Team Sirius

Software Testing Plan

3/26/2021
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 1.3

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. Observers and researchers will be able to use our web dashboard 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. 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 easy to digest web application. To ensure our product can operate successfully for our client, we will conduct extensive testing with multiple forms of testing to achieve this.

Testing is an important part of any project as it determines what constraints or requirements it needs. With an immense amount of testing a team can feel confident their code works properly. Testing can also save money as the further the project goes the more expensive a bug is to catch if not caught early in the implementation stages. Finally, testing is one of the most important parts to a project's success as it really can illustrate the quality of the product.

2. Unit Testing

The first step of our testing starts with unit testing. Unit testing is simply testing a unit of your application, and this unit is the smallest piece of code that can logically be isolated in the system. In programming languages this is commonly found as a function or method. Unit testing specifically requires that the test does not require external systems, as the goal of unit testing is to isolate each part of the application to show that they are working correctly individually. Each unit test provides a strict written contract that the piece or section of code must satisfy. With the use of unit tests, the team is able to find problems in the application early in the development cycle. With this concise testing style, during development, it allows for easy detection of the exact method or function that is not working as it should.

For our project's unit testing we will primarily create tests for the front end of our project. This front end testing will focus on the user input and communication with the client database. Due to our team utilizing Django as our framework, we can implement Tox, which is a tool for running tests in different virtual environments. Django actually includes a basic tox.ini which automates some checks and testing that our build server performs on pull requests.

Tox runs these unit tests that test things such as import settings, the document spelling checker, and code formatting. We can run and install the Tox command from any place in our Django source tree. By default, Tox runs their test suit with the bundled test settings file for SQLite. Django additionally includes a set of JavaScript unit tests for functions in certain contrib tools. These JavaScript tests are not run on default by using Tox because they require Node.js to be installed. When executing the Tox command, it runs npm install to ensure test requirements are up to date and the runs npm test. Below will list out the unit tests that will be testing the functionality inside the three main modules of our project:

2.1 Database

Establishing a connection with NPOI's database will be the first step in utilizing our web application. The database connection is needed for our graphs to be able to display data for the data graphs, as well as, showing the latest version of NPOI array.

- Validating a connection with NPOI's database
- Check if main data is being stored in the database:
    - Array status and comments
    - FDL station data
    - Vacuum and pressure data

2.2 User Login and Authentication Privileges

The user login page is where the user with a known account given by NPOI can log in to view our dashboard. The authentication system is through Django so using their testing frameworks like Tox we will be able to verify which user has logged in. While also verifying what privileges each user has access too. As an admin you can create new users to the dashboard as well as remove users and their specific privileges.

- Verifying the account that logged in
- Check for observer or test user
- Check for admin or engineer

2.3 Data Graphs

The data graphs page will show the important metrics needed for any observer or employee at NPOI. If the corresponding data from the database has not been imported then no user will be able to view the data illustrated in our graphs. Having each categorical measurement on their own specific page will continue to reduce load times. This also allows any observer or engineer at NPOI to view each specific instrument measurement at the same time if they have multiple tabs open.

- Check if the vacuum data is imported from the database
- Check if the pressure data is imported from the database
- Check if the FDL station data is imported from the database

2.4 Array Status Graph

The array status page will allow any user the ability to view the health and maintenance of the array at NPOI. If the corresponding data from the database has not been imported then no user will be able to view the current health and maintenance of the array. Whether you're an admin or observer you can hover over each station on the array to show the current status and most recent comments posted about the station. Although, being admin grants you the privilege to update the status and or make a comment about that specific station.

- Check to see which user is viewing the graph
  - If admin: display status buttons and comment boxes to allow for changes in the status and comments
  - If observer: only display the status and comments most recently pushed by an admin
- Check if each array status is imported from the database
- Check if each array comment is imported from the database

With a web based application for our project, our unit testing is primarily needed to test the front end by performing double checks for data storage and retrieval. After testing the individual modules of the project, the next step is to test how each of these modules will interact and integrate together.

3. Integration Testing

The second step of our testing is conducting integration testing. This type of testing is a method of testing the overall compatibility of modules inside an application. This project will conduct this type of testing by building tests to ensure that each of the pieces of an application link together properly and integrate into the overall system. The goal of integration testing is to guarantee that all of the processes that operate between modules function properly and contain the correct data at the end of the operation.

