Testing Plan

March 29, 2021

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 2
2. Unit Testing 4
    2.1. Flask 4
    2.2. Dash 8
    2.3. User Class 9
    2.4. PyMongo 10
    2.5. FlaskMail 10
3. Integration Testing 11
    3.1 Page Navigation 11
    3.2 Graphing Functionality 11
    3.3 Logging in and out 12
    3.4 Administration Functionality 12
4. Usability Testing 13
5. Conclusion 16

1. Introduction

Since the discovery of the first exoplanet in the early 1990s, society has been infatuated with the exploration of outer space with the goal of finding more and more planets outside of Earth's solar system. A vast number of these exoplanets have been discovered, but none have been found to be Earth-like planets orbiting sun-like stars. Assistant astronomer Joe Llama and his team at Lowell Observatory specialize in the study of exoplanets and they aim to find a planet that is similar to Earth. Dr. Llama's research has brought him to gather data on the Sun in order to better analyze its radial velocity. By gathering data on the Sun, we may more precisely detect the presence of an Earth-like planet around a foreign star. Yale University has developed the EXtreme PREcision Spectrograph (EXPRES), which is a tool to measure such data. Lowell Observatory has partnered with Yale University to use their EXPRES tool on the Lowell Observatory Solar Telescope (LOST) to gather data on the Sun.

Dr. Llama is leading the team in gathering the data, and they gather data at a rate of 40GB per day. This is because Dr. Llama is taking 5-minute exposures with EXPRES, for around 8 hours a day. Now that Dr. Llama and his team have begun gathering and storing data, they have encountered three areas in which their workflow is lacking. These areas are:

1. **The ability to quickly and conveniently interact with data.** Once the day is over, all data collected from that day is store in FITS files on the Lowell servers. There is currently no way to access visualization of the collected data without manually parsing through large FITS files and rendering graphs individually.
2. **The ability to share select data with the public.** As a public institution, it is important to Lowell Observatory that the public be able to learn about research that the Observatory is involved in. Currently, data from LOST is not being shared publicly.
3. **The ability to share all day with researchers at other institutions.** Along with the public, other research institutions should have full access to LOST data in order to facilitate research.

In order to provide Dr. Llama and his team with interactive data visualizations and a platform for sharing data, LOST EXPRES has developed a web application that will provide a clean and intuitive user interface for both researchers and public users to view and interact with data. Key features of the application include:

1. **A graphing page complete with interactive capabilities.** The application allows users to view a radial velocity graph and dive further into the data by viewing a one-dimensional and two-dimensional spectrum for each radial velocity. Each graph allows users to zoom in and out, adjust the axis, adjust the density of data, and record pictures of the graph. In addition, two-dimensional spectrum graphs have the option to display one or more orders.
2. **Role-based permissions.** Anyone from the public is able to view and interact with a limited amount of data. Researchers looking for a larger amount of data are able to request researcher-level access with a valid email address and associated institution. The sole administrator can approve researcher requests.
3. **An administrator dashboard for controlling information.** The sole administrator is able to determine the range of data that is currently available to the public. The public is not able to see the entire data set collected by Lowell, only a portion of the data.

Other features include a glossary page for looking up important definitions and a news page containing current stories and information about Lowell Observatory. Both pages will be editable by the administrator. In addition, data is not only available visually, but also available to download from the application.

To ensure our application meets all the requirements of Dr. Llama and functions correctly, efficiently, and in a user-intuitive manner, our team has laid out a software testing plan. Software testing introduces a series of tests that examine the integrity, quality, and functionality of a piece of software. The purpose of testing is to uncover any unwanted behavior or expose shortcomings in the functionality of a product.

LOST EXPRES is implementing unit, integration, and usability tests. Unit tests will test individual sections of our code to ensure each section executes as it is intended. We will be writing unit tests to ensure navigation through our application is correct, that the application reads from the database correctly, data is plotted correctly, and users can properly interact with the application. Integration tests will examine how well these separate components work together to produce the desired functionality. For our application, this will include testing the experience of using the application as a whole and navigating between different features. Usability testing will assess how user-friendly our application is, by asking people unfamiliar with our project to navigate our application and complete a variety of tasks. This will simulate how end users will interact with our

application and will reveal any issues with the interface and other software components. Now that we have introduced our product and given an overview of the tests, we plan on executing, we can look more closely at the details for unit, integration, and usability testing.

