American Astronautical Society

Field Session Team

Maddie Geesen
Anthony Arellano-Gandy
Milo Dietrick
Jason Dishongh

1. Introduction

The AAS Rocky Mountain Section seeks to put on a successful conference year after year. The success of the conference is closely related to the attendee's overall experience and the amount of attendee engagement since conferences are all about collaboration. We want to provide attendees with all the tools necessary to have a good experience and contribute to the advances of the community. To do this, the planning committee also needed the tools to put on a good conference by easily soliciting papers, engaging with their authors, and running their session. All of this can be accomplished with a good website.

The old website interface worked and met the requirements set forth by the conference board members, but many features needed improvement. The document submission and review interface were restrictive and lacking in features as well as usability. The goal of the AAS team was to improve these interfaces for a more fulfilling user experience. Specific goals included improving the flexibility/functionality of the document submission interface, improve the conference schedule, account home page, and website appearance. The required experience for this project included work with PHP, JavaScript, HTML, and CSS.

2. Requirements

The requirements for the website were broken into two distinct sections: functional and nonfunctional requirements. The functional requirements refer to the improvements that had to be made to the website itself and were the main focus of the AAS team to meet the "definition of done". To accomplish the functional requirements several steps were required to ensure proper processes were followed. These steps were considered nonfunctional requirements and pertained to development and proper coding practices. Both sections are described in detail below.

2.1 Functional Requirements

- Web-admin functionalities.
    - Set up a test environment for making modifications to the website offline.
    - Provided documentation so anyone can set up the test environment.
    - Database schema updated.
        - Connected abstracts to papers.
        - Paper number field added in submission table.
        - Connected paper to schedule.
        - Connected Author/Bio to schedule.
        - Presentation.
- Session chair interface.
    - Now able to get zip format of all files per session for session chairs to download.

    - Allowed session chairs to move abstracts/papers to other sessions.
    - Provided interface for accepting papers.
    - Session chair “run session interface” with everything needed: presentation, bio, timing, and schedule.
- Attendee interface.
    - Allowed authors to confirm their attendance at the conference.
    - Allowed attendees to resubmit/Edit submissions.
    - When abstract is accepted assigns paper number.
        - Auto populates paper number on paper submission form when authors go to submit paper.
    - Secondary author able to edit and submit paper to corresponding abstract - One author uploads an abstract but then the other author wants to upload the paper.
        - Shows all of the user’s papers when the user log in.
        - Implemented field to re-assign/share paper to other author.
    - Bio upload (needed a Bio for every author, forces them to upload one with their paper).
    - Able to reuse bio from previous year if present.
    - Shows preview or abstract on schedule.
        - All Papers available online.

- Nice to have.
    - Automated program generation.
    - Get metrics for page visits.
    - Session chair interface.
    - Social Media presence.

2.2 Non Functional Requirements

- Created a testing environment so that team is not working on the live website, which could result in breaking the entire site.
- Implemented version control using Git so that team can all be simultaneously working and so that merge conflicts were easily resolved.
- Project done using PHP, JavaScript, JSON, and SQL
- Website needed to work on all operating systems, including mobile devices
- Utilized the already existing database, which contains information and documents from the 2019 conference. Needed to overlap information between these tables and the new 2020 tables.
- There was little to no documentation from last year’s field session group. As our team worked through the project, we needed to add documentation so that in the future, the structure of the website is more easily understandable.

2.3 Use Cases (correspond to each Functional Requirement above)

- Web admin functionalities - Administrators should be able to make updates to the website without working on the live website.
- Session chair interface - Session chairs needed to be able to download one, multiple or all of the submissions to their session, sort them by type, or date submitted. They needed to be able to modify the submissions, just as an attendee would, but with even more

privileges, interface with the authors in their session and use the website to run their session, come time for the conference.
- Attendee interface - Attendees needed an easy way to upload their submissions, modify submissions that have already been uploaded (except for fields the committee does not want attendees to be able to modify) and share their submissions with co-authors. Web admin needed to be able to enforce rules on these submission such as naming conventions and required fields as well. The ability to upload submissions currently exists (but the code is sloppy with hacked together, late-add functionality).

