CSCI 370 Final Report

Team: RMS-AAPG

Nicholas Herbic
Raymundo Corona Nunez
Mikayla Sherwood
Josue Milenga
Alexa Nelson

Revised June 15, 2022

CSCI 370 Summer 2022

Dr. Paone

Table 1: Revision History

Revision	Date	Comments

Rev – 1	5/20	Completed Sections:
		    I. Introduction
		    II. Functional Requirements
		    III. Non-Functional Requirements
		    IV. Risks
		    V. Definition of Done

Rev – 2	5/25	Updated Sections:
		    II. Functional Requirements: Reformatted to be a subheading section rather than a bulleted list.
		    III. Non-Functional Requirements: Reformatted to be a list section.
		    IV. Risks: Added likelihood, impact and migration aspects.
		    V. Definition of Done: Added to clarifying features from our client.
		
		Completed Sections:
		    VI. System Architecture
		    XII. Team Profile
		    Appendix A
		    Index

Rev – 3	6/1	Updated Sections:
		    II. Functional Requirements: reformatted to make clear the 3 separate sections
		    III. Non-Functional Requirements: reformatted to fit advisor's criteria
		    VI. System Architecture: Renamed figures and added one more.
		    Appendix A: Adding more terminology.
		    Index: Fixed so the figures have the corresponding page.
		
		Completed Sections:
		    VII. Software Test and Quality
		    VIII. Project Ethical Considerations

Rev – 4	6/6	Updated Sections:
		
		Completed Sections:
		    XI. Project Completion Status
		    X. Future Work
		    XI. Lessons Learned

Rev – 5	6/10	*Updated the style of formatting.
		Updated Sections:
		    IV. Risks: Changed from list format to a table
		    VI. System Architecture: Updated figure layout and captions. Also added a leading text.
		    VII. Software Test and Quality: Added a leading text.
		    X. Future Work: More information was added.
		    Index: Was changed to Appendix B and updated.
		
		Completed Sections:
		    Appendix B
		    Appendix C

Rev – 6	6/15	Updated Sections
		    II. Functional Requirements: updated the layout and added to the user info page.
		    VI. System Architecture: moved where our website images were placed (spaced them out)
		    VII. Software Test and Quality: Elaborated upon the opening paragraph
		    X. Future Work: Added elaboration on future works.

Table of Contents

I. Introduction

The American Association of Petroleum Geologists (AAPG) is a massive organization with thousands of professionals. Every year, AAPG hosts a nation-wide conference that gives many different people the opportunity to showcase their interesting breakthroughs in research and technology via oral or poster presentation. This year, the annual conference is being hosted by the Rocky Mountain Section (RMS) here in Denver.

In years past, AAPG has always used a printed Excel sheet that is handed out for each session (typically 8 total sessions throughout the conference) to each judge. These sheets would then be turned in and tallied up by hand. This is a tedious and outdated process that is subject to human error over the course of 8 sessions for both oral and poster presentations.

To transition to a more modern method of judging, RMS-AAPG wants to develop a web-based application that allows not only designated judges, but any attendee of the annual conference to submit peer reviews of any presentation. It must be simple for users to navigate to encourage this transition and allow for admin access to view the final average scores.

II. Functional Requirements

The three main components needed for this project include the following:
(1) A website that interfaces with the customers; (2) a database that stores all the user information, presentations, and QR codes; and (3) a script that pulls all the scores that are stored for each presentation in order to provide accurate results. The scores are determined by crowd voting, and the judges determine the best presentation from the result of the presentation with the highest score.

II.A Website

The website must allow the creation of users in order to properly keep track of judging scores as well as display a schedule when needed. The goal when it comes to judging speakers in particular is that attendees can "sit in" for entire blocks of time in order to judge a section's individual speakers and not individual people alone. Finally, the website must also have a way to keep track of scores per event since presenters can present at multiple sections and have multiple posters.

Home Page:

The home page contains a menu screen with options to access the schedule or proceed to the judging form. This space is mainly to establish that this website belongs to the client and that its sole purpose is to judge the RMS-AAPG conferences for this year and years to come. The schedule option redirects the user to another website containing the schedule pdf, per the client's request, and the judging form option leads the user to the user info page.

User Info Page:

This page is used to create users in the database's users table based on gathered data. Here, information such as first name, last name, email, and company name are entered by the user and stored in order to create unique users within the database. Because each email is unique, the user's email is used as the primary key in the users table in the database.

Selecting the Conference Time Section:

Here the user selects the time section they are currently attending in order to display relevant information only. This means that once the user selects the section they want to grade, the next page for picking a presentation displays only the presentations that are within the time frame they selected.

Evaluation Selection:

This page allows the user to further narrow down their options based on their previous selection and what presentation or poster they plan on judging. Here the user can judge a poster or individual speaker based on predetermined criteria. The rating system allows for 100 points total and a section for comments. It consists of four separate sections for overall presentation; content; overall impression; and comments/suggestions. This page also confirms a score is submitted, and inputs the information into the database. Using the composite key of the user's email and the title of the presentation they are evaluating, ensures that a user does not judge a presentation more than once.

II.B Database

The database has 3 important functions: storing user information after it is input into the user information page; storing presentation information after admins enter the presentation csv; and storing the presentation scores that the users input.

II.C Retrieval of Scores

The conference runners require a way to retrieve users' evaluation scores of each presentation, as they will announce which presentations received the highest scores. In years past, AAPG calculated the average score for each presentation and used that as their final score. The webapp handles the calculation of presentation score averages and sends them to the database. The clients are then able to access the SQL database through Azure Data Studio and download the scores table as an Excel or CSV file.

III. Non-Functional Requirements

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. The non-functional requirements of the project, based on the functional requirements, are as follows:

- The program must be a web application.
- The program must be accessible from any web application (including mobile).
- The program must be easy for new users to understand.
- The program must use the client's cloud database, Azure, but does not need to connect to existing 3rd party host websites.
- The results of the judging must only be made known to the main clients (Ellen and Topher).
- The RMS logo must be present on the home page.

IV. Risks

There are several technology risks that had to be considered during the development of the web app. Most of the risks for this project involved dangers within the database. The database is something that should only be supplied with accurate data, and must not be accessed by unauthorized users. The technology and skill risks for this project are included in Table 2 below.

