Course Description. Here is the catalog description:

Project-oriented course designed to implement concepts developed in other computer science courses. Principles of software engineering will be emphasized throughout. Consulting Laboratory.

This course is unique: your work in this course will be similar to a real-world software job in many respects. You’ll work primarily on one major project, you’ll work with others on a group that simulates a company, and you’ll seek to understand and satisfy the needs of a person who is your “client”. Your work will involve all major aspects of software development.

Software Engineering is the name for the process of developing software. Without a course such as this one, it’s likely it would only be after many years working in the software industry that you would gain an overview of the overall nature of Software Engineering. That’s because most entry-level software positions involve making many small changes to existing software (called “software maintenance”) or testing software; also often software for a new application is developed by integrating several different pieces of already-developed software.

The standard Software Engineering “life cycle” consists of these phases:

- interviewing the client
- writing a technical requirements document for the client
- performing Object-Oriented Analysis
- designing an Object-Oriented software system
- implementing the design through programming
- testing and modifying each of these phases of work, and
performing post-delivery software “maintenance”, which means modifying the software. (Trying to reduce the often-great cost of this phase is the reason for much of the methodology of software engineering.)

Emphasized within the discipline of Software Engineering is the necessity of using a methodology that accommodates the inevitable changes that occur, largely due to human oversights and other errors. In fact, due to such inevitable changes, the sequence of phases must be iterated more than once. Since perfect foresight is impossible, Software Engineers routinely discover decision errors they’ve make in their analysis, design, or implementation work that require iteratively redoing work on prior phases. Thus:

In general, students can best learn to apply much of the material defined in the Software Engineering KA [Knowledge Area] by participating in a project. Such projects should require students to work on a team to develop a software system through as much of its lifecycle as is possible... there is increasing evidence that students learn to apply software engineering principles more effectively through an iterative approach, where students have the opportunity to work through a development cycle, assess their work, and then apply the knowledge gained through their assessment to another [iteration of the] development cycle. (ACM-IEEE Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science, December 2013, p. 174.)

Aspects of typical CS courses by Charlesworth and others:

- The instructor does any programming assignment in the course, prior to asking students to do so.

- The instructor provides a clear description of what a computer program should do, prior to asking students to write the program.

- The instructor will know what are the best decisions, relating to how to write any assigned computer program so it be successful. (If for some reason that isn’t the case, the instructor will at least have sufficient knowledge of the context of the program to give wise insights to students about how to write the particular programs in the course, when students need a hint.)

None of those three aspects of typical courses should be the case in CMSC 322, for students to effectively learn! That point is clear from the ACM-IEEE Computer Science Curricula Guidelines for Software Engineering and the additional explanation earlier on this syllabus. Students are expected to be able to explain that point on quizzes and the final exam, as well
as at any time during the course.

**Additional course content** includes (among other things) parts of the following chapters of Stephen Schach’s book *Object-Oriented and Classical Software Engineering*, 8th Edition:

Chapter 1 The Scope of Software Engineering, Chapter 3 The Software Process, Chapter 13 Object-Oriented Analysis, Chapter 14 Design, Chapter 15 Implementation, and Chapter 16 Testing.

**Strict Attendance Policy.** (The reasonableness of this attendance policy should seem natural to you, once you’ve carefully read the above description of the special nature of this course.) Students are expected to attend each meeting of the entire class and each meeting of their group. A group meeting time should not be changed unless all members of the group are informed and agree to the change. Students should check email daily to learn when the entire class will meet. (At the beginning of the course, until further notice, we’ll have our regularly scheduled class sessions in addition to consulting laboratories.)

The instructor reserves the right to give the V grade to any student who misses 3 class and/or group meetings during the semester without an acceptable excuse. The V grade is explained in the college catalog as “failure because of excessive absences” and 0 grade points are received for a grade of V. That is, receiving a V has the same effect as receiving an F in the course.

**Grading (in addition to the info on attendance).** At this time, the tentative plan is to weigh the parts of the course used in determining semester grades roughly as follows:

- 31% class discussions, homework quizzes, assignments
  (other than assignments related to the group project)
- 8% review quizzes
- 50% related to group project
- 11% final exam; Thu Dec 8, from 9 to noon.

A: 90-110; B: 80-89; C: 70-79; D: 60-69; F: below 60.

Your grade on the group project depends in part on the confidential evaluation of your peers. Each of you will be asked to evaluate the work of your peers, but the instructor will make the final judgment. Peer evaluations will ask you to what extent each other person on your team came to each group meeting and was well-prepared, encouraged others to work without doing all the work themselves, put in the time and effort required, were talented (and in what ways), etc. While you will certainly be told your group’s grade on an assignment, to keep peer evaluations confidential, you will not know your individual grade.

In other courses my students can compute their overall grade after each stage of the course. In the real-world the summary of your performance cannot be determined until the end of the project. Both because of that and because con-
fidential peer evaluations are instrumental in determining much of your grade, in this course you won’t know with confidence your grade until the end of the semester. However, to keep you reasonably informed, after each stage of the course I’ll try to notify those students who aren’t doing A or B work up through and including that stage.

Studies show each of us has a much better recall of things we explain to others than things that are merely explained to us. Furthermore, when things are explained to us, we’re much more apt to incorporate them into our way of thinking if we must actively discover them as we go along or at least answer questions about them as they are being explained. So your explaining things, such as how certain Java classes and methods work, to your teammates not only will help them, but it also will help you in a significant way. And the best way to explain is to season your explanation with questions, such as “now how could we do thus and so?”.

So whenever possible and appropriate, try to engage in such explanations with the other members of your group. Ideally this process will be a constant give-and-take, where each team member is taking turns figuring things out and helping others discover them. Also, it may be that some members of your group are stronger in writing and documentation, some in coming up with sound testing suites, some in programming, some in organizing software, and others in organizing the people within the group. While all students are expected to do all these activities reasonably well, the people in your group who are in the best position to “explain” might vary depending on the activity and stage of the semester.

During Fall 2016, the course uses Jepson G30 on Monday and Wednesday from noon to 1:15am, and Jepson G20 on Friday from noon to 1:15am. We’ll use some of those times for class sessions and some of those times for consulting laboratory sessions. It won’t be important for us to distinguish between when we are having a “class” session and when we are having a “consulting laboratory” session. Later in the semester students will have some of their team meetings at those time slots. In view of this course’s unusual emphasis on a group project, most of our “class sessions” will take place in the first half of the semester.

IMPORTANT: The Math&CS Department Chair has asked faculty to alert students that any request for special accommodation for quizzes/tests (with prior written approval from CAPS or from a Dean, etc., of course) must be made by the student to the professor at least one week before the quiz/test. Space in the Math&CS Department previously used for giving quizzes/tests has now been reallocated to offices for new faculty in the Department.