Final Report

6/10/2020

CSM Williams (Dr. Tom Williams)

Benjamin Jessing, Jensen Eicher, and Daniel Garcia

Conference Cycle Visualization (CCV)

Introduction

Our client, Dr. Tom Williams, expressed his vision for our product as a web-based visualization for conference details. Williams is an associate professor at the Colorado School of Mines, who desires to upgrade his website with a conference visualization platform. Conferences in his field of computer science repeat yearly and present potential attendees with various scheduling dates to consider. With the large quantity of conferences that a researcher must track and manage, organization and timing can be difficult. The desired program, being developed in HTML and Javascript, will provide an easy-to-use interface with visual aids to communicate details and provide researchers an easier way of keeping track of each conference and their deadlines. Williams's platform design describes concentric rings with bolded sections and a time marker that revolves about the center point. By clicking a ring, users may easily view automatically scraped conference details from WikiCFP, the site on which these conferences exist. The initial project vision that guides the development of this project is shown in figure 1.

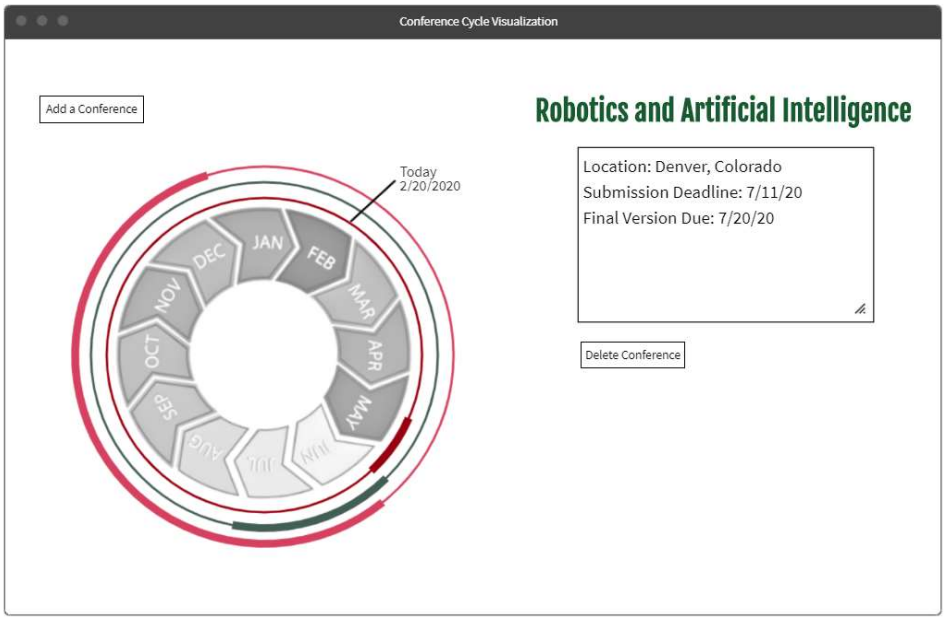

Figure 1. Initial Product Vision

Requirements

Functional Specifications

Data collection along with visual aid and display is essential to the functionality of this product. There are currently two methods of collecting this data in order to display to the user. The first is by scraping data from WikiCFP by URL to include in the visualization upon the user's request; the other option would be to input the data manually. The data then needs to be provided in some visual form. To this effect, conferences are symbolized in the visualization as rings. Each ring includes a bolded section representing the abstract/submission-to-decision period, along with the submission deadline, if there is in fact an abstract, submission, and decision deadline. Conference details are then provided in a separate portion of the web app by a simple click. In this manner, conference data are efficiently navigated and categorized in intuitive, custom

formats. In parallel to considerations of functionality, ease of use is an important aspect of the project as well. This was the reason the option to add a conference by URL was implemented. If the conference is on WikiCFP, the user must simply copy and paste the web URL, and our scraper will pull that data for them. In addition, the user may also delete a conference entirely with the click of a button. This clear, user-friendly display communicates conference details in an orderly and refined style.