Table 2: Technology and Skill Risks

Risk	Description	Type	Likely to Occur	Impact	Risk Mitigation Plan
User-interface Problems	Website component functionality might fail (e.g. buttons, page errors, etc.).	Technology Risk	Very Likely	Moderate	Make the application as simple as possible and use an intuitive design.
Database Leak	The database could leak sensitive information about users (e.g. names and email).	Technology Risk	Unlikely	Minor	Microsoft Azure has built-in security protections, such as encryption.
Flawed Data	The database could encounter flawed or skewed data.	Technology Risk	Very Likely	Major	Create an application that prevents the users from submitting multiple forms for one presentation or entering scores out of range.
Dropped Tables	The database could experience a dropped table, or deleted table, accidentally.	Technology Risk	Somewhat Likely	Major	Include a SQL command "if it exists" which is a clause that prevents a table from accidentally being deleted. It also prevents the making of an extra table if it already exists.
Azure Cloud Issues	Azure cloud is a specific type of cloud that the client uses, and has different syntax and tools that need to be learned.	Technology Risk	Very Likely	Major	Familiarize ourselves with the software by doing research.
JavaScript and Website Management	Not all team members have a background in website management and using JavaScript.	Skill Risk	Very Likely	Major	Familiarize ourselves with the language and application by doing research.
Database Management	Not all team members have a background in database management.	Skill Risk	Very Likely	Major	Familiarize ourselves with using databases and implementing them on the Azure cloud by doing research.

V. Definition of Done

In order to make the expectations of both the clients and the developers clear, a set of criteria were created to outline exactly what needed to be accomplished for the project.

- The website's functional/non-functional requirements are met in a way that is efficient and user friendly.
  - Efficiency would refer to minimizing the amount of clicks a user needs to make in order to submit an evaluation.
  - User friendly would refer to making the design of the application intuitive for users to use correctly.
  - The information is stored properly in the database the developers created.
- Create an environment where this evaluation process and website can be repeatable for future conferences, different companies, and events.
- Software demonstrations for the clients satisfy the client's wants/needs.
  - Tests display accurate results in a digestible matter and can be exported if needed for future analysis.
  - Tests would include setting up a "conference", judging the "presentations", and displaying the results.
- A final URL/working product is provided to the clients.
  - There is no need for technical documentation describing the website's functionality, but source code may be included.
- Documentation for potential new admins in the form of code comments, and a final copy of the code to give to the clients.
  - The final copy of the code will be on the Azure server that the clients and the developers both have access to.

VI. System Architecture

The importance of the system's architecture is very high since many users will be interfacing with the web app. The web app should ensure an easy flow for the users to comprehend and minimize the use of clicks. The system architecture should also ensure that the frontend, the web app, and the backend are communicating properly. Therefore, the system architecture plays an important role in the project.

VI.A Technical Design Issues

- SQL server knowledge problems
  - Problems creating an initial SQL server for our database.
  - Partially linked to the communication issues.
- Non-specific testing issues
  - Testing is done visually by looking at the local website instead of creating dedicated testing scripts.
- Communication issues with our client's technical staff member
  - Delayed information on how to gain server access.
- Problems with knowledge about React
  - Coding is more difficult due to the lack of experience for most of our group.
  - Figuring out what the developers need to do is time-consuming.

VI.B System Design

Figure 1 is an introductory page that functions to signal to the user that they are currently on the evaluation website owned by our client. From here, they have the option to move forward to the evaluation process or see a schedule of presentations.

Figure 2 is a page that gathers the user's data in order to create and populate the entries in the SQL database. These entries are used to keep track of users, which help keep track of evaluation forms and ultimately calculate scores.
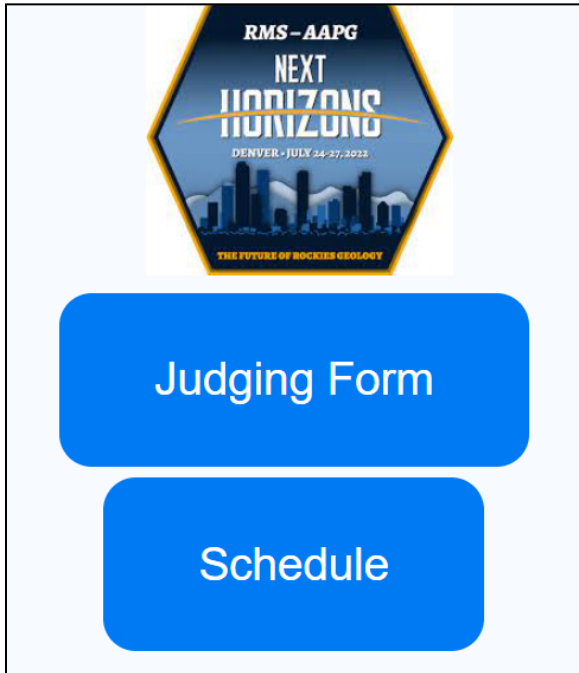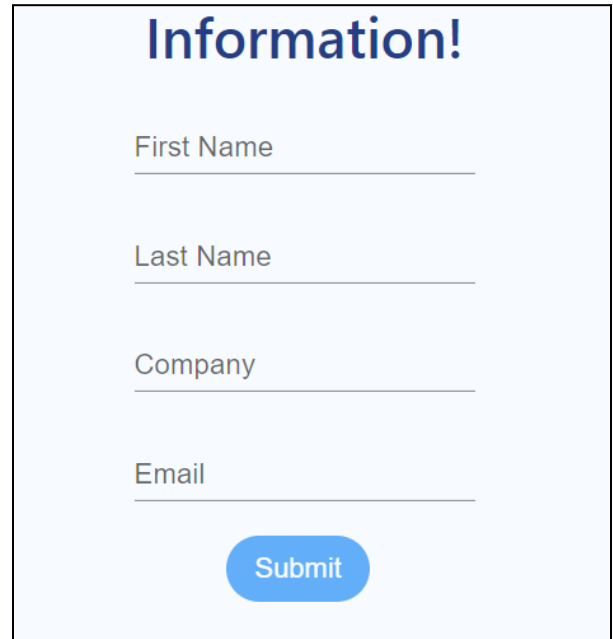

