CS 3240-001 **Advanced Software Development** - Spring 2018

ENGR (17963)

INSTRUCTORS: **Sherriff, Mark (mss2x)**

Respondents: 90 / Enrollment: 146

Summary: CS 3240-001 Advanced Software Development - Spring 2018 (17963)

Overall Course Rating	**Overall Instructor Rating**
CS-3240-001 Mean 3.68 CS-3240-001 Std Dev 1.11 CS-3240-001 Response Count 445	*INSTRUCTOR:* Sherriff, Mark Mean 4.14 Std Dev 0.96 Response Count 624
SEAS, 3000-level courses Mean 4.09 SEAS, 3000-level courses Std Dev 0.99 SEAS, 3000-level courses Response Count 12906	SEAS, 3000-level courses Mean 4.20 SEAS, 3000-level courses Std Dev 0.93 SEAS, 3000-level courses Response Count 22603

~ QUESTIONS AND DETAILS ~ | *~ ANSWER MATRICES ~*

1. How accurate is this statement for you: The project was of acceptable length.

~

Question Type: Likert

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-001, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
90	4.00	0.91	29 (32.22%)	40 (44.44%)	13 (14.44%)	8 (8.89%)	0 (0.00%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
90	4.00	0.91	29 (32.22%)	40 (44.44%)	13 (14.44%)	8 (8.89%)	0 (0.00%)

2. How accurate is this statement for you: The project was of acceptable difficulty.

~

Question Type: Likert

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-001, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
88	3.89	0.89	22 (25.00%)	42 (47.73%)	16 (18.18%)	8 (9.09%)	0 (0.00%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
88	3.89	0.89	22 (25.00%)	42 (47.73%)	16 (18.18%)	8 (9.09%)	0 (0.00%)

3. How accurate is this statement for you: The project helped me better understand the phases and intricacies of software development.

~

Question Type: Likert

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-001, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
90	3.53	1.30	25 (27.78%)	28 (31.11%)	16 (17.78%)	12 (13.33%)	9 (10.00%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
90	3.53	1.30	25 (27.78%)	28 (31.11%)	16 (17.78%)	12 (13.33%)	9 (10.00%)

4. Which topic/lecture in this course was your favorite and why?

~

Question Type: Short Answer

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-001, Sherriff, Mark

Total	Individual Answers
72	*See below for Individual Results*

I thought the topic on design patterns was very useful. I also thought the activities helped enhance my understanding for the subject as well.

I did like the Git lecture earlier in the semester. I had used Git before, but I had never bothered to get an overview of how Git works from a higher-level. I thought it was fairly useful to learn it, because now I have a better idea of what is happening when a merge conflict happens, a branch is rebased, etc.

~ QUESTIONS AND DETAILS ~	*~ ANSWER MATRICES ~*
	Ethics - it opened up deep questions about human behavior and morals
	The software maintenance lecture because it was interesting to hear why it's important, why there's a stigma, and the activity (fixing the meme code) was good practice.
	I enjoyed the lecture on requirements and specifications because for the in-class activity we acted as the customer and the developers which really helped me understand the unit and the concept behind requirements vs specifications.
	I enjoyed everything except for the adapter units
	Design Patterns, I felt like learning more of those could be really useful in the future.
	Having a lot of freedom on the project
	I enjoyed the lectures on project management and risk management. I'm CS but I am definitely not going into software engineering so I was honestly worried about having to take this class but those topics are relatable across any discipline and I will definitely use those concepts and ideas so I appreciated learning about all of the different perspectives and ways that people manage work and other people.
	I really loved almost all of your lectures. I know that you said that they were all 'bland' and that this is an uninteresting class, but it was honestly my favorite class that I have taken here at UVa.
	Software Architecture and Design, as this particular topic is one I have had trouble with in the past (understanding REST, MVC, Design patterns, etc).
	I enjoyed the section about evaluating and dealing with risks associated with software failure.
	I really liked the different software methodologies. It was interesting to see how different ways of developing a project contribute to different requirements of the project
	Ethics, What if scenarios are fun and interesting. It also shows how people really feel about certain scenarios.
	I didn't really have a favorite topic, if I'm being honest. This is a class to fulfill a requirement for me, that's all.
	Source Control - I have never understood GitHub or AccuRev (the SCM the company I worked at used before) until I learned about this in this lecture! it was so interesting and clarifying!
	The series of lectures on design patterns
	github, useful in industry. getting to know the importance of spceifications
	Models/Django
	I thought the design patterns were cool, and seemed useful for real life.
	Agile development. Real-world industries use it all the time!
	SW Methodologies
	Code Smells because it was just enjoyable and I believe everyone got a lot of technical debt if the project was released to the state of Virginia lol
	N/A
	I enjoyed the lecture on software development methodologies
	Rest because I had always heard about it and never understood it.
	Security
	Maybe I'm just a nerd, but I liked the discussion on software development methodologies. This was something that hadn't been taught in other courses, and that I think will be pretty useful post-graduation.
	Software Design patterns was an interesting subject. Not one that I would look deeper into, but was more interesting at a base level than most of the material.
	I think my favorite topic was probably going over MVC and REST- I think this was really useful in understanding how typical projects are designed and structured.
	My favorite part of the course was learning about the different methodologies of software development, more specifically the difference between Agile versus Plan-Driven software methods
	Adapter Patterns
	Software Engineering Ethics

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
	user stories rest Probably working on the scenarios because I liked the problem/opinion solving aspect. Agile because its relevant to cs jobs ---, because --- I actually really liked the UML stuff and design patterns. I'm a visual learner and thinker so I was able to visualize every thing quite well because of those topics. The in-class activity was a little difficult but overall I think it helped me learn the difference between the two adapter patterns Security lectures, I just find the whole topic interesting and not well covered elsewhere in the CS curriculum I liked the topics in the beginning of the course such as the model and structure of a software engineering environment and development process. I enjoyed learning about the intricacies of what it takes for a software engineering company/team builds and deploys products and how the team works cohesively as a team to get the job done. I feel like these topics were things that I thought about but didn't give to much thought to so it was nice to actually learn about these ideas. I really liked discussing software ethics because that is just something that I generally always find to be really interesting. I love hearing the different sides of an ethical decision in my field and thought that the discussion style of these lectures brought some interesting opinions to the light. I really liked the lecture on software maintenance, probably because maintenance is such an underdog. I like design patterns because it was the most technical. requirements elicitation Design patterns because we actually got to try out with coding some basic ideas behind them, and got to most opportunities to understand them through activities. REST Apis Design Decomposition; it really made me think harder about how I organize the modules for my codebase I liked the project. I found the lectures to be pretty boring and trivial. No particular favorite lecture. In class activities were very useful and engaging. Ethics I quite enjoyed our 2-lecture discussion on Engineering Ethics, as it was intellectually stimulating and had multiple possible right answers. I really liked the lectures on Git, MVC, and the different types of development methods just because it was all very new to me and will be beneficial in the future. I really enjoyed ethics and wish we had spent more time on it. I enjoy talking about issues like that. Ethics-I really enjoyed looking at the difficult parts of software development, beyond the nuts and bolts of programming REST API because I finally learned what this "buzzword" means in industry. I don't really think I enjoyed any of the topics we discussed. This class is the boring side of SE I enjoyed the topics regarding design methodologies and source control. I liked learning how to work in a collaborative group programming assignment. I never went to lecture so I pretty much have no clue. I watched everything before the first midterm and literally none of it was even remotely interesting so probably some topic from after the midterm. I guess version control is a cool topic but way cooler at a low level, not just conceptually. I liked the ethics lecture because that makes for interesting discussions (also I remember it better since it just happened). Lectures 20 and 21 where we learned about design patterns by coding very simple and short solutions making use of the Professor's skeleton code were the most useful. They were engaging which made them enjoyable. MVC - it cleared up concepts that I have previously learned from an internship that were still confusing to me.

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~

The different methodologies, because it represents different leadership styles and general solutions to tackle problems that can apply to things outside of software dev.

Learning about agile and plan-driven methodologies, since I was given a case interview on when to use one or the other for my current internship

Requirements because I find it to be the most interesting because it's like, psychology, economics, and computer science all in one.

I think ethics are interesting because everyone can have a different opinion of what is the right thing.

Software development patterns - they seemed very applicable and like "oh that's what that is!"

Wearables - i think IoT is a field with sooo much potential

The final lectures about ethics. I think its an interesting perspective

None

Design Patterns.

5. Which topic/lecture in this class do you think you will find the most useful in the future?

~

Question Type: Short Answer

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-001, Sherriff, Mark

Total	Individual Answers
72	*See below for Individual Results*

The majority of it will apply to my career.

working in a group

Pretty much all of them

Agile

No particular lecture, but I learned so much from the project.

Just the software development lifecycle in general

Agile, IDEs, Youtrack, Git

Either design patterns, risk management or security as no matter what profession in CS we take, that these topics will show up in some fashion.

The different kinds of methodologies for team work (plan-driven, agile, etc.).

Ethics-Before this lecture, I didn't even know there was a code of ethics. Even if I won't keep that tab open in my browser as I'm writing code, it's helped in that that lecture got me to think about the moral ramifications of the actions that people take as software developers

Again, I think MVC was probably the most useful, especially considering in a recent interview, I was asked if I had worked with any MVC projects (this was before the class and I honestly didn't know anything about MVC, so this really helped!).

The methodologies one

As I said in the previous question, I am not going into software engineering. However the lectures on project management and risk management are applicable in any discipline and I know that I will definitely use those concepts in the future.

I think the system architecture and MVC lectures will be most useful to me in the future because of how much I will need to think about these when I get a job. I feel like I understood what these were from a high level, but it helped to get into the nitty-gritty of it in lecture and to discuss it in regards to our projects.

