Updating the Scientific Usefulness of the Splatalogue Database

Daniel J. Lopez-Sanders\textsuperscript{1,2} and Anthony Remijan\textsuperscript{1}

\textsuperscript{1} National Radio Astronomy Observatory
520 Edgemont Road
Charlottesville VA 22903, USA

\textsuperscript{2} Benedictine College
1020 North 2nd Street
Atchison, KS 66002, USA

ABSTRACT

Splatalogue is an online database for astronomical spectroscopy created and maintained by the National Radio Astronomy Observatory (NRAO) that contains over 11 million spectral lines of over 1300 different species; these lines are from over 15 different linelists. The Splatalogue user interface (splatalogue.online) was built on an old interface and while it has worked very well for the community over the course of the past 15 years, it does lack basic functionality that will allow for more scientific analysis done with the data contained in the database. This project involved working to complete four goals to facilitate both future development of this website and the work of astronomers who use it; the first, second, and fourth goals were finished within the time of this summer Research Experience for Undergraduates (REU) project.

1. **Security:** Ensuring the site was safe from attacks so it can be consistently available as a secure resource for the astronomical community.

2. **Documentation:** Documenting the project code to facilitate future development of the site.

3. **Modern Technology Base:** Switching the base of the web server technology from PHP to Python for security and feature integration reasons.

4. **User Interface (UI) Improvements:** Facilitating the use of the website through a user interface style improvements.

1. INTRODUCTION

Spectral line data is an invaluable asset in the field of molecular spectroscopy. As with any other study, the data must be able to be interpreted and used with relative ease. It is for this reason that the Splatalogue database and user interface were built. The Splatalogue website contains information detailing the motivation behind the creation of these resources for the astronomical community:

The NRAO Spectral Line Catalog (Splatalogue) is an attempt to collate, rationalize and extend existing spectroscopic resources for use by the astronomical community. The Jet Propulsion Laboratory (JPL) database, the Cologne Database of Molecular Spectroscopy (CDMS) and the Lovas/National Institute of Standards and Technology (NIST) database provide an enormous amount of data collecting the data in all three online databases together generates over 5.8 million transition data entries across 1038 individual entries. As part of the ALMA and EVLA scientific plans, one of the main goals of telescope operations is that each new instrument be easy to use by a novice observer. Specifically, in the ALMA Project Plan v2.0, it is stated clearly that:

"The final major scientific requirement affects the diverse community that will use and benefit from the scientific capabilities that ALMA brings to extend their research endeavors: ALMA should be easy to use by novices and experts alike. Astronomers certainly do not need to be experts in aperture synthesis to use ALMA. Automated image processing will be developed and applied to
most ALMA data, with only the more intricate experiments requiring expert intervention."

To that end, we interpret the above statement to also apply to an observer interested in spectral line astrophysics. A spectral line database needs to be available that is descriptive as possible in the way it represents molecular, atomic and recombination line transitions.

Currently, the commonly used databases that are used for this search do not describe transitions in a user-friendly way, and where the catalogs overlap, the descriptions have to be compared and resolved to be consistent. Furthermore, the Lovas/NIST database tabulates only observed interstellar transitions, but it does provide the user with a much better representation of molecular transitions by using a full description of the molecular transition. As such, The North American ALMA Science Center and the Data Services Working Group is dedicated to generating a collated and rationalized database of spectral line frequencies, transitions and line strengths from radio to infrared wavelengths that can be used by the entire astronomical community interested in spectral line astrophysics.