Figure 1: Home Page Display

Figure 2: Evaluation Form Page Display
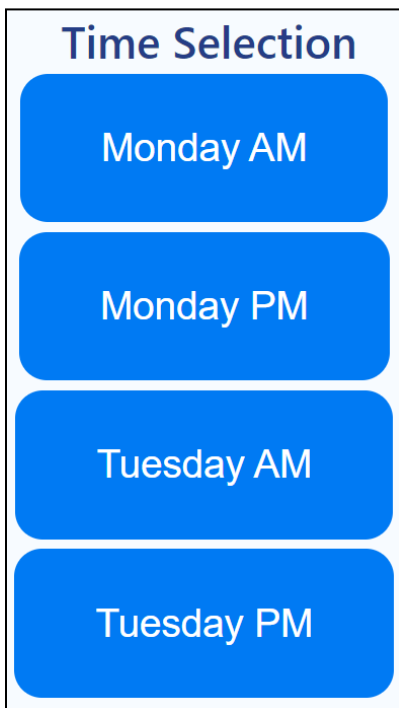
Figure 3: Time Selection Page Display
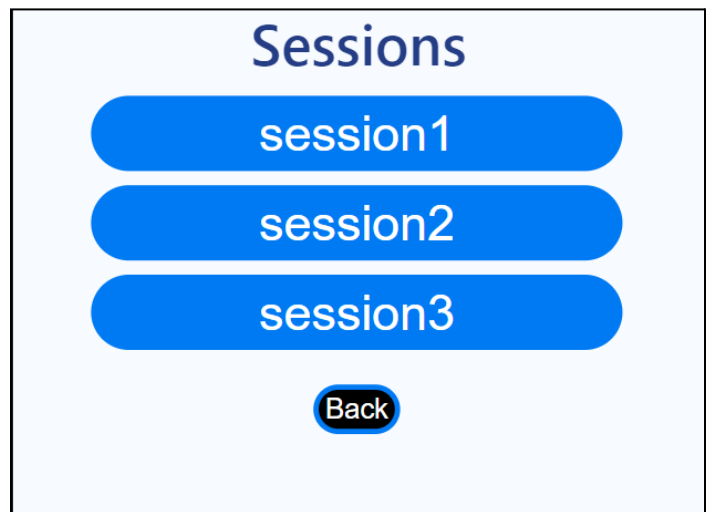
Figure 4: Sessions Page Display

Figure 3 is the Time Selection page that comes after the user is created and presents the options of a time to select from.

Once the user picks a time, they are moved to the Sessions page, Figure 4. From here, the user decides what session to select.

Once the user has selected a time, they are prompted with various presentation panels. Figure 5 is the Presentation page that layouts each presentation in panels for a single session. The developers also suggested that our client use a panel format when listing the presentation to make it more appealing to users.
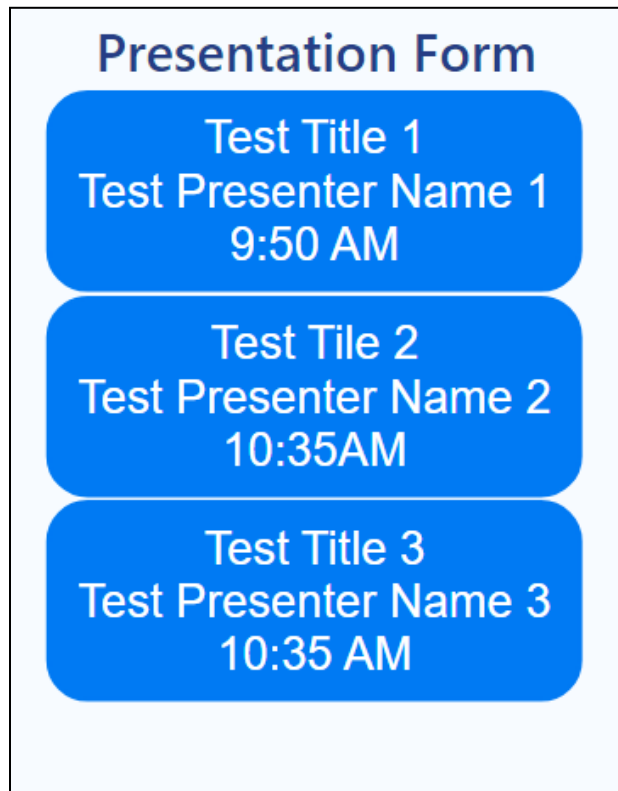