2. Unit Testing

Unit testing is a type of software testing where code is sectioned into individual components, or "units" to be individually validated. The purpose of these tests is to verify that each piece of code runs as expected; doing this on a small scale should make a difference on a large scale. The thought process is that if each unit of a project is tested to check it works, you can be more assured that the code will work how you expect it to, and to make it more clear about where future bugs appear from in the code.

One of the important things to define for unit testing is what constitutes a “unit,” and how it changes over different areas of the project; this will affect how we write our tests and where. The way we will break up our code is first by library, and then by unique component, as most of the libraries used are extensions of Flask. Our file structure uses a factory pattern to apply behavior to the underlying Flask server, meaning the code for different libraries are in separate sections.

2.1. Flask

Flask acts as the main server functionality, and uses several global variables that are necessary to test. For Flask unit testing, we will be using a combination of the Pytest library, as well as the Selenium library to provide a webdriver. For each of the files in the file structure, we will have a separate file with the “test_” prefix to run that file.

2.1.1. Routing

Url directing in Flask is handled by initializing functions specifying the path of the server they route for. Routes can parse types of routes to pass as parameters into the function.

Unit	Description	Bounds	Input Example	Exp. Response
Static Pages	Static pages are rendered from templates and should not throw errors from inside Jinja.	-A valid path: slashes and alphanumeric characters	-Request: /	-The index page is loaded correctly with all necessary libraries included.
Dynamic Pages	Dynamic pages require the server to send a request to the database before the page is rendered to the user.	-A valid path: slashes and alphanumeric characters	-Request: /glossary	-The glossary page renders with records loaded from the database
Error Handling	If a route is invalid or forbidden to outside accounts, the server will return an error page.	-An invalid path -A valid path without a valid login	-Request: /invalid	-The server returns the appropriate error page for the error being thrown.

2.1.2. Post Requests

Handled inside route functions, types of requests can be filtered and parsed with Flask's request variable to execute other code. Pytest and Flask's test client will provide dummy request contexts.

Unit	Description	Bounds	Input Example	Exp. Response
Sign In	The login page will attempt to authenticate the email and password with the database.	-A valid email -A valid password: Alphanumeric characters	-Email: abc12@nau.edu -Password: 3xamp1e	-The server initializes the user class and runs it against the database for a singular match. Not finding it will result in a failed login attempt.
Register	The register page sends received post requests to the administrator's email for verification.	-A valid email -First name: Characters, <20 long -Last name: Characters <20 long -Institution: Characters	-Email: abc12@nau.edu -First name: Geoff -Last name: Jefferys -Institution: NAU	-The admin receives an email containing the applicant's input information, with a link to confirm their account.
Forgot Password	The forgot password page sends an email to a user if their email is registered to the server.	-A valid email	-Email: abc12@nau.edu	-If the email exists in the database, it will send an option to reset the password to that email.
Add to Glossary	The "add terms" page adds definition records to the database to be rendered by the glossary page.	-Term: Unicode characters -Definition: Unicode characters	-Term: Soup -Definition: A consumable liquid	-The server adds a record for soup in the glossary collection. -The request will fail if not authenticated as the administrator.
Add Article	The request adds a new article to the database to be rendered in the news section.	-Title: Unicode characters -Subtitle: Unicode characters -Author: Unicode characters -Content: Unicode characters	-Title: Example -Subtitle: subtitle -Author: Geoff Jefferys -Content: This is a sentence, but it could be a page	-If the current user is admin, the server adds a new record to the article collection.

Change Email	The request updates the user's email, which will be verified by sending an email to both old and new records	-Old email: alphanumeric characters -New email: alphanumeric characters	-Old email: zyx90@nau.edu -New email: abc12@nau.edu	-If the old email belongs to the current user and exists once in the database, but the new email doesn't exist, update the account's record in the database.
Change Institution	The request updates the user's institution. This does not need to be verified	-Institution: Unicode characters	-Institution: NAU	-The server will update the institution record of the current user if authenticated

2.1.3. Jinja

Jinja uses Flask's request context and renders variables to construct web pages. We will be using Pytest and Flask's test client to verify the output.

