems, this system is maintained by the FRoST research team.

Once the Crunch Machine is done processing the data it will then send it to the NAU physical science server where we need to provide a centralized hub to coalesce the data from ~~Lowell Observatory and the~~ FRoST Observatory . Our hub (server) will have to provide the framework for data processing and data access to a centralized database . This is crucial to the functionality of the website and its ability to display the desired data sets and to give public access to the database.

We will be serving the data gathered to the public, this naturally leads to a client server model. Our data will be centralized allowing us to implement our server hub application with a RESTful API also known as a representation state transfer(REST) application program interface. This will allow for us to easily receive and send data between all clients and our server. Using a RESTful architecture allows for flexibility, modularity, and scalability and if implemented correctly is easily maintainable. There are many ways to implement this type of architecture the following are some viable options.

Alternatives

Django:
Django REST framework is a python language toolkit for building Web API's. The code base is actively maintained and open source. The documentation is very thorough and complete. JSON support is built into the api. The Django framework is used by Mozilla and Red Hat.

Jersey:
Jersey is a java language toolkit for JAX-RS. JAX-RS is a java api for developing RESTful web services. Java EE web applications are everywhere on the web. Java documentation is thorough and complete. Jersey is a high level reference for JAX-RS so there is room for over complication in the implementation. Sometimes overly abstracting code from its original intention makes for unnecessary run around to get work done. Jersey must be run on a jvm.

.Net Framework:
.Net framework is a C# framework. It is in current development at Microsoft. Microsoft's documentation is poorly maintained and incomplete. The best documentation usually comes from their tutorials which are 'hello worlds' of very complex problems. Can only run on windows environments or windows virtual machines.

Zend Framework:
Zend Framework is a PHP 5 framework. PHP is a dead technology and shouldn't be considered for anything other than teaching students the overall concept of server side scripting.

Chosen Approach

Django/Python:
The maintainer of the code will be an in house researcher in the Physical Science Department at NAU. Their preference was stated as 'python' because of their future involvement. Django REST framework is the obvious choice if maintainer preference was the only requirement. It has other benefits as well. It is high level implementation of a RESTful api but is not too high level as to not leave room for customization. It can be easily tailored to our simple need of moving data from one server to the other and presenting that data to our web application for public access. It is written in Python, and though it is not the fastest language out there, it is one of the fastest to develop prototypes on and full on finished implementations. This speed allows for features to be added and removed quickly to suit everyone's needs.

Proving Feasibility

To show that we can move and publish data with our Django REST API we will develop simple application modules to test each HTTP web request, GET, POST, PUT, DELETE. We will deploy these applications on API endpoint urls on localhost for easy debugging and tweaking. Simple requests can be entered through a browser where the expected JSON data will be displayed in text format. We will develop JSON object standards for the data that will be published from the Crunch Machine to the local database. We will write scripts to show that JSON data can be parsed and inserted into the database correctly. Once we have input and output database information correctly we will display pictures and data pulled from the database on simple website proving the feasibility of a web API implementation to provide correct data to a web user interface to provide public access to the database.

Database and Web Application Back-End

We need to get access to the machine where the website will be hosted and get access the the crunch machine that is located on the NAU network while pulling data from the Lowell network. An issue that we are facing now is that the crunch machine is not actually running, therefore it is not pulling any data from the crunch machine. We also need to figure out a way to create user accounts. We are thinking we will make one admin account that has access to the database to create other accounts, but we will not be creating a functionality for users to create accounts on their own volition on the page.

Alternatives

Create Dummy Data:
An alternative to not getting the necessary information needed to create the database that the website will pull information from is creating dummy data to mimic the data that will be pulled from the Lowell database. We have been informed by our sponsor that they will provide this if needed.

A con to not being able to get access to the database/not getting the data we need is that we would need to create a database that acts like the one we would be pulling information from. Instead of just being able to access the data and pulling it into our database, either us or our sponsor would have to create a fake database and add data to it. This makes it more work for us and does not guarantee that the database will be exactly the same, potentially causing merging issues.