Figure 5: Presentation Page Display

Figure 6.a: Presentation Page

Figure 6.b: Content Page

Figure 6.c: Impression Page

Figure 6.d: Comments Page

Figure 6: Evaluation Multi-Step Form Page

Figure 6 shows the pages that make up the entire process of the Evaluation form. Once the time, the session, and presentation are selected then the user can access these forms. Per the client's request, the same flow and information were kept; the developers simply made it a bit more sleek as well as suggested adding a small "information" button, shown in Figure 6.a and Figure 6.b, which displays a short description of what is evaluated for each category.
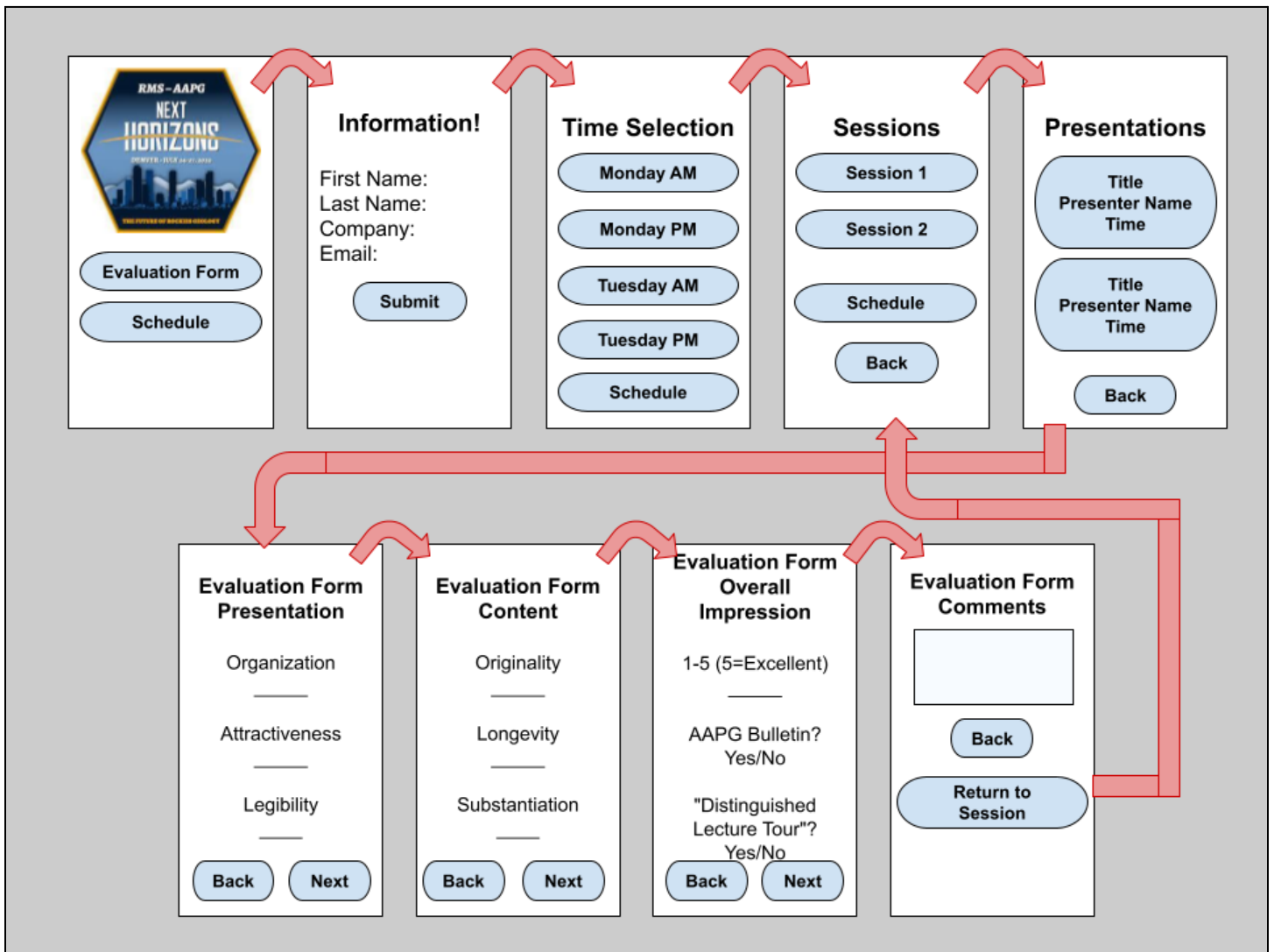
VI.C Page Flow

Figure 7: Website Flowchart

Figure 7 shows the web flow from one page to another. Essentially, the user starts at the home page where they have the option to move to the Evaluation Form or see the Schedule. Clicking on the Evaluation Form button takes them to the next page where the user enters their information. Once the user is finished, they submit and move on to the Time Selection page. This page gives the user the option to choose what time they are evaluating or the option to look back at the schedule. The user then selects a session on the Session page where they are still able to look back at the schedule if need be. Next, the user selects a presentation on the Presentation page and proceeds to the Evaluation Form. From the Evaluation Form, the user is guided through a series of evaluations for the presentation they selected, and once they fill everything out, they are returned back to the Session page.
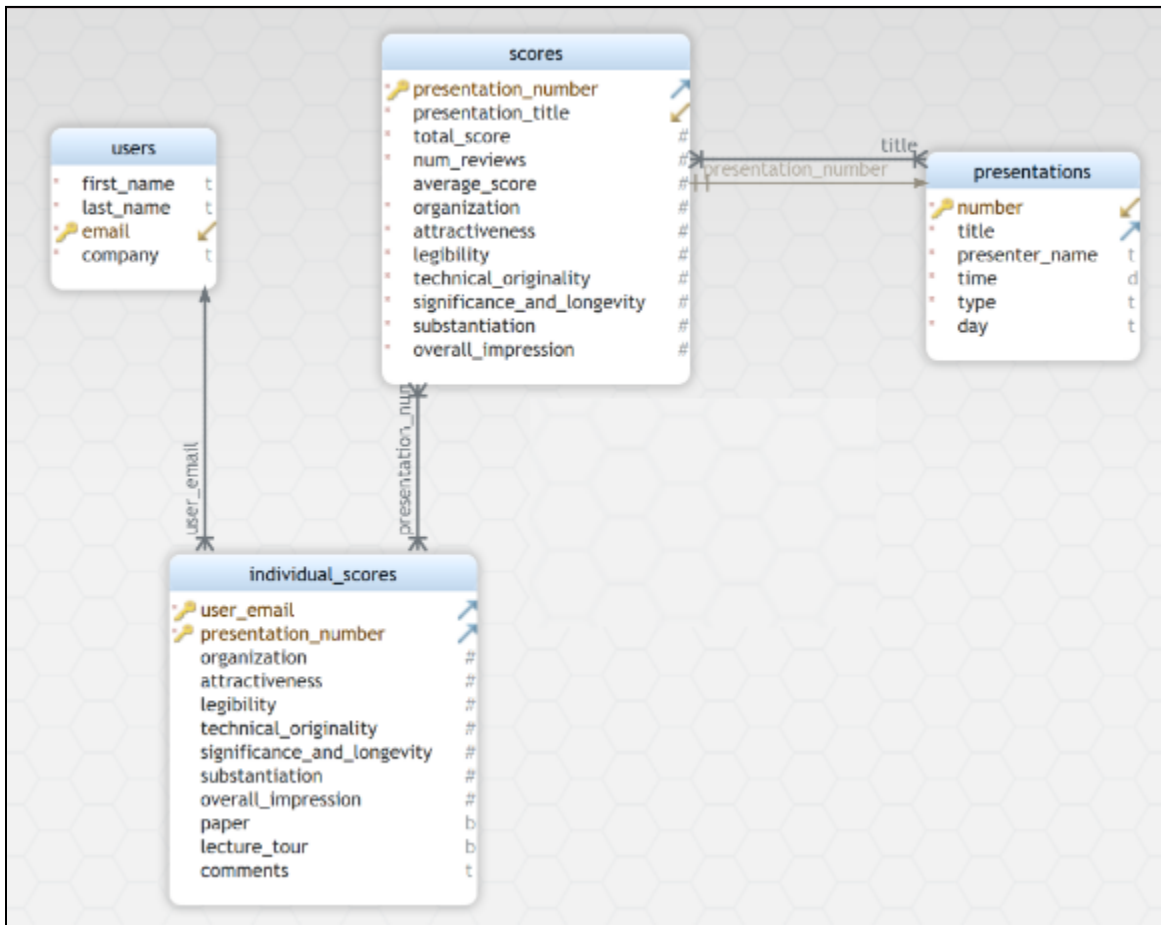
VI.D Database Schema

