

# Guide to CS39Q

Dept. Mathematics and Computer Science

August 9, 2006

## 1 Introduction

CS39Q is the group project course in the Computer Science curriculum. It is a required course for all students majoring in Computer Science. It is intended to be a capstone course that will bring together many of the topics that were covered in the rest of the curriculum. For this reason, we expect that students take this course in their final year, or at least after they have completed all of the core courses of the curriculum.

### 1.1 How It Works

There are no formal lectures for CS39Q, and it is not assessed by the traditional final examination by which most of the other courses offered by the department are assessed. The department announces a date within the first week of registration for a new semester when the course will officially begin with its first meeting. The prescribed schedule of events is set out in Table 1.

Week	Activity	Description
1	Orientation	Students identify groups or indicate that they need one. All groups should be established before the next meeting.
2	Project Identification	Groups identify the projects they are considering. Ideas are discussed and assignments to supervisors are made.
3	Proposal Submission	Each group submits its proposal to its supervisor.
6	Midterm Presentations	Each group makes a presentation of what its project is about, what the objectives are, and what the current progress is.
12	Final Presentation	Each group presents its project and gives a demonstration of its operation.
13	Report Submission	All project reports are to be turned in one week after the final presentations have been made.

Table 1: Schedule of Meetings and Activities throughout the semester.

The above schedule is modified a little during the Summer because there are only nine (9) teaching weeks during the Summer. In that case, the first three items are completed within two weeks (the first two are done on the first meeting, and students use the first week to come up with their proposals. The midterm presentation is moved up to week 5, and the final presentation to week 9. The reports are actually collected about 1 week after the official end of the Summer registration period.

## 2 Assessment

There are three components to your raw project grade.

1. The midterm presentation is worth 10% of the total grade. It is further broken down as follows:
  - 5% for how clearly the project is defined
  - 5% for the progress towards a solution
2. The final presentation is worth 30% and is broken down as follows:
  - 10% for the quality of the presentation
  - 10% for the amount of the project that is functional
  - 5% for the usability of the software developed
  - 5% for the quality of the technology used in the implementation

There is a subjective level of difficulty rating that each grader makes for each project. This rating is on a scale of 1 to 3, 1 being the least difficult and 3 the most difficult. The rating is used to adjust the contribution of the quality of the presentation and the functionality. A project with a difficulty rating of 1 will have 5% contribution from the quality of the presentation and 15% contribution from the functionality. A project with a difficulty rating of 3 will have it the other way around: 15% from the presentation and 5% from the functionality. The rationale behind this is that a more ambitious (therefore difficult) project will have a lower chance of fully working, and therefore would have been more heavily penalised if the functionality counted for 10%. On the other hand, any lack of functionality will have to be compensated for by a good presentation. (Suggestions for improvements on this modifier scheme are welcome).

3. The final report is worth 60% of the total grade and it is divided as follows:
  - 10% for the quality of language used: this is a technical document and the language used should be appropriate. Proper English is expected.
  - 10% for the problem definition and investigation: it should be clear what the project ought to do, why it is useful and what the context of its development is (i.e. what are the relevant technologies to the project, and to what extent were they used?).
  - 25% for the description of the solution or implementation: this includes the design decisions that were made, and any clever ideas that were brought to bear in the implementation of the project.
  - 15% for the functionality, results and analysis: the degree to which the project was a success, how thoroughly it was assessed, and what improvements could have been made with the benefit of hindsight.

## 2.1 Student assessment

At the end of the project, each student assesses himself and his other group members in each of three categories: programming, write-up and general participation. For each category, each group member has 12 points that he/she must distribute among all group members, including himself/herself. Note that the total number of points given by each group member for each category is a constant 12. The total number of points given to each member (including those given by himself) are then totalled to produce a weight. In any sized group (2 to 4), if each member gives all members an equal distribution of his 12 points for each category, then the total given to a single member will total 36 points (12 for each category). The final score for an individual is computed by multiplying his weight divided by 36 by the raw project total.

For example, suppose that the members of a group of 3 have allotted their points as follows:

Member	Wts Assigned By Member 1			Wts Assigned By Member 2			Wts Assigned By Member 3			Total Weight
	Prog	Writ.	Particip	Prog	Writ.	Particip	Prog	Writ.	Particip	
1	6	4	4	5	3	4	6	4	4	40
2	3	4	4	4	5	4	3	4	4	35
3	3	4	4	3	4	4	3	4	4	33

Notice that each column sums to 12, which represents the total number of points that each member had to give out. In this example, member 1 has clearly done most of the programming, though the members disagree slightly on the relative excess that member 1 has contributed to the programming. Note also that in this example, all three members agree that all three of them participated equally well overall (group meetings, software development, design specification, brainstorming, etc). Both members 1 and 3 thought that all members contributed equally to the writeup, but member 2 thought that he contributed a tad more than his group members to the write up. The total weights are shown to be 40, 35 and 33, respectively, for members 1, 2 and 3. Say that the raw project grade is 72%, then member 1 will actually get  $40/36 * 72 = 80\%$  and member 3 will get  $33/36 * 72 = 66\%$ .

