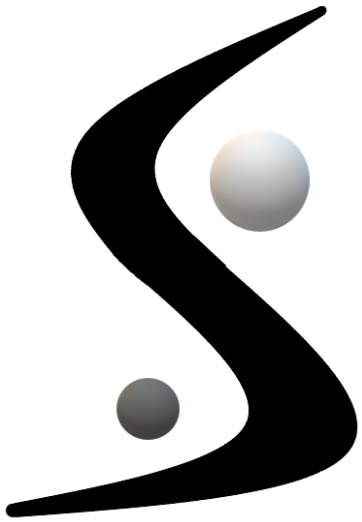
Team Sirius



Software Design

2/12/2021
Project: NPOI Dashboard Web Application

Team Members:
Mario DeCristofaro
Cameron Hardesty
Hannah Park
Matt Rittenback

Sponsors:
Jim Clark
Henrique Schmitt
Adam Schilperoort

Mentor:
David Failing

Version 2.2

Table of Contents

1. Introduction

The Navy Precision Optical Interferometer (NPOI) is the largest optical Interferometer in the world. The NPOI is an astronomical long-baseline optical interferometer that has been in operation in Anderson Mesa since 1994. This NPOI dashboard web application is sponsored by Jim Clark from the Navy Precision Optical Interferometer Naval Research Laboratory, Remote Sensing Division as well as Adam Schilperoort who is a software engineer at Lowell Observatory. This dashboard will provide valuable information to observers at Lowell and researchers at NPOI. Observers and researchers will be able to use a web system in which they would be able to interact with graphs of star data relevant to building a better understanding of the space we seek to explore and travel through in the future.

NPOI and Lowell Observatory work together because of their common interest in space research and collecting data to better understand binary systems.The shared data comes from an array of six mirrors spaced tens of meters apart to gather data rather than a single telescope, and is so large because it combines star light collected to form a high resolution aperture. NPOI currently collects 3 major pieces of data; instrument data (temperature, humidity, and pressure), observational (date, time, and operator), and pipeline data (condensed text format of the collected data). The collected data is invaluable to inform instrument health, performance, and other diagnostic data. As, it remains unusable to administrators, engineers, observers, and researchers alike because this data is so dense and time consuming to parse. Also the location of the data is spread across different machines as well as on directory trees across the network. Making it difficult and sluggish to analyze.

Team Sirius's solution to this would be at minimum provide a webpage that will be deployed to the server at NPOI with graphs that users will be able to interact with, displaying the most current and previous observational data collected. This webpage will also display the instrumentation data as plots organized by date. Storing this information in a database and not as one big html dump will also increase the efficiency and speed of the website. This will issue an unlaggy graph to illustrate the data collected years prior up until the most recent. We will be using MySQL to store the collected star data, giving us the ability to use Django's python extensions to retrieve that data to then be neatly organized in a graph. The parsed generic text data will be stored on MySQL using the different identifiers. Having streamlined the requirements stated by the client, we are confident as a team that we will be able to efficiently and neatly illustrate the data onto an appealing web application.

2. Implementation Overview

The NPOI dashboard will be a more efficient and reliable tool for researchers and engineers to gain the information needed to complete their daily tasks. The dashboard will produce multiple graphs to show vacuum pressure, temperature and other diagnostic readings to be analyzed. It will also allow engineers to adjust the array configuration so that observers can determine which stars to look at. Finally our web application will produce a display of star observation data. The assortment of graphs and charts will have the capability to scale easily and flexibly, as well as be reorganized by date or observer. These visualizations will be easily reconfigurable to suit each user's individual needs. Team Sirius utilized a process similar to the producer-consumer pattern where we will break down our project into two categories. The first is the back end database production, storage, and retrieval. The second is the front end web dashboard data analysis and organization for consumption.