3. System Architecture

The website is built in a web framework called WordPress which is structured from a combination of JavaScript, PHP, HTML, and MySQL. The WordPress framework allows one to create a custom website using provided templates with a secure database on the backend. WordPress features an online graphical user interface that allows the user to create custom templates and edit the styling of the web pages. Once built, the website is then hosted by BlueHost which is a paid hosting service provided by the Rocky Mountain AAS conference.

To develop the required features without affecting the live website, our team had to create a localized test environment. Each team member needed to have their environment so that each component of the website could be worked on without creating conflicts during the development process. For this local testing environment, our team decided to use XAMPP. XAMPP is a wrapper for PHP, MySQL, and Filezilla. Filezilla is a file transfer protocol system that allowed our team to seamlessly transfer files from our local testing environments to the live website on BlueHost. WordPress is designed around using the online GUI to create webpages, however because of this editing features and styling in code is somewhat challenging. However, using XAMPP with a variety of debuggers our team was able to find and address the major components that were requested by our client.

To address our client's requirements, our team needed to thoroughly understand the workflow on the website. The flowchart in Figure 1 on page 5 represents the process that a typical conference attendee goes through, creating an account and submitting and editing documents for the conference they are attending. Many of the current issues with the website revolve around this workflow and restrictions our client wanted on this flow, so much of our time was focused on perfecting the details of this process.

