CSM Claussen:
Teaching Triangles

Created By:
Tyler Quast, Tyler Brown,
Tomek Prussak, Tim Cranor

June 22, 2017

Introduction

Teaching Triangle's primary clients were Dr. Claussen and Dr. Mehta at the Colorado School of Mines. Dr. Claussen and Dr. Mehta wanted to create a website based on the Teaching Triangles program. Teaching Triangles is a group program that sets out to enhance the teaching experience by organizing schedules so that teachers can group up in pairs of three to review and critique each other's classes. This will allow teachers to have a wider range of opinions on how to improve their teaching proficiency and make a better overall learning environment. The CSM Claussen group has created a website that allows users to input their schedules and run an algorithm that will group the teachers based on scheduling conflicts to make sure all participants have the smoothest transfer when beginning this program.

Requirements

While planning our design decisions and how we planned to go about solving this problem we came up with a list of both functional and nonfunctional requirements that the website had to meet in order to be successful. The functional requirements are as follows:

1. Have user input their schedule
2. Have separate admin and general permissions
3. Allow user to specify professors they do not want to work with
4. Allow user to specify if they want to work in their department or approach the entire campus.
5. Send information to a text file
6. Complete an algorithm that sorts the schedules and puts teachers in groups of three.
7. If it is not possible to only have groups of three, allow the program to be flexible to create groups of four to avoid conflicts
8. Create and return a list of schedules to the admin user only

All of these functional requirements were met upon completion of the project. In addition, there are nonfunctional requirements, or requirements that must be met, that are not necessarily completed through coding. The nonfunctional requirements are as follows:

1. Must be a web service
2. Must be able to complete within a few hours
3. Have a server instance running via working with CCIT

4. Must be deterministic

These requirements were met with the exception of #3. This is due to new and evolving security rules that we would not have been able to meet in the six week session, as CCIT is still deciding upon how PHP will be implemented on the campus network. However, it was promised that after we turned our code over to the client, the client would work alongside CCIT and field session faculty to implement our code on a CSM server.

System Architecture

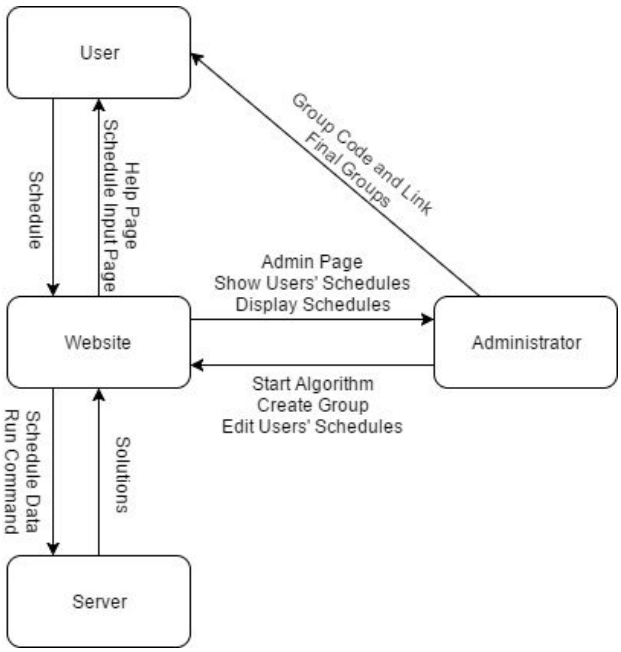

Figure 1. System Architecture Diagram.

Figure 1 is a basic diagram of our system architecture. system architecture is as follows: The website is used as the main hub for all the other components in the project. To start, an admin will come in and create a new group. This information will be sent to the website which is then stored in the server. Next, users come and and fill out a schedule. Once completed, the information will be sent to the website which will be stored in the server, same as before. Additionally, through routing of the website, administrators can see and edit schedules of specific users which can then be stored in the server. Finally when the process is complete, an admin can generate and receive the final groups of triangles. It is then the admins responsibility to send the information to the groups manually (this is due to CCIT restrictions).

Technical Design

The Teaching Triangles assignment can be cut into two parts, the front end website and the back end algorithm, both will be discussed in much greater detail below. Not only was the

entirety of this project written in five different languages, but they all had to communicate and work together in order to generate the user experience we originally envisioned.

The front end of our product is a website, which was created in HTML, CSS, Javascript and PHP. PHP is the backbone of the site. It has the most code of the four, completing many basic functions. For example, data is frequently passed from page to page to keep a consistent and coherent user experience. Much of this was done with PHP. Additionally PHP was used to complete data storage, data reading, and file creation and deletion. PHP was also used extensively for completing form verification. Every text box assures that email address are valid, that multiple user names are not in use, makes sure that every field is filled out and much more. While not the most complex of the four languages used it was the language that the rest of the languages work off of.