For the first category of database production, storage, and retrieval Team Sirius has decided to utilize MySQL for a multitude of reasons. NPOI is utilizing PhpMyAdmin to administer their current database. PhpMyAdmin is a software tool written in PHP, intended to easily handle administration of MySQL over the web. Utilizing MySQL in our project will allow for easy transition between our web dashboard and the currently in place system. Furthermore MySQL is by far one of the most popular database management systems out there, boasting one of its largest selling points being extremely fast read operations. It contains all of the features we need for a database management system. MySQL is designed with focus on the Web, Cloud, and Big Data. MySQL provides improvements in scale-up and scale-out performance, high availability, and data integrity, monitoring, and resource management, development agility, and security. MySQL additionally does not demand much of your computer resources meaning utilizing MySQL will cost less computer memory or CPU usage compared to other Relational Database Management Systems. Finally MySQL is compatible with our choice of Django as a web framework.

For the second category of front end web dashboard data analysis and organization for consumption, Team Sirius has decided to utilize Django as a web framework. We decided to adopt Django as our web framework due to the fact that it is developed in Python and its built-in full stack capabilities are easy to use and well documented. Django comes as a complete package so that once installed and configured we can start developing right away. Django follows a Model-View-Template architecture. This Model-View-Template architecture was critical in our choice of web framework because this style of architecture is essential for us to produce the best product based on our requirements. The Model helps to handle our database which will store the collected data. It is a data access layer which handles the collected data. The Template is a presentation layer which handles the User Interface portion in totality. The View is used to execute the business logic and interact with a model to carry data while rendering a

template. Django offers our team a clear and organized way to carry out the best project possible while fulfilling our requirements.

3. Architectural Overview

One of the key components in developing an application is building out the architecture that the system will follow. This allows for an easy to understand vision of how each of the components of an application work together.Below is a graphical representation of our system diagram and how each component will be communicating with each other respectively.

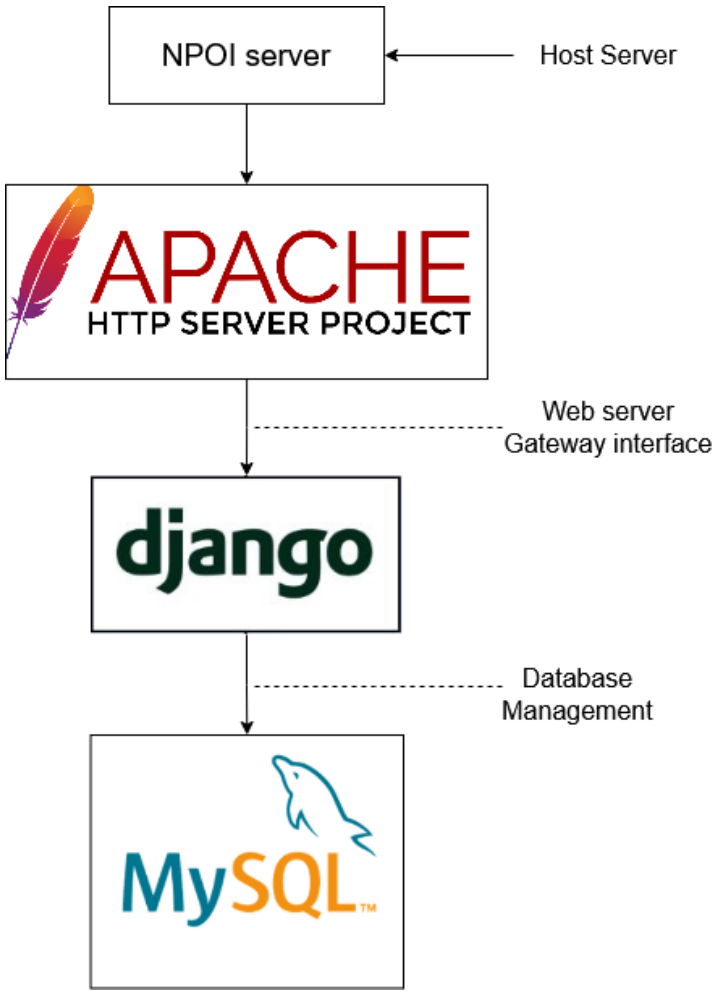


Figure 3.1 Software Architecture Diagram

As shown in Figure 3.1 NPOI will be providing the server spaces that will allocate resources and space to house the dashboard application. From there the application will be built on the Linux platform provided, which is the Sextons server at NPOI. The application will be hosted on an Apache server also located on the Sextons server. From there Django will be the