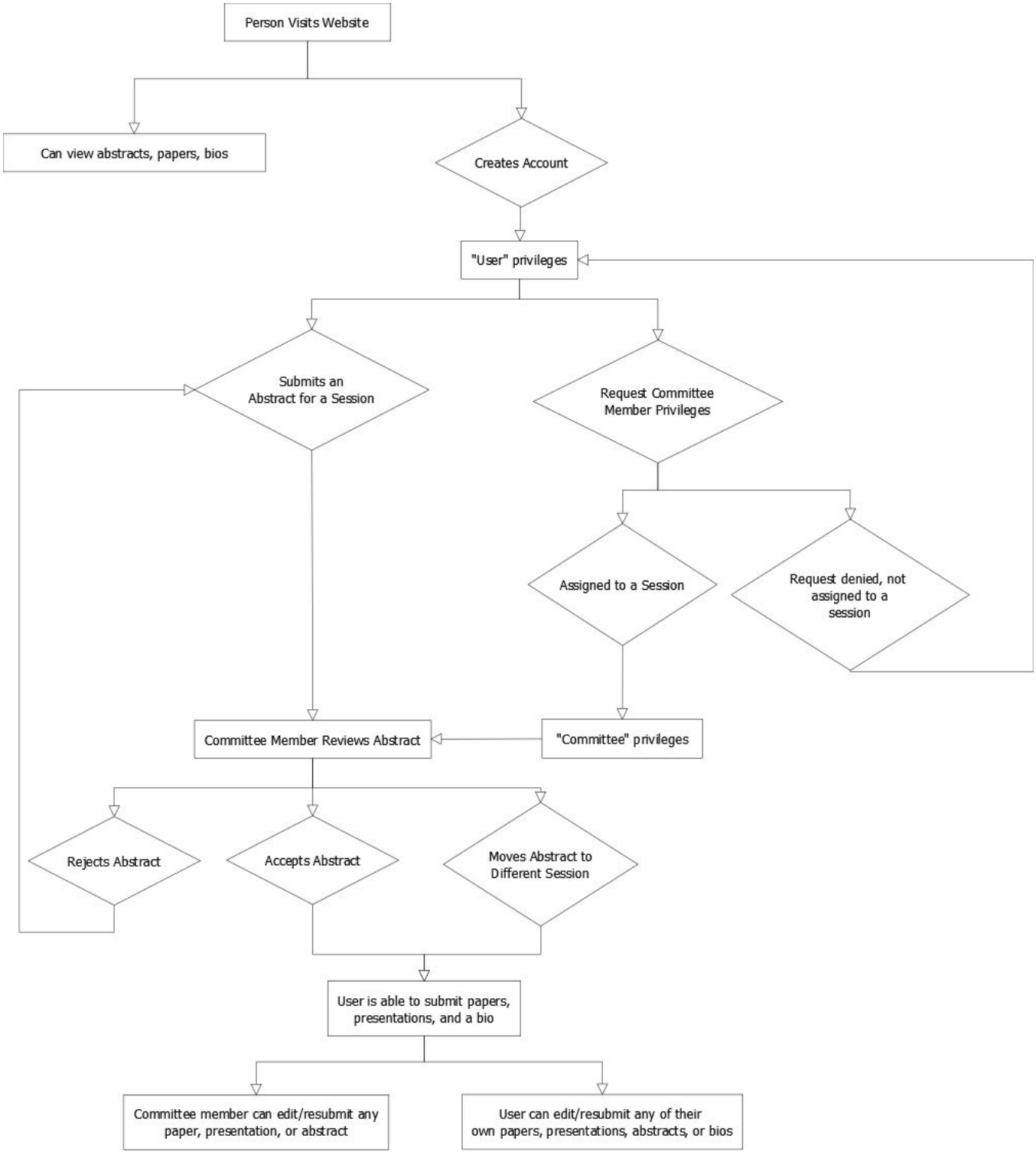


Figure 1: Typical workflow of website user

4. Technical Design

The first step in developing our product was setting up a test environment, debugger, file transport protocol, and version control for the website. Instructions for setting up each of these are included in Appendices 7.1-7.3. Next, we restructured the database to suit our clients requests and to be easier to expand upon in the future. From there, we improved queries across all pages and were able to develop new pages. We changed the Document Submission page, as this page was previously filled with bugs. Our next step was creating a Document Resubmission page, where both User members and Committee members can edit submissions. We updated and added functionality to the Account page; anyone who is logged in can view all of their submissions here. From there, we developed a Review Publications page and an Events page. On the Review Publications page, committee members are able to download submitted papers and abstracts, view information about the submissions, and accept abstracts. The Events page will be the go-to page during the conference itself. It displays a schedule of the current conference, with clickable links for each session, which then display the abstracts for that session along with their presentation times and locations.

4.1 Database Schema

One of the first components to address for the project was to update the database schema for the 2020 conference. The previous database schema was missing many relational components, and tables, and table columns that would create a better user experience. There were also a few features of the original database that were confusing and poorly constructed, so our team sought to improve these features. The first step was reconstructing most of the database to make it more robust and to improve dependencies. An Entity Relationship Diagram of the database schema is shown in Figure 2 on page 6. To start, each year has its own conference in the Conference table. Conferences are made up of several events, which can be of a type Session, Poster, Social or Other. Events that are of the type Session are composed of several sessions, which are held in the Session table.

When a visitor to the website creates an account, entries in both the Account and Member tables are populated. Their privileges default to User, and they have the option to request Committee privileges. Members with accounts can submit abstracts to sessions. Committee members can accept abstracts, edit the submissions for abstracts, and move the abstracts to different sessions. Once a member has an abstract accepted to a session, they can submit other related publications, such as papers, posters, or presentations. These additional submissions populate the Publication table and must be tied to an accepted abstract. Members with User privileges can edit their submissions, and committee members can edit anyone’s submissions. Committee members can be assigned to sessions, which are reflected in the Session Chair table.