To meet this goal, the Splatalogue online interface was developed, with two ways to access the data: (1) an Application Programming Interface (API) called SLAP, which allows a program to get data from Splatalogue, and (2) a PHP-based website interface. The web interface was put into production in 2005, and was the focus of this project. This interface contains two pages for searching: a Basic version and an Advanced version. Both were created to allow searches by different criteria, with the Advanced version allowing the user to search by many more criteria and customize the displayed data more than on the Basic page. Over the years, this website has been updated with more functionality, and new data have been added to the database. While this website has served the astronomical community well over the past 17 years, there are certain aspects of the website that possibly contained security risks, hindered future website development and were not providing information contained in the database in ways that best met the needs of the astronomical community. It is to solve these problems that this project was undertaken. The website was developed with the Django web framework with Apache as the web server. Django is an open-source, Python-based web framework that facilitates the web development process through functions and packages that handle more low-level tasks for Hypertext Transfer Protocol (HTTP) Request and Response handling. Also, the new version of the site was deployed on a RedHat Enterprise Linux 8 (RHEL 8) server. The old version of the website would receive the HTTP request, and PHP would build the page for the user based on the user’s input parameters.

Specifically, how the new version of the website was designed to work is as follows:

1. The user (client) makes an HTTP request to the web server (Apache).
2. The web server receives the HTTP request and parses it so it can figure out what the client wants.
3. The web server passes the request to Django through the use of a Web Server Gateway Interface (WSGI) Application, which allows it and the Python-based Django web framework to interact.
4. Django receives the request from the WSGI application and returns the appropriate response. It is in here that the database server is queried and the results used in building the response.
5. WSGI passes the response to Apache, and Apache returns it to the user.

2. SECURITY

2.1. Issue

Security is an essential consideration when building any computing service with which public users can interact, and Splatalogue is no exception. Furthermore, for such a widely-used spectral line catalogue, is is of the utmost importance that the data on the site remain constantly available and its integrity remain intact so users of this astronomical tool can count on the site’s data in their research. One specific way sites can be exploited is through Structured Query Language (SQL) injection. Now, SQL is the language used to query the actual Splatalogue database to get information for a user’s search. And, since a user can enter in any certain parameters to construct a search, such as variable frequency ranges, for example, the SQL query to the database is constructed dynamically from user input. The security risk for SQL injection comes from directly inserting user inputs into the query without escaping them (removing dangerous characters) or validating user inputs (making sure that the user data is valid before inserting it into the query). For example, let a query for a login to some website be structured as follows:

```
"SELECT * FROM users
WHERE username = "' + username + "' AND password = "' + password + "'"
```
where 'username' and 'password' are user data. If these data are put into the query and parsed as code, this would allow the user to enter in any database commands desired. For example, if the user set the username to be "bob"; DROP TABLE users; --" and set the password to be any text (let it be "forest1234" as an example), the query, built and with user inputs processed as code, would be as follows:

```sql
SELECT * FROM users
WHERE username = 'bob';
DROP TABLE users;
--" AND
password = 'forest1234';
```

Anything after the second single quote and the semi-colon, if processed as raw SQL, would be run by the database, whether it be dropping the "users" table as in this example or anything else, including deleting from, modifying existing data, or adding any data to the database. The two dashes mean that everything after them is treated as a comment. Now, such an issue must not be allowed to occur with Splatalogue so the user community can count on data being available and its integrity ensure. The previous version of the Splatalogue website interface was built on a server-side scripting language called PHP. While PHP is good for getting data from the server and building an HTML page to be returned to the user, it is easy for security issues like SQL injection to arise with the use of PHP, even with the security improvements that have been made in PHP in updates over the years. Also, in the previous version of Splatalogue, the user inputs were directly editable in the URL string for the Basic page, further making easy the ability of anyone, including a hacker, to modify the data put into the query string.

An additional security risk that can be avoided was a Cross-Site Request Forgery (CSRF) Attack. In this, another web server can submit data to Splatalogue and Splatalogue assumes that the inputted data is okay and processes the request. For example, if a hacker builds a website where you enter in your banking credentials such as username, password, account number, and routing number, if the banking website is not secure, the hacker’s website can submit a request to the banking website to transfer money from your account to the hacker’s account. This issue could pose a risk for Splatalogue by allowing other websites to interact with the web server to submit a form and put Splatalogue data from the Splatalogue web server on their website in an unauthorized manner.