There really are no pros of creating a dummy database, but in this situation, it would make it possible to finish the project and the database would be pulling information. In this case, the database we created would just need to be connected to the database it should be pulling information from.

Chosen Approach

Custom Database:
We are going to assume for now, that the crunch machine will be pulling data from all the correct locations when the time comes to start creating the web application. We will also be able to create a database on our own regarding the administrative user accounts so that the login functionality will be working. These administrative accounts will need to be secure because they will have functionalities that regular users should not have.

Proving Feasibility

We are going to talk to our sponsor about this issue and make sure that they will get the data on their crunch machine so that we can pull the data from it, letting us test the site as effectively as possible.

Web Application Front-End

Because the main purpose of this project is to create a dynamic dashboard that contains and displays various data to users, the front-end design of our web application is extremely important. Our web application must be able to display data to our users in a clear and meaningful way while remaining responsive and accessible to all devices and users.

Creating a responsive web application is crucial. Because many end users have different devices they may be using to access our web application, our application must be able to dynamically display its content in a clear way regardless of the device being used. Creating a dynamic web application can be done by creating responsive code by hand, or by using popular existing frameworks such as Bootstrap, or Foundation.

Alternatives

Custom Development:
Creating custom code to get a web page to behave dynamically is useful because when creating the code by hand, the creators have control over every aspect of what should be included and discarded. Having custom code developed for a project also gives the benefit that the creators of the code will know how to easily make changes or edits at any point in development. An issue with creating all custom responsive code for a web page is that it may take more time than just using an existing responsive web framework. Another issue with developing custom code is that the quality of the code depends greatly on the development experience of the team developing said code.

Bootstrap Framework:
The Bootstrap Framework is one of the most popular responsive/dynamic web development frameworks available. Bootstrap allows a user to quickly set up a web page that is fully responsive right out of the box, and thus is able to save precious development time. Because Bootstrap is so popular and has been used for many years, there is the benefit of documentation and code reliability. An issue with using the Bootstrap Framework is that Bootstrap by default includes many things that not all users would use within a specific project, thus making the project files unnecessarily large and numerous. Another issue regarding the use of Bootstrap is that some aspects of Bootstrap are not very easy for users to customize, which can lead to frustration when wanting to develop custom features.

Foundation Framework:
The Foundation Framework is similar to Bootstrap in that it allows users to quickly create a web page that is dynamic and responsive on mobile and desktop devices. Foundation differs with Bootstrap slightly in that it has slightly easier customization, which allows for smoother custom development. An issue with Foundation is that it does not have the online popularity of Bootstrap, and so the documentation and examples are more limited.

Chosen Approach

Bootstrap Framework:
Having limited time and resources, we have decided that we will be using the Bootstrap framework to enable our web application to be responsive and dynamic. Because of the popularity and proven success of Bootstrap as a responsive development framework, we believe we will be able to best create a dynamic and mobile friendly web application with the use of Bootstrap. Having limited time, using Bootstrap will allow us to create a dynamic and responsive web application without having to waste time on developing custom code. Although making custom changes to existing Bootstrap styles can sometimes be an issue, we believe that any changes we may have to make will be small enough that custom development will be minimal and simple.

