Andrew Granger, Jarad Hosking, Zihao Jia, David Jonson
Dr. Mahdi Roozbahani

Final Report — Simple Schedule

Introduction & Problem Definition

Figuring out a class schedule can be one of the most troublesome and painstaking processes students deal with each semester. They have to wade through hundreds of classes and thousands of sections, search for objective and subjective data on professors and classes, and ensure there's no time conflicts. Simple Schedule hopes to streamline this process, automatically finding non-conflicting schedules optimized by a 'score' system we create. Several possible schedules will be presented to students along with the score, which accounts for grade point average (GPA), time of classes, rate my professor reviews, etc.

There are several apps available to aid schedule creation, but they are often nothing more than GUIs to help explore offered classes. A student still has to manually peruse other tools like Rate My Professor and Course Critique to actually find information about their classes. Our approach is different from other applications in that we will integrate data from multiple sources like Rate My Professor (scrape), Course Critique (API), and Oscar (API) to allow the user to make scheduling decisions based on desirable metrics. For example, a student may want to minimize the number of classes that occur during the afternoon, maximize the average GPA of the classes, and also only take between 13 and 16 credits. Our system will allow for this along with many other schedule tuning tools. The system will integrate these features so that students do not need to think about various complex possibilities themselves, allowing them to build the best schedule for potential academic success. The impact of the app can be measured via user studies asking for students' feedback on the app. Potential risks may be bugs hidden in some unimportant features and unexpected issues while trying to access certain data sources. As well the system will have to account for biased review data, ensuring that the information made available to students is as accurate as possible.

For the user, the service is free. All data we access is available for free and the only cost involved is the production time to create the product. We estimated that we would have a functional build available after a month of work and a fully featured app functioning by mid-April. In order to maintain pacing on this project, it was important to set up a midterm and final exam of capabilities to ensure that the project was a success. Our midterm exam was a test of the system's ability to generate all possible schedules for a student's specific needs. It was crucial that we met this goal to provide a strong foundation for future development. Our final exam focused on ensuring that any additional features and filters had been added and were functional. This included things like sorting by grade point average, creation of aggregate "scores" for schedules, and a variety of smaller features.

Survey

A poor class schedule can drastically change a student's semester and workload, and hence their mental health. Simpson finds students often feel "high levels of stress" due to having "very little resources" when it comes to schedule creation. She believes this adds to the already immense pressure on students and can lead to dangerous reactionary coping mechanisms such as self-medication. Ma argues making scheduling easier would even lead to higher retention rates for engineering students, and that a good schedule greatly increased working efficiency.

Our app will not single-handedly solve mental health, but any process to help automate a stressful part of a student's life is a useful one.

To better understand schedule formation, it is important to understand the other side of how schedules are created. In "Stochastic Optimisation Timetabling Tool for University Course Scheduling," the authors note that the problem universities face in order to create a timetable that incorporates all classes for a given semester is NP-hard, and so a variety of different approaches to avoid the exponential nature of compute time need to be incorporated. This type of problem solving is rather applicable to our project, and may involve a similar diversion to different approaches. Kuldeep summarized some effective timetable generation algorithms involving linear programming. Moreover, the implementation of an information system which could generate automated timetables is given.

To create a subjective understanding of student opinions on professors, we plan to perform sentiment analysis on reviews from Rate My Professor. Pang and Lee discuss the various techniques available for NLP and the viable issues that arise. They argue the phenomenon of insincere and biased reviewers does not greatly harm data on the assumption that all subjects are relatively affected by such issues at the same rate. Regardless, addressing these issues is helpful. In addition to biased reviews, Liu classifies three types of "spam" reviews: fake reviews, reviews about too broad of a subject (e.g. "I love Georgia Tech" as opposed to "I love Dr. Roozbahani"), and non-reviews (e.g. advertisements, no opinions). We can easily identify the latter two types of spam by searching for external links, running sentiment analysis and finding neutral opinions, and determining subjects of the text. Liu notes the difficulty in detecting the first type (fake reviews), but for the most part we will not concern ourselves with these, since they are unlikely on a platform such as Rate My Professor (professors are unlikely to spam fake ratings about themselves). Greimel-Fuhrmann's paper on student review behavior, explores this further, analyzing the ways in which students evaluate professors. They assert that positive reviews for professors are much more often a reflection of good teaching behavior, and in fact less influenced purely by what grade was received. This gives credence to our Rate My Professor data, potentially mitigating one of the major sources of bias for or against professors. As an aside, we argue that despite potential negative bias in RMP reviews, the information we gather from it is still useful. Some professors, dubbed "SET (Student Evaluations of Teaching) Deniers," argue against the utility of SETs, but as Carlozzi found, these professors on average have lower scores from services such as RMP. So ironically, there is bias amongst those claiming bias in the SETs. Granted, there are often low sample sizes in RMP and the usual issue of often having a bipartite group of reviews (extremely positive and extremely negative). Goos and Salomons discuss the drawbacks of SETs and their inaccuracies (primarily the result of low sample size). As such, RMP will of course not be the only factor we take into account when ranking our class schedules. As discussed before, location proximity, time, and average GPA are some other factors to consider. Further, we can use classification and regression trees to help work with incomplete data, as described by Loh. These can be used to determine average sentiment/opinions for professors, even when data is sparse. Further, we can weigh how much RMP factors into our overall schedule score by weighing in the sample size / number of responses. Most of the data processing / backend will be done using Python.