framework used to house the application allowing easy access to both front end and back end operations . MySQL is the back end chosen by the team to hold all of the diagnostic and research data that will be accessed by the application. The MySQL database will link directly to the Django framework allowing for easy querying and editing of the database.

3.1 Django Architecture

Django uses an architecture pattern known as Model View Controller or MVC. An MVC pattern will allow the developer complete control over what the user sees by allowing them to manipulate every step of the process, from receiving the request all the way through to sending the data back to the user. This control makes our application highly modular without having to change the functional foundations.

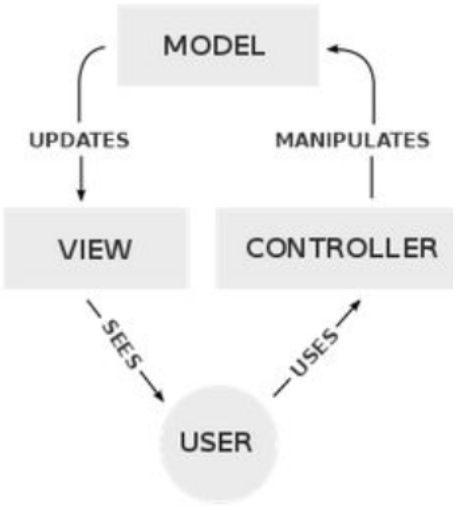


Figure 3.2 Model View Controller architecture diagram

In Django the view and controller aspects of the architecture are handled by the views.py file inside of each sub-application. This Python file doesn't only handle what is shown on any given page but also the functionality and specific details of the page. It works in tandem with the models.py file to query the database and present needed information for any given page. Django then takes the information built into the views.py page and renders the requested web page presenting it to the user. The web pages for our application will implement a common design with the use of a Django feature that allows the developer to set a base page on which all other pages are built off of.

4. Module and Interface Descriptions

After covering the overall architecture of how the application will function, now this document will break down the architecture into five main modules. Below is a detailed description of each of the modules needed in the application. Along with the description are any needed diagrams to show how each of the modules communicates with the overall system.

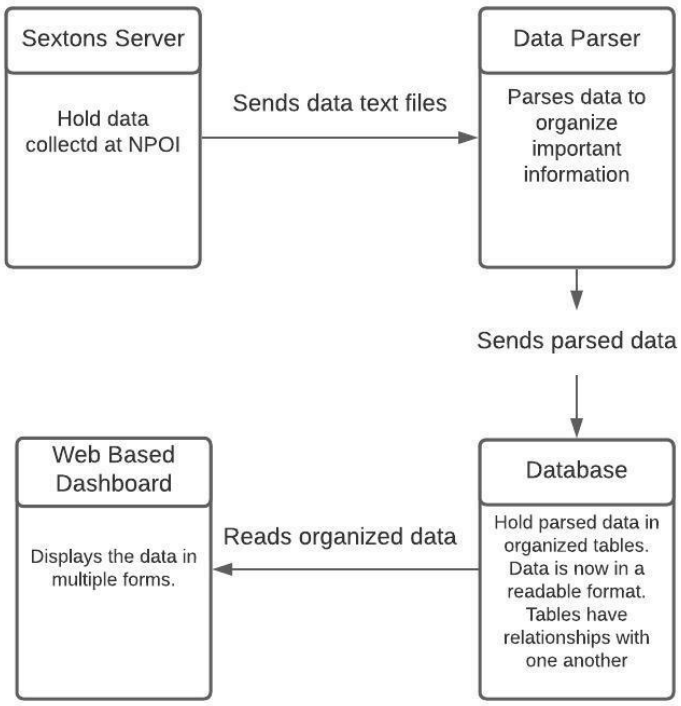


Figure 4.1 Lucid Chart showing the flow between the project's modules

4.1 Data Parser

The data parser will be responsible for parsing through the data collected at NPOI. The currently collected data is stored in generic text files. These text files contain essential data, but are dense and unorganized. In this current form, the data can not be used to its full potential. The data parser will parse through these text files and pull out data and store it in an organized and sorted manner in the database.