These database updates allowed for all tables to be related to one another via foreign keys, so no one table is isolated, as tables were in the former database. It also allowed for us to not have unnecessarily repeated columns across tables. The only repeated columns are foreign keys to other tables. This allowed us to write much more efficient queries. For example, in the old database, all documents, including abstracts, were stored in one table. So, there was no way of knowing what publication corresponded to which abstract. Additionally, there were many untracked errors in the data, for example there was no correlation between abstracts, their publications, and information such as the author, presenter, and session. If this information did match, it was repeated across many rows of the table. Splitting this one table into two, Abstract and Publication is more efficient and allows for the correct information to be extracted from the database. Each abstract has an abstract I.D. and stores information including the presenter, submitter, if the abstract has been accepted, the session it has been accepted to, and the URI for the abstract document itself. All other publications are stored in the Publication table. This table is comprised of only the abstract I.D., which is a foreign key, the publication type, and the URI for the submission. Our client requested that all the information about the submission be tied to the abstract I.D., which is now possible.

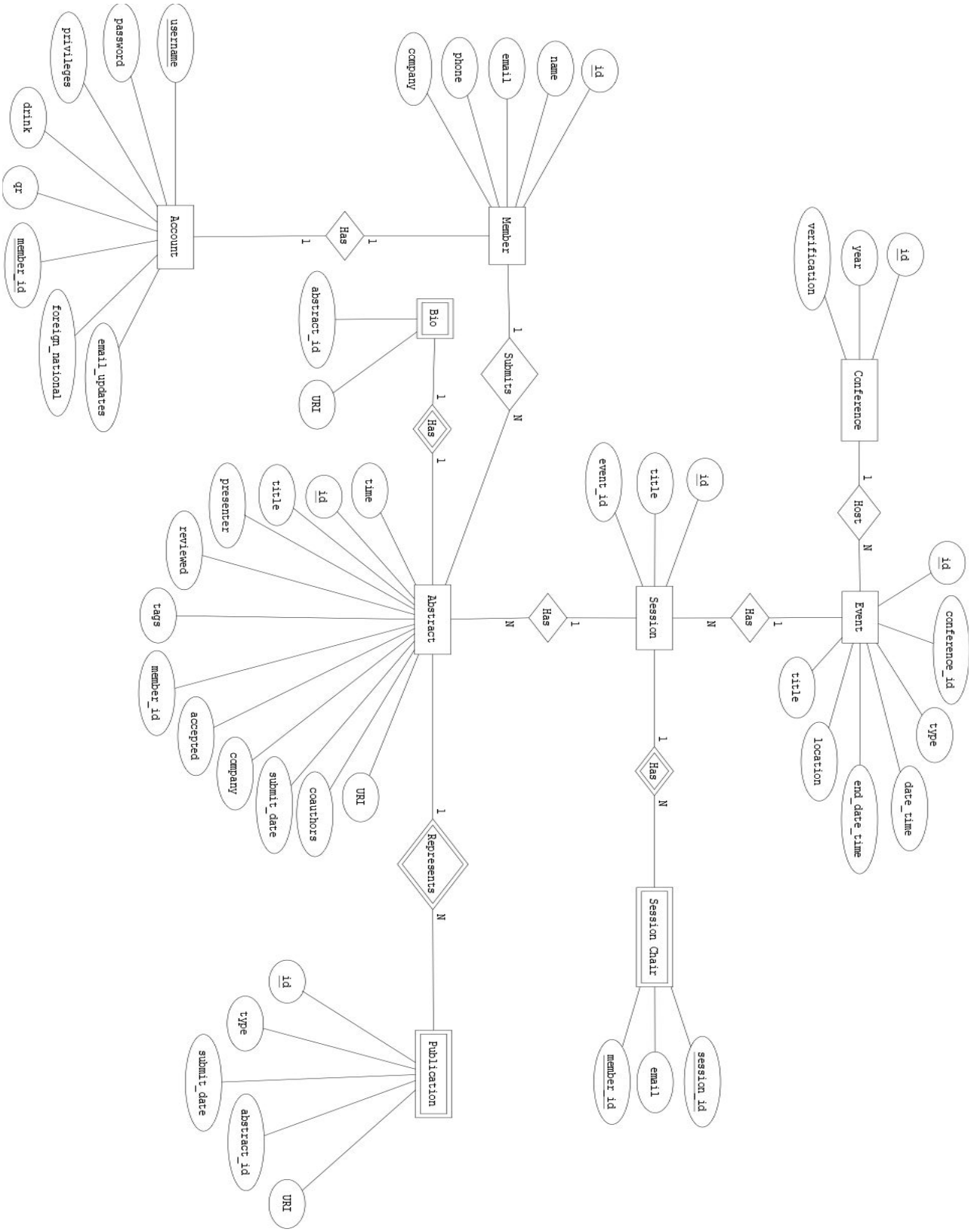


Figure 2: Entity Relation Diagram for 2020 Database

4.2 Document Management Interfaces

Another large aspect of our project was improving and updating the document management and approval interfaces. The new user interface allows users to upload, edit, and resubmit documents which include abstracts, associated research papers, and presentations for the annual conference. This feature was present in the existing website but it was dysfunctional and restrictive. Mainly, there was no way to edit a document after it was submitted or to tell if a submitted abstract had been accepted. There was also no interface for committee members to review new abstracts, accept them, edit them, or move them to different sessions. To address these issues we had to modify or create three pages: the document submission page, resubmit page, and the document review page, otherwise known as review abstracts page. The features of each page are described in detail below.

4.3 Document submission page

All members, both normal users, and committee members have access to the Document Submission page. There are user input fields on the page including text fields, dropdown menus, view-only fields, and a file upload button. Before our team made changes to this page, every text field was simply editable and blank. The user also didn't know what fields were required until they tried to submit their document. Our main goal with this page was to make it more user-friendly. At each text field that is required for the user to move on, we added stars in a red font at the end