Unit	**Description**	**Bounds**	**Input Example**	**Exp. Response**
Base Page	Jinja templates can extend each other to replace/add content. Every HTML page extends the base page in some way.	-Request: A valid path on the server	-Request: /account	-The base page will be rendered with content on top of it; all valid pages will have a navbar.
Macro Functions	Jinja allows functions to be defined to automate repeatable elements	-Navbar Elements: list of ordered pairs with path and navbar display.	-Navbar Elements: [("/admin", "Admin")]	-The base page adds an "Admin" button to the navbar.

2.2. Dash

Dash unit testing relies on Pytest to check expected outputs, and Selenium to interact with control elements. The tests revolve around expected outputs of client and server-side callbacks. The located file will be named with the “test_” prefix.

Unit	Description	Bounds	Input Example	Exp. Response
Layout Render	Dash requires elements to be defined in python, so we must test that it renders the correct HTML.	-Layout: Python rendered HTML elements	-Layout: html.Div(className=”class, children=[“hello world”])	The Dash renderer will return hello world
Choose Spectrum	Test that clicking a point of the graph returns its filename.	-RV plot: an HTML element -Click data: JSON information	-RV plot: -Click Data:{index:0, filename”stuff00”}	The server returns an existing filename to the page.
Choose Date Range	Test date selection sends an ordered pair of dates within range and zooms in the plot.	-RV plot: an HTML element -Max date: date of the latest entry -Min date: date of first entry	-RV plot: -Max date: 29/3/2021 -Min date: 1/1/1970	Zooms in RV plot to selected date range
Render Spectrum	Test correct interaction with the graph, including switch and range inputs.	-Spec Data: an HTML element -Resolution: Ratio, 1:1 to 200:1 -2D: boolean -Orders: ordered pair of ints, 0-85 -Log: boolean	-Spec data: -Resolution: 50:1 -2D: True -Orders: 40-45 -Log: False	Graph renders with selected orders in multiple colors, with a reduced resolution to affect its speed

2.3. User Class

User class unit testing will be using the same technologies as testing Flask, that being Pytest and Selenium libraries. This testing in this section will revolve around user permissions and login capabilities. The same file structure will be used as well, by using the “test_” prefix before the testing files.

Unit	Description	Bounds	Input Example	Exp. Response
Sign In	The user class must be able to authenticate the input user and update the Flask session variable.	-Email: A valid email with one record in the database -Password hash: Unicode characters	-Email: abc12@nau.edu -Password: 3xamp1e	The user class is authenticated against the database, and the session variable is updated to match.
Sign Out	The user class must be able to sign out if it is already authenticated. A user must be logged in to log out.	-None	-None	The user class is logged out, and the session variable is updated to match.
Roles	The user class has multiple roles, so both the researcher and admin roles must be tested	-Email: A valid email of a specific role, repeated for other roles	-Email: abc12@nau.edu	The different roles have access to their respective tasks, these tasks operate as required.
Register	The user has the ability to change roles through registration to have more access within the site.	-Email: A valid email with no current record in the database. -Institution: Unicode characters -Name: Unicode characters	-Email: abc12@nau.edu -Institution: NAU - Name: John Smith	The user class information is sent to an email accessed by the administrator of the site awaiting approval.
Generate Password	A random password is generated and referenced to the database.	-None	-None	A password is generated and added to the user class database entry.

2.4. PyMongo

In order to test interaction with our MongoDB database, we use Pytest by itself to verify success. The same file structure will be used as well, by using the “test_” prefix before the testing files.

Unit	Description	Bounds	Input Example	Exp. Response
Connection	Test if the Python interface has connected to the MongoDB database.	-URI: The identifier for the MongoDB database holding the current data for the website.	-URI: ‘mongodb+srv://Name:name@someLocation’	A valid connection to the identified database.

2.5. Flask Mail

The server requires it to be capable of sending emails to the administrator. Flask mail will use Pytest to test authentication; the same file structure will be used as well, by using the “test_” prefix before the testing files.

Unit	Description	Bounds	Input Example	Exp. Response
Send	The mail server authenticates messages at runtime, sending registrations to the admin’s personal email.	-Mail server: A valid email address -Port: Integer -Username: A valid email address -Password: Alphanumeric characters -Recipient: A valid email address	-Mail server: test@gmail.com -Port:200 -Username: lost@gmail.com -Password: eggsample	The server sends an email with the configured credentials, and the email is received by the chosen email.