To create an attractive app, it is important to design a good user interface (UI) which facilitates interactions between the users and our app. Debasmita has identified different issues in designing an efficient UI. This paper could help us to avoid making mistakes in UI design. Moveover, Guntupalli introduces the strategies used for successfully designing UIs and makes suggestions for UI designs which serve as a good guide for us. A great deal of the visualization will be done through Javascript and D3.

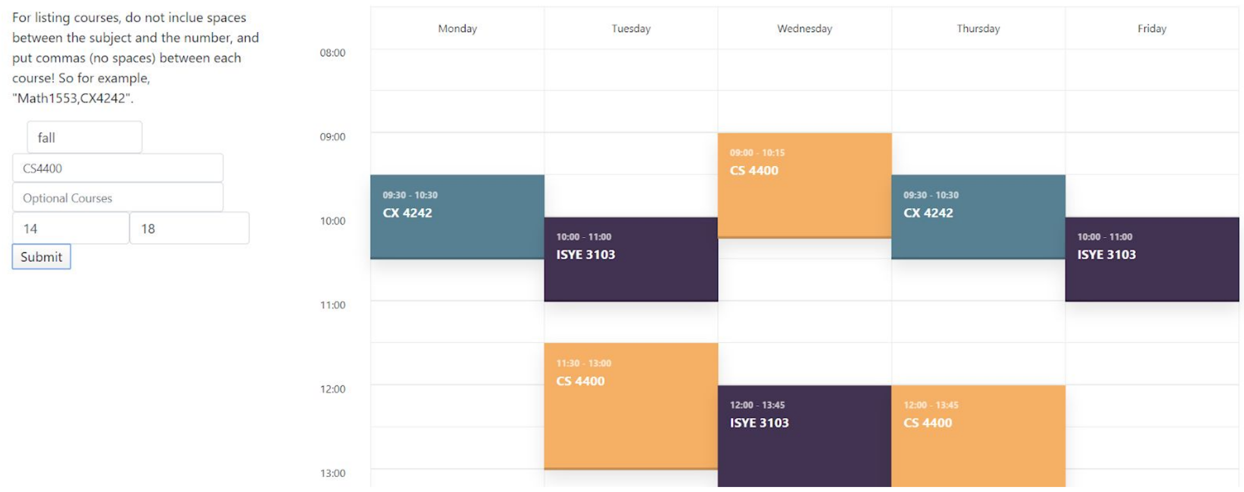
Proposed Method

It is crucial to understand why our system is better than its many similar counterparts. As previously stated, there are several apps available to aid in schedule creation, but they are often nothing more than GUIs to help explore offered classes. A student still has to manually peruse other tools like Rate My Professor and Course Critique to actually find information about their classes. Our approach is different from other applications in that we will integrate data from multiple sources like Rate My Professor, Course Critique, and Oscar to allow the user to make scheduling decisions based on desirable metrics. The system will integrate these features so that students do not need to think about various complex possibilities themselves, allowing them to build the best schedule for potential academic success. We anticipate that this application could reduce schedule creation time as much as 50%, while also ensuring that a student is apprised of all their possible schedule options. In order to better highlight how our system is different, we felt that outlining an explicit list of innovations would be useful. Our innovations are as follows: Combines a variety of data sources to give students all streams of information in a digestible format, Creates the schedule for you to save on time, Integrates a scoring system that allows for confirmation of best schedule creation.