of the fields' title. We added dropdown menus for the user to select the type of document, such as abstract or paper, and for the session that they would like to submit their abstract to. The submitter and company fields are auto-filled and not editable, as this information is pulled directly from the database and cannot be changed. The presenter field is auto-filled to be the same as the submitter but is editable because sometimes people submit other people's documents for them. For example, sometimes a professor will submit their student's work for them.

4.4 Resubmit page

The Resubmit Publication page has the same format as the Document Submission page. We created this page from scratch and the page is only accessible from two other pages. It is accessible from the Account page, where all users can edit and resubmit their documents. It can also be reached from the Review Publications page, which is only accessible by committee members. From here, committee members can edit and resubmit anyone else's documents. This page has all of the same fields as the Document Submission page, with the addition of a "Page Number" field, which all members can view, but only committee members can edit. Every field on this page is auto-populated with search queries made from the database. The "Type" field is no longer a dropdown menu but is a view-only field. The "Session" field is a dropdown menu for committee members but is otherwise not editable. This allows committee members to move abstracts to different sessions. The rest of the fields are the same as on the Document Submission page. Users have the option to upload a new file, although this is not mandatory.

Once the submit button is clicked, the values all of the fields are recorded and the database is updated accordingly.

4.5 Document Review Page (Review Abstracts)

Only committee members have access to the Document Review Page. This page allows committee members to download and review submissions, view and download the presenter and their biography, and accept abstracts. Users first choose a conference and then select a session from a dropdown menu populated for the chosen conference. Once a session has been chosen, a table is generated with all types of documents for the chosen session. The table provides information about each document, including its title, session, presenter, and type. The title and presenter columns are hyperlinks that can be clicked to download the document itself and presenter's biography respectively. The edit column is also a hyperlink that takes the user to the Resubmit page, where they can change information about the document. One of the more complex parts of the table is the "download" column, which allows the user the mark checkboxes for titles they wish to download. Once they have selected all the files they want to download, the user then clicks the "download" button and it will create and download a zip file to the user's computer. Finally, the "accept" column allows committee members to mark abstracts as accepted and updates the database accordingly.