I think understanding the whole software development lifecycle was useful.

The different project methodologies - plan driven vs. agile methodologies. I also think that the requirements engineering section of the course will also be useful for working with clients.

I think the ethics discussion was very helpful.

Architecture and MVC. It helps that I now know how the internet works.

~ QUESTIONS AND DETAILS ~	*~ ANSWER MATRICES ~*
	I think for me personally the lectures on different methodologies will be most helpful. Understanding why, for example, an agile methodology is best for a certain type of situation will be very important in getting behind or perhaps trying to change the development strategies of a company I work for (or perhaps start) in the future. Verification and Validation The topics on agile vs plan-driven development because I feel that the material in that unit will be relevant when I am interning at a software development company. I feel that the topic of design patterns will be helpful, as knowing the conventions for widely-accepted problem solutions will likely come up. MVC, REST Architecture, and SOA N/A ^ ditto to the previous question. The discussion of software development methodologies. User stories Definitely REST and MVC because I learned those for the first time and now can understand a lot of descriptions of projects for my internships or systems I have to use! The decorator and the adapter class I think that the information we learned about using version control will be the most useful in the future. Probably learning about design decomposition because it's an important way of communicating design decisions among software engineers. Also talking about Agile vs Plan-driven. Software Engineering Ethics github and design decisions rest Security, smelly code Once again the methodologies section and probably the adapter patterns since it's a solution to many problems. Also source control Django/Models When to do agile vs plan driven. However, you'll catch me with a bullet in my brain before I go join a NoVA Gov't contractor that uses plan-driven and a 20 year old tech stack. The early lectures on requirements and how to stage them felt both important and something that is frequently glossed over, meaning the time taken to look over it was even more valuable. I don't want to be a software engineer, I want to be more of a project manager/analyst, so I think requirements will be most useful for me in the long run because that'll be a crucial aspect of my job Ethics MVC and REST; these are real technologies used in software engineering workplaces, so they will be highly applicable if the job requires this knowledge Risk Assessment and Mitigation Source Control Django ---, because --- I think the project and actually working within a team environment while working towards an end goal related to CS will prepare me more for the future than any topic covered in lecture. MVC was super useful and cool to learn about. Rest and MVC REST or MVC SCM: Github The topic on design patterns was the most useful. Software processes, Source control, User stories/requirements Design Patterns, Ethics

~ QUESTIONS AND DETAILS ~	*~ ANSWER MATRICES ~*

Design patterns probably. That one will be a big help in designing later systems.

Models because we will need to plan how to write software

Design Decomposition While this wasn't a particularly fun topic, I felt like I was learning something that would directly help me in my career.

Maintenance and Ethics

I thought that your lectures on software design patterns was very useful. This is a problem that I have seen in the real world and am interested to see how I can better help implement these

Software Engineering

organizing a project though stories, using version control, dockerizing

design patterns

Literally none of them. I've done a ton of internships already. You learn way more about software engineering in the first week of any job than you could in a classroom and it really doesn't matter what kind of classroom software engineering experience you have going in. That pretty much has zero effect on work performance. If I was one of the kids who somehow didn't know git going into this class, I'd say git is far and away the best/only topic that's relevant to the real world. A lot more time should be spent on version control and students should be tested on git commands and conceptual understanding. It's just SUCH an important part of the real world and so many people never really understand, or they use git GUIs and get laughed at when they show up at their first job. If I was one of those kids who didn't know anything about web and wanted to work in web dev, I would say overall web dev concepts are useful as well.

Requirements.

Requirements

Choosing a methodology

Different Software Development methodologies - a lot of companies I have interviewed at look for some experience with this and getting the chance to develop this and apply it to a project in a team environment I think will go a long way

I really couldn't say most seemed fairly useful. I imagine that it will depend on the type of work I end up doing in the future. If I end up in some sort managerial or sales role the information about requirement elicitation will be most helpful, and if I end up doing security the information on security will be.

Probably the design patterns lectures

Also software development patterns, same reason as before

Git

Maybe MVC along with knowing the methodologies.

I think that the adapter units will be most useful (but I would suggest better ways of presenting the information)

None

MVC/REST/Design Patterns

6. What lecture/topic(s) in this class "did not work" or were not seen as useful in the long run?

~

Question Type: Short Answer

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-001, Sherriff, Mark

Total	Individual Answers
67	*See below for Individual Results*

The adapter pattern lectures, while useful, didn't really fit in with the rest of the content. All of the other lectures focused on what could be considered "soft skills" of the software industry, while this was more of a technical skill

I would suggest better ways of teaching adapters

Design patterns.

It's all useful, it's just I figured I would learn all of this stuff once I'm on the jobsite. It doesn't feel very applicable right now and none of it really stuck. I can relate topics to what I have learned through internships, but the material feels too abstract when applied only in the classroom.

~ QUESTIONS AND DETAILS ~	*~ ANSWER MATRICES ~*
	requirements stuff N/a NullPointerException on cs3240.Evaluation().cs:6 I didn't get much out of the lectures. We basically observed the approaches that software engineers use in the field, but the project was a better learning experience for this than any of the lectures. I found the software design pattern lectures to be particularly confusing I don't think anything was NOT useful, I just think that it would have been more concrete if we used more real life examples (for instance, I really liked the amazon url demonstration when we talked about rest framework). I felt like a lot of it was more theoretical unless we used specific examples. Every time we had a "case" or scenario to work with, that was when I understood the concepts the best. I am not going into software engineering and I know that it's important to understand how it works but there was a lot of specifics that I just won't use in the field that I'm going into. But that's on a personal level - I think you did a fantastic job covering the material and making even the most mundane topics at least a little bit interesting - much appreciated. I'm sure that adapters/adaptees are really useful, but these were very difficult to grasp from a fundamental level so I'm still fuzzy on why they are important. I also personally thought it was a dry subject but I don't think there is anything that can be done about that. MVC- I had a hard time following it. The adapter pattern activity was a bit strange. The project wasn't very well organized, but since this specific course was the pilot course, it was to be expected. REST/Architecture-- I guess I just don't understand it well enough Specifics about policies Not sure~ SE Methodologies Nothing really, pretty much every topic was either interesting enough for me to pay attention to. engineering secure software I believe all of these are useful in the long run, however they are just boring topics to learn about in general Adapters N/A N/A N/A I didn't find a lot of the topics in the class useful in the long run all of the intricacies of the work flow Adapter Patterns YouTrack and user stories. We had no idea how to make effective user stories as no one had worked on such a project before and no one understood the language well enough to make good user stories. Adapters - These seemed so useful but I can honestly say the way everything was presented was so confusing. The in-class activity even honestly just confused me more and I had to ask a billion questions to understand anything. N/A none of them were problematic. The security lecture probably should have come a little earlier thoguh. Adapter patterns... sorry, it just didn't make a lot of sense. The general idea made sense but actually trying to code them and stuff was a mess. REST, MVC. I felt that we went to fast on these topics. I still don't have a good grasp on them since we had to learn on our own for the most part. The whole adapter activity class simply did not work and nobody understood what they were doing.

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
	MVC
	Lecture: Course Intro
	The material on IDE's felt shoe-horned in, especially since most, if not all students at this point have used and are familiar with IDE's.
	security didn't really work because there was not a lot of focus on it ever
	I don't feel that there were any particularly useless topics.
	See my previous response... This class as a whole is SO much less necessary than the faculty likes to pretend it is.
	As much as I enjoyed learning about the different design patterns, the in-class worksheet on the adapter pattern and its variants was not helpful for me. It made me more confused than anything, and pivoted around the knowledge that you can't instantiate an interface in Java.
	It's too early to say
	For some reason the lectures on class adapters seemed really out of place to me.
	I did not experience this
	I just think that the information could have been squeezed into a much shorter time period, as it made the information feel a lot less dense and less useful, even though everything probably was really useful!
	The design patterns discussion was not structured all that well. It felt too much like an "add-on" to the course and wasn't very well integrated with the rest of the material. It resulted in a huge gear shift for the course, and I don't think enough time was devoted to actually explaining the hows and whys of the material. Design Patterns are something that are super important, and indeed there can and is a course almost entirely dedicated to design patterns. My point on this is that if there isn't enough time to dedicate to design patterns effectively, then it should honestly be left out of the course curriculum. Otherwise, it will continue to feel like an add-on, which is detrimental to conveying the importance of this topic.
	functional/non0functional requirements
	I didn't understand the point of the different types of object vs class adapters because up until this point, the class was seemingly theoretical about software development as a whole, while class and object adapters have to do with physical implementation of the code.
	REST and MVC stuff, but that's just because I have taken the mobile class.
	I didn't like class activities. When I am in classroom, I like professor to talk all the time. I can do different activities at home on my own.
	I feel like the design patterns just did not seem to fit in the class. We weren't really programming something to use those designs
	I'm not sure yet, honestly I'll just have to wait and see.
	I thought all the lectures was useful.
	3/29/2018: Software Design Patterns: Adapter This lecture did not work out that well. My peers and I felt confused at the end and there were still many things we didn't learn well enough by ourselves. We would have much rather liked to get an overview of the Adapter Design Pattern from you, since you usually explain topics pretty well, before moving onto the activity. There weren't any topics that were not useful. I thought each topic deserved its lecture.
	I can not think of any that were truly terrible. The one day we did an in class activity about adapter patterns wasn't the best, but afterward we thoroughly covered the material. Though I must admit that I am still a bit foggy on some of the information from that section.
	Methodologies seemed sorta "duh"...
	The lectures about REST just didn't help me learn it as much as I would have liked. A similar activity to the ones we did at the end of the class would have been helpful.
	Design patterns
	Design patterns
	I thought the whole idea of not explain the project details was complete horse-shit.
	n/a
	did not particularly like things such as working with stakeholders for making the design, just some very hand wavey topics

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~

Maybe some of the intro lectures like Software Engineering in general.