Figure 8: Database Schema

Figure 8 is an ERD the developers created of the final database schema. It includes tables for *users, scores, presentations,* and *individual_score*. Each one of these tables is added to or queried from at some point during the use of the app. *users* contains all users, and the email is later used in conjunction with presentation_number to create a composite key for *individual_scores*. This composite key is queried whenever the user is viewing the list of presentations, as its existence/non-existence determines if the user has already submitted an evaluation for a particular presentation. This is vital to know, as the developers don't want users submitting multiple evaluations of a single presentation. The *individual_scores* table keeps track of each single review every user leaves; it is the largest table by far. The *scores* table tracks the total scores, number of reviews, and score averages for every presentation. This is what is used at the end of the conference when the "best presentation" for each category is announced.
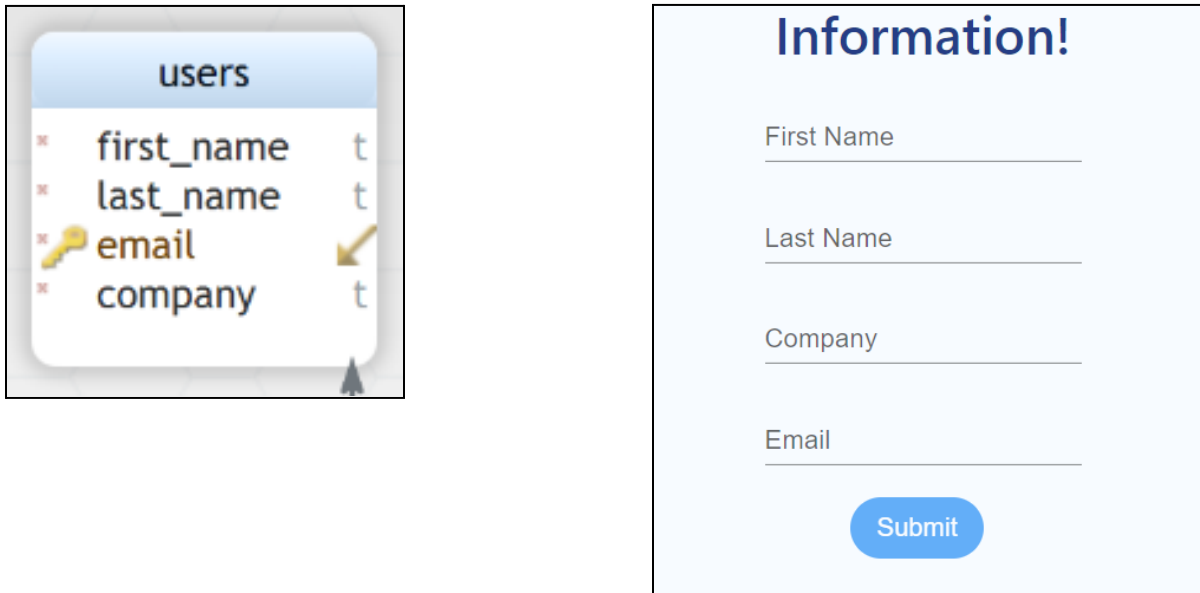
VI.E Database and Page Relation

Figure 9: User Table and Judge Form Correlation

When the user fills out the judging form, each entry is used to populate the *users* table with the appropriate attribute. This is the table in the database that houses our user's information. Figure 9 shows how the two components interacting help the frontend communicate with the backend.
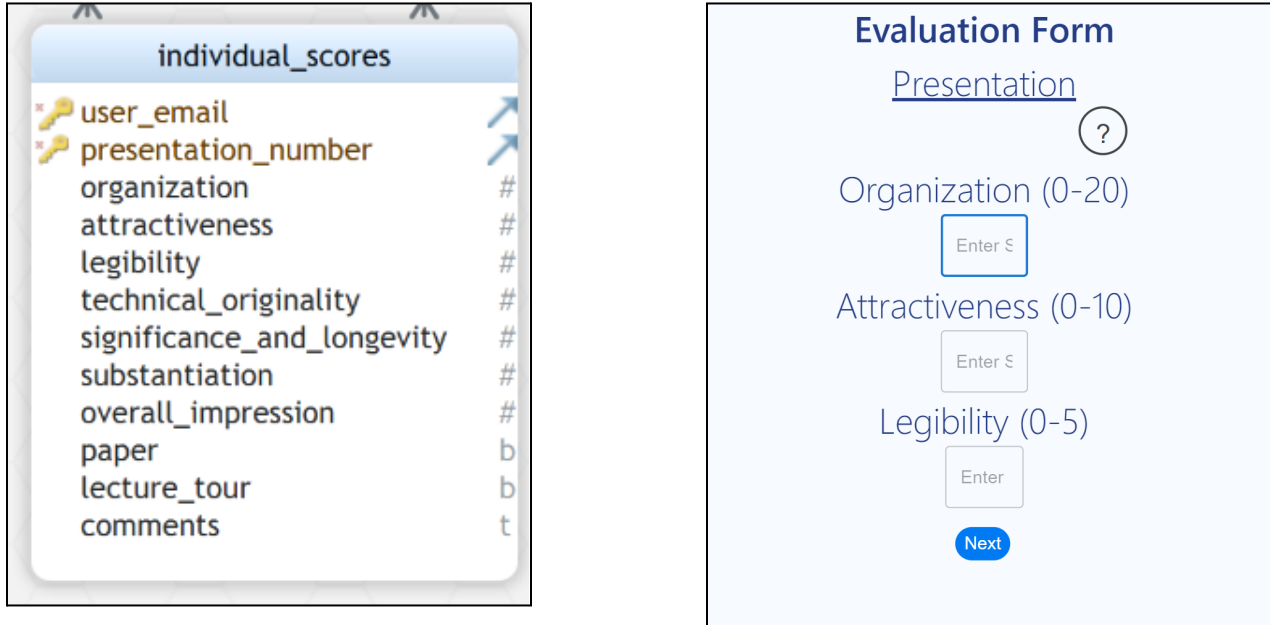
Figure 10: Individual Scores Table and Evaluation Multi-Step Form Correlation