Conferences are stored as an array loaded from an imported package. Packages store conference details, ring sequence, and color preferences. Necessary conference information includes relevant attributes collected from WikiCFP and the user's combined edits. The public data is either manually entered or searched through the app and pulled from WikiCFP.

Non-Functional Specifications

Data scraping is performed in JavaScript using libraries from NodeJS. Data scrapers, also called crawlers, are programs written to collect information from structured websites. WikiCFP is the website from which our code will scrape text in the form of a JavaScript object type.

The visualization is easy to use and understand for new users. UI/UX is attractive to view and conference rings are distinguishable based on color and order from the middle. The navigation interface is intuitive and natural to navigate without distraction, enabling unfamiliar users to adjust quickly.

Overall, integration into an existing website is straightforward and effortless. The product acts as a "widget" and can plug-in to existing HTML smoothly. The product is programmed in HTML and JavaScript per the client's request. The Definition of Done is formally met and the web app is located in a public GitHub repository where all source code and documentation will be provided.

System Architecture

Figure 2 below explains the process non-admin users follow when interacting with the web app. Upon website startup a file living on the domain that contains the stored conference data will be used to populate the conference visualization. Simultaneously, conference objects will be created along with associations to their relevant visualizations. This then allows non-admin users to click on the created visualizations and view the relevant stored information.

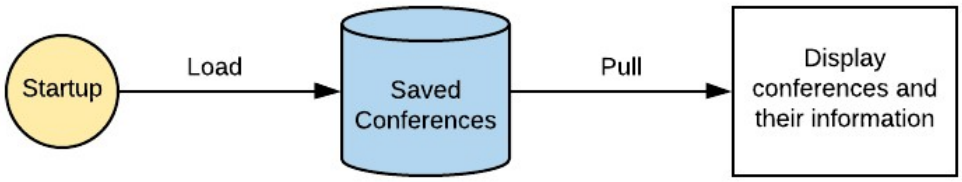

Figure 2. (Non-Admin) User Interaction

Figure 3 explains the process admin-users follow when interacting with the application. Following the diagram from the yellow Startup node along applicable paths demonstrates how admin-users can manage what is revealed to non-admin users of our product. These various options include the deletions and additions of new conferences along with scraping WikiCFP for relevant conference data.

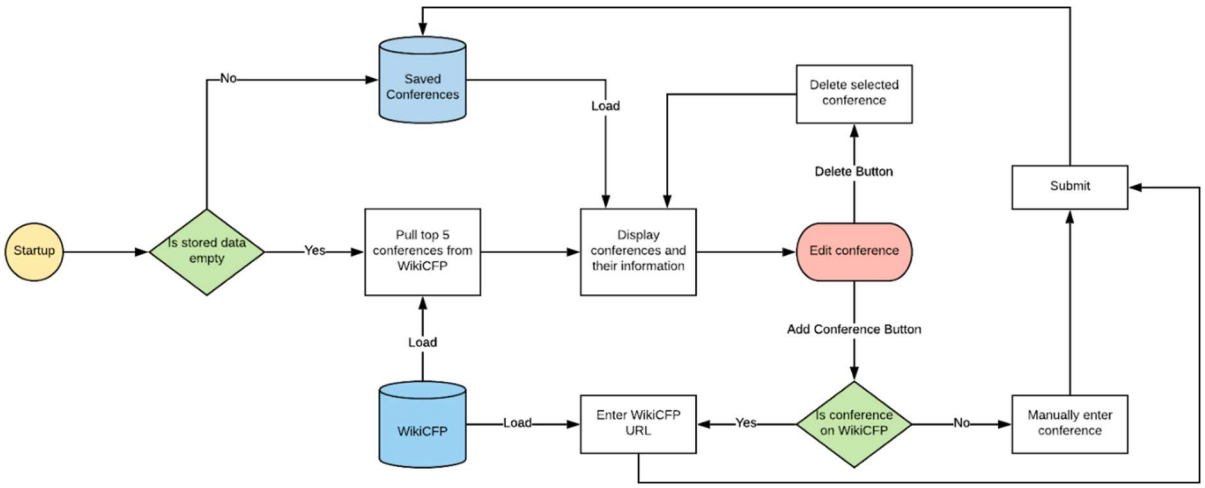

Figure 3. Admin-User Interaction

Figure 4 below shows the architectural design of saved conferences and their integration with the larger web application. Text files will store data of conferences scraped from WikiCFP or be edited by users manually. The data in each text file summarizes all added conferences and their relevant information.