2.2. Solution

The decision was made to move away from PHP as the web-server base to Python. For security reasons, this was made to avoid the possible security issues of PHP. Specifically, SQL injection is avoided through the use of custom validation functions, through the use of which user inputs are only inserted into the query if they are valid data. While this requires more work on the developer’s part to ensure the names of the inputs called on the server side match up with those given on the client side, such names likely will not be changed that often. Now, users can employ the site to meet their needs because they can trust that the website will be available and its data is valid. Also, to avoid putting the user input parameters in the URL string, the searches of the spectral line data were made to occur via a JavaScript request to the server that receives the spectral line data and puts it in an HTML table for the user to see.

To prevent a CSRF attack, a value called a CSRF token was put on every page with a form. This token is unique, and is a feature of Django that allows forms to be submitted if and only if they contain a CSRF token that the Django web framework generated, which means that forms can only be submitted if they came from the Splatalogue web server.

Additionally, as with the previous version of the site, Hypertext Transfer Protocol Secure (HTTPS), a secure version of HTTP, was the only one allowed to be used across the site for security purposes. HTTP requests to the site were set up to be forwarded to HTTPS.

3. DOCUMENTATION

3.1. Issue

Collaboration is often a part of scientific research, and the development of the Splatalogue website is no exception. While the new version of this website was developed in the summer of 2022 by a summer student, the project will also be maintained going forward by others; specifically, members of an NRAO development team. When completing a redo of this website with Django/Python as the web server base, one of the issues encountered was that the code for the previous, PHP-based version of the website, which was reviewed while constructing the new version of the website, was not well-documented. There were some comments throughout the code, but not everything was well-described so anyone unfamiliar with the code would clearly know what was going on. Documentation was identified as a project goal to facilitate future development. Without it, bugs and possibly security risks could arise from a lack of understanding of how the code works, and with it, developers can modify or add to the existing code with relative ease, thus allowing them to best serve the astronomical community. An example function from a JavaScript file is below to demonstrate the need for documentation in the redone version of the site.

```javascript
function reset_chemical_name_top20() {
    document.qpsearch.chemical_name.value = "";
    for (var i = 0; i < 8; i++) {
        document.qpsearch.top20[i].checked = false;
    }
}
```

From viewing the function and the page on which it is designed to operate (the Advanced page), the task the
Motivation

Source: Project Abstract

The spaltproject user interface (spaltprojectonline) was built on an old interface and while it worked very well for the community over the course of the past 15 years, it does lack basic functionality that will allow for more scientific analysis done with the data contained in the database. The current program will not only work on updating the old php code to a more modern interface but also to improve the scientific usefulness of the database by adding additional functionality.

Code Style

General

- Comments: Comments should not be too large, and the comment rule should be descriptive. It is a lot harder to track down errors and bugs if code is in large chunks.
- Python: All Python decisions should be done according to the specifications made here or in the Python section below.
- Comments: Please leave descriptive comments next to your code! This can help you remember what you did, and help others understand your code if they are reading it. If something is very easy to understand, or calling the “obvious” function within a Django app (like a function to display an HTML page, you don’t have to comment what that function does. In this case, you should however, leave a comment describing what page this function displays. Any line after the first one of comment should be indented by the default amount of the language it’s written in plus a space (e.g. if the default indentation is 4 spaces for that language, use that; after the comment begins syntax e.g. if Python comment).
- Names: These should be descriptive but only long enough to be as descriptive as is necessary to clearly identify the function. For example, page is not a good name for a function that renders the HTML page for the basic version of Spathologica because it is not clear, and the naming that displays the basic search results page is descriptive but is longer than is necessary to convey the information if function accomplishes can be determined, but only after some time of inspecting the site and the code.

3.2. Solution

In order to make future development of the site easier, a documentation scheme was derived, and descriptive comments were placed throughout the code. First, a "README" Markdown file, which contains the project description and is a common feature of computer science development projects, was written for developers to read. It outlines the basic project structure, the motivation behind it, how to set up the project on the production server, and rules for code structure, among other helpful content. An image of a portion the rendered version of this file is included in figure (1).