To consolidate data for the UI, we first had to scrape the data from Rate My Professor, Course Critique and Oscar. Initially we planned on using an API for Course Critique and Oscar, however scraping turned out to be easier. To download the desired data from Rate My Professor and Course Critique we used Python's requests, BeautifulSoup, and regex libraries to collect and clean the data. After collection we used Python’s pandas library to consolidate the data into one large data frame that was exported to a JSON file that is easily integrated with JavaScript for the front end. We initially thought retrieving live data from Oscar using JavaScript client side would be a straightforward task, however we encountered a seemingly unavoidable CORS error. This error arose from an Access Control setting disabled on our local server. Without technical knowledge of managing requests server side, our team faced major delays. In an attempt to complete a proof of concept, we abandoned the idea of scraping oscar live and decided to revisit data collection using python to create a static JSON file with limited courses. After creating this file we encountered more unexpected problems, specifically with professor names and online courses. The helper classes we created to reflect the framework of our UI assumed that courses would not be online and assumed that professors' names across Rate My Professor, Course Critique, and Oscar would align. To overcome this hurdle we decided to revisit data collection and restructure our JSON files to allow for smooth creation of the schedule objects.

To find feasible schedules, we first take in a few parameters as user input: the classes they MUST take, the classes they want to take (although they do not necessarily have to take all of them), and the range of credit hours they are comfortable taking. At this point, all possible combinations of their required courses will be found. Our program has a “Schedule” class which contains “Section” objects. To find possible combinations, we initialize several schedules, each with a single section (one for each possible section of each possible required course). From there we attempt to add sections of other required courses (checking to ensure there is no scheduling overlap) to each schedule. This is recursively done for all required courses, and the function returns a list of valid schedules. Next, we figure out possible combinations of optional classes the user wants to take such that they take their desired amount of course credits. These combinations are then run through the same function as earlier, and a final list of possible schedules is created. The Schedule class keeps track of the expected GPA for that given schedule, the times and locations of each class, and the ratings of each professor, all of which are used to assign a general "score" to each schedule so they can be ranked. Sample size of professor ratings is accounted for to ensure a small response rate does not lead to score deflation or inflation. The user is then given a list of all possible schedules that fit their initial request.

The user interface is in the form of a downloadable web app developed using the d3 knowledge and skills we have accrued throughout this class. It displays a user's agenda Monday through Friday with times on the left (as featured below). On the left sidebar, a user may enter the semester they are planning out, the classes they need to take, the classes they want to take, and the credit hour range that they are comfortable taking. Ideally upon submission, either an error message pops up (e.g. "No possible schedules found!" or "No classes entered!"), or the schedule with the highest score appears. The user should then have been able to flip though a few different schedules, sorted by highest score to lowest score. We capped the limit of possible schedules at ten so as not to overwhelm the user.


For listing courses, do not inclue spaces between the subject and the number, and put commas (no spaces) between each course! So for example, "Math1553,CX4242".