I didn't like most of the paper activities (except the coffee one, that was okay). Usually they didn't help me to understand the material.

I think more time should've been devoted to learning the best software practices when writing Django projects and RESTful APIs.

Code Smells because even though its used in industry, they haven't been relevant in my internship experiences.

7. How accurate is this statement for you if you used the podcasts from this class: Podcasts were useful to catch up on material that I missed due to absences.

~

Question Type: Likert

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-001, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
88	4.18	0.77	18 (20.45%)	25 (28.41%)	5 (5.68%)	2 (2.27%)	0 (0.00%)	38 (43.18%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
88	4.18	0.77	18 (20.45%)	25 (28.41%)	5 (5.68%)	2 (2.27%)	0 (0.00%)	38 (43.18%)

8. How accurate is this statement for you if you used the podcasts from this class: The podcasts were useful to review material that I was unclear on.

~

Question Type: Likert

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-001, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
90	4.04	0.82	15 (16.67%)	26 (28.89%)	7 (7.78%)	3 (3.33%)	0 (0.00%)	39 (43.33%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
90	4.04	0.82	15 (16.67%)	26 (28.89%)	7 (7.78%)	3 (3.33%)	0 (0.00%)	39 (43.33%)

9. How often did you listen to the podcast for a lecture?

~

Question Type: Multiple Choice

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-001, Sherriff, Mark

Total	Every lecture (NA)	Nearly every lecture (NA)	Whenever I needed to review a topic (NA)	Only when I missed a class (NA)	Randomly just to see what it was like (NA)	Never (NA)
90	3 (3.33%)	0 (0.00%)	26 (28.89%)	14 (15.56%)	7 (7.78%)	40 (44.44%)

Results for SEAS, 3000-level courses

Total	Every lecture (NA)	Nearly every lecture (NA)	Whenever I needed to review a topic (NA)	Only when I missed a class (NA)	Randomly just to see what it was like (NA)	Never (NA)
90	3 (3.33%)	0 (0.00%)	26 (28.89%)	14 (15.56%)	7 (7.78%)	40 (44.44%)

10. Do you have any suggestions/comments that we should take into account for future projects for this course?

~

Question Type: Short Answer

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-001, Sherriff, Mark

Total	Individual Answers
67	*See below for Individual Results*

I know you guys think it was "too short" but I think it was good that it was the way it was because organizing 8 people was a whole new territory for a lot of us.

I think that the 8-person team size was a good idea, but his project simply didn't have enough things that could be done in parallel to work flawlessly with eight people. My teammates and I all had sprints or partial sprints where we simply weren't needed, and thus didn't accomplish much of anything.

~ QUESTIONS AND DETAILS ~	*~ ANSWER MATRICES ~*
	no no Create the tech stack for the students Have the rubric very clear for the final project. Or if you don't, don't penalize students nearly 25% during a demonstration for a project that's worth 50% of the total grade. No, I think that this was a pretty good project and it incorporated a lot of elements of web development that were helpful to learn. HTML was very unfamiliar to me and I feel MVC was covered as a concept, but not on a practical level that could be used effectively for students new to this. I don't think you ever provided information on how to document our APIs like you said you would. I think that the size of the group was too big for the project we worked on. I think the size and difficulty of the project itself was perfect for what we could do in a semester, but we definitely did not need 8 or so people on our team to get the project done. I think 4 or 5 would be the right amount for this project. Honestly, because of the number of people in the project and the size of it, a few of the members in our group didn't (or couldn't) participate as much in the project because of that. The project seemed like a bit of just coding and coding, but in industry there would be more meeting with the client. Thought out better ahead of time, instead of playing catch-up. Make it longer or decrease team size. There were a lot of issues in various groups, not just my own, of people either having to conjure up tasks that were not really related to the project, or of simply doing nothing. Granted the folks who did nothing probably still would have done nothing but the people who just made there own tasks were more or less just unable to help with anything from fear of stepping on someone else's toes. Maybe make team sizes smaller. The difficulty is appropriate for this kind of class however, the team size maybe too big which caused some people to do less work (making it seem like the project was too easy but really it was not). 9 people in a group is too much. It felt like I was competing with my own group members for points, which should never be the case. This course is meant to be teaching us what the real world is like. Well I can say with 100% confidence that if I've ever in a situation with a company where I feel like I am in competition with my own project members as opposed to being in competition with other companies, then that is a company that I will not be working for for long. This is partially a symptom of the way the sprints were graded (it felt like there were grading relative to group members rather than relative to other groups). But is also a symptom of there being entirely too many people on project groups. Maybe let students choose what they want to build, instead of doing a generic project. Lower the group size, I always felt like I was not doing enough since 1-2 people in the group would push huge blocks of code, or get to things before I did, so I would be behind and could not contribute as much as I wanted to. Maybe have less people on the teams Take some more time to explain the resources (django, etc.) made available to us. I learned plenty with the project but it made the class seem like just a web development course...just one that didn't cover a lot of web development topics in class. Grocery store idea would've been so much more fun/interesting, voting system is complex as it is and while there wasn't as much work as past projects, it look a really long time to understand how the voting system works. Make it more organized and planned. I think it would be nice if they were cooler projects. A voting system was hard to get excited about. Maybe creating a business website, but I know e-commerce does this as well. This project was weird - the parts were not very compartmentalized, making it hard to work on in a team setting. The project, I feel, should stress the software development lifecycle for a product with many parts that must be integrated. I would highly suggest something simple that achieves this goal as to promote no "cramming" as well, especially because the point of the course is *not* Django or web development, but instead the life cycle. I would love to see smart-home projects, something mechatronics related (I'm a little biased...), or something that has various parts. I'm more hesitant to go with a true piece of software because it would be a LOT of work. BUT, the project as it was this semester did not teach me about software development lifecycle and good practices.. I've learned more using my personal website than making this one. Clearer timeline with the instructor providing what he repeatedly promises on time.

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
	The group size felt a little bit too big. There were some people in my group that were at a loss for what to do and ended up really unable to do anything since all the main aspects were picked up by other members. I would recommend looking into software that doesn't require a UNIX structure. I had to dual boot Kubuntu to be able to work on the project because Docker Toolbox had too many issues when working alongside docker proper.
	Should have "client" change functionality during mid-development etc. to see how teams respond to change. Possibly more real-life scenario or challenges.
	Definitely provide more detailed requirements so that students know what exactly they need to accomplish.
	A group size of 8 was not a great idea. I felt that this project could have been done by 3 people and that's kinda what happened. One other person and I carried our group very hard... Having groups be determined by things like GPA only hurt people who are smart and help people who are not going to try in the class. I think a good idea would have been to allow each person to agree on a partner to be in the group with them. Overall, smaller group size, similar difficulty, choose a partner, don't let groups carry. Also if you do carry then you should not be penalized because your group was stupid.
	Make it more concrete from the start. The amount of misinformation we received from either Sherriff backtracking or TAs not hearing about changes led to a lot of communication errors. I know it's "realistic" because clients change what they want all the time. But it would have been more useful if we were left more unhindered and able to do work for it.
	The evaluation process needs to be revised. It's very hard to distinguish between lack of progress and lack of effort. Someone could spend hours trying to fix one bug while another person changes the UI 10 times, and it would appear that the latter put more effort into the project. Sprint and individual scores did not reflect your contribution to the project.
	Have a suggested way to split up the project among members. My group split up the group among frontend and backend which ended up with front end doing a lot of work and backend not really sure what exactly they should be doing (as a lot of the backend work would depend on the frontend work). I think if our group split up the project based on functionality (login, voting, API) rather than "frontend" and "backend" then I think my group would've done the work much more evenly split among members rather than 2 people doing most of the project.
	I wish there was more structure for the projects- I know that you want us to be able to do that ourselves, but there was an obvious disconnect between what the teams ended up doing with what you all expected from us, as shown from the final exam- I think there was an expectation that we should have each covered all of the front end and back end and everything in between as "employees" of the company, but we ended up splitting up into front end and back end groups, so we gained expertise in that area but understand nothing else. And I think what I really wanted to gain out of this project but didn't was a full understanding of the project and a full understanding of everything required for the back end.
	I think having smaller groups (4-5) with a slightly smaller project would work better. I feel as though with groups of 8-9, it's very difficult to contribute equally. Furthermore, some people have background experience in databases, Django, Docker, Heroku, that makes them contribute far more to the project than students who only have 2150 experience, Python. This isn't really the fault of the students; there is a wide variety of previous development background from 2nd years to 4th years. From a personal group experience, we had an excellent source control manager, one guy with Docker experience, two people who had taken databases, and a CEO who didn't know any non-Python technology but self-taught himself on the way. Everyone else knew nothing about web development. Maybe that's the way the course was designed to force, but ideally a more uniform background would allow a better distribution of code commits, and so that some people aren't floundering and constantly trying to understand all these new technologies. As to how to ensure this in the CS curriculum, I have no idea.
	The whole point of the 9+ person teams is to make it difficult, but god damn was having a 9+ person team difficult. My TA had no idea whatsoever as to what was going on in the project and was extremely vague on the requirements. I would ensure that your TA's are good project managers before putting my grade in their hands.
	For professional apps, we should have question/answer phase with customer to get an idea of the specifications more clearly
	If it fits your vision, give us some sample scenarios so that we have some semblance of an idea what the demo is going to be like.
	9 people in a group was too many! I think no one really felt responsible for the project, since there were so many other people. I felt like I was the only one trying to coordinate it as a whole, so I had a pretty bad time. I think smaller groups would have helped, and are more realistic anyways.
	Offering a big'ol list of resources/tutorials that apply to the topics you want us to explore. I understand the idea of making us all figure it out on our own, yet it seems somewhat damaging to our learning to go through this process of trying to make ends meet by using bad coding practice/ignoring good pracitce
	too many people in a group. 4 people did absolutely no work the entire semester, and the TA knew and didnt try and help. If people do no work, and we make it known, the TA should intervene.
	I know it's extremely hard to create groups but my group was a pretty weak group and we had less members than most groups so the project was a lot of work.
	maybe have a project that's a little better defined at the start