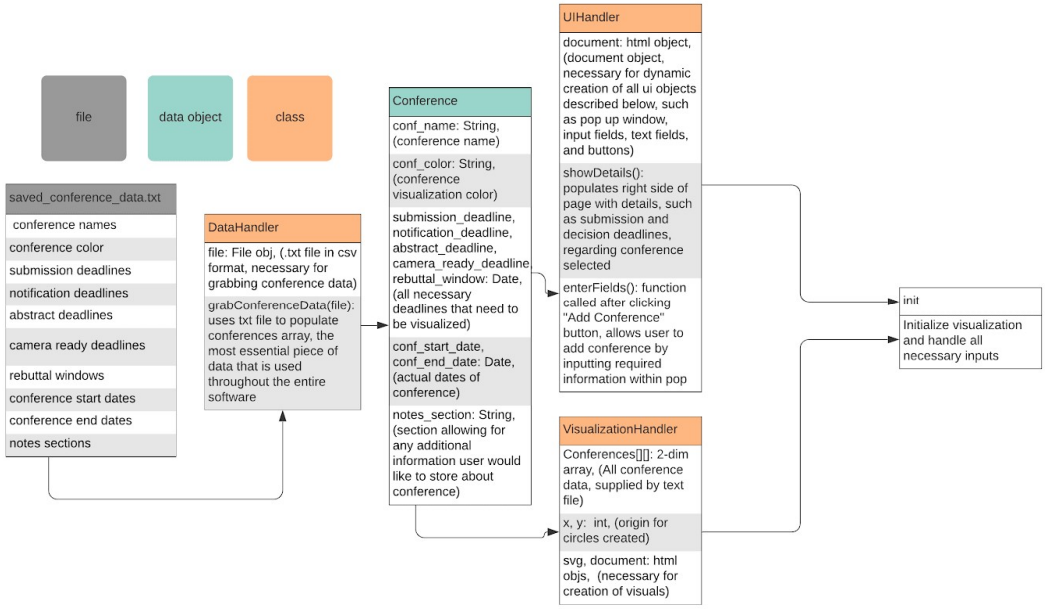

Figure 4. UML Diagram of Saved Conference Data Integration with the Web App

Technical Design

The format for saved conference data is a text file. Figure 5 below demonstrates this concept:

```
3 <-- Number of conferences
IEEE CAMAD 2020,http://www.wikicfp.com/cfp/servlet/
event.showcfp?eventid=57737&copyownerid=81368&skip=1,Robotics,#f08950,24,32,48,100,120,Right There,These are some notes
                                                                                                    <--
RAI--Ei and Scopus 2020,,AI,#9ce490,46,80,102,20,30,Over Here,And here are some more notes          <-- Example Conferences
IWSMR 2020,,Physics,#90c6e4,52,96,256,24,48,Under That Bridge,Eventually you get tired of writing notes <--
```

Figure 5. Text File Format

The application's homepage displays the conferences as concentric, uniquely-colored rings. Title, dates, and other relevant information are displayed beside the visual aid with buttons for adding and deleting conferences. The entire ring represents a calendar year and the bolded sections represent time between deadlines. Figure 6 shows these representations below:

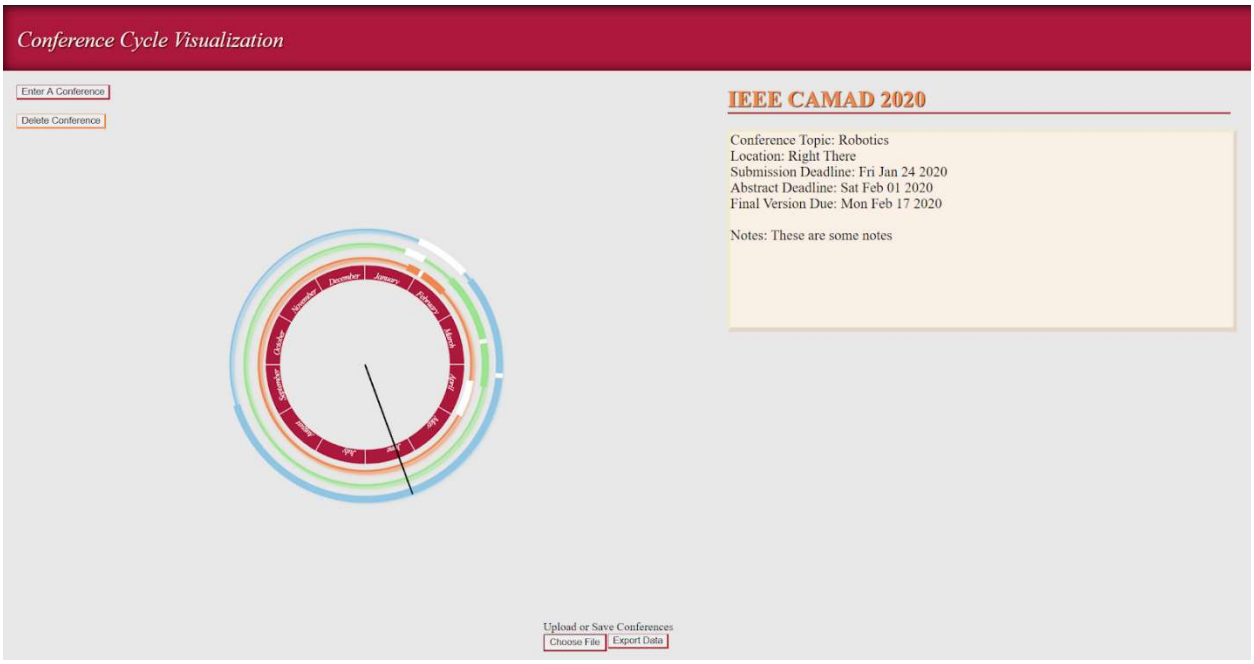

Figure 6. Conference Cycle Visualization Homepage (Admin)

When the "Add a Conference" button is pressed, the following pop-up window appears (shown in figure 7). This window allows users to click a hyperlink that directs them to the WikiCFP website. After the user finds their desired conference, they enter the URL into the specified box to load all details automatically. For further customization of conference details, or if the particular event does not appear on WikiCFP, the user may enter the details manually.