fall

CS4400

Optional Courses

14 18

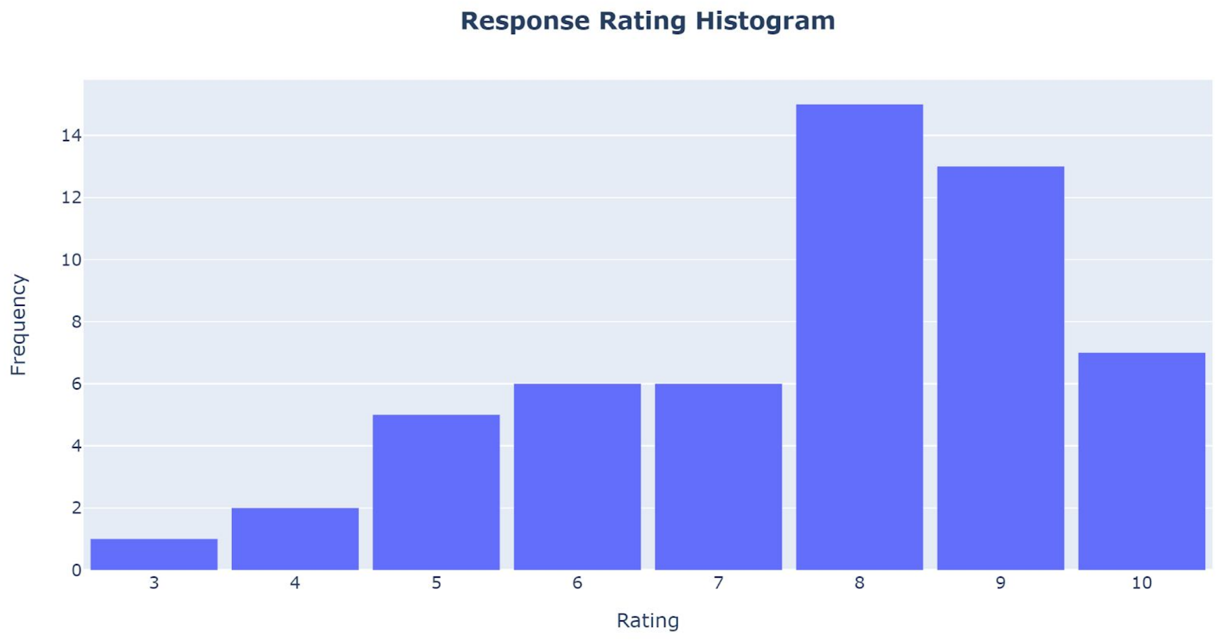
Submit

With regards to the division of labor, Jarad and Andrew developed the back-end of the system, David handled data collection and cleaning, and Zihao developed the front-end UI. It is the opinion of the group that this distribution of work was fair and even, and that all team members have contributed a similar amount of effort.

We ran into a significant amount of issues due to the COVID-19 outbreak. Of course, none of us were able to meet in person, but to further make things difficult, Zihao/Roger lives in Singapore, Andrew was intermittently on the west coast, and Jarad and Manny were in Georgia which made meeting, even virtually, quite difficult. Roger also fell ill toward the end of the semester and was admitted to the hospital. This made communication difficult, and at the end we were unable to fully integrate the back end with the front end. After contacting Dr. Roozbahani, he advised us to not stress significantly about the final product, and that the progress we have made was important in and of itself. With extra time and better conditions, we of course would have integrated the back end and front end, and we would have a server running to allow us to have dynamic data and live-scrape oscar as opposed to working with a static subset of data. We have finished the algorithm that finds feasible schedules, the metric that scores schedules, and the function that sorts schedules. We have also finished data collection, and the script that creates objects from our data necessary to run the said algorithms. Lastly, we have completed the visual aspects of the UI, although it is currently independent of the back-end system.

Experiments & Evaluation

One of the first things we tested was whether or not people would actually use this product. During the initial development phases of our project, we created a poll which was completed by a variety of students at Georgia Tech. We reached out to students in class group chats we're in, fraternities, and friends. They were provided with a brief description of our product and were asked, assuming the product works as advertised, to rate on a scale from 1 to 10 how likely they would be to use our product. In this questionnaire a 1 represented a student who would be very unlikely to use the system and a 10 was a student that would be very likely to use it. The results were very encouraging, with 54 students responding with an average score of 7.648. This data is visualized below, and the raw data is available in the appendix.



Testing occurred constantly and thoroughly throughout development. Any code change meant, of course, that at least some minor tests were run. So we ensured, in small experiments, that all our python classes could be initialized as expected, all our equality and string-printing equations worked as expected, our UI inputs worked correctly, etc.

On a broader scale, our larger experiments ensured that the whole architecture was designed correctly and that our greater algorithms ran smoothly. We started with a slightly smaller dataset, starting off with just ISYE classes. We evaluated our results with smaller test cases of theoretical schedules involving just ISYE courses. Certainly, some bugs were expected. The primary sources of these bugs came from (i) missing certain possible schedules (ii) undesired/incorrect scores for schedules. Error of type (i) were mostly caused by small recursion and looping errors in our algorithms, and errors of type (ii) were not even necessarily errors but just unexpected consequences of the way our scoring algorithm worked, which we modified as necessary (by weighting things like GPA and rate my professor scores/sample sizes differently).

We then added in the rest of the courses/sections (including non ISYE courses). We next ran some experiments by putting in our own classes for the semester to see what schedules the program recommended. Indeed, we found our actual schedules somewhere in the results, but beyond that, we were able to compare the schedules it offered and the score it gave each of them. As discussed before, we toyed with the scoring metric until we considered it agreeable to our subjective opinions.