Proving Feasibility

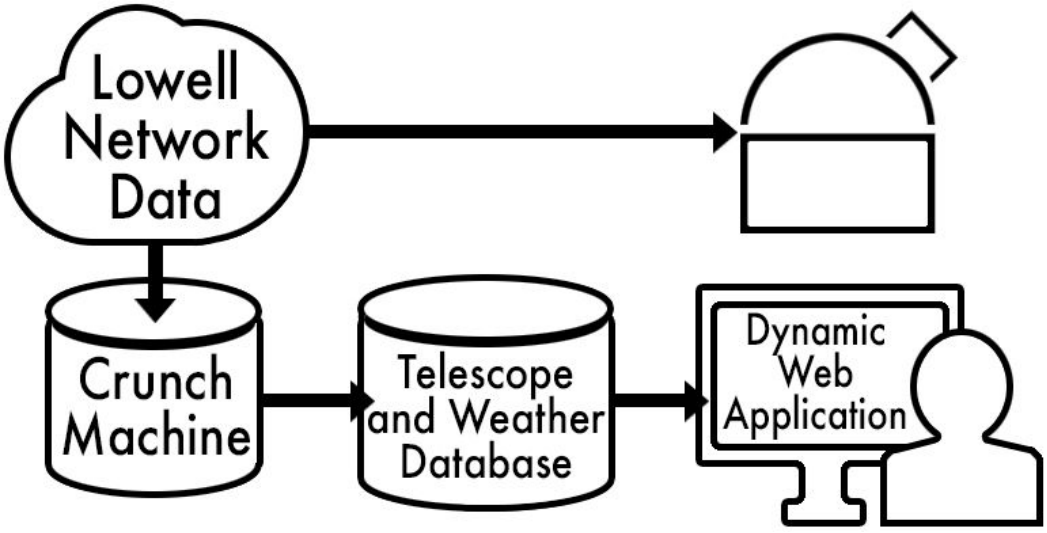
Proving feasibility of the Bootstrap framework will be quite simple as the development and test cycle of a web page is quite rapid. Once we have an initial web page developed for testing, we will demo the use of Bootstrap and how our web application will be fully responsive. There

are also many examples that showcase the many features of the Bootstrap framework online, so demos of specific features and elements of the framework are readily available at any time.

Technological Integration

Although our project has a few challenges, we believe we have viable solutions to most. After coming up with solutions to each individual problem we are finally able to start seeing how our the individual parts of our project are going to come together to form a single system.

The system diagram below depicts how our team believes the final system will look like once we are able to integrate our solutions together.



Conclusion

The purpose of this document is to outline the challenges of this project, and the possible solutions that we have discussed. We have analyzed several candidate solutions for each problem and chosen ones that we believe to be the best given the restrictions of this project.

Project

In review, our project is to deliver a web application for our client that showcase the diagnostics for their telescope data, such as images captured by an all-sky view cam, weather

conditions, and a target list. Our web application will also provide basic observatory functionality, including emergency shutdown and the ability to update the target list queue.

The project requirements have posed a number of different challenges. Our web application will require front end development to display all of the telescope data. A database will have to be created to store all of the collected data. We must find a way to network our web application to the client crunch machine to obtain the data.

Challenges and Solutions Table

The Challenges and Solutions table below explains our teams technological challenges, proposed solutions, and our confidence level for each proposed solution/problem.

Tech Challenges	Proposed Solution	Confidence Level
Network Accessibility	VPN Access	Low/Moderate
Data Accessibility	Django/Python	High
Web Application Front-End	Bootstrap Framework	Very High
Web Application Back-End	SQL Database	High

Solutions Summary

As far as solutions are concerned, Networking is the only thing that is still very much in the air. We are promised certain data will be available on a crunch machine, but the machine is still not quite finished, and the location is still a bit ambiguous (Note: it will definitely reside in the NAU network). All of the other solutions seem to fall into the vicinity of an average web app. Some of the requested features such as an all-sky view overlay, or visual target list will take some extra effort, but seems well within the range of possibility at the moment.

We are confident in our solutions to the requirements given to us by our client. With the exception of the network accessibility, we have a very good grasp of what the challenges entail, and of how to best solve them.

Our feasibility research has left us with a few questions to ask at the moment:

- Will we have to train the client in maintaining or modifying the site? Yes, please :)
- If we can't have python running on the web server, what will we have to use instead? PHP should always be able to run, even though it isn't the best solution.

We will probably have more questions, but if nothing else, this document has been a solid starting point and will allow us to ask the right question.

Closing

The skill set in the group seems varied enough to at least give us a great starting position for planning the project. Most of the problems seem to able to be broken up into much more manageable chunks. As research and planning progresses, more of the unknowns will become clear, and we can make changes as necessary. For the time being, everything seems to be on track.

For the foreseeable future, we would like to stay in communication with our clients to ensure that the progress we are making is satisfactory. If the basic requirements are met much earlier than anticipated, it will give our project team much more room to develope the above and beyond features which would greatly improve the quality of the finished product and increase the possibility that our system will be a successful one.