4.2 Database

The database will be responsible for holding the data collected at NPOI in an organized format, making it possible to read and write to and from the data. We will read from the database in order to create the graphs, charts and plots which will be displayed on our web based dashboard.

4.3 Web Based Dashboard

The web based dashboard is responsible for being the first resource that engineers and researchers go to find information necessary to complete their individual tasks. Hosted on server space provided by NPOI the web based dashboard application will be an easy to use and well organized first point of contact for necessary information. This application will be built on the Linux platform and hosted on an Apache server located on the Sextons server. The web application will pull data from the database which stores the data collected at NPOI.

4.4 Diagnostic Data

The core module of the project needed by the client is a way to graphically display the variety of readings that they collect from multiple different sources along the interferometer array. The centerpiece of this module will be a page to monitor the pressure and temperature readings for each of the lines along the array as necessary. This data will need to be graphed and displayed quickly so the engineers can make any adjustments necessary for the interferometer to function properly. This portion of the application will need to interface with the database in order to pull the collected data and display it in an easily readable format.

4.5 Array Status

Part of allowing the array to perform smoothly will be ensuring that the operators are able to understand the current array configuration and status of various portions of the array. This will entail needing a map of the array with interactive elements to display what portions are currently operational. With given permissions certain users will be able to edit the configuration if the status of the array has changed.

5. Implementation Plan

With the project broken down into smaller core modules, the team created a schedule for different tasks to be completed in the upcoming semester. We analyzed which modules needed to be completed first and prioritized those in our planning process. Breaking down the project into smaller modules allows the work to be split amongst team members and promotes project parallelism by having more than one task being completed at once. For example, in Figure 5.1 below, the team has been working on Database Integration and while members are wrapping up the process of that task, other team members have begun working on Array Status Page with the data from the database.

Figure 5.1 Gantt graph of passed and upcoming milestones of project

As described in the example above, the modules are planned to end as another one starts to keep the team active and productively working on the project. After all the project's modules are implemented by week nine, then the team will work on testing and refining the project along with creating final report documentation. The team is working diligently to complete an alpha prototype before March 15th, to share with the team mentor and client. The team is confident that following the schedule created in this section will guide it to a successful project completion.

6. Conclusion

The goal of this project is to assist the clients at NPOI with a web-based dashboard that will allow them to view and monitor instrument data of the different metrics the NPOI collects. Whether it be vacuum pressure, temperature, humidity, or a miscalibration of a computer our solution will list all the analytics in a neat way. The dashboard will provide a new solution that allows the client to better maintain the calibration of the NPOI with much less effort and cost of time. With this more streamlined workflow, the NPOI will produce more reliable data for astronomers and researchers to pull and utilize from. Without needing all the time it took previously to dissect and analyze the current generic text format of each star log data.

In conclusion, having the next steps finalized allows us to begin focusing on the main implementation of the project. Having the project broken down helps the team visualize how the application will fundamentally work. The application will be implemented using a Linux based system on Lowell's Sexton server that then collects the star log data to be parsed by the local parser. The parser then sends the data directly to the MySQL database. The application will utilize Django framework to allow for easy integration of the database and the ability of querying and editing. The front-end of the application will use the integrated database along with Python graphing extensions to graph the data with interactive features desired by the client. The team is confident that our provided solution will fit and meet all the functional requirements that were agreed upon last semester. We are excited to continue the work on the prototype and to be able to help streamline the work for observers and researchers at Lowell Observatory and NPOI.