The course coordinator, who is responsible for final grades, reserves the right to modify a student's assessment if it is deemed to be unreasonably harsh towards another student. This right will be exercised only in extreme circumstances though, so students should try to collect their final results as soon as they are ready to ensure that they have been treated fairly by their group members.

## 3 The presentations

The midterm presentation is only about 15 minutes long. In that time, you should focus on making a clear statement of what the project is about. In particular, it is important for you to be able to speak to the scope of the problem you are working on (what will you definitely not be doing, what will you be definitely doing, and what might you be doing). You should try to minimize the uncertainty in what you plan to do.

The second aspect of your midterm presentation should address the challenges that you expect to face. This is where you should bring out why the project is non-trivial. Sometimes it is useful to ask yourself "What about this project would prevent a group of average high-school Computer

Science students from being able to do it?” If you cannot identify something then your project is probably aiming too low. At the end of the day, we want to see that you have brought knowledge from the courses in the curriculum to bear on the problem that you are addressing.

Another aspect of the midterm presentation is your progress to date. We do not expect that you would have had much to show for your time up to that point, so we are not looking for examples of running code. However, we do want to know that you have given the implementation enough thought to have been able to identify the areas that might pose a technical challenge. You should note here that your having to learn a programming language is not quite the kind of technical challenge that we are looking for. The aim of asking you to consider the challenges is to get you to engage in problem solving activities where you might exercise knowledge gained from the rest of the curriculum (or elsewhere); the aim is not to get you to produce a list of your personal limitations.

### 3.1 The final presentation

The final presentation is expected to be polished. It should take about 35 minutes in all. Roughly, this should be broken down into 15 minutes of presentation and 15 minutes of demonstration, and questions and discussions during and after the presentation should take about 5 minutes, and not more than 10 minutes.

Note, that you do not have to give a complete presentation and then a demonstration. It might be the case that you give only a brief introduction and orientation to your implementation, and then transition to an in-depth showcase of the features of your implementation that essentially presents what was aimed at, and how well it was met in one fell swoop.

Here are some tips for giving your final presentation:

- In giving your presentation, you should be very careful to say what was accomplished within the first 3 slides.
- Sometimes it helps to produce a screenshot of what was accomplished to illustrate what was intended.
- Rehearse your presentation, try to fit it within the given time limits.
- Do not be embarrassed about self-imposed limitations on the scope of the project. It is better to have a project that works that could have done more still, than to have a project that tried to do everything, but failed to do even one.
- Do not demonstrate anything that you have not verified before (unless you cannot help it, e.g. when data is supplied by someone in the audience). You might be surprised to learn how software can fail in ways that you never anticipated, especially when the need for it to work is high.
- Be excited about your work, and present it in as positive a light as possible. Even if your project does not fully work, emphasize the things that do work, and celebrate them. You should also deal with the parts that do not work, but do not let them dominate your presentation. This also implies that when you are still developing your solution, and you realise that you may not complete everything, you should focus on getting something working that is easy to show.

## 4 The final report

The final report is supposed to be a technical document that presents the work done by a group. It should be thorough, but not long-winded. I expect about 20 pages with a reasonable font size (11 or 12) and reasonable spacing (single spacing). It should not exceed 35 pages without good reason. Nonetheless, the number of pages in the document is not an important consideration in the assessment of the document.

The final report should be broken down in approximately five sections:

**Introduction:** Present the problem being solved, why it is useful or important, and an overview of what you managed to accomplish for your project.

**Background:** Describe the existing relevant technologies that impacted (or ought to have impacted) your implementation. Also include mention of any products or tools that are similar to what your project aimed to accomplish.

**Method:** Present your design (and architecture), highlighting its positive features; discuss the tradeoffs that were made and the reasoning behind some of the decisions that were not completely constrained; describe the implementation of the design, including any technical challenges that arose, and how they were surmounted. Describe how testing was done, and if any automated evaluation was done, how it was carried out.

**Results:** Present data that measures various aspects of your implementation. Use the objectives that you used in the definition of the project as your benchmark for assessing the degree of success of your project. This is where you would present screenshots of the user interface at significant points in the program. If performance is an issue or if the quality of your implementation can be measured quantitatively, you would also present data that reports on that in this section.

**Conclusions:** Give an assessment of the success of your implementation; to what degree was the original problem described solved? Given all that you have found in the results section, and those decisions that were made in the Method section, are there things that would have been done differently? how so? What lessons can be taken away from the implementation of the project (e.g. don't try to use a certain technology because it is too hard to debug, or use a particular programming technique because it is highly effective)? What extensions could be made to the solution or what modifications could have been made to the original problem definition to produce a better degree of success?

Along with your reports, you should also submit softcopies of the software you have developed (preferably on CD-ROM). Since there are no other means by which this course is assessed, you will not be given back your projects. You should take the opportunity to examine your report when it is graded, and see the comments written by your supervisor. One positive consequence of this, is that you have available for your perusal, the project reports that were submitted in the past – complete with grades and comments. You are encouraged to examine these before you write your own reports so that you have a good idea of what is expected of you.