Javascript was implemented mostly in the schedule editor page. The Javascript code was the most technical part of the website, as we wanted the site to be as dynamic and flexible as possible. For example, in the schedule editor, users are initially given a single class box to fill out their schedule. If so desired they can add one, two, or as many additional schedule boxes as they need to input all their information. This gives the site a clean and minimalistic feel which makes the user experience as simple as possible. Javascript was also used to create functions that could be called throughout the rest of the code, similarly to functions in Java or C++ for example. Javascript was also used in the admin page to allow the admin to see which users have registered in real time. When combined with PHP admins are also given the option to edit, and delete specific users schedules within the group.

The HTML was used for general formatting of the website. HTML was the most simple of the three languages used, being used primarily to display text, and format things such as the table and positioning of various objects on the website. HTML was used primarily on the index page. It was very simple to create a clean landing page that was visually appealing. We also used HTML to group objects into classes so they could be edited with greater ease and efficiency.

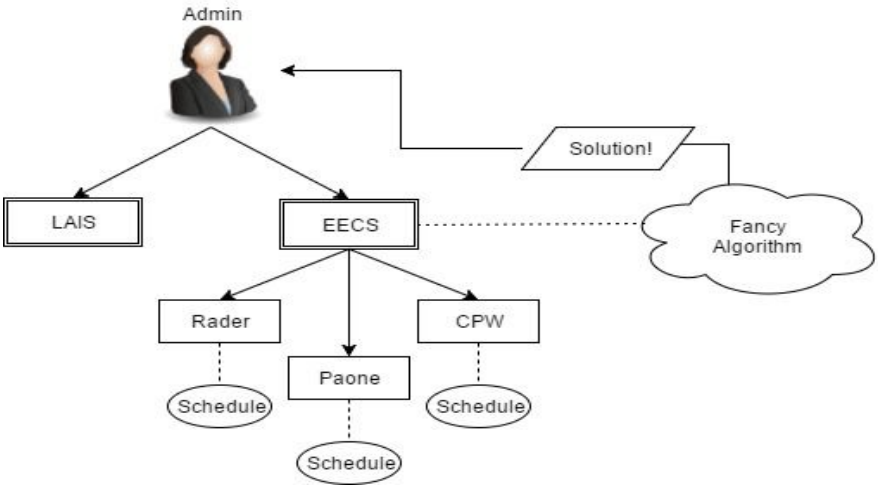

Figure 2.

CSS was the last of the languages used to complete the web site. CSS is a styling language that works with HTML to provide style and aesthetic appeal to webpages. We used CSS to style the text and background of every page. We also used CSS to create our stylized buttons and text boxes throughout the website.

As for the website itself, the design is quite simple (Figure 2). An admin can create multiple groups. All of those groups contain users who input their schedules. Once the admin is satisfied, they can process schedules. The data is sent to the C++ algorithm (explained below). Once the algorithm has finished, it sends the results back to the admin who can do what they would like with the information.

The C++ algorithm reads teachers' schedules, preferences, names, and emails from a text file then attempts to find a solution that allows all of these teachers to participate in teaching triangles. If no complete solution exists, it will generally produce ten unique solutions. In cases with smaller groups, some of the incomplete solutions produced will repeat. It will also produce incomplete solutions if no solution is found within a certain amount of time, three hours by default.

We wanted the algorithm to be as fast as possible while still adhering to both hard and soft constraints. We divided this into three major steps: parsing the data from the input file, determining which instructors can be paired, and forming a solution from the instructors.

The data parsing is fairly simple. It reads the data line by line, and each line has several major sections delimited by "!,". These sections are, in order, the instructor's name and email, constraint 1, constraint 2, etc, preference 1, preference 2, etc. The constraints and preferences are internally delimited by "!.". Constraints contain a name, start time, end time, days of the week, and a cost. The cost is used to determine how important a constraint is. Constraints with a cost of 5000 are priority classes, or classes that the professor would like to see reviewed. Constraints with a cost of 1000 are ordinary classes, which cannot be cancelled in order to form triangles. Constraints with lower costs are soft constraints, such as office hours, which could be rescheduled. The algorithm supports constraints with negative costs as well, but these are not used by the website. They could be used to show times when instructors prefer to work. Preferences contain an email and a cost. The email is the email address of the person who the instructor would prefer to or not to work with. The cost is just like the constraints, with positive costs for people to avoid and negative costs for people to prefer.

The cost is used when generating pairs. The algorithm iterates over each instructor and each other instructor. It then checks to see which of the first instructor's constraints conflict with the second instructor's priorities, and adds their costs to a total cost. It then repeats this process using the second instructor's constraints and the first instructor's priorities. If these conflicts occur on multiple days of the week, the cost is added multiple times, once per day of conflicts. It then checks to see if either of the professors have a preference for each other by comparing the email in the preference to the email stored for the professor. The reason we chose to use email is because different professors may enter the same name differently. Some people may choose to use titles or abbreviations or misspell part of a name, which is much less likely with an email address.