~ QUESTIONS AND DETAILS ~	*~ ANSWER MATRICES ~*
	I wasn't able to contribute as much to my team as everyone else since I am taking this class relatively early and haven't had as much experience. Team size might have been too large Would be more helpful to bring in an "actual" client that wants some type of software. Then allow us to spend a class asking questions to create requirements/user stories. Have well defined instructions on how to install the components needed for the project. By sprint 2 many of my teammates were still having trouble installing them. not to do Web Design especially which so much technology that new to us. I personally felt that the group size was too large, as it allowed people to slip through without repercussions as there were enough people that they would just make progress anyways. I also believe that maybe either a smaller group size or an even larger project scope would be better for future classes? I also thought that I didn't feel too much pressure to contribute to the group project when everyone else was already handling everything, leading back to the people slipping through the cracks without much repercussion point where it didn't feel like people not doing much was hurting the group effort too much, which can be both a good and a bad thing. Needs to be much more organized earlier in the year. I understand the project is supposed to mimic a real design contract which undergoes changes, but I've worked for a software development firm and this was just disorganized by comparison. n/a n/a I liked the interfacing with the receipt printer as part of the project, because it gave the computer engineers (and more hardware-minded CS majors) something they could really tackle. I would just keep in mind that CpEs are also required to take this course, and not to make the projects ~too~ web-dev-y. more clear instructions Honestly consider shifting the focus entirely to the project. It was a really cool introduction to Django and working in a software development team and teaches you the core concepts better than any of the lectures. Please go over the technical aspects more for how to build the projects. I still don't feel like I know Django as much as I thought I would when I first stepped foot into this class. Maybe a compilation of any useful suggestions/comments that students put here? / posts from piazza Like if you decide to use the same project I know that we had a few technology bumps with realizing that Docker works better on Linux and Macs and depending on the Windows version you could accidentally destroy VirtualBox's operations by enabling Docker Our group also compiled some documentation into quick slides for our group for an overview of Django and Docker and the best way to install/set them up (they're in a google drive linked in the README of our github) if you wanted to offer them as a starting point for groups (learning is learning, sometimes it helps to have a place to start, especially if it's already been documented) Make the team smaller, too many people are hard to manage and not doing much. The project was really interesting! I would like to see more creative projects like this. The laws are a pain though lol Don't make it 8 people teams Smaller groups and less in class activities. The project this semester was more of a 4-5 people per group size project. Project should allow more people to have things to work on without making it a lot harder, or in other words add more possibilities for different aspects that make your design unique. Also if you use Docker again, please be more helpful in getting it set up and working for everyone. 8 people was wayyyYYYYyyyy too many It honestly would've been easier if I just did it all by myself I think you should make smaller project groups. This project honestly ended up being a few people doing the majority of the work because there was not enough work to be passed around. I think that there shouldn't be such a push at the end to finish. From what I saw at the demos not all groups were prepared. Myself and another teammate basically built our project and dedicated 2-3x the amount of time that anyone else did and we did great! I just think that the TA's said the groups were done because we were told one sprint to early that our grade depended on it. Overall I loved the project Smaller teams would be better. Also better definitions of requirements and expectations. While I was able to work with the flexibility, I saw lots of people who defaulted tot doing nothing because there were no specific tasks enumerated by instructors. It was incredibly frustrating to deal with the lack of motivation and initiative as a result of vagueness. And I don't think it was super reflective of real-life software development struggles either.

~ QUESTIONS AND DETAILS ~	*~ ANSWER MATRICES ~*

Organize it way better and make the groups way smaller. I totally get what you're trying to do by being vague about requirements and making it so the class has no clue what's expected all semester and then putting us in huge groups (this is what happens in the real world blah blah blah). It's just not fun though. On paper I actually do understand why it looks like a good idea but I feel like it's just one of those things that shouldn't actually be done. There are just SO many differences between industry and the classroom; I really don't know if there's any version of actually accurately simulating the shitty parts of industry in a manner that's constructive and will improve students experience in the real world. Trying to do this in the classroom setting just causes students to dislike each other, absolutely hate their TAs and dislike the professor. (I don't dislike Sherriff; I'm speaking about what happens to the general student in this course).   Putting A students in the same group as C students is infuriating. Again, I totally understand the goal here. In the real world, the average engineer will have to work with people who aren't motivated or aren't competent. It's REALLY hard to do eight people's work, especially when everyone wants to participate because they want to be able to tell the TAs what they specifically worked on.

I think many of the TAs were too subjective with grading the sprints. This wouldn't have been a big issue, but the sprint grades were worth so much of our overall grade. One peer told me that his TA gave his whole group a 2/5 in order to "motivate the team to try harder" even though their work satisfied the expectations of the sprint. Without any other knowledge on that particular issue, that didn't sound very fair to me.

11. During the project, how many hours per week did you dedicate specifically to project work?
~
Question Type: Multiple Choice
~
contributed by Sherriff, Mark (mss2x)

Results for CS-3240-001, Sherriff, Mark

Total	0-2 (NA)	3-5 (NA)	6-8 (NA)	9-12 (NA)	13-16 (NA)	17 or more (NA)
90	13 (14.44%)	48 (53.33%)	22 (24.44%)	3 (3.33%)	3 (3.33%)	1 (1.11%)

Results for SEAS, 3000-level courses

Total	0-2 (NA)	3-5 (NA)	6-8 (NA)	9-12 (NA)	13-16 (NA)	17 or more (NA)
90	13 (14.44%)	48 (53.33%)	22 (24.44%)	3 (3.33%)	3 (3.33%)	1 (1.11%)

12. How would you rate the availability of TAs?
~
Question Type: Likert
~
contributed by Sherriff, Mark (mss2x)

Results for CS-3240-001, Sherriff, Mark

Total	Mean	Std Dev	Excellent (4)	Good (3)	Average (2)	Weak (1)	Very Poor (0)
89	2.75	0.90	15 (16.85%)	47 (52.81%)	19 (21.35%)	6 (6.74%)	2 (2.25%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Excellent (4)	Good (3)	Average (2)	Weak (1)	Very Poor (0)
89	2.75	0.90	15 (16.85%)	47 (52.81%)	19 (21.35%)	6 (6.74%)	2 (2.25%)

13. How would you rate the helpfulness of the TAs?
~
Question Type: Likert
~
contributed by Sherriff, Mark (mss2x)

Results for CS-3240-001, Sherriff, Mark

Total	Mean	Std Dev	Excellent (4)	Good (3)	Average (2)	Weak (1)	Very Poor (0)
89	2.49	1.16	19 (21.35%)	28 (31.46%)	27 (30.34%)	8 (8.99%)	7 (7.87%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Excellent (4)	Good (3)	Average (2)	Weak (1)	Very Poor (0)
89	2.49	1.16	19 (21.35%)	28 (31.46%)	27 (30.34%)	8 (8.99%)	7 (7.87%)

14. How often did you make use of the TA office hours?
~
Question Type: Multiple Choice
~
contributed by Sherriff, Mark (mss2x)

Results for CS-3240-001, Sherriff, Mark

Total	Every week (NA)	Every other week (NA)	Once per assignment (NA)	Rarely (NA)	Never (NA)
90	5 (5.56%)	9 (10.00%)	14 (15.56%)	38 (42.22%)	24 (26.67%)

Results for SEAS, 3000-level courses

Total	Every week (NA)	Every other week (NA)	Once per assignment (NA)	Rarely (NA)	Never (NA)
90	5 (5.56%)	9 (10.00%)	14 (15.56%)	38 (42.22%)	24 (26.67%)

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
15. Any specific comments about the TAs you would like to share? ~ Question Type: Short Answer ~ *contributed by Sherriff, Mark (mss2x)*	Results for CS-3240-001, Sherriff, Mark

Total	Individual Answers
60	*See below for Individual Results*

no

N/a

Divya's cool

Even though some of my labmates did a lot of work on the lab, they only got 3.5 of the 5 because a TA thought that a 5 is only worthy when something truly extraordinary was done. I think it's fair to say that if we were on track or ahead on the project, we should deserve 5s if we put in decent work on the project.

Make sure your TA's know what is going on and have the interpersonal skills of a regular human being. I don't know if my TA was extremely autistic or what but lord have mercy on my team.

not at all happy with TA's. Were inconsistent with providing information (said our front end was great one week and took off for it the next week), were extremely subjective in their grading for sprints, and provided no concrete guideline or grading criterion. At one point we were docked a point on our sprint because the TA thought our work was great but "just wanted to motivate us" a bit - that one point had a significant impact on the final grade due to how the latter is calculated. one of the weakest links in an already weak course.

TAs did their best, but they weren't given much information and they did a different project as students than now. They weren't very helpful but it wasn't their fault

For whatever reason it felt like all the office hours were on Sunday, they could have been spread out better.

They couldn't know how to help with anything.

They need to be more coordinated in terms of sprint expectations and grading.

They were awesome! Very helpful!

I wish they had shared with us what exactly they were looking for during a given sprint or that all of the TAs had a uniform set of things they were looking for. For example, we lost a point during one sprint for not having our site deployed on Heroku yet but my friends in other sections said they received 5/5 and did not have this done yet either. This was just a little frustrating since we did not know that it had to be deployed yet at the time.