Additionally, throughout the code, descriptive comments were given to code features so future developers could understand the code. Specifically, named custom functions were given a documentation style that is as follows. As designed, it contains three different sections: Function Inputs, Function Outputs, and Function Notes. The Function Inputs and Outputs sections both contain the name, data type, and description of their values; for the Function Inputs, this is done for each of the arguments in the function call (it must be specified if the argument is optional), and for Function Outputs, this is the set of return values. The Function Notes section contains where the function is used as well as anything else the developer might need to know in general about the code. A sample Python function from the new version of the website can be seen in figure (2).

Figure 1. A rendered version of the README project documentation file, containing "Project Motivation" and "Code Styling" sections.

Figure 2. Documented Python code from the new version of the website. Converts the specified frequency or wavelength value from MHz, which is what it is saved as in the database, to the user-specified unit. Contains the named custom function documentation at the top of the function and descriptive comments throughout the function code.

Figure 3. A CDMS website graphic; a PNG plot of relative intensity vs. frequency for a Carbon Monoxide (CO) search.

4. MODERN TECHNOLOGY BASE

4.1. Issue

As previously discussed, the switch was made from PHP to Python as the base of the web server for security reasons. Another reason the switch was made from PHP to something else is because the data, as it was rendered to the user, was just text. While this, including the ability to download results, was very helpful to the user community, the data could be easily displayed to the user in helpful ways, such as plotting data of frequency vs. intensity. PHP, while good at getting data from the database and putting it into a page returned to the user, did not offer much at all in the way of displaying the data as Python did. Other existing databases, such as the Cologne Database for Molecular Spectroscopy (CDMS), offer such functionality. An example graphic from CDMS can be seen in figure (3). An example search for Carbon Monoxide on the Basic page of the old version of the website can be seen in figure (4). This demonstrates the previous extent to which data were rendered for the user.
4.2. Solution

The switch from PHP to Python allowed the new version of Splatalogue to best meet the needs of the astronomical community. Packages such as astropy and molsim, for generic astronomical work and molecular spectra simulations, respectively, exist in Python but not in PHP. With a well-documented Python-based web server for the new version of the site, these packages could be implemented more easily. As Python is one of the most popular programming languages in the world, it has many packages and functions that have been developed that facilitate programming in it. Also, Python has the added benefit of being easy to read like PHP.

While an external Python package, mysqlclient, had to be installed in order to allow the Python web server to query and get data from the database server, this dependence on an external package is far outweighed by the versatility and the power of Python, as anything that can be done in Python–code from molsim, astropy, numpy, and more–can be done with the Splatalogue data in order to render it in a way most useful to users of the website. The CDMS graphic (figure (3)) demonstrates some of the functionality that Splatalogue can offer with Python as the base of the web server. Additional functionality was not able to be implemented during the course of this research project, but has been planned to be done in the future. This work will allow Splatalogue to more flexibly and easily meet the needs of the astronomical community, and Python is a significant part of what makes this possible.

5. USER INTERFACE (UI) IMPROVEMENTS

5.1. Issue

While the Splatalogue web interface has served the user community well over the past 15 years, the "look and feel" of the site does not facilitate the use of the site as well as is desired so users can most easily take full advantage of this tool for their research. For instance, some of the redshifted frequency text (seen in figure (4)), which was red, could not be easily seen when a different color was behind it. Also, the search button on the Basic and Advanced pages was in location on the page and had to be scrolled to in order to submit the search button. The same goes for the table row containing the column headers for the search results table, which stayed right above the first row in the table and had to be scrolled to in order to know what a particular value in a row was (e.g., a certain frequency or intensity value). Also, there was information about the database, such as the linelists used and the format of the quantum numbers, that was in different locations on and not accessible from all pages of the site.

