# Winter 2019 Graphical Response Report for EECS 398-001: Special Topics (Nicole Hamilton)

Project Title: **Winter 2019 Teaching Evaluation**

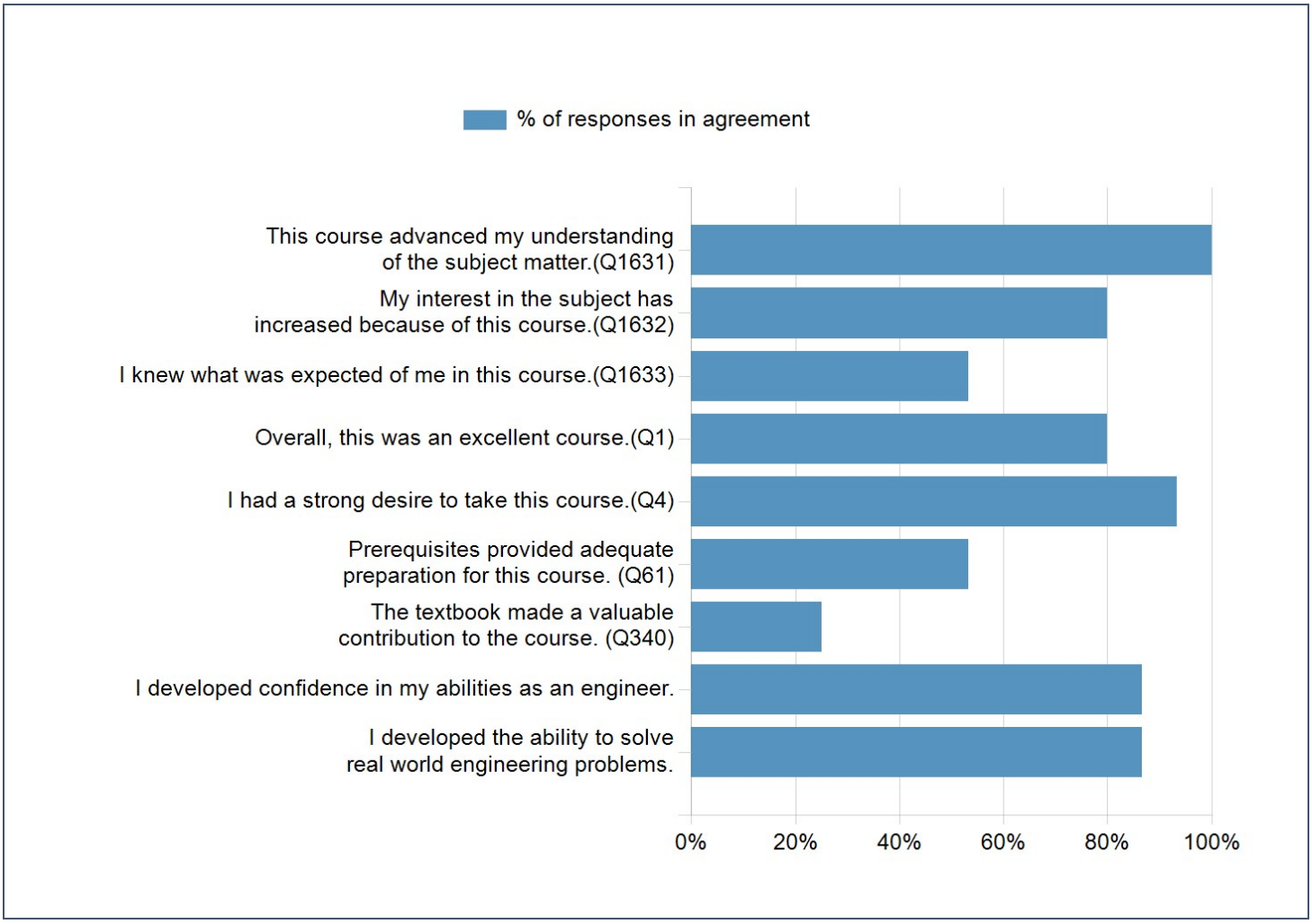
Course Audience: **61**
Responses Received: **15**
Response Ratio: **24.6%**

