Control System for a Robotic Spectrograph at Lowell Observatory

NAU NORTHERN ARIZONA UNIVERSITY

School of Informatics, Computing, and Cyber Systems

Team: Henry Fye, Nhat Linh Nguyen, Jacob Penney, Jakob Pirkl, Kadan Seward

Client: Dr. Joe Llama & Dr. Gerard van Belle, Lowell Observatory, Flagstaff, AZ

Team Mentor: Rudhira Talla

Motivation

Introduction to Spectrographs

Professional astronomers have the important but difficult task of gathering information about astral bodies, objects they can scarcely conveniently observe. Among the many tools they use for this task, **spectrographs** allow them to learn about the qualities of these bodies, such as chemical composition, density, temperature, and speed of movement, among others. This information can yield important discoveries, such as **habitable planets**, **black holes**, or **the sources and machinations of strange phenomenon**.

Problems and Our Role

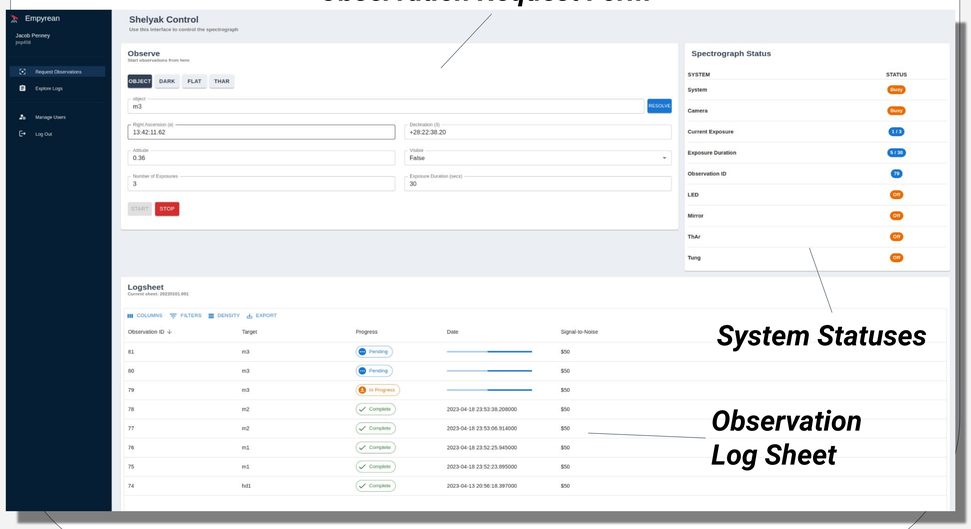
Our clients at Lowell Observatory use some of the finest commercial spectrographs in the world to conduct research just like this every night, but these tools can be unwieldy to use because of their complexity. Software solutions exist which abstract away these complexities, but they are **dated**, oftentimes **break**, and cannot be readily improved upon by those that use them because **their source code is proprietary.**

Solution Overview

Our Web Application includes:

- A **modern and secure UI** which allows astronomers to take observations and receive real-time system and observation updates
- A messaging server **API** which routes user requests and resulting data
- A **database** which stores and allows for querying observations.

Observation Request Form



Technology

React

React is a Javascript library designed for easing web development. It also features many **component** libraries, letting future developers easily add features without expertise in more complicated technologies.

Flask

Like React, we chose Flask because of its popularity and ease in learning. Additionally, it has many desirable features built-in, like **flask-socketio** and **flask-sessions,** which allowed quick implementation of needed features.

Websockets

Without websockets, this project would not have been possible. Normally, all web data must be sent as a response from a user request, but websockets allow for **asynchronously pushing information** from server to client.

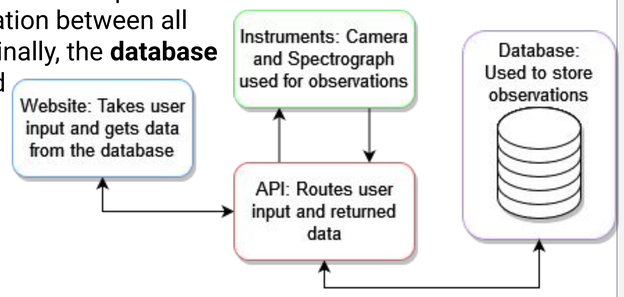
PostgreSQL

With the structure of our data and our client's skill set, we required a popular open-source **relational database**. PostgreSQL was a clear choice because it is very popular, well-supported, and relatively easy to use.

React JS Flask PostgreSQL websockets

Architecture

The primary workflow of this project was designed to be simple and is made possible by only three components outside of the spectrograph and camera: a **user interface**, **API**, and a **database**. The **interface** allows the user to command the instruments or view past observations. The **API** acts as a router and sends information between all the other components. Finally, the **database** stores all of the observed astronomical data.



In this primary workflow, the user selects an object to observe and provides instructions on how, all of which is sent to the API and on to the camera and spectrograph. The instruments fulfill this request, after which the resulting data is returned to the API and sent to the database for long term storage.

Key Features and Outcomes

The various components covered in the architecture provide key features that were specifically asked of us by our client. These were created as project outcomes and they include:

Login System
- The ability to login and manage users on a table on an admin account.

Create Observation
- The ability to input specific information to the spectrograph and create new observations

Explore Past Observations
- The ability to advance search for previously made observations

Status Updates
- The ability to view live status updates as the spectrograph and telescope operate

Future work

- Consider ways to integrate the application with other astronomical tools
- Identify areas where the application's performance could be improved, such as reducing load times, or optimizing database queries