5.2. Solution

The new version of the website was designed with the Basic and Advanced pages having the same general layout and user inputs from which to choose as those pages on the old version of Splatalogue. For the new version, as with the old version, custom styling through Custom Style Sheets (CSS) was used. One additional tool that was used to make styling easier was the CSS/JavaScript library Bootstrap, which contained several styled classes that made it much easier to design a sleek UI. The following sets of images give a comparison between the old and the new versions of the site to demonstrate the improved UI. Figure (5) demonstrates the significant changes in the homepage of the site. While the links to the Basic and Advanced pages remain from the previous version, the Basic page being the homepage of the site was no longer the case, with an entirely new page with a sleek design containing links to the different pages on it. Not visible in the image of the new Splatalogue’s homepage was a welcome message for the user.

Next, figure (6) demonstrates the stylistic changes between the old and new Splatalogue Basic page, both searching for Carbon Monoxide. Note the minimalist layout of the Basic page’s search interface and the fact that the search button and the column headers for the table were at the top of the page even though the middle portion of the table was being viewed. The new Splatalogue’s user input section was in the same place on the page (not shown in the figure) but was created with a more sleek, minimalist design.

After this, figure (7) demonstrates part of the search interface and search results for the H-alpha Recombination Line. The search interface on the Advanced page on the new version of the site can be seen to have a minimal-
A comparison between the homepage for the old and new versions of Splatalogue, with the old version’s homepage on top and the new version’s homepage on bottom.

As with the Basic page, the Advanced page has the search button and the row of table column headers stay at the top of the page when users scroll down the page. Also, on the new version of Splatalogue, the grey button that says "Select Species" is the header for that input section and can be clicked to toggle that section’s display between open and closed, using the Bootstrap library to do this in a smooth manner. This functionality is the same for all other input sections and their headers on this page.

Finally, figure (8) demonstrates the difference between the old and new Splatalogue versions’ site information styling and organization. The old Splatalogue had information about the site both on pages linked at the top of the Advanced page and in sections at the bottom of the Advanced and Basic pages, whereas the new version of Splatalogue was designed with most of that information (with one internal page linked) in a series of Frequently Asked Questions (FAQs) one page. Each question was styled so it must be clicked to view the answer, and opening one question’s answer closes all of the others.

Overall, the minimalist interface of the new version offers advantages over the previous version by more strongly encouraging users to return through the ease of use and the minimalist design while still providing the same useful layout and site information offered for over 15 years by the previous version of Splatalogue.

6. CONCLUSION

The new version of Splatalogue was undertaken to provide improvements in the following four areas: Security, Documentation, Modern Technology Base, and User Interface (UI) Improvements. While the feature addition described in the Modern Technology Base section, specifically plotting user search results on the user’s inputted frequency ranges, was not implemented as planned during this project, the future considerations for this project include the completion of this task. The website accomplished the other three tasks in the following ways:

2. **Documentation:** Documented the project code, both with in-code and overall documentation (in a README documentation file), to facilitate future development.

3. **Modern Technology Base:** Switched the base of the web server technology from PHP to Python for feature integration as well as security reasons, with plans to plot user data using the molsim package in the future.

4. **User Interface (UI) Improvements:** Improved the organization of site information and overall page styling while maintaining the general layout of the Basic and Advanced pages. Used custom Custom Cascading Style Sheets (CSS)/JavaScript as well as a CSS/JavaScript library called Bootstrap to do this.

Overall, this website has been and will continue to be a valuable resource for the astronomical community, and by accomplishing the aforementioned goals, this project has helped support that mission.

7. **SOFTWARE AND THIRD PARTY DATA REPOSITORY CITATIONS**

We thank all the people that made and have maintained Splatalogue (the database, the web interface, and the API). This includes but is not limited to Tony Remijan.
APPENDIX

Software: astropy (Astropy Collaboration et al. 2013, 2018), Bootstrap (Bootstrap Authors 2011), Django (Django Software Foundation 2011), mysqlclient (Naoki 2012)

REFERENCES

Bootstrap Authors, T. 2011
Django Software Foundation, 2011
Naoki, I. 2012