The integration process for our application is fairly simple given that all of the components have been built from the ground up during our development process. As shown below in figure 3.1, there are only two major components that need to be tested when it comes to integration. First, the actual application being integrated into the client's system and second, the hosting of the application on an Apache server that the client will be providing.

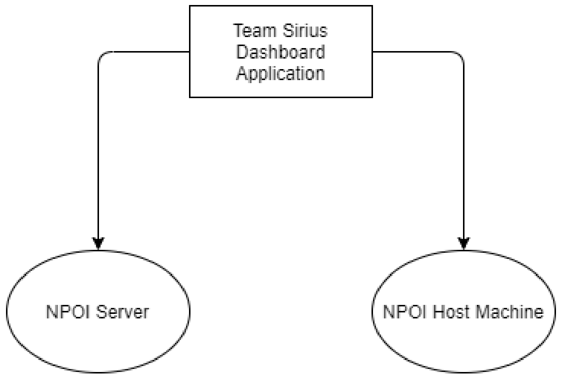

Figure 3.1: Diagram showing the primary integration events of this project

The integration of all of the Javascript modules and the backend database was tested and successfully integrated during the development of the alpha prototype. In this testing the main edge cases that we considered were the attempted querying of data that did not exist in the database, however despite our own testing, all of those errors are already handled by the Django framework. After the edge cases of attempted queries were resolved, the next piece to test was to check to see if data was being passed between the database and Django properly. For testing this, we developed a few test queries that checked the different data types that are present in the database and displayed them in a webpage to ensure that they were translated correctly.

Handling the integration to the client system and the hosting of the webpage on an Apache server the testing is a straightforward task. To test the system on an Apache server we will implement through the Django built in web server gateway interface(wsgi) settings to attach the application to the server and ensure multiple computers can access the application at the same time and that data changes will persist between client machines. The only testing left to be completed after accomplishing the network interface is to implement the whole application on the client's machine and repeat all of the previous testing to ensure that each individual module is working properly.

Once the individual modules functionality and integration is successful, the next step in testing is working to test how the project handles in the hands of the user.

4. Usability Testing

For the third and final step of our testing, is to conduct usability testing. Usability testing is a method of testing the functionality of an application. This type of testing is conducted by observing real users as they attempt to complete their allotted tasks on it. The users are observed by the researchers or engineers of this application. Based on the results of these tests the researchers report their findings to the engineers who then can make the required changes to their application in order to improve and better serve the users. Usability tests identify areas where users may struggle with the application, and help to get recommendations for improvement. The goal of usability testing is to better understand how real users interact with the final version of the application and to improve the application based on the results of this testing. The primary purpose of a usability test is to improve the overall design of our application.

Given the nature of the project being primarily based on the end user, a majority of our usability testing will be focused on the client feedback and their user stories. This testing will be primarily focused on the usability of our front end of our application since that is where the majority of users will be operating. The purpose of our application is to present data in a much more digestible format than the client's current workflow. Thus when the end users are testing our project, they should understand what to do with the data they are presented with. In order to make other operations as seamless as possible, we will be analyzing and noting the use of our product by an end user. This will be accomplished by meeting with our client and allowing them to use the system to see if they are satisfied with how the application completes certain tasks and how easy they are to perform. The team will then take this feedback and improve the application by implementing changes based on the feedback then presenting it to the client again in a follow up test to ensure the change is a positive one.

Our usability testing will take up the majority of our testing time since almost all of the integration testing and unit testing has been completed already. The goal of this usability testing will be to make the application as easy to use as possible while still maintaining full functionality. Working with our client we will conduct multiple rounds of usability tests to collect feedback and implement changes between tests. These changes can address how the tasks are accomplished to make them more simple or efficient. The overall goal of this usability testing is to ensure that our client and any of the other end users can pick up the application and accomplish what they need to right away using our dashboard.

5. 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 helpful manner. 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.

In conclusion, having the testing plan finalized allows us to improve upon an already functional project. Having the testing plan broken down helps the team to conceptualize the improvements needed by the users. The application will be implemented using a Linux based system on Lowell's server that collects the 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. Based on the results of our testing, our team is prepared and eager to improve upon our already working project. 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.