Figure 10 shows the *individual_scores* table that gets data from the evaluation page. The users give a score for each category. Each score is entered, and when the evaluation form is submitted, each attribute is populated with the individual scores for that presentation. Since each presentation has unique numbers, the developers decided to make this our primary key as each individual score is linked to that number.
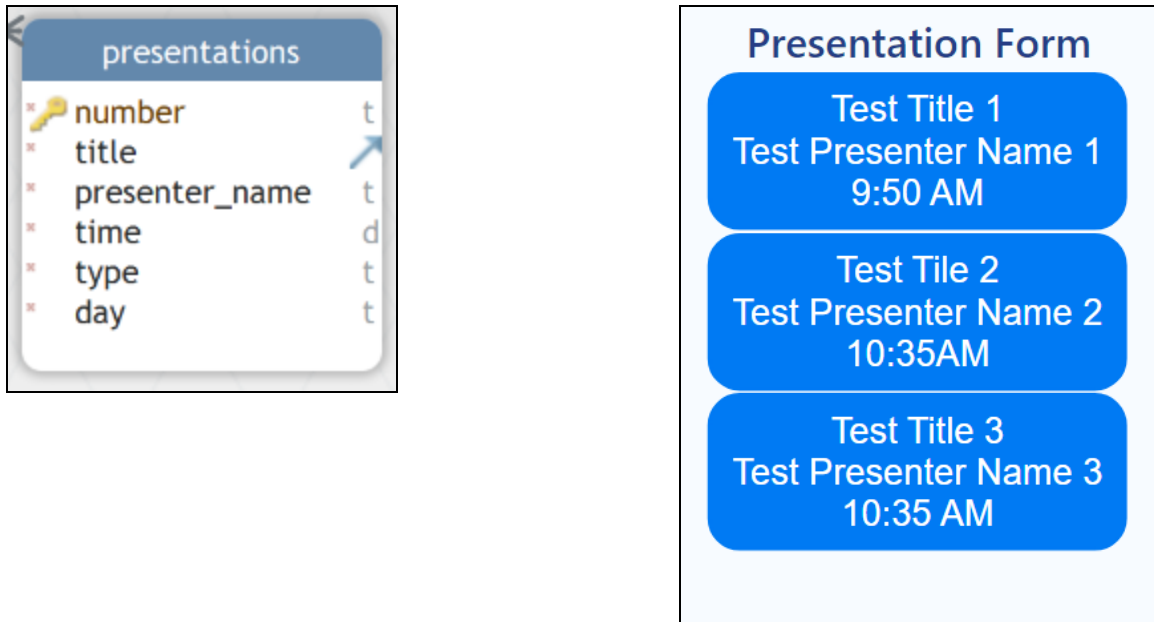
Figure 11: Presentations Table and Session Correlation

On the Presentation page for each time/presentation type combination. First, the type is queried to filter out only presentations of the matching type (oral/poster). Then, the day and time are queried to filter out only presentations in that specific session (Monday/Tuesday and AM/PM). The page is occupied with a separate button for each presentation, ordered by time. Within each button, the time, presenter_name, and title are queried to display to the user. Figure 11 shows the page and table that communicate with each other.

VI.F Interaction between Systems

Figure 12: System Interaction Diagram

Figure 12 shows that the web app and database live within an Azure server provided by the clients. It also displayed how the frontend and backend interact with each other, as well as how different types of users interact with the web app differently.

VII. Software Test and Quality

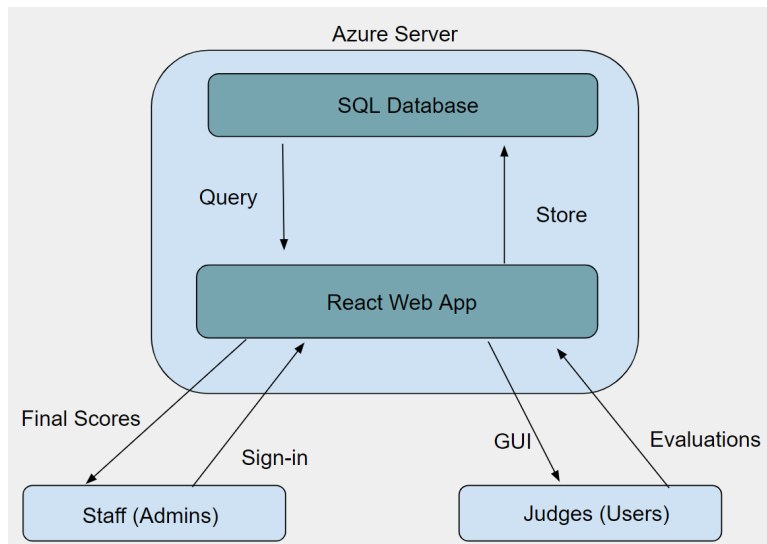Software tests help determine the reliability of a piece of software. The testing process is crucial to ensure the web app is able to flow correctly and users are able to use the web app without a problem or error. Users are not always going to act perfectly; therefore, the testing phase ensures that they are able to communicate with the web app even if complications were to occur. Table 3 lists our testing methods and results. There are two main testing environments that were utilized in order to ensure correct results: manual/visual testing and the Postman API.

Manual testing consists of console logging, browser alerts, and the web app itself. The browser console allows the developer to check that the frontend is running properly and in the way that is expected. Visual testing was used primarily to check the style and the flow of data through the webpage. Also, using browser alerts and the inspection tool within the Chrome browser, we checked accessibility and user interaction to further nail down our goal of making this web application user-friendly.

The second testing method utilized was the Postman API. This platform provides a simulated browser environment and only requires the localhost URL on which the React app runs in order to connect to the app itself. From there, the developer can take the APIs they have created in the React app and test them to see if those APIs work as intended. One way to test the API is to see if the HTTP request is processed by the server successfully. The expected result should return an instance with a 200 series message saying whether the request is "OK" or if there was an error. "200 OK" means that the server processed the request successfully and can now fetch or post the required data. An error message means that the request was not processed and that the API may need to be re-worked in order to make sure it works. Fetched data will be returned as raw JSON data, in which the developer can "beautify" that raw JSON data into a more readable version.