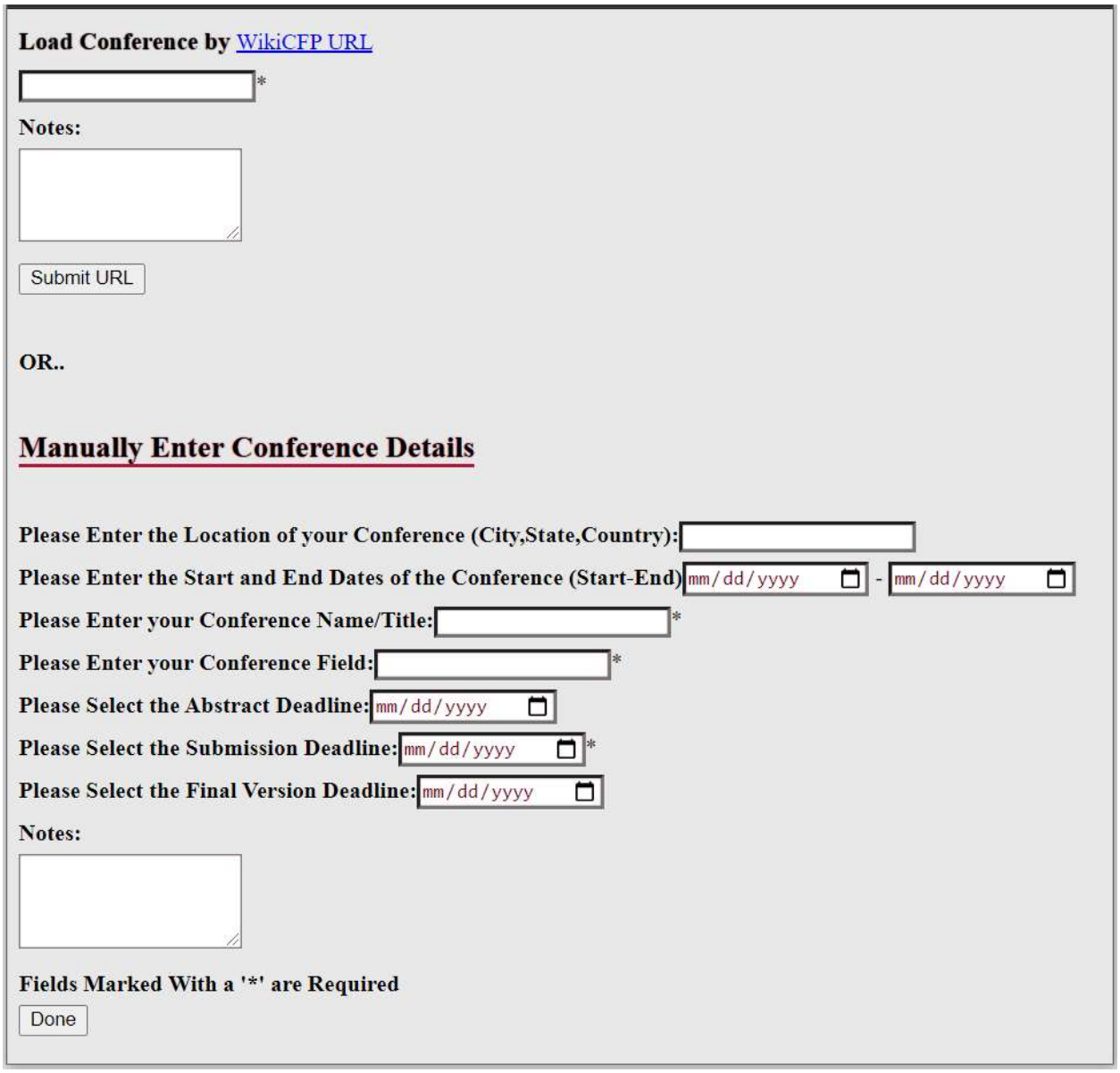

Figure 7. "Add a Conference" Pop-Up Window

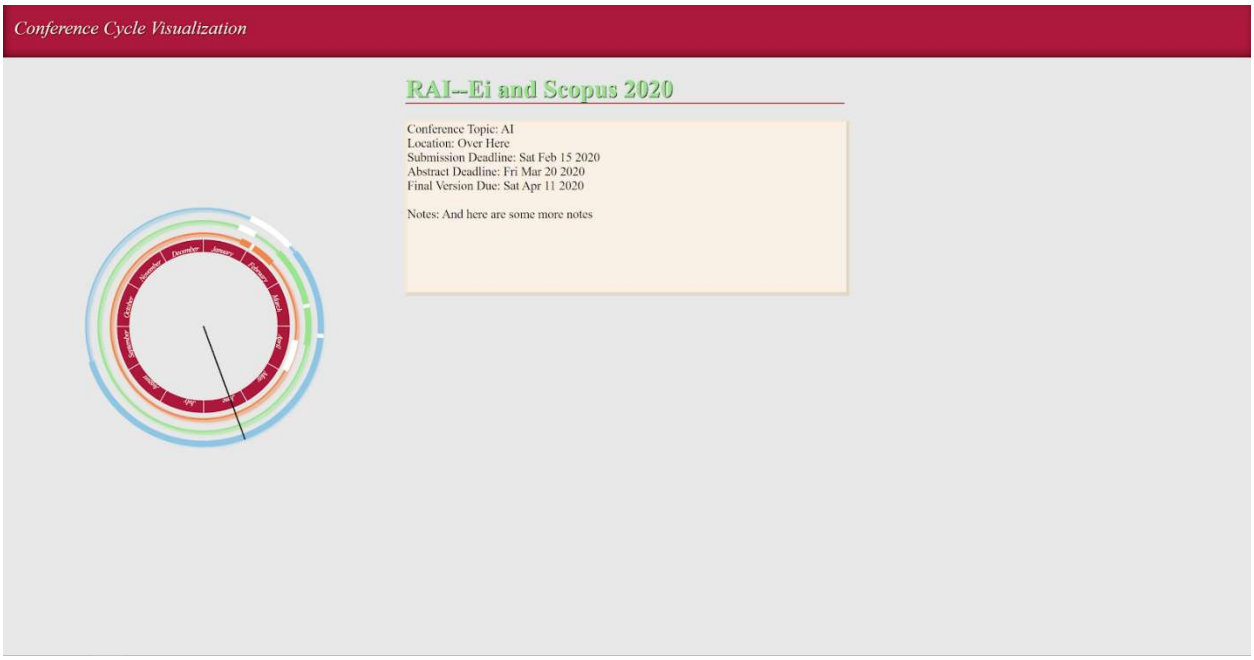

Figure 8. Conference Cycle Visualization Homepage (User)

Figure 8 shows the forward-facing application viewed by a regular user. This version of the program does not permit any customizations and simply displays the admin's data and preferences.

Software Quality Plan / QA

The table below describes all aspects of the plan to promote the software quality of this project:

Software	**Affect on end product**
Dev-Quality	
Detailed documentation	High-quality documentation on all code will ensure ease of understanding and aid in a quick integration process.
Concise, non-cluttered code	Code that follows best practices will be easy to read, making later additions or revisions possible.
Streamlined code	Keeping code streamlined and simple will result in lower usage costs and better user experiences.
Code reviews	Peer-reviewed code creates opportunities to enhance the program.
Pair programming	A collaborative work environment increases productivity and velocity of development and well as the quality of the final product.

Security	We aim to provide a program that limits the probability of a security breach by following best practices, ensuring the safety of users' emails and conference preferences.
QA	
Unit testing	Unit tests via MochaJS will prove our code is accurate, correct, and robust, resulting in a high-quality product.
Client testing	Client testing via MochaJS will ensure client satisfaction and allow for revisions prior to the end of the course.
UX testing	There will be tests for usability via MochaJS to provide improvements on the application's user-friendliness.