N/A

N/A

N/A

N/A

I think they're trying pretty hard. As a TA myself, I know how hard it is to be in their position, especially with super subjective material having such a large impact on our grade. I would do a better job than the current TAs but I also would never ever ever want to TA for this course so I kinda see why those types of students end up being the 3240 TAs. The grading in general though needs to be a lot more objective. It would take a lot of stress off your TAs, and would cause way fewer students to be pissed off on a weekly basis. The TAs opinions factoring into the grade is perhaps the most frustrating thing about the course. I know you're reading this and already doing the "well in the real world this is how you're judged too" thing in your head and while I still completely disagree with the class being run this way, I don't have concrete evidence that it doesn't work so I'll let it go till the next text area.

Did not make very clear what they were looking for when grading during lab.

I felt that they knew as much as I did most of the time.

Our TAs (Zach and Bobby) were awesome, thanks guys.

I think the TA you get for your project really really really affects the grade that you get on each sprint and the feedback you receive. The TA that I had for our project was not very helpful at giving us feedback for our mid-sprints, and then docked off points at the end of our sprint for not doing things that they never told us to do. The other TA in our lab section, from what we could tell, did a lot better in terms of giving their teams

I generally just think that there was an expectation from the TAs of the teams that we didn't quite get a hold of until the end, giving us relatively poor grades in the sprints.

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~

The TA's were helpful. Except our lab TA who was very vague and then when we came to demo to our knowledge we thought we had eveyrthing working because he told us we were fine even though we were not. Daniel was really vague and is the reason we lost some points on our demo due to his ambiguity.

Our TA didn't always clearly tell us what was expected each sprint

Whenever our team asked a question the TA's answer was always I don't know. Especially if it came to why software wasn't working.

No.

No.

For the project, the TA's didn't seem to know all the details for the project and it was a factor in our product being a little clunky. (Other parts include laziness on our part)

Most had Office Hours only on Sunday. Few new how to use Django well. Further many did not know anything about the project as a whole. On multiple occasions I received incorrect or poor information from TAs

Sam was awesome.

The TAs were pretty good, nothing to comment really.

We felt like our sprint scores were often given harshly without any indication when we were being assessed.

Nope, they were all generally good in my experience

They were great.

Marissa was really nice

Not really, the two I interacted with were both good.

The TAs often seemed clueless about the overall state of the project and the requirements

After talking to friends in other lab sections, the grading for each sprint was widely different.

I always went to Bobby and Zach's office hours. They were extremely helpful and fun to be around. They also cared about the dynamic of my group so they would ask me to make sure everything was fair and the work divided equally.

Our TA was extremely unhelpful. I am not sure if you instructed them to give us vague answers to any questions we asked but they were pretty much useless. Also I was also confused why we were given 4s during the sprints that we accomplished the most work and put in the most time. We seemed well ahead of most other groups.

Very helpful!

She was nice.

Our groups's TA was very helpful and nice, but I personally didn't need to interact with him very much.

They were not too helpful, and it seemed like they were out of the loop all the time.

Grading was extremely subjective and they did not understand how much effort I was putting into the project compared to my team members...

Seemed not only to have imperfect knowledge, but also didn't seem willing to correct that knowledge. A few questions asked got answered as a middle-man to the professor, but other than that the answers were wishy washy. Do we need primaries? Well maybe I don't really know he hasn't specified. Sorry, but at that point it's not an issue that will be faced in the real world, but instead an issue where we're given requirements that make intuitive sense, ie we need multiple elections going on at once and voter-location specific elections, and makes it NOT intuitive because the TAs just didn't help.

None.

Sprint 3 specifically, where you tried to light a fire on everyone's butts seemed a bit unfair. Many groups were graded poorly only because we didn't meet the requirements set at that time. This was because the requirements were not explicitly stated in class or in lab. It was a bit of a change from going from sprint 2 to sprint 3. If you talked about what should have been done then the task would have been done.

Sam did a good job of recognizing that there was a difference in the amount of work being done. I think that grades could be based more on Github work and less on what individual people say

No

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~

n/a

n/a

It would've been helpful for them to know more about the project

Bobby was really great for Django help. He helped explain confusing concepts that even the Professor couldn't help our team with.

Nope.

Divya and Danny are the best! They were both super helpful and made the project and the class a better experience.

Zach and Bobby were cool / knowledgable! I think in general, TAs weren't entirely in the loop with project requirements / expectations. Most times when we asked "do we need to do this?" they were unsure. Usually when we asked them specific questions about technologies (ex. the printers) they knew about as much as us. That's understandable, since there were a bunch of technologies, but it would have been nice to get some help with stuff like that.

Good tas

zach is the best

Zach and Bobby are the bomb

16. What other topics do you wish we had time to cover or which topics did we cover that you wish we could have covered more deeply?

~

Question Type: Short Answer

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-001, Sherriff, Mark	
Total	Individual Answers
52	*See below for Individual Results*

Probably testing... I think it would've been nice to dive into Travis or something

Honestly not sure, I was just kind of along for the ride

Software Testing

Maybe bring in a guest lecturer to talk about how they make a software product from start to finish to give us a better real world idea of how we can apply what we learn.

The security information and the classes on ethics were both very fun and a more in-depth look at either would have been fun.

Design Patterns - as stated before it should either be more integrated into the course or dropped altogether.

Source control and its different types.

none

No opinion here.

Security is a big one. How to implement Agile methodologies. Hmm idk maybe teach us a little about Django?

Security but that's just because I'm a security person

Can't think of any.

N/A

N/A

I would've loved to see more team management covered, specifically splitting teams into subgroups to tackle specific problems. Our project team had assigned roles, but those had very little to do with the actual coding each individual was responsible for. Finding out how to best divide and use assets would have been helpful.

Definitely just more in depth information about the project- I guess that would just be adding more requirements to the project.

Design patterns!

More design patterns

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
	Maybe some topics about the industry that we need to know as engineers. Probably a more miscellaneous lecture on what to expect in the workplace could have been useful. I don't know what other topics could have been covered. NA Day-to-day professional life Deeper dive into architecture of websites. E.g. How to design a website like spotify I wish we could have covered the Singleton, Abstract Factory, and Decorator patterns a bit more in depth. I loved that in-class activity we had that day. probably would've been nice to have some industry speakers come in I wish we had learned what django/docker actually does. Ethics. Not sure, a lot of content is covered in this class and it is hard to tell until I have experienced a large software dev environment what I do and don't need to learn. more technical stuff Good software practices Ethics I wish we talked about Django in class more so that we could all have a similar introduction. I wish that we could have covered other technologies that are used in the 'real' world I wish we had gone further into the technical details of web development. As a CpE, I'd had no exposure to anything resembling web development in this class, and had we not had a couple of people on my project team who knew Djano super well, the project would have been a huge time suck. Not really any. Some of the things you think are super cool that we see in some of the popular apps Adapters You should've done security at the beginning because te design patterns No comment. I'm honestly not sure. I'd like to see examples of workplaces with certain methodologies, and analyze them I personally wish that we did a bit more with maintenance. While we got to learn about it, we didn't really do too much with it as the end of the semester was already approaching. n/a n/a more technical aspects of Django More into github and django, it felt like I was tossed into the deep end to learn these two types of software N/A (Idk what we've done in class yet since I never ever go) none, covered a lot in this class It is what it is. I don't want to learn this, you don't want to teach this, but it's important that UVa kids leave C'ville with an idea of what agile is. Distributed computing More on security might have been useful, and sooner. Also maybe a crash course in DJango/Docker/etc.

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~

17. To what degree do you agree with this statement: the team size from the project was appropriate (please elaborate in your class comments).

~

Question Type: Likert

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-001, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
90	3.22	1.25	14 (15.56%)	31 (34.44%)	15 (16.67%)	21 (23.33%)	9 (10.00%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
90	3.22	1.25	14 (15.56%)	31 (34.44%)	15 (16.67%)	21 (23.33%)	9 (10.00%)

18. This semester, we doubled the size of this class to accommodate as many students as possible. Please comment on this. Did it affect the quality of the course? Did it not matter to you? Did it work?

~

Question Type: Short Answer

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-001, Sherriff, Mark

Total	Individual Answers
76	*See below for Individual Results*

I thought it was fine. More people allowed for a more balanced work distribution.

I doubt it changed the quality at all, but I wasn't in the other course so it would be hard to compare.

I don't think it affected it at all. I never had trouble in class or in lab trying to get the assistance of the professor or a TA

I was happy with the size

I didn't think that the quality of the course was affected and appreciated that I was able to get into the course.

This made the course exponentially worse. It did not work at all. By the way, I realize I'm leaving a bunch of savage comments. I don't dislike my group, and we're not doing bad. I'm on pace to get an A in this course. I realize it's really easy to come off as some butthurt kid who isn't doing well int he class when I have this many negative comments, but the class is just really way more stress than it needs to be for what it is and the gigantic team sizes is one of the many things that plays into that issue. Also on a team size of 8, it's so much harder to establish yourself as a leader. This is the first CS group project I've ever not been the unofficial leader on and it's about to be the first one I'm not gonna get a 100 on (not a coincidence).

Sometimes meant certain people didn't have work

I found it fine. Lecture was fine.

Given the increasing size of the CS student body, I think that increasing the student cap was much needed. I do not believe that increasing the size hurt the course in any way, though I don't know how the increase affected managerial issues (such as TAs)

Yes because I think teams were double the size as a result and that was terrible. I think there should be a way to strike a good balance there.

It didn't affect the quality of the course but doubling the team size left people without things to do.

I didn't care, except size was way too big in teams, making alot of redundancy in work. THe project length was good and everything. But it was mostly just 2 people that does everything.

Thank you! CS department is underemployed and class sizes should be increased to fit people in!

No No Yes

I thought the course was fine, and I honestly kind of liked having a larger team size, because it did sort of make us have to assume roles on the team, instead of just doing whatever needed to be done.