5. Software Quality Plan

5.1 Pair Programming

By utilizing pair programming, we were able to be more time-efficient while writing higher quality code. We found that when we worked in pairs, we ran into much fewer bugs. And, when we ran into bugs, it was much easier to find and to fix them. We also created a more well-written and robust code, since we are combined our knowledge and unique strong suits.

5.2 Code Reviews

One of the main ways that we ensured we delivered our client a quality product was through code reviews. The practice of pair programming lent itself very naturally to frequent code reviews. We found these to be extremely helpful because we were able to address issues such as merge conflicts or inconsistencies across website pages proactively. We gave each other advice on how to improve each others code or to get past blockers before too much time was wasted on any one task.

5.3 User Testing

We set up local test environments through XAMPP where each one of us worked separately and tested exactly how our code affected the website. This proved to be very valuable to the quality of our product because we were able to constantly test the website and

its functionality. It was also extremely important in all of our weekly Sprint meetings with our client. We were able to show her first hand what we did on the website and she was able to work with the updated version to test changes and to show us exactly what she wanted to be done in the next Sprint. Our weekly direct user testing with our client ensured that at the end of the field session, she was not surprised by our product, since she saw it, worked with it, and gave feedback every step of the way.

5.4 Daily SCRUM Meetings

Every morning, our group met for a SCRUM meeting, where we first discussed our progress, blockers, and plans with each other and then, most days, called our client and summarized our teams' progress, blockers, and plans as a group. These meetings served a very similar purpose to the code reviews and Sprint meetings, as they helped us not waste time when we are stuck, keep our big-picture goals in mind, and make sure that the website is functioning smoothly and as intended.

5.5 Documentation

In receiving the software that last year's field session group had developed, we found that there was little to no documentation on both the large scale interactions between different pages or between the HTML and PHP code and the database, and on the small scale within the code itself. Because of that, we spent a significant amount of time just trying to understand how the software worked. So, throughout the field session, we worked on improving the documentation of the software, with diagrams on the large scale and detailed comments on the small scale. We believe that good documentation is a critical part of a high-quality software project, which is why this was such a focus for us. This will make it much easier for future developers on this website to update and improve upon it.

6. Results

6.1 Incomplete features:

Although we were able to complete a majority of the required tasks for the website one of the major issues faced by our team was the overall styling of the website. This styling is governed by the CSS components which were found to be very difficult to edit. The website was originally built using a web design service called WordPress which allows a user to style their website using preset themes in an online editor. This interface is fairly limited without paying for more themes and plugins. The CSS components of the website were managed globally which made it very difficult to change any one part of the website without affecting the entire site.

Also on the list of things that did not get completed in time were some of the document management interfaces. This included a feature where the client wanted a single button to

choose and upload files, a common practice in many web pages. Despite multiple attempts to create a single button to choose and upload a file in the document submission and resubmit publication pages. Due to the communication across multiple files in PHP and JavaScript, any attempt to implement his functionality resulted in the webpage crashing

6.2 Performance testing results:

So far our testing has shown that the live website works in all three browsers that we have tried which includes Firefox, Google Chrome, and Microsoft Edge. After migrating our work to the live website we found that some of the links we created for page navigation, logins, and downloads were not quite working. Testing in the live website environment allowed us to see how the website will work in real-time and we were able to address these issues by modifying links and managing conflict between our version-controlled website and the live one hosted by Bluehost. Several bugs were identified in the process, which included a button on the document submission page which would simply refresh the page instead of submitting the document, the login page also would not work until we cleared the cache, and all of the links in the code had to be updated. After merging with the live website were able to address these bugs and more.