**Report Comments**

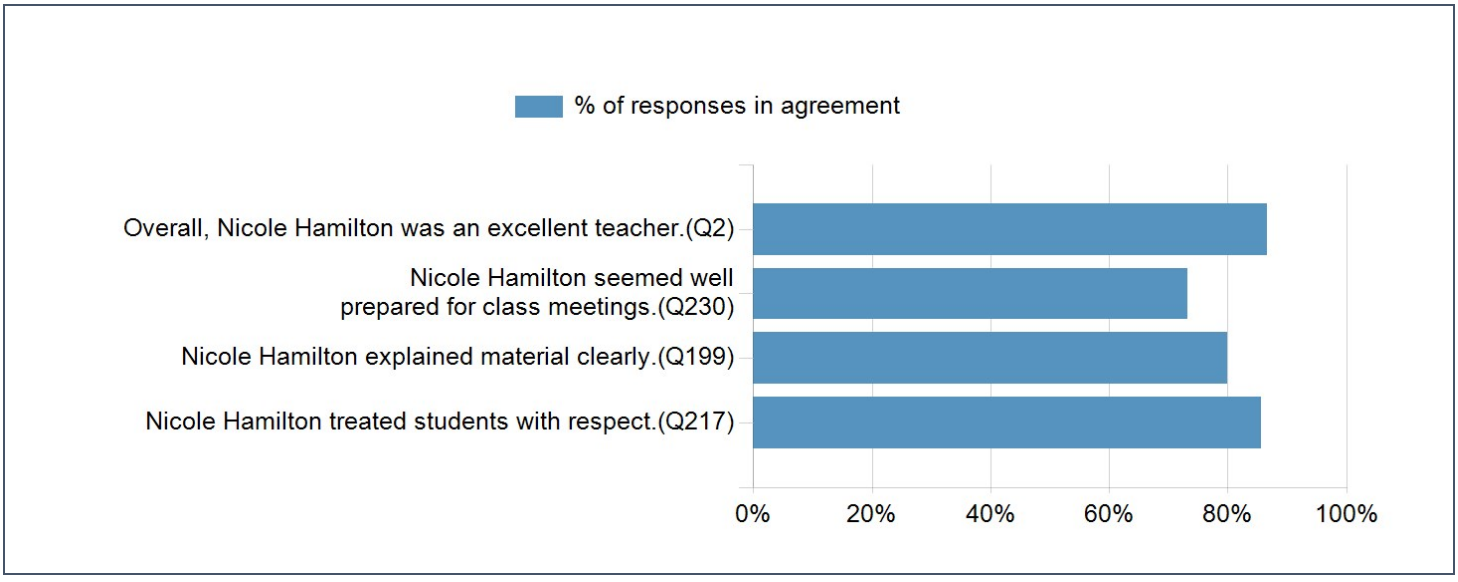
This report is a summary that tabulates the percentage of "Strongly Agree" and "Agree" answers for the quantitative rating questions that appeared on your evaluation. Any questions using an alternate scale or any questions added by the instructor appear after the main charts. Responses to the open-ended questions appear at the end of the report. Ratings are from the Winter 2019 teaching evaluations of EECS 398-001: Special Topics.