It was fine. Some ppl didn't even show up anyway.

I do not think that this effected the class, or the quality of the course. I really loved the course, and if the team sizes were any smaller I could have gotten a very bad group

It didnât matter to me.

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
	If you're talking about the size of the lecture, it didn't matter to me. If you're talking about the size of the project groups, I thought it was a better experience. It matches up more closely with a real team scenario with around eight people per team. I thought it was a good number. I have no frame of reference as far as the amount of students or the previous quality of the course, but I felt that it didn't make anything any more difficult. I felt it was effective. I do not think it mattered Due to the nature of the lecture and Professor Sherriff's delivery on the topics, the size of the class did not affect the quality of the course at all. Regarding team size, I think that it was an appropriate size that allowed the work distribution to be not overbearing on the members. I'm not sure it changed anything. I doubt it did to be honest. Dont know what it used to be like. So not sure Inflating course sizes is destroying the CS department. I hope we cut and fail many many more kids with the pilot redesign. A lot of kids give up on their chem/comm/whatever major and choose CS because its easy and you'll get rich. I don't enjoy having these failures who had a major 2nd year crisis and turned to coding in my group as they are typically unmotivated and have weak coding skills. I didn't notice the large class size. It didn't matter to me. I suppose that having a large class allows for more lab time slots, which may be helpful for students' schedules. It made the group sizes very large, however the lectures seemed to be okay since many people skipped anyways It made it so that every person had one task that was there own. We really didn't see a lot of what other people were doing nor did we see any side of the project that wasn't our own. If you coded the ballots page you didn't do any documentation and vice versa. I would say it would work if the project was larger, but as it stands I think shortening the teams would be good, or increasing the length and difficulty so that everyone has to help with everything. Lecture was fine as far as size. Many issues with sharing printers and the barcode scanner which made work hard. If class size also meant larger teams please refrain from have 8 in the future. Since we all need to participate and the TA's only have a small window into our work, I often felt that I was competing against my teammates. Why help someone when I can do it myself and get the commit credit. 9 people made splitting tasks difficult. Didn't affect me, thought it worked well. The increase in class size did not really matter to me as I was still able to pay attention during lectures. I do not think the quality of the course suffered -- Sherriff did an excellent job keeping the class engaged throughout lecture. I also like working in such a big group for the project. It was challenging, but a cool thing to figure out how to work around. it was fine I do not feel like it affected the course i didn't notice I think it was ok. I assumed it was always this size because it was well-managed I think the course size was fine! I don't think more or less people effected this because there were so many office hours and sheriff made himself availible so often. It didn't matter. It was still a great class. I felt that the class size worked well. There was basically no interaction between different teams so I don't think it really mattered at all. If it affected the project team size, then no, it did not work. Some people insisted on doing every task themselves and left little room for others to contribute. didnt matter Size of the class did not matter, it was more about the size of the teams for the course long project that did matter Personally I felt as if the groups were too big with 8 members. However, that did end up with no one doing too much work for the project giving people time to work on stuff other than 3240 project. The team size was too big for the given project (4-5 people made more sense) but working in a team of 8 is good practice, so the project just needs to be better. Otherwise class size was fine.

~ QUESTIONS AND DETAILS ~	*~ ANSWER MATRICES ~*
	Did not matter to me.
	The size of the course was fine with me.
	I felt that the team size was good
	I did not like it. We needed more attention from the students. And the TAs and Sheriff needed to communicate more. The TA okay'ed us for our project, and then Sheriff shat on it during the actual demo. If there was clear communication, we would have been able to fix the problems. It wasn't as though we were too lazy to implement certain features, but that we didn't know what features to implement. It doesn't help that the TAs ok stuff during our sprints and Sheriff asks us during the presentation why our project is shit.
	I don't think it affected the quality of the course. The lectures were fine.
	I think it worked however there we lack of in class activities earlier which meant that kids could not show up
	My comments in regards to this question are contained in my answer to question 10.
	No. Quality was the same.
	I don't think this mattered to me, except for the team size (mentioned above)
	I can't really say how this has affected the quality of the course because I never heard anything about this class prior to taking it.
	The large class really did not bother me in the slightest and it especially did not bother me in lecture. I thought that our team sizes were appropriate because with a given team there is a big range of skill-levels and also a range in the amount of effort given by each member. This larger team size helps make up for certain members that may not be as knowledgeable or that may not be very dedicated.
	I don't think it noticeably affected the quality of the course.
	class size was fine
	It made the projects more difficult to evenly contribute to. See the above comment. Besides this, there was no difference in lecture quality.
	It didn't really matter to me. The team size was obviously larger, but I think the project was properly scaled to match that.
	The class size in lecture did not really affect my own personal learning. However, the group size was very difficult to manage, as it made it hard to hold every person accountable for a given task.
	I thought the class size was fine. I feel like the team size was also good because it was realistic.
	It did not, I'm thankful that the size doubled otherwise I wouldn't have gotten in.
	Size wasn't a factor
	It doesn't bother me as long as i always find a spot to seat.
	It probably affected the quality of the course a little bit, but it was also helpful cause people need to take it before they graduate and more room is better for that
	It made the projects harder to split up, I felt like sometimes I really didn't know what to do after a couple of my lab mates would take all the functionality and finish it all, it was like I was left with nothing to do or couldn't think of anything to do
	I think doubling the class size was not a problem.
	I don't think there was a problem with the size of the class- I know I was just sitting around distracting people, but that's really just an issue with any class.
	The class size did not bother me.
	Not Matter
	It doesn't matter for lecture, it just sucked in lab with having large groups.
	The team size was definitely too large for the project. I believe only 4 people were doing the heavy lifting in terms of coding. One team member just copied Professor Sherriff's code for the printer the whole semester and barely attended lab and when he did he just did work for other classes and didn't participate. We ended up scrapping his copy paste code because it didn't work. So maybe the teams could be smaller.
	I think smaller groups would've been a way to make people more accountable. I do understand that a large team is meant to challenge SRC control and what not, however, I think 1 or 2 less people wouldve made a big difference

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
	It made class seem less involved. It is easier in a bigger class to do other work instead of focusing on the subject matter.

19. The course addressed technically rigorous subject matter consistent with the course objectives.

~

Question Type: Likert

~

contributed by Dean of the School of Engineering and Applied Science

Results for CS-3240-001

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
89	3.94	0.93	23 (25.84%)	49 (55.06%)	8 (8.99%)	7 (7.87%)	2 (2.25%)	0 (0.00%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
2575	4.41	0.73	1314 (51.03%)	1068 (41.48%)	124 (4.82%)	40 (1.55%)	21 (0.82%)	8 (0.31%)

20. The instructor used methods other than/in addition to traditional lectures (for example, active learning, in-class problems, collaborative learning, in-class discussion) effectively in this course.

~

Question Type: Likert

~

contributed by Dean of the School of Engineering and Applied Science

Results for CS-3240-001, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
88	4.24	0.77	35 (39.77%)	43 (48.86%)	6 (6.82%)	4 (4.55%)	0 (0.00%)	0 (0.00%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
3237	4.04	1.05	1263 (39.02%)	1155 (35.68%)	399 (12.33%)	201 (6.21%)	113 (3.49%)	106 (3.27%)

21. There was a reasonable level of effort expected for the credit hours received.

~

Question Type: Likert

~

contributed by Dean of the School of Engineering and Applied Science

Results for CS-3240-001

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
89	4.09	0.78	28 (31.46%)	44 (49.44%)	14 (15.73%)	3 (3.37%)	0 (0.00%)	0 (0.00%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
2582	4.14	0.98	1074 (41.60%)	1085 (42.02%)	195 (7.55%)	151 (5.85%)	73 (2.83%)	4 (0.15%)

22. The homework assignments helped me learn the subject matter.

~

Question Type: Likert

~

contributed by Dean of the School of Engineering and Applied Science

Results for CS-3240-001

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
88	3.17	1.24	12 (13.64%)	26 (29.55%)	18 (20.45%)	18 (20.45%)	9 (10.23%)	5 (5.68%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
2582	4.15	0.93	1020 (39.50%)	1049 (40.63%)	244 (9.45%)	126 (4.88%)	49 (1.90%)	94 (3.64%)

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~

23. The textbook increased my understanding of the material.

~

Question Type: Likert

~

contributed by Dean of the School of Engineering and Applied Science

Results for CS-3240-001

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
89	2.95	1.38	7 (7.87%)	9 (10.11%)	11 (12.36%)	7 (7.87%)	9 (10.11%)	46 (51.69%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
2584	3.62	1.20	503 (19.47%)	541 (20.94%)	393 (15.21%)	215 (8.32%)	116 (4.49%)	816 (31.58%)

24. The course material was well organized and developed.

~

Question Type: Likert

~

contributed by Dean of the School of Engineering and Applied Science

Results for CS-3240-001, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
90	3.82	1.08	24 (26.67%)	42 (46.67%)	13 (14.44%)	6 (6.67%)	5 (5.56%)	0 (0.00%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
3226	4.07	0.99	1209 (37.48%)	1293 (40.08%)	382 (11.84%)	188 (5.83%)	81 (2.51%)	73 (2.26%)

25. The instructor was knowledgeable about the subject matter.

~

Question Type: Likert

~

contributed by Dean of the School of Engineering and Applied Science

Results for CS-3240-001, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
88	4.42	0.83	49 (55.68%)	32 (36.36%)	4 (4.55%)	1 (1.14%)	2 (2.27%)	0 (0.00%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
3230	4.54	0.67	1951 (60.40%)	1029 (31.86%)	131 (4.06%)	26 (0.80%)	17 (0.53%)	76 (2.35%)