3. Integration Testing

Integration testing is used to determine that all functionalities in your application are working properly and also connected to the correct places. The goal of integration is to ensure that the software matches the schema that was created during the design phase of development.

To perform this type of testing, modules that are linked to one another will be examined in order to ensure that everything in our program is interacting with each other correctly. The modules that integration testing needs to be used on for this project include page navigation, graphing functionality, logging in and out for users, and administration functionality.

3.1 Page Navigation

To test page navigation, the tester will simply need to double-check that we are properly routing between all of our HTML pages. Our tester will need to be able to access the home, news, data, login, logout, account, register, admin, and glossary pages from the navigation bar at the top of each webpage. Each label in the navigation bar should bring the user to the corresponding webpage.

There are a few other places where navigation can take place between two pages without the navigation bar. This happens when an administrator is updating the glossary or news sections, or when a user is logging in or out. These integration tests will be covered more thoroughly in Login and Admin integration tests.

The purpose of the integration test on this module is to ensure that all directories on the website route the user to the intended destination.

3.2 Graphing Functionality

To test graphing functionality, the tester will be required to manipulate the graph data in various ways. This includes zooming in, zooming out, changing from 1d spectrum to 2d spectrum,

setting resolution, selecting orders, and resetting the axes of the graph. The tester will know that the graphing functions are properly integrated when every graph can be correctly displayed and manipulated, as this will demonstrate that the graphing page and database are communicating with each other.

3.3 Logging in and out

To test logging in and out, a user will attempt logging in as both a researcher and administrator. When logged in as a researcher, the user will be able to see graphing sections that were not set to be viewed publicly. When logged in as an administrator, the user will be able to see everything a researcher can access, as well as additional administrator privileges. To ensure that logging out is properly integrated into the system, the user will logout. After that, they will try to access researcher and administrator only data, which they should be prevented from doing since the user is now logged out.

The goal of testing this module is twofold. The first thing to test is that the users we are logging in as are actually located on the database, in order to make sure that this module is integrated with it. The second thing to test is that our researcher user has researcher privileges, and our administrator user has administrator privileges. If this is the case, then our login module is properly configured to pull specific data about the logged-in user from the database, ensuring integration.

3.4 Administration Functionality

To test administration functionality, the user will first need to be logged in by using an administrator username. The user can then attempt to add items to the glossary, news articles to the news section, and designate which data is set to private or public.

Testing the integration for glossary and news pages will be similar for each page. First, the user will access the admin webpage they are performing the integration test on. Next, the tester will try to add an item to the glossary/news page. After that, the tester will go to the webpage that lists out the glossary/webpage to check and see if the item they just added is there. If the item is present, then the test passes.

To test the integration for changing the accessibility of data, the tester will use the admin page and attempt to set the graphing data of a specific day from private to public. After this change has been made, the tester will log out, navigate to the graphing page, and check to see if the data that was just set to public is visible.

4. Usability Testing

Usability testing implements the use of end-users with software testing. In this portion, there won't be any kind of programming or code testing. Instead, we will be using ordinary people who will represent the general public. The main goal of this section is to attain feedback on the usability of our product. This will include the user experience within the UI and will help us to refine and test the ease of use of the LOST EXPRES web application. This will not only help to improve the usability of our web application but will also help to detect and fix bugs that weren't already discovered by unit testing and integration testing. Usability testing is especially important for our purposes because a large portion of our web application is used by the common public.

In order to effectively test our application, we need some criteria for the people used in testing. The plan is for members of the group to gather family members, friends, and other people willing to test the web application, and to simply have them use the web application. For the most part, members of LOST EXPRES will have to host the web application as well as the server to run the application, so most testing will be done locally. One of the main criteria for selecting testers will be that they have little or no exposure to the web application. The best way to attain fair, accurate feedback is to get it from someone who has not had the opportunity to familiarize

themselves with the web application, because they won't have any prior experience or opinions on the web application.