Part of our testing process was to bring the website to the conference chairs to get some feedback on the work we are doing. After demonstrating our work at the meeting we were able to receive some feedback on the features of the website that we implemented and which features they would like to see. This was very useful for our team as it gave us some more detailed insight into what the board is looking for and what their priorities were regarding the website. The features that they wanted are included in the “future work” section for this report, however, it should be noted that our team is working on getting these features up and running as this report is being written.

6.3 Future Work:

Future work includes a lot of the feedback we got from the client board team and includes mainly styling of the webpage and some improved functionality of certain webpages. However, there were some main bullet points that they wanted to include. Most of these features were fairly simple and straightforward to add to the code, however, some of them were more complicated, such as adding sessions and submit dates on the account page which is a fairly simple task. Some other simple tasks included adding a download all row in the document review page and a select all feature for that page as well. Eliminate the reviewed column from the review submissions, and add a loading bar to the submit page that ensures the user does not repeatedly press the submit button.

There were a few more tricky features to implement which involved some database queries, some were straightforward, others were not. This included adding a list of emails to the session chair interface. Adding dates for the submissions and sessions on the account page. The board also wanted to have another feature associated with the paper documents which

would have a manually selected paper number used to reference the papers throughout the website. There was a request to add a feature to export the entire database as a CSV file, however with the complexity of this task, it was determined that this task is not feasible with the remaining time left for this project.

6.4 Lessons learned:

Throughout the process of working this website, our team overcame a fairly significant learning curve as none of us had any experience with web development before this. The first obstacle was simply navigating through the WordPress files and determine which file has which purpose through the file structure. Our client provided us with some starter information on this but most of our understanding of the website came from just messing around in the live website hosted on BlueHost.

The next challenge we faced was finding a way to work on the website without affecting the live site. This required finding a way to run the site in a local test environment where we could work the components of each web page without bringing down the live website. This was by far the most challenging aspect of the project. We settled on working with XAMPP which gave us a wrapper for hosting PHP, JavaScript, and MySQL on our local machines. Setting this up on Windows and Mac was fairly straight forward however one of our group members was working on a Linux distribution and despite multiple attempts, Linux would not run the website due to conflict between the different versions of PHP. This was resolved when that member switched over to a mac for development.

Once the local environment was set up the next challenge was to modify the code for the PHP pages, JavaScript functionality and revamping some of the features in the MySQL database. Our team had some limited experience with JavaScript and HTML but none of us had experience with PHP. We overcame this with extensive online research and learning how to work with the WordPress framework which thankfully was used quite commonly in the web development world. One teammate focused primarily on setting up the new database and creating the appropriate queries for the pages while the rest of the team focused on the front end framework. Through extensive research, our team was able to gain an understanding of the website framework and accomplish many of the goals set forth by our client.

7. Appendices

7.1 How to Set Up Local Test Environment Using XAMPP

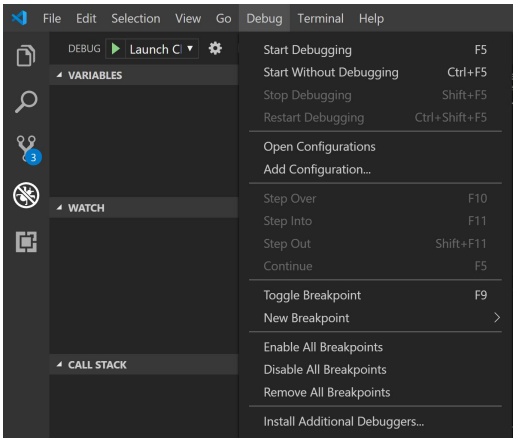
For Windows:

1) Install XAMPP for Windows https://www.apachefriends.org/download.html
2) Download SQL file from database
3) Download the latest public_html zip file from bluehost and extract into (xampp/htdocs)
4) Change the following properties in php.ini (xampp/php/php.ini)
   max_execution_time = 5000
   max_input_time = 5000
   memory_limit = 1000M
   post_max_size = 750M
   upload_max_filesize = 750M
5) Start XAMPP control panel and start both MySQL and Apache
6) Open admin for MySQL
7) Once that is open on the left hand side click new to create a new database
8) Once created click on the name of the database and click on import from the top bar and select the SQL file you downloaded from the database
9) Once the database is imported click on the home icon
10) Click the user account tab on the top bar
11) Find the add new user link on the webpage
12) Fill in the following fields
    User Name: **`aasrocky_ss_d0c1`**
    Host Name: localhost
    Password: hCU6d9X9gxwF
    Then mark the checkbox for global privileges and hit go at the bottom of the page

13) Open wp-config in (xampp/htdocs/public_html) and change the following lines
    define('WP_HOME','https://localhost/public_html');
    define('WP_SITEURL','https://localhost/public_html');
14) Start a new tab in a web browser and type localhost/public_html and the website should load

7.2 XDebug Installation

1) We recommend installing Virtual Studio Code and XAMPP
2) Hover over the Debug bar on the top left corner of the screen then select Install Additional Debuggers



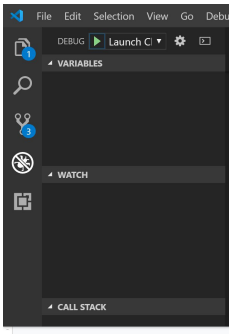
3) Find PHP Debug and then click install

4) Once you have installed PHP Debug, write a simple PHP program to open up php info

```
<?php
phpinfo();
?>
```

Then copy and paste all of the information on the screen and paste it into the white text box on https://xdebug.org/wizard.php and click analyze

5) The website will then provide the rest of the instructions on how to set up XDebug for your version of PHP, follow those instructions
6) Once you have completed those instructions click the bug on the left hand of the page and then select the gear icon.



7) Then verify that you have the following lines of code within the configurations

```
        {
            "name": "Listen for XDebug",
            "type": "php",
            "request": "launch",
            "port": 9000
        },
        {
            "name": "Launch currently open script",
            "type": "php",
            "request": "launch",
            "program": "${file}",
            "cwd": "${fileDirname}",
            "port": 9000
        }
```

8) Restart your local host
9) Open run the same code above to open phpinfo() and if the debugger is installed properly you should see the following display

7.3 How to Connect to FTP (using FileZilla)

1) Download FileZilla (https://filezilla-project.org/download.php?type=client)
2) Set up your FTP Account from BlueHost
    a) Login to cPanel
    b) Click “hosting” link at top of page
    c) Click “cpanel” link at top of page
    d) Under the file section click “FTP Manager”
    e) Under “Add FTP Account” add username and password
    f) Leave the directory field black. It automatically will set it up so that you have full access to all files.
    g) Select unlimited Quota and click “Create FTP Account”
3) Connect in FileZilla
    a) Within FileZilla, go to “File” > “Site Manager”
    b) The information for each field should go as follows:
        i) Protocol: “FTP- File Transfer Protocol”
        ii) Host: “aas-rocky-mountain-section.org”
        iii) Port: “21”
        iv) Logon Type: “Normal”
        v) User: “username@aas-rocky-mountain-section.org”
        vi) Password: the password you set up in BlueHost
    c) Click “Connect”
    d) You should now see your Local information in the left panel and the shared website files in the right panel under “Remote Site”

7.4 Known Bugs

Resubmit Publication Page

There is a bug with the Presenter text field. If you change the presenter of the document and click the Submit button, the new changes will update the database. If you edit that same document in Resubmit again, the new presenter will appear for less than a second but then appears to be overridden to the original presenter that was used in the Document Submission page.

Document Review Page

When a Conference and Session is selected, all of the documents fully appear only some of the time. What fully appears is all columns before the presenter column for the first row only. It seems that sometimes the code has problems getting the bios from the getBios function in function.php. The function gets called from custom.js ajaxReviewPublications. We tried updating the query and different type of ajax calls, but none fixed the issue.