26. The instructor was well prepared for class.

~

Question Type: Likert

~

contributed by Dean of the School of Engineering and Applied Science

Results for CS-3240-001, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
90	4.18	0.98	40 (44.44%)	35 (38.89%)	9 (10.00%)	3 (3.33%)	3 (3.33%)	0 (0.00%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
3233	4.37	0.80	1632 (50.48%)	1192 (36.87%)	234 (7.24%)	58 (1.79%)	39 (1.21%)	78 (2.41%)

27. I received adequate preparation from the prior courses in the curriculum to be successful in this course.

~

Question Type: Likert

~

contributed by Dean of the School of Engineering and Applied Science

Results for CS-3240-001

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
90	3.85	0.93	21 (23.33%)	38 (42.22%)	20 (22.22%)	4 (4.44%)	2 (2.22%)	5 (5.56%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
2583	3.97	1.00	823 (31.86%)	1084 (41.97%)	355 (13.74%)	169 (6.54%)	71 (2.75%)	81 (3.14%)

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~

28. The grading policy was fair.

~

Question Type: Likert

~

contributed by Dean of the School of Engineering and Applied Science

Results for CS-3240-001, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
90	3.64	1.17	26 (28.89%)	27 (30.00%)	20 (22.22%)	13 (14.44%)	4 (4.44%)	0 (0.00%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
3230	4.01	0.99	1096 (33.93%)	1351 (41.83%)	405 (12.54%)	198 (6.13%)	90 (2.79%)	90 (2.79%)

29. The instructor responded adequately to in-class questions.

~

Question Type: Likert

~

contributed by Dean of the School of Engineering and Applied Science

Results for CS-3240-001, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
90	4.30	0.88	45 (50.00%)	33 (36.67%)	7 (7.78%)	4 (4.44%)	1 (1.11%)	0 (0.00%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
3230	4.27	0.86	1446 (44.77%)	1296 (40.12%)	257 (7.96%)	97 (3.00%)	51 (1.58%)	83 (2.57%)

30. The instructor effectively used technology in support of the learning goals for this course.

~

Question Type: Likert

~

contributed by Dean of the School of Engineering and Applied Science

Results for CS-3240-001, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
88	4.39	0.69	42 (47.73%)	40 (45.45%)	4 (4.55%)	2 (2.27%)	0 (0.00%)	0 (0.00%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
3217	4.12	0.95	1234 (38.36%)	1286 (39.98%)	344 (10.69%)	146 (4.54%)	74 (2.30%)	133 (4.13%)

31. The average number of hours per week I spent outside of class preparing for this course was:

~

Question Type: Multiple Choice

~

contributed by Office of the Provost

Results for CS-3240-001

Total	Less than 1 (NA)	1 - 3 (NA)	4 - 6 (NA)	7 - 9 (NA)	10 or more (NA)
90	10 (11.11%)	36 (40.00%)	31 (34.44%)	8 (8.89%)	5 (5.56%)

Results for SEAS, 3000-level courses

Total	Less than 1 (NA)	1 - 3 (NA)	4 - 6 (NA)	7 - 9 (NA)	10 or more (NA)
2583	88 (3.41%)	704 (27.26%)	1052 (40.73%)	432 (16.72%)	307 (11.89%)

32. I learned a great deal in this course.

~

Question Type: Likert

~

contributed by Office of the Provost

Results for CS-3240-001

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
90	3.62	1.16	20 (22.22%)	39 (43.33%)	14 (15.56%)	11 (12.22%)	6 (6.67%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
2577	4.08	0.97	977 (37.91%)	1098 (42.61%)	285 (11.06%)	159 (6.17%)	58 (2.25%)

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~

33. Overall, this was a worthwhile course.
~
Question Type: Likert
~
contributed by Office of the Provost

Results for CS-3240-001

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
90	3.53	1.25	20 (22.22%)	38 (42.22%)	10 (11.11%)	14 (15.56%)	8 (8.89%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
2579	3.97	1.10	980 (38.00%)	957 (37.11%)	332 (12.87%)	196 (7.60%)	114 (4.42%)

34. The course's goals and requirements were defined and adhered to by the instructor.
~
Question Type: Likert
~
contributed by Office of the Provost

Results for CS-3240-001, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
90	4.12	0.92	35 (38.89%)	39 (43.33%)	9 (10.00%)	6 (6.67%)	1 (1.11%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
3216	4.26	0.82	1384 (43.03%)	1449 (45.06%)	257 (7.99%)	84 (2.61%)	42 (1.31%)

35. The instructor was approachable and made himself/herself available to students outside the classroom.
~
Question Type: Likert
~
contributed by Office of the Provost

Results for CS-3240-001, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
90	4.17	0.90	38 (42.22%)	36 (40.00%)	9 (10.00%)	7 (7.78%)	0 (0.00%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
3230	4.20	0.93	1449 (44.86%)	1243 (38.48%)	349 (10.80%)	112 (3.47%)	77 (2.38%)

36. Overall, the instructor was an effective teacher.
~
Question Type: Likert
~
contributed by Office of the Provost

Results for CS-3240-001, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
90	4.19	0.93	41 (45.56%)	32 (35.56%)	11 (12.22%)	5 (5.56%)	1 (1.11%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
3237	4.11	1.00	1367 (42.23%)	1213 (37.47%)	399 (12.33%)	165 (5.10%)	93 (2.87%)

37. Please make any overall comments or observations about this course:
~
Question Type: Short Answer
~
contributed by Office of the Provost

Results for CS-3240-001

Total	Individual Answers
49	*See below for Individual Results*

N/a

I think more work needs to go into assigning team members. Only one or two people on my team had crucial prior technical knowledge and spent the first week or two trying to catch the rest of the team up. This frustrated them and drastically slowed down our progress.

Sherriff is a great professor and obviously makes lots of efforts to make this class more useful and pertinent to our education. But even though he said he would change the course a lot, I really don't see much change based on things I've heard from previous semesters. I just think the course itself needs to be strongly re-evaluated.

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
	Amazing professor! Overall, this is a pretty dumb class. I think the idea of a project is good but the execution is pitiful. One thing he says is that we will not be able to choose our groups in the real world; however, in the real world, if you have idiot group members that do not perform, they will be fired. I wish firing group members was a thing. Me and one other person carried our group and that is the only reason we did well. GPA should also not be a factor when picking groups. You should get one partner to be on your team that you get to choose. Also I don't understand why we had to make user stories when in the real world that is more of an interaction with your client. We were just making up BS along the way. Overall, this class was just dumb. Lectures were ok but I feel like everything in this class is what you will learn in the real world. Project was stupid. Grading was extremely subjective and arbitrary. Grades did not fully show how much effort I put into the class. I'll still likely get around an A but so will my group members despite them being dumb. You always think everyone at UVA is smarter than you until you get into a class with them... Finally I think Sherriff is one of the best professors. He is one of the best at truly understanding his students. Alright I'm going to end on that. I hope you will take what I said into consideration. This felt like a boring and dry course and Professor Sheriff did an amazing job to make it fun and entertaining. The class content itself does not give a fun outlook of the subject. It seem like a difficult thing to make enjoyable and should just be integrated into other classes as I don't see a way to make this fun. Once again, Sheriff did his best to make the lectures funny and entertaining and he was fairly successful in it. Great Professor, but a challenging course to teach. The TAs were not very available or helpful, unfortunately. This class only sucks because so many people are at different levels with respect to web development and django familiarity. Therefore it was very hard for me to shine because i didn't know what I was doing most of the time relative to a lot of other students. And that's not for lack of effort, either. N/A I understand that Sherriff expected us to learn a lot of things in this course by ourselves, however he was extremely unhelpful when I went to his office hours. He essentially said I don't know you try to figure it out by yourself. In addition it seemed like he was always changing objectives in relation to the project 10 hours for lab. Sherriff is good and made this class bearable, but it still doesn't seem to teach much of anything important The course overall felt well structured and well presented. The lecture material was interesting and presented well. I've already stated my issues with the project, however, so I believe that software dependencies and group size should be looked at for future semesters. Loved the class! Hi you're an awesome professor and person - thanks for being great. I think this course should be moved up to 2nd year curriculum. A lot of topics in this course have already be covered in other classes or an internship. Sherriff: 5/5 stars, good lecturer Material: 1/5 stars, holy shit this is so boring Project: 1/5 stars, no surprise a 9+ person team with no project manager was a shitshow. Overall: Sherriff you are the only good thing about this course <3 Professor Sherriff is great, and I wish I had more professors like him. The only thing I would change about the course is the name. Advanced Software Development should be changed to something regarding Software Engineering since that's more inline with the information that we covered. Decent course. The material is not too exciting but important to learn. The project was difficult without prior background experience with the technologies. Prof. Sherriff is a great, entertaining lecturer. I would certainly like to take another class with him, preferably something more interesting than CS 3240, but the waitlist for Mobile is sitting at 120, so RIP. I think 5 is a good team size for the project. I don't feel like I gained anything from this class lol I'm sure other people are going to hit the points of this course I lot better than I could. So I'm going to omit any sort of rant-type response here. I can say that this course needs some work. I kinda wish I had waited until the next semester to take it, because this course kind of felt like a guinea pig this semester. It might be better if there was more introduction to materials such as github and docker during early labs, instead of having the python labs, as I think python was easy enough to learn on our own, and could have been assigned as early homework instead. Good course to take wish I learned more technical aspects such as Django development. Learning felt like it was dependent on group. Grading was not very clear.