Table 3: Software Tests

Test Name	Purpose	Description	Environment	Expected Result	Actual Result
Buttons (e.g. Back, Next, Schedule, and Return to Session buttons)	Test the functionality of the buttons and redirect the user to the correct spot.	When a user clicks a button, they are redirected to the appropriate page.	Manual Visual Testing	The component links to the correct page or pdf.	Components were linked correctly.
UserInfo Form	Tests the user is entering the correct data.	Checks that the user supplies enough and valid information.	Manual Visual Testing	Doesn't let the user continue if the information is invalid or not provided.	Submit button is disabled and doesn't allow the user to continue if the information provided is invalid.
UserInfo Database	Tests that the user's data is being entered into the database.	Once the user submits their data, the application ensures the data is input correctly into the database.	Postman Testing	Raw JSON data with the appropriate values.	Appropriate values were present in the JSON data.
Panels	Tests the functionality of the sessions and presentation panels when a user clicks on them.	When the user clicks a panel, they are taken to the corresponding presentation list for that session or judge form for that presentation.	Manual Visual Testing	The panel links to the correct presentation or judge form.	Session panels are correctly linked to presentations and presentation panels are correctly linked to the judge form.
Judge Form	Tests that the user is entering the correct format for scoring.	Checks the user supplies all and valid scores.	Manual Visual Testing	Check the scores before moving through the multi-step form to ensure scores are valid.	Throws errors when a user enters an invalid score and doesn't let them continue.
Judge Database	Tests that the user's scores are uploaded to the database.	Upon clicking submit, the scores entered by the user are pushed into the database	Postman Testing	Raw JSON data with the appropriate values.	Appropriate values were present in the JSON data.
Schedule	Tests that a schedule (pdf) is shown when the user clicks on that button.	When clicked, the button redirects the user to a link that contains the schedule.	Manual Visual Testing	Schedule is displayed in the application.	Schedule is linked correctly.

VIII. Project Ethical Considerations

When creating software, some important aspects to think about before putting your code out into the world are the ethical implications that it may have associated with it. This is because code has the ability to be misused or created for the wrong purposes, resulting in events that could negatively affect the public interest. To help address these issues, the Association for Computing Machinery (ACM) and the Institute for Electrical and Electronics Engineers (IEEE) have

developed the Software Engineering Code of Ethics and Professional Practice. This section discusses a few of these codes; how the project relates to them; and how they are addressed.

VIII.A Active Principles

2.01. Provide service in their areas of competence, being honest and forthright about any limitations of their experience and education.

> When working with a client who may not be familiar with any software development, it is important to be as transparent as possible in order to keep them well informed and updated with any major decisions, technologies, resources, and limitations regarding their desired product and the process of its creation. In order to keep a transparent development process, weekly meetings were held with the client where topics such as current status, future plans, and foreseeable obstacles were discussed. This allowed for the client to ask as many questions regarding the development process as needed as well as provide immediate feedback on anything the client may have changed their mind on.

7.05. Give a fair hearing to the opinions, concerns, or complaints of a colleague.

> When developing software, it is important that the opinions, concerns, or complaints of colleagues are heard. The core of any good development team is that every developer can give their input on the direction that the project is going towards. A good way to maintain a checks and balances type of system in the development cycle of a project is if each developer can voice their opinion, concern, or complaint if necessary. It is critical that the development team gives a fair hearing to that developer's opinions, concerns, or complaints as this could either benefit or hinder the project. The developers for this project recognize that this principle 7, section 7.05, completely applies to this project. Violating this principle can have grave ramifications for both the team and the project. To prevent the violation of this principle, each developer recognizes that at some point they will have an opinion, concern, or complaint about the project. It is the job of the other developers to make sure that they allow that person to voice their complaints, concerns, or opinions to the team in order to adhere to this principle.

VIII.B High Concern Principles

3.14. Maintain the integrity of data, being sensitive to outdated or flawed occurrences.

> This is an ethical standard that the developers must be very cautious about. Having only minimal experience with databases, particularly ones with a large number of simultaneous queries, could lead to an unstable product when active. There has to be no flaws with the SQL database and the queries written in React to ensure that the data is not flawed or outdated. After every conference, the database must be wiped so that there is no accidental overlap when the next conference comes around. There also have to be security checks in place to prevent users from entering numbers beyond the range of the highest/lowest possible score, as well as to prevent a single user from evaluating the same presentation more than once.

IX. Project Completion Status

The goal of the project, RMS-AAPG 2022, was to create a full-stack application that would allow users to score presentations and have those scores aggregated in order to determine the best presentation. All the front-end features have been implemented and work to the client's specifications. The routes to the pages work correctly, moving from one page to the next seamlessly. All buttons are implemented and fleshed out, resulting in the desired effect, i.e. submissions or continuing on to the next page. The user information form is fully functional in that it allows the user to enter their information into the form and results in the data being sent to the database when submitted. The visual aspect of the application is successful; the CSS elements are laid out correctly in both Chrome and Firefox, and Microsoft Edge. However, the application does encounter some sizing issues regarding mobile view. Some of the CSS elements do not lay out correctly for the mobile view layout, specifically for the browser. This may require adding bits of a CSS framework

such as bootstrap in order to handle this issue, but for the most part, all functional and non-functional requirements have been met. In addition to this, testing on the server side part of the project has yielded successful results. Testing the current API in Postman has returned the expected results. A post request sends the correct data to the database and results in a 200 series message and the raw JSON data. A "200 ok" message, lets us know that the request was processed successfully and the raw JSON data was returned. These same results were also seen when testing directly from the browser. However, additional APIs will require more testing to determine if they are successful or not. Even though the current API being tested yields the correct result, the response time from the API call was above 1200ms when posting data to the database. This could result in data being posted to and fetched from the database taking much more time than expected.

X. Future Work

With such a short period of time to work on this project, it was to be expected that not all necessary requirements were met in order to have a fully finished product. We managed to replicate the flow of the website that our clients were looking for, as well as create the skeleton for other future implementations, but this still leaves some key work that needs to be added for full website functionality. These mainly include: setting up an admin portal for easier access to the results, implementing a search engine for presentation selection, and writing a script to display results using graphics.