Prepared by: **Office of the Registrar**
Creation Date: **Tuesday, May 7, 2019**

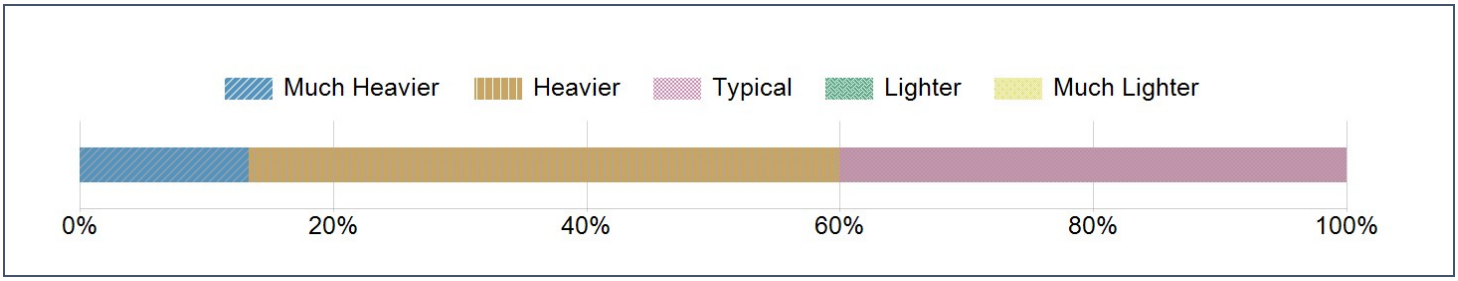
## Responses to questions about your course:




## Responses to questions about the instructor:



## As compared with other courses of equal credit, the workload for this course was:




## Responses to Open-ended Questions

### What were the strengths of the course ? (Q953)

| Comments |
| --- |
| The labs were extremely helpful. The lectures were interesting and the instructor was enthusiastic and open to answering questions. Taking this course exposed me to doing large system design projects with large groups of people. |
| It's cool learning about the parts of a search engine. Also the labs were very helpful in teaching content. |
| Professor Hamilton's real life experience gives great insight into what it is like to work in the real world. The class is hard but gives you something to show during interviews and is the reason I have a job now. |
| The project itself and the lecture material that went along with it provided a great opportunity to learn how to build a search engine from scratch.<br><br>My strategy for projects previous to this would be to combine existing libraries. This is something that I learned in EECS 485. However, I do feel that Professor Hamilton's philosophy of doing everything from scratch forced me to be more comfortable in doing that, which I think is valuable for the kind of engineer I want to be.<br><br>Having never taken 482, I found the operating systems topics covered in this class to be very useful. |
| The project. Absolutely made this course worth it. I admit, I was feeling significantly regretful at one point for taking this class. As time went on, and our search engine wasn't coming together they way we wanted it to, I was feeling frustrated at the lack of helpful direction provided, especially when the lectures covered material that was completely useless to what we actually needed. Near the end of the semester, however, I looked back and realized all of my accomplishments this semester. I remember feeling proud of having achieved so much in such little time. For that simple fact, I feel this class was worth it. |
| The biggest strengths are the concept of the course itself, the fact that a career engineer is teaching it (knows what matters in real life), and the fact that Nicole is willing to listen to what students think (this has become much more evident this semester). |
| The labs and office hours |
| The project based focus of the course as well as the significant industry experience of the professor.<br><br>In this course, practical application is the single most important facet of the lectures. |
| The topics covered in this class are practical and applicable to industry. The class also provides good experience in working with large–scale projects. |
| The course provided a superb opportunity to architect, build, and test an entire system from scratch. This is incredible important skill in industry and was important to practice in course work. Not only was it challenging and engaging but the end result of the course (a functional search engine) made the entire project extremely rewarding. |
| The strengths of the course were the labs, the homework assignments, the availability and willingness of Professor Hamilton and the IAs to meet in office hours and discuss design and strategy, and the project itself. The milestone reviews were helpful in keeping us on track and discussing our progress with Professor Hamilton. |
| The course covered a strong base curriculum while allowing students to go above and beyond in various disciplines. |
| Getting experience with a large programming project. Learning a lot of low–level C++/C. |
| I think the idea behind the course and the actual problem itself are very interesting. Making a product that people can talk about technically and nontechnically is a great thing for interviews. I also think the labs were incredibly helpful and made those parts of the project that were covered way clearer. |
| The labs are excellent and help students get a better understanding of the topics in lecture. |