For each testing session, we will provide the participant with a set of directions that will vaguely explain to the participant what they should be doing. This will be written carefully as to not give the participant any direction. The site itself should be very clear and explain how to accomplish the tasks through the web application's design. We will test participants one at a time, as the site should be easily navigated and the directions given to participants should be easy to perform by one person on their own. Most, if not all, tests will be done in person, because of the complexity of running the code to host the server, so all participant findings can be easily observed and documented.

In order to get accurate feedback from participants, they will test the web application in 3 separate steps.

1. **Testing for the general public, using the web application as a public user with no privileges, and only having access to admin deemed public information**
2. **Testing for researcher, using the web application with limited privileges and access to all information**
3. **Testing for administrator, using the web application with full privileges and access to all information in the database**

User Instructions:

Hello, thank you for participating in the LOST EXPRES application testing process.

The following instructions are provided in order for our team to properly collect accurate data. Please follow the instructions to the best of your ability and if you cannot perform an activity, skip it and continue on with the test.

Terms to know:

Public User: a user that is part of the general public, this role is designed for an ordinary everyday visitor of the website, only has access to public information.

Researcher: a user who has requested researcher rights, and has been accepted by administrators to be recognized as a researcher, has access to public information as well as private information.

Administrator: a user who has been designated as an administrator has access to public information as well as private information, and also has the ability to add, delete, and modify the web application's content. An administrator can also accept/reject researcher requests.

Knowing this information, we ask that you place yourself in the position and mindset of each role while completing each task that pertains to the role. Throughout testing the web application, take mental notes of any aesthetic or design implementations that you feel might improve the application.

Public User:

- Navigate to the News page, and open up a news article
- Navigate to the Data page, scroll down to the bottom, notice there is no graph displayed at the bottom, scroll up and select a point on the top graph, now scroll down to notice the selected point being graphed
- Zoom in, zoom out and reset a graph on the data page
- Navigate to the glossary page

Researcher User:

- To begin, navigate to the login page
- Check to see if an incorrect log in will create an error
- Login with username: <username for premade account> and password: <password for premade account>
- Test all of the actions listed in the “Public User” section
- Log out of Researcher account

Administrator User:

- To begin, navigate to the login page
- Login with username: <username for premade account> and password: <password for premade account>
- Test all of the actions listed in the “Public User” section
- Notice a new selection on the navigation bar, titled admin. Select this and you will notice fields for posting news articles, create a new news article, and check if it was added to the news page
- In the admin controls, you will notice a new sidebar to the left of the screen, add a new glossary term

5. Conclusion

The design of the LOST web application has been carefully thought out to ensure the most efficient use of resources and the fastest user experience. By using new and modern frameworks, Lowell will have a state-of-the-art data visualization tool for analyzing solar data. LOST EXPRES's web application will deliver an interactive experience to multiple types of users. Public users will gain insight into Lowell Observatory's research, satisfying their mission of public education and outreach. Meanwhile, research users, approved by a Lowell admin, will be able to view all of LOST's data, advancing space exploration and scientific collaboration. The sole admin user, our client Dr. Llama, will have the satisfaction of sharing his research findings in a modern way while maintaining control over the application. Overall, the design of the LOST web application ensures that Dr. Llama will receive an elegant solution to his current problem: the lack of any simple solar data visualization. The technologies we chose will all contribute to a simple, yet pleasing application that is reliable and scalable to the future of Lowell Observatory.

Through this document, we have outlined our process for testing our software and have provided detailed examples of tests that will be implemented. All our tests can be classified into three types: unit testing, integration testing, and usability testing. By using a combination of these three types of tests, we will be able to discover any unexpected problems, incorrect behavior, or poor design in our application. Unit testing will ensure that each individual piece of code functions as it is intended. With each piece of code working independently, we will conduct implementation testing to verify that all components work together and communicate properly. Last, our usability tests will provide our team with feedback from real end-users. Specifically, we are looking for suggestions regarding the user interface and how intuitive it is to navigate our application. This will include feedback from all three types of users, public, researcher, and admin. Most importantly, the usability of the interactive graphs will be tested.

Using results from our tests, we will update our application by fixing any found bugs and errors, and improving user-interface components. As a team, we are confident that our testing plan will uncover any room for improvement within our application. We are eager to enhance the functionality of the site in order to deliver the most useful and powerful data visualization tool to Dr. Llama.