Finally, we started testing edge cases. For example, we created error notifications for when no possible schedules were available, and we ensured students could not register for courses on the wrong campus.

Another way we tested and evaluated our system was through a user survey. Upon completion of the system we sent out our program to a sample of students, again through class group chats, fraternities, and friends. After interacting with our system, we talked with each person and asked them to evaluate both the functionality and usefulness of our system. The response was great; people liked what we had created and approved of our product.

Conclusion & Discussion

While the completion of this project was a difficult yet fun process, it was also a little bit disappointing. As mentioned previously, due to the onset of the Coronavirus and subsequent spreading of our team across the Globe the team ran into a lot of issues. First and foremost, one of our group members was forced to return home to Singapore, where he was then on a significant time difference to us. This made collaborative work quite difficult and slowed our progress tremendously. As well, the inability to work together in person made the development of the back-end rather cumbersome. Andrew and Jarad were planning to meet weekly throughout the semester to build the system together, but since they were forced to seperate, the rather intricate system had to be built in seperate pieces, which made things move significantly slower. The final nail in the coffin was Zihao getting sick. Since he was the one tasked with developing the UI, he had been researching and compiling the knowledge necessary to build it throughout the semester. With him unavailable so last minute, the group was unable to finish the integration of the User Interface with the back-end. If we had more time and hadn't been forced off of campus, the team is certain that this feature would have been integrated properly and we could have produced a more complete system.

That being said, the Coronavirus epidemic also provided us with an interesting opportunity to experience how some Computer Science work occurs in the real world. In the future, we may very well have to work on large-scale projects in a partially or fully remote fashion. Having to adapt to this new paradigm was difficult, but also provided us with a unique look into our potential futures working in the Computer Science field.

In the end, our final product was still an impressive achievement. As demonstrated in the Experiments and Evaluation section, the remaining parts of the system work as intended. The user is able to input the classes they need to take and want to take, and the system returns all the possible options and allows for further schedule optimization. While this project was very challenging, it was also quite rewarding. Seeing the system come together over time, and finding all the ways we could use the skills acquired in this class was refreshing. Overall this project was a positive experience, and provided an opportunity to utilize our skills that many other classes can't offer.

Works Cited

Bettina Greimel-Fuhrmann & Alois Geyer (2003) Students' Evaluation of Teachers and Instructional Quality--Analysis of Relevant Factors Based on Empirical Evaluation Research, Assessment & Evaluation in Higher Education, 28:3, 229-238

Carlozzi, M. (2017). Rate my attitude: research agendas and RateMyProfessor scores. *Assessment & Evaluation in Higher Education*, *43*(3), 359–368.

Chaitanya, Guntupalli. (2008). User Interface Design: Methods and Qualities of a Good User Interface Design.

Goos, M., & Salomons, A. (2016). Measuring teaching quality in higher education: assessing selection bias in course evaluations. Research in Higher Education, 58(4)

Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, *5*(1), 1-167.

Loh, W.-Y., Eltinge, J., Cho, M. J., & Li, Y. (2018). Classification and regression trees and forests for incomplete data from sample surveys. *Statistica Sinica*.

Ma, Shuai & Akgunduz, Ali & Zeng, Yong. (2015). COURSE SCHEDULING ACCORDING TO STUDENT STRESS. Proceedings of the Canadian Engineering Education Association. 10.24908/pceea.v0i0.5849.

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, *2*(1–2), 1-135.

Pongcharoen, P., Promtet, W., Yenradee, P., & Hicks, C. (2008). Stochastic Optimisation Timetabling Tool for university course scheduling. International Journal of Production Economics, 112(2), 903–918.

Sandhu, Kuldeep. (2001). Automating Class Schedule Generation in the Context of a University Timetabling Information System.

Saha, Debasmita. (2015). User Interface Design Issues for Easy and Efficient Human Computer Interaction: An Explanatory Approach. Jan. 2015.

Simpson, S. (2018). Murray State's Digital Commons Stress Triggers, the Effects Stress Has on Social, Mental and Physical Behavior in College Students, and the Coping Mechanisms Used.

Appendix

Raw Likeliness to use survey results:

8,9,8,8,6,9,10,5,8,6,8,7,7,9,10,4,8,9,3,5,6,5,8,9,10,9,7,9,10,6,8,5,9,9,8,9,9,8,8,5,6,9,10,7,8,7,8,6,
9,10,10,8,7,4