Disability accommodations	Since the design is based around separating and distinguishing conferences based on color, individuals who experience color-blindness could find the UI challenging. In addition to color blindness, other visual impairments might hinder the application's usability as well. We recommend additional investment in program development to address accessibility issues.
Peer testing	A mock client test will be performed during which a fellow computer scientist (outside of Field Session) will download, run, and implement the JavaScript from the public repository. This real-world test will ensure the integration process is as easy as possible.
Course Documents/ Deliverables	
Proper grammar	All deliverables use proper grammar and professional verbiage.
Proper audience tone	Deliverables accurately target the desired audience, providing dense descriptions and high-level details, when necessary and/or appropriate.
Highly informative and properly conducted presentations	Presentations provide an accurate, descriptive, engaging, and professional overview of the necessary topics.
Overall document quality	Documents and deliverables reflect a high standard of quality in development and client satisfaction.
Project Documents	
README	An intuitive and descriptive README will provide proper context, descriptions, and instructions for implementation.

Table 1. Software Quality Plan Details

Results

Features Deemed Out of Scope

Some initially-planned features were deemed out of scope, unnecessary, or unimplementable by the client and/or our team. The final product will not store user data, overleaf links, or track papers. A database of conferences will not be stored by our app due to storage constraints; instead, this will be scraped on-demand. By removing these features from the project scope, our product can stay within schedule.

Testing

Testing is important for ensuring the reliability and performance of a program. The following subsections describe the approaches the CCV team has taken to thoroughly test and ensure the utmost quality of the project.