The solution generation is the longest and most complex part. It is implemented with a recursive function that begins by iterating over the list of instructors. This list is sorted by these

criteria, in order (if there is a tie, it moves to the next one): bias, number of connections to other instructors, and finally email address. Bias is used to make the ten incomplete solutions different from each other. Starting with the lowest number of connections allows the algorithm to use the instructors who have the fewest possible triangles first. This makes the algorithm much more likely to find a complete solution in one of the early iterations. It then sorts all connected instructors by their scores, and iterates over those instructors. In the next layer, it iterates over the second instructor's connections, making sure they have a connection to the first instructor before summing the second and first instructors' scores for the third and sorting them. If there is a number of instructors that is not divisible by 3, then it runs another layer and attempts to make a group of 4. In either the group of 3 or 4, it then forms a triangle, creates a copy of the set of instructors and removes the instructors in the group of 3 or 4 from it, then recurses using the reduced set of instructors. If it finds a complete solution, it returns that solution, otherwise it continues until it exhausts all possibilities. If the recursive call did not find a complete solution, then it continues iterating until it finds one.

Decisions

Language Choices:

We debated between Java, Python, C++, C#, and Ruby but C++ was chosen for its versatility, ease of running on windows machines, and widespread accessibility. In addition, C++ was the only language presented that all group members had had some experience in.

For the website, PHP was chosen as the easiest way found to implement sending data to the algorithm via form entry into a text file. HTML and CSS were used by necessity to implement the visual portions of the site. Some Javascript was used in order to dynamically generate our scheduling page.

User Schedule Storage:

User schedules are stored as text files. This decision was made in order to maintain simplicity. The goal is to create easily maintainable and readable files. Additionally, the algorithm uses text files as its input making it much simpler to store user info as a text file.

Roles:

Admins create groups that will contain a group of schedules to be processed. Admins can have as many groups as they would like. Admins also have the ability to create additional

groups, delete users from a group, edit specific users' groups, and finalize the process by generating the schedule.

Users are given a group ID and will go in and fill out their schedule. They will use the group ID so the schedule will be added to the correct group. Users can enter multiple groups. Users have the ability to create a schedule. They can also come back and reedit their schedule at a later date using our load schedule feature.

Results

Performance Testing Results:

Our stress tests proved to be successful. We found that with a possible solution, the algorithm can complete in a matter of seconds. However, if there are no perfect solutions, the algorithm could take hours. We have set a cutoff that, after three hours, the algorithm will terminate and generate ten incomplete solutions. These tests were performed with over sixty instructors, a sample far above the predicted group size.

Results of Usability Test:

Feedback from our real-world beta test, where the website and algorithm were tested by several faculty members of the EECS department, was received and implemented. Many of them were very pleased with the final product. Some suggested that we make it easier to navigate the site. To accomplish this, we added back buttons throughout the site to help users navigate to previous pages.

Features Not Implemented:

Trailhead integration was not implemented due to time constraints. The client had indicated this was the least preferred feature to add. Trailhead integration would require adding an entire suite of security features to the website, and implementing a full login system. Logins are not needed for non admin users and data can be entered via form submission. In addition, passwords are not currently encrypted, and stored in plain text.

Future Work

Other user preferences apart from preferred professor could be added. In addition, allowing members from 2 groups to form a triangle across groups could be added. Trailhead could be integrated to allow easy login and retrieval of schedules. The algorithm could have significant time optimizations at the cost of much higher memory use if it pregenerated triangles rather than pairs, then used triangles in the solution generation.

Lessons Learned:

Extensive amounts of HTML and PHP were used in this project, which our group had little experience with. We also learned about algorithm optimization as our algorithm is very complex, so we have designed it to run as fast as possible. We learned that developing code to run on a virtual machine is quite difficult, as usually it will have a unique file permissions setup. Generally G++ can be used to compile C++ code to run on a Linux environment, but this is very machine specific, and requires ensuring that all makefiles will produce functional executables across machines.

Appendices

Product Installation Instructions

Once the code is handed off, installation should be simple. Just set up the server, and keep all the files in their original folders that they appear in in the original turn over. However, if during useage, if something seems amiss, just delete the file corresponding to the group and recreate it and it should be fixed. In addition, the file path on line 35 of algorithm.php (currently /home/ubuntu/workspace/algorithm/data.dat) needs to be updated to match the file system. The link in the default email also needs to be changed. It is located on line 304 of adminPage.php. The makefile located in the algorithm folder will likely also need to be run to recompile the algorithm. If you need additional support you can contact Tyler Quast at tylerquast@mymail.mines.edu.

Development Environment Description

The website and the server were both built in the Cloud9 IDE. This IDE was great for convenience. Because we used this service, our code will not come with a prebuilt server, as we used the one provided in Cloud9. For the algorithm we used Visual Studios. Visual Studios was great for organization and ease of use, providing many keyboard shortcuts and auto completes.