## What suggestions would you make for improving the course ? (Q955)

| Comments |
| --- |
| I would suggest clearer guidelines on where students should be in the project at different parts of the semester. Our team definitely had trouble with time management and ended up doing a lot of work towards the end. I also suggest developing the project management aspect of the course a little better, and maybe swapping out Gantt charts for scrum boards, or something more widely used in industry. I realize that it is not the most interesting part of the course, but it is still important for people to learn in their MDE's. |
| Let us use the STL when we want to, implementing classes is a huge time sink and it would have saved us a lot of time and stress. |
| It was a bit raw, could use more homeworks and labs. |
| There a few issues that I had in this course.<br><br>I had a difficult time knowing what was expected of me<br>– Announcements were done through Piazza. I like this, however, the important announcements need to bypass student notification preferences. Announcements were not going to my email inbox, and I would frequently miss them.<br>– I was also late to notice Canvas assignment postings such as the presentation slides. Ideally I'd like to get emails for important things.<br><br>Large teams for undergrads are very difficult to manage<br>– I like the idea of this class having less structure. However, I think it's a difficult structure for 6–person teams.<br>– People spend years learning how to manage teams of engineers in a workplace setting where it's everyone's job to be available 9–5 during the week.<br>– In EECS it's rare to have more than 2–3 person teams for a class. We learn to manage these teams okay. Six–person teams is an entirely different ballgame. We all have busy conflicting schedules and it was too hard to manage effectively.<br>– Because of the challenges described above, I think our hands need to be held a little bit more. Perhaps this means more concrete deadlines (e.g. this many pages crawled by this deadline). Perhaps, to help us with this, instead of 4 hours of lecture per week some lecture time could be dedicated to time for teams to be meet in person together. Maybe more labs could cover more of the search engine components that don't vary so much in functionality and implementation, allowing us to spend more time on parts that involve design decisions.<br><br>Labs<br>– Labs covered difficult content and students should have been given more time to complete them. A single weekend was challenging to manage, especially when we sometimes weren't sure when the labs were going to be. |
| More labs! Less assignments such as Gantt chart or simple coding exercises. Rather, introduce the concepts of the search engine through labs. The labs we had were extremely helpful when developing the project components. |
| Most of my suggestions haven't changed since last time (making 370 required, not requiring students to implement standard library containers and utilities or just using C), but I do have a couple more. I feel that teams would heavily benefit by having hard deadlines on some of the parts of the project, such as having to give a demo of their ability to crawl the web and parse HTML pages by say, halfway through the semester. Then with four weeks left they should probably be required to show at least the beginnings of an index / query compiler / front end. Additionally, I think you kind of confused some people about how grades would be determined. At the beginning of the class, you seemed to stress that the class was "competitive". I think this made a lot of people feel that only people who "beat" everyone else could earn a good grade. As the course progressed (especially near the end), you have taken a more lax attitude towards the competitive spirit of the class. I don't think that this class needs to be competitive to motivate people to put their maximum effort in. I think you should say something to the effect of "if you put in a lot of effort, and contribute a large amount, you will do well in the course." Therefore, I think you should make it clear that this class is not a competition – it would probably lead to you having far fewer headaches when having to deal with students who are very concerned about grades. |
| Went more in depth in final essay but move labs up to first month and align teams based on what they want to get out of the course |
| I would recommend allocating Amazon EC2 compute time for each team in order to equalize resources available to each team, especially network speed. |
| As I'm sure others have mentioned, evenly distributing the workload over the semester is something to work toward.<br>I think the code in lecture slides is helpful, but not as lecture material. I think it would be better used as reference for labs or small scale components that can be incorporated into the search engine. |
| See my personal essay for details. Generally more transparency on what pieces of the STL we could use (as there were some just not a lot). Also getting people started earlier on the Crawler. Also more labs and "hands on learning" outside of the capstone project itself. |
| Here are my suggestions, in no particular order:<br>1. Work to get teams to coalesce and develop a unified vision for their project at the beginning of the semester. Meeting with teams right after team photos/the project plan is due would work wonders in keeping teams on track.<br>2. More labs and homework assignments.<br>3. Take away Windows API calls except for gracefully failing with debug information when C++ exceptions do not suffice.<br>4. Don't present code in lectures, have students develop the code themselves in labs and homeworks and keep the lectures more |