Performance Testing Results

The UI is designed to be quick and responsive without allowing redundant files or functions to disrupt runtime. Quick, direct, and simplified data-scraping procedures ensure fast and accurate results of conference searches. Lag and latency are negligible, allowing full enjoyment of workflow for program users.

Data Scraping Test Results

Meticulous tests were run to assert the validity of the information presented to users of our app. The NodeJS server runs as intended, allowing the HTML code to speak to the JavaScript node server effectively. The Search by URL field scrapes data from WikiCFP almost instantaneously. “Undefined” is returned instead of a JS object if the URL searched for does not exist on WikiCFP. If an invalid URL is entered into the previously mentioned field, an error is yielded; the checks performed for this test make sure that manual entry cannot break the visualization by analyzing the input string for correct formatting. The JS object returned by the data scraping method is complete and accurate to its crawled source. Inputted and scraped strings are trimmed of leading and trailing whitespace. Lastly, the top-five-URL method in the back-end scraper does, in fact, return the top five conferences from WikiCFP’s homepage. MochaJS unit tests ensure the accuracy and robustness of the web app. The back-end code loads and stores data that is good for use and the front end displays it accurately and completely. All of these tests demonstrate the precision and accuracy of our web app.

Usability Test Results

Workflow and speed-of-understanding are integral parts of our web app. The user experience and interface are easily navigated. The program has not been found to be “crashable” by typical methods of usage. Lag and latency are negligible, providing a smooth user experience. The workflow is fluid and adaptable to a user’s preferences. Auto-filled data upon searching for

conferences are manually editable if the user desires changes. These important results prove the ease-of-use to users of the app.

Front- and Back-End Compatibility Test Results

Our visualizer app relies on the flawless presentation of back-end data by the front end. After text file imports were clear, it was verified that arrays populated conference instance members. Variables were not null, if applicable, and angle values were clamped from zero to two-times-pi. Date values were clamped between zero and 366 as necessary for the circular nature of the generated diagram. Conference dates were converted to numbers and vice-versa without error. The date object was returned when convertNumToDate was called; similarly, an integer was returned when convertNumToDate was called. The text file export button correctly formatted the text file and placed data in the correct positions for later reading. Most importantly, the conference radii were evenly spaced.

Recommended Future Work

If work is to continue on this project, a few design additions might improve it. If user- and admin-facing apps shared a database, security could be monitored and users could save personalized data such as notes, ring colors, and links to papers and external articles. Animations would improve the visual quality of the app by providing smoother transitions and UI feedback. Lastly, email notifications could be beneficial to remind users about approaching deadlines. It is recommended to implement these and any other design features that would make this app more enjoyable to administrate and use.

Lessons Learned

This project was full of moments during which new skills were needed in order to continue forward with its scope. Because of this, the time spent was both difficult and rewarding. Our team overcame significant issues during development. Specifically, we realize that, while temporal skills vary, it is essential and beneficial for the team to consistently communicate. Along with this, attempting to develop and implement a plethora of features will never be of greater reward than getting a small number of features to work *perfectly*. In other words, always go for quality over quantity when it comes to delivering products to clients.

Understanding the aspects of the project that are unimplementable in the relative time frame is also essential. Without this, teams could spend long hours working on code that is never implemented. Time saved can be used to write documentation, comments, slideshows, and other deliverables. Without comments and documentation, even small projects become incomprehensible.

A unique perspective and lesson learned could also be taken from the current climate we are working in. As a computer scientist, most of your work is done behind a screen and keyboard, however, communication is more important now than ever before. Without the ability to meet face-to-face, many concepts and ideas about discrete components of the project are lost in translation. Inability to communicate quickly and effectively can lead to larger issues: team morale, project workflow, and agile velocity. Overall, this field session project has helped us learn and experience things we would not have had the chance to learn elsewhere.

Appendix

Links Relevant to Development & Libraries Used

- WikiCFP
- Web Scraper (Browser Extension)
- NodeJS
- Puppeteer
- Cheerio
- Request
- JavaScript
- W3Schools
- HTML
- CSS
- MochaJS