The first goal to achieve if the project were to be expanded is to create a functioning admin page. This would require a separate login for whoever needs to see the results of the conference or whenever a new set of presentations needs to be added to the website. Theoretically, this would be implemented by adding a new table to the database that contains admin information. This table would contain an email and a password key. From the home menu, a new dropdown menu would be added in the top right corner that would link to an admin login. This page would require you to enter the two admin credentials, namely the email and password. Upon submitting, a query would be run to verify that the email exists in the admin page table, and ensure that the password corresponds to the password for that email. Once the proper credentials are verified, it would take the user to the admin page, where they would be able to download an excel file containing the tallied results, display the results on the webpage, or upload an excel file containing a new list of presentations. The latter would be used to delete all the data in the database and replace the current presentations table with presentations in the submitted excel file. This could be used to add more presentations to the conference by including the current presentations and the new ones in the submitted excel file, or it could be used to submit a new presentation schedule for a future conference.

Another idea to expand the project would be to implement a search bar that would allow the user to navigate the vast list of presentations. This would enable the user to select a desired presentation faster in order to save them the hassle of scrolling through all of the different presentations if they already know which one they want to attend. Something extra that we could implement, which we briefly touched on with our clients, was the usage of QR codes. This would allow mobile users to quickly access the evaluation form of a specific presentation without having to manually navigate the web page. We originally thought of implementing this, but settled on making this a stretch goal. Fortunately, as it stands, we do have the framework to possibly implement the usage of QR codes properly. It would just be a matter of extra time to accomplish this.

The last goal that we have to achieve is to display the results, specifically in a way that is visually easy for the administrators to read. We would most likely have the admin page display graphs that would categorize the presentations by score, from least to greatest. We would also display other analytical data, such as pie charts showing

the sessions with the highest percentage of scores. Besides graphical representations of the results, results can also be downloaded to a csv file in order to be used in a program like excel or spreadsheets.

XI. Lessons Learned

- React is a very useful tool for implementing web applications. Because it incorporates tools from both JavaScript and HTML, it is very easy to learn and teach yourself.
- Node.js is an essential tool for connecting a React web application to a server/database. It allows for communication between the JavaScript web code and an SQL database, giving the ability to send queries to the database.
- Javascript is a great language to use for a project like this because of its ability to both create web applications and run commands to communicate with connected databases.
- Microsoft Azure takes complicated processes, like creating a database or website, and helps to simplify them. While it is not the easiest platform to navigate and learn on your own, with time and practice, it is a great skill to have under your belt.
- Although this project was executed in under 5 weeks, something that proved to be helpful with team organization was sprint planning. This allowed for clear expectations to be set and met by each member of the group.

XII. Team Profile

Nicholas Herbic
Senior
Computer Science
Hometown: Centennial, CO
Work Experience: SUMMET Program Mentor, Generation Teach Math Teaching Fellow, DECTech Camp Teacher
Hobbies: Camping, Fishing, Guitar, and Video Games
Fool me once, shame on you. Teach a man to fool me, I'll be fooled the rest of my life.

Josue Milenga
Junior
Computer Science
Hometown: Aurora, CO
Work Experience: Electronic Sales Associate at Costco
Hobbies: Drawing
A wise man once said, if you write a book, write it with no words.

Alexa Nelson
Junior
Computer Science
Hometown: Benicia, CA
Hobbies: Art, video games
Clubs: Sigma Kappa
Applications are my passion.

Raymundo Corona Nunez
Senior
Computer Science: Computer Engineering
Hometown: Thornton, CO
Work Experience: Encon Design Intern, Challenge Mentor
Hobbies: Reading, Cooking, Night Hikes
Clubs: Multicultural Engineering Program, Society of Hispanic Professional Engineers
I am Groot.

Mikayla Sherwood
Junior
Computer Science: Computer Engineering
Hometown: Aurora, CO
Work Experience: WORTH Xilinx Lead TA, Micro.Bit Lead TA and Course Development, Computer Science Department Intern, Desk Assistant for Residence Life
Hobbies: Film Photography, Botany, Stationery, Cultural Embracement
Clubs: Multicultural Engineering Program, Women's Opportunity to Redesign Technology and History, President of the American Indian Science and Engineering Society
One day… I will make an onion cry.

References

"Getting started," *Getting Started | Axios Docs*. [Online]. Available: https://axios-http.com/docs/intro. [Accessed: 15-Jun-2022].

"Getting Started," React. [Online]. Available: https://reactjs.org/docs/getting-started.html. [Accessed: 15-Jun-2022].

"Installation - material," *UI*. [Online]. Available: https://v4.mui.com/getting-started/installation/. [Accessed: 15-Jun-2022].

MashaMSFT, "Azure SQL documentation - azure SQL," *Azure SQL | Microsoft Docs*. [Online]. Available: https://docs.microsoft.com/en-us/azure/azure-sql/?view=azuresql. [Accessed: 15-Jun-2022].

Node.js, "Documentation," *Node.js*. [Online]. Available: https://nodejs.org/en/docs/. [Accessed: 15-Jun-2022].

Postman, "Postman Documentation," *Postman Learning Center*, 22-Apr-2022. [Online]. Available: https://learning.postman.com/docs/. [Accessed: 15-Jun-2022].

"React-cookie," *npm*. [Online]. Available: https://www.npmjs.com/package/react-cookie. [Accessed: 15-Jun-2022].

Appendix A – Key Terms

Term	Definition
Azure	Azure is a public cloud platform that provides software solutions such as infrastructure as a service, software as a service, and platform as a service.
React.js (React)	A javascript based UI development library.
Node.js (Node)	An open source, cross platform, back end javascript runtime environment designed to build scalable network applications.
ERD (Entity Relational Diagram)	A structural diagram used for designing a database that shows the relationship of entity sets stored in a database.
JSON (JavaScript Object Notation)	A type of file where data is stored in a readable format for developers to look at and manipulate when creating their software.
HTTP (Hypertext Transfer Protocol)	A set of rules for transfering files over the web.
API (Application Program Interface)	A set of definitions and protocols for building and integrating application software

Appendix B – Figures

Appendix C – Tables