| Comments |
| --- |
| conceptual.<br>5. Try to cover all of the conceptual material necessary for the project by the midterm, then make the midterm test everything that lectures have conceptually covered to see how well students learn the material and ensure everyone is on the same page. (It may be necessary to move the midterm date fairly late into the semester to make this possible.) Then, don't have a final exam — it is too much to have presentations, final code submissions, final reports, individual reflection essays, private team demos for just the professor, and a final exam all in the last two weeks of class as teams are struggling to finish up the project.<br>6. Keep students up to date EVERY WEEK, or at least very regularly, on where they are should be progess–wise at this point in the semester. This would help avoid my team's scramble to finish in the last three weeks of the class.<br>7. Change the order of topics to let teams get to work right away and provide suggestions on what they can do from week 2 of the semester. (Teams could start beginning to parse HTML then.)<br>8. Have some larger–scale, team lab assignments that involve downloading webpages or extracting links, as well as keeping the other, individual lab assignments.<br>9. Encourage teams to work on data structures they want to use from the beginning of the semester, and clarify several times what is appropriate versus non–appropriate use of the STL. This became clear to everyone eventually, but some were confused until relatively late in the semester.<br>10. Let team members rate each other confidentially, and award all team members the same grade for the project unless there are significant deviations in terms of team members who contributed far and above from the rest or those who were free riders. Merging the team and individual components of grading could help promote more unity among team members and less competition to get the most LOC. |
| The course had a few kinks due to pacing of lectures and assignments, but these will be smoothed out over time. |
| I know this has been said multiple times before, but I really think students should be allowed to freely use the STL in this class. While I do agree that there's benefit to learning how to implement or customize STL classes, I think students have enough on their plate trying to implement an entire search engine. For example, my group dealt with a lot of memory errors because we used C strings instead of STL strings. We could have spent all that debugging time on implementing cool search engine features. Just my 2 cents. |
| I think that teaching each section very separately and making it clear when people should be able to start and finish parts of the project would be helpful. The schedule online said everything should be started way sooner than we actually had the knowledge to do so, and that resulted in our team being super stressed and trying to establish an index structure before that had really been covered. |
| I've discussed these in office hours with Professor Hamilton, but I'll list some suggestions here as well. More labs throughout the semester, less giant chunks of code on the lecture slides, fixing the style guide, allow some use of the STL and have that defined clearly at the start, and way more progress reports/check ins with the teams. |

**Comment on the quality of instruction in this class.**

| Comments |
| --- |
| I thought the instruction in the course was generally quite good. The lectures were engaging and clear, and the instructor was always willing to answer questions we had. |
| Hamilton is a cool person who genuinely cares about students, but I don't think she's a very good teacher. That's understandable for someone who's still relatively new to teaching. She's also very receptive to feedback which tells me that she's invested in making this course the best she can. |
| Professor Hamilton is unique among instructors at U of M. She comes from industry instead of academia, and with that comes a different style of teaching. It takes some time to get used to, but gives a feel for what it is like to work in industry. If you want to go into industry instead of grad school I recommend this course. |
| I found the lecture content to be very well–done. It was fairly comprehensive and clear. I think Professor Hamilton did a great job of leveraging her expertise here.<br><br>Overall, I'd say this course is clearly a work in progress. However, I respect how welcoming Professor Hamilton is to feedback and can definitely see that big improvements have been made from the first semester. I'm grateful for all of Professor Hamilton's efforts in ensuring that her students have a good and productive experience. |
| COuld be more organized. Lectures need more structure in my opinion. |
| Instructional quality is very high in my opinion. Some parts need more coverage (crawling, html parsing), but overall you explain concepts very well. I thought the RAs did a great job as well. I think the class can benefit from having labs earlier and by potentially having more of them. |
| Quite repetitive, but insightful. Did not like the large amount of code on the slides |
| The professor was superb. Her real world experience is a breath of fresh air compared to almost every professor I have ever had at Michigan. She seems to care about the students more than my other professors have. In other courses I felt like a student ID and a grade. Professor Hamilton lectured one of my favorite courses ever at Michigan.<br><br>The course content is relevant to software engineering, unlike almost every other course I have taken here. |
| Nicole knows a good deal about the projects/problems she has worked with. Her insight into industry is something that other lecturers can't always offer. |
| Professor Hamilton cares very deeply about her students and works very hard to teach as well as possible. However, I think that pacing lectures more slowly and not covering the same material twice unless everyone is really struggling would greatly improve the quality of instruction. I expect this course will only improve. The IAs both had wonderful labs, and all of the instructors did an excellent job. |
| The class was very well–instructed. Some lecture material could have been covered more in–depth and slower the first time, but it was great overall. |
| I don't think the issue was ever with the explanations but with the order of material. |
| Professor Hamilton is an instructor that wants to see her students succeed. She fully acknowledges that there will be a lot of issues at the beginning of a course as ambitious as this one, and she has been very willing to listen to student feedback and adjust the course accordingly. |

**Among the courses you have already taken, which proved the most (or least) effective in preparing you for this course, and why? (Q1098)**

| Comments |
| --- |
| EECS 482 was useful because I came into the class already understanding sockets and multithreading. |
| No course really prepares you for the scope of this course. It's a semester long project with no hard deadlines so everything is up to how disciplined you are. |
| No classes I have taken adequately prepared me for this class. 482 and 485 would both help as these topics come up regularly. |
| EECS 281 because of its emphasis on algorithms and C++ programming<br>EECS 280 for its focus on C++ language features |
| 482 was definitely helpful with the operating system calls needed to develop a multi–threaded system, as well as having already developed the intuition to think in a multi–threaded way. Other than that, I think 482 isn't really necessary and this course stands alone and should be very accessible after 281 (and possibly 370). |
| 370 was a must have in my opinion, 482 was extremely helpful, 490 and 483 were helpful for the concepts of parsing and grammars etc, 486 was useful for IR knowledge and more exposure to search specific concepts in this class |
| EECS 281. Did not feel prepared for OS stuff, but 281 gave me confidence to implement data structures |
| EECS 482. EECS 482 covers the operating system constructs required to build a search engine. |
| EECS 482 is almost required for writing a good crawler in a reasonable amount of time. I would suggest to next semester's groups to try and find at least one person who has taken it. |
| I believe 281 was the most effective in preparing me for the course due to the strong emphasis on data structure's development |
| EECS 280 was very helpful in thinking about data structures and knowing the basics. EECS 485, which I took concurrently with System Design of a Search Engine, was extremely helpful in that we learned about processes, threads, and sockets, and learning the same material in two different languages with two different teaching approaches was very helpful. EECS 381 was very helpful in learning how to think about design and to write high–quality code.<br><br>EECS 376 was not directly useful for System Design of A Search Engine. |
| 482 was definitely helpful. Our socket code strongly resembled our 482 socket code and knowing how threads and mutexes worked is always a plus. |
| EECS 281 is the most effective course I've had for preparation, but I also haven't taken any upper level EECS courses besides this one so there are likely others that help prepare for this course. |