~ QUESTIONS AND DETAILS ~	*~ ANSWER MATRICES ~*
	Thanks for the wonderful class It is a bit hard to let everyone gets involved in the group project. Since we are all new to this kind of working style, cooperation is not easy. I think the team size could have been a little bit smaller, or the project could have been a little more intensive. At points we split up tasks that could have easily been accomplished by one person into tasks multiple people could work on, just to give everyone a fair amount of work. Great course!!! Fun project. One of my better classes this semester. Professor Sherriff makes the course a lot more enjoyable. It is a shame to see how my views on software development and Sherriff changed over the term. Up till test one I was really on board with the course and interested to see how the project would progress. As time when on I was let down again and again. Sherriff in the guise of friendliness fosters an atmosphere of unprofessionalism and emotion. Sherriff all to often got angry at questions of "when will the API be given" or "are there more printers" or "how should we go about doing this" and then tried to blame it on us that were where being jerks and or impatient. Other times when questions where asked on the project that Sherriff did not have the answer to yet he brushed them off and said he'll tell more later or when its "done". Our team nevertheless our team had to make assumptions to move forward. When these assumptions turned out wrong it was still our fault for having made poor assumptions in the first place. Sherriff also took it as a badge of honor as for the dryness of software development. He said that it is in many ways similar to broccoli. I like broccoli. Is it my favorite food? No, but I have no issue with it. Similarly I have no issue with the content of this class but THE WAY it was force fed to us by and overgrown man child who did not have plan and put the burden of responsibility to use the broccoli on the students it too many cases. Furthermore the EVIL hat to me felt like a poor way to cover up slipups on the professor's part in the notion of portraying a character. ***** 5 star rating It makes sense that the class was a little disorganized given the short prep time, but the project was really difficult with so many people and so little information. The grading is also really subjective and you can lose so much credit for one person on the course staff not feeling like giving full credit. I would design a much more interesting project than a voting system for next semester to at least make it easier to want to work on the entire semester. Sherriff is an absolutely fantastic lecturer. He's the best in the department. Actually Bloomfield is probably the best, but Sherriff is a close second. I never went to lecture, but it was so easy to focus through his podcasts (even though I have severe ADHD, and the material is boring as hell) since he's such an engaging lecturer. The project sucks. The team size sucks. Trying to simulate the real world is super unnecessary. Students will learn these skills better in their first week of work anyway. I'm very, very glad the CS curriculum is being redone. I think it would be very wise to factor just how insanely unnecessary this class is into the redesign. Take git and very very basic web stuff (intro html/js) and put in into a much earlier prereq and then take 95%+ of the software engineering knowledge stuff in the curriculum and put it in some kind of class that isn't required for every CS major. Unrelated, but make BA CS kids take OS. The one Homework Assignment that we completed before we learned anything in the course shouldn't have been weighted as heavily as it was, nor should it have only checked three test cases. The class was generally enjoyable. - I liked Sherriff's lectures. - I didn't like paper activities (maybe except the coffee one, that was okay) - I liked the coding activities like the square peg round hole activity, the maintenance activity, and the singleton activity. Fun, helpful, and non-stressful! - I was afraid to ask Sherriff questions. In my experience he makes you feel kinda bad when you ask him stupid questions, and I'm pretty shy. - I thought grading was fair - it would be cool if you did unity demo for decorator pattern! (assuming I understand the decorator pattern correctly) Project team size for this semester was probably one to three people too many. The project difficulty was on par for what this course is defined as. I would like to say that changes to the syllabus should be announced in advanced rather than just done(like switching to more in class activities instead of attendance...). I wish the course came full circle and talked a bit more about software testing (highly important, especially if the aim is to replicate bits of the real world). I thought that the project was great. It was a good length for the number of people because yeah, even though we probably could've gotten it done with 4 people, having 8 people made the coordination and communication much more difficult and I think the difficulty was worthwhile because it was realistic to what we would see in the real world. It was really annoying because we never got clear information about the project but again, I think it was worthwhile because it was realistic. You should probably grade assignments a little faster though because we waited like, 2.5 months for our homework which was annoying... Overall it was good, you did a good job. :)

~ QUESTIONS AND DETAILS ~	*~ ANSWER MATRICES ~*
	Professor Sherriff, I hope you do not write this comment off as a bad semester or a pissed off student. It was a bad semester but that had nothing to do with how bad of a professor or person you were. I remember how you mentioned in the beginning 4 lectures about how you would never grade attendance in the class. Then, at the end, you changed the rubric from 20% final exam to 15% final and 5% class work. I didn't have a problem with that because I agree with you: attendance was way too low. But, it was the fact that you changed it sneakily and never told anyone about it. That was incredibly dishonest and lacking in integrity and I think that tells a lot about you as a person and as a professor: you let your emotions control you far too much, it's childish. In fact, I didn't even take your ethics lecture seriously because, to be frank, you don't have the right to lecture anyone on ethics after doing what you did. Furthermore, I have talked to you outside of lecture and the fact that you get childishly angry about nothing is painfully obvious. In fact, I have friends who have told me you yelled at them during OH for asking clarifying questions on a project that was so clearly unorganized and disappointing as a whole. Seriously? I never want to take a course by you again after hearing that and honestly, you need to grow up as a person. Learn to control your emotions, you are not a child anymore. Anyway, I hope you actually read this and take it seriously because you can improve a lot as a professor and a person if you stop letting your emotions control you so easily.
	The project needs to be rethought, or at least curved. Project grades were entirely dependent on your TA, and not all TAs graded fairly. One groupâs project didnât even work and they got mostly 5s on the sprints, while another team met all the requirements and got some 3s and 4s. Also, several times throughout the semester, a team member would do my part of the project while I was still working on it, and I was forced to scrap everything. It was very frustrating, and because of this I donât believe my commits or scores accurately reflect my effort. The course was interesting and I liked the concept of the project, but found many issues with its implementation and evaluation.
	This class just felt super arbitrary and unplanned. We spent a bunch of time doing failsafes I don't know we needed to do or not. I have no idea if you guys actually looked at github/youtrack, and if so how specifically you evaluated us from it.
	Major structural concerns with the course the can only be traced back to Sherriff's failings as a professor. 1. Didn't really teach. Probably could have gotten more information from reading a book about software dev. 2. Horribly unclear on deadlines and expectations. 3. Evaluation metrics were a black box. 4. Material was poorly organize and actively took away from learning experience.
	I think that this is a solid class. The project we worked on this semester was interesting and allowed me to learn a lot. The class material, too, was insightful and I have learned new ways to think about and tackle problems. However, the class and project began to grow apart and become two distinct entities as the semester progressed. I feel that a bit more unity between class and lab could be beneficial.
	Lectures were engaging and Sherriff made the material interesting to learn by either tying it to personal stories from working as a software dev or by giving us in-class activities. While a lot of people probably dreaded the in-class activities, I actually thought it helped foster conversation with the people sitting near me and actually helped me learn the material a lot better. The lecture slides were not too dense so that the lectures were boring, rather they had the key points and Sherriff did a great job of expanding on each one.
	Lighten up on the project. You're penalizing students without giving them adequate guidance. We would've been able to build for the feature you were looking for if we knew what you were looking for. I understand that that is a real-world concern, but I also have a GPA concern as well. You're ruining my future.
	Sherriff is the best but TAs were really bad (didn't know how to help, unfair grading). Also the team was too big.
	For one, I think it's worth noting that I personally didn't like the course material (thought it was rather useless and didn't seem like a good use of 3 credits, more on that later), but Sherriff did teach it well and was helpful at certain points. However, the organization of the course was quite terrible. The grading for the first HW wasn't provided until way later than it should have been, and the guidelines provided weren't clear enough - TA's took off on very subjective criterion because of vagueness in how it was explained. I thought the project was awfully implemented - no concrete guidelines as to how it was going to be graded overall, how to implement it, and what our individual grade would consist of were given. There was a disconnect between what the TAs told us and what Sherriff told us in class, sometimes with Sherriff telling us certain things the TA's would then take off for on the sprint grade. The TA's were quite awful - they had entirely subjective criteria for grading, both for sprint grades and for individual grades, and were not reliable in any sense to ask questions for. In reality, none of the software development techniques we learned in class were used on the project - making me think that the content would be better off restructured to teach modern frameworks and techniques used in software development (Docker, Django, REST API's, etc), the same way we learned about GitHub. The assumption that people would know these techniques off the bat/could learn them on their own makes no sense to me given the time required to self-learn such things far exceeds that to learn, say, the principles of agile development or how to do a functional decomposition. In general, many software professionals talk about how the in-depth knowledge they learned for their degree wasn't useful in "the real world", and I was hoping this class would be a remedy to that problem. Instead of learning useful stuff for real-world work however, I am left with a feeling that I completely wasted my time in learning things that I will probably learn anyways from a company's policy training, and bitterness towards a project, TA's, and professor that forwent standard conventions of providing a proper timescale, grading rubric, and defined criterion, all for a facade of "emulating the real world".
	The group was too big. I know the goal was to make us work in a team like environment, but we are being graded as a team. I had team members do nothing. The TA knew, and nothing was done. I could not get team members to contribute, and the TA's should have tried to do something in this case.

~ QUESTIONS AND DETAILS ~	*~ ANSWER MATRICES ~*
	The project just didn't hit with me. I ended up doing a fair amount of the work NOT because I was bossy/my team sucked, but because the project just wasn't good for a team setting in my opinion. I have nothing against most of the team, but me and one of the others could have gotten the project done in less time, with better coding standards, and with better techniques than the 7 - I believe that's because the project's scale does not match what the course is meant to teach. This might just be with my group, but the attempts to make us use "good software development techniques" (such as Youtrack, github branches, multiple methods of communication, standup meetings, etc) all just felt like they were bs'ed away. It's because they were extraneous and unneeded. A real workplace will have much better defined goals and a much better structure; this did NOT simulate that well in my opinion. Furthermore, the TAs simply were not helpful. Part of this is with the project not being fully developed, but most groups likely fell victim to overengineering their system because of the crappy requirements. Why is there only one election a month? Why does this one election not have different ballots based on location, as most elections do? My team spent most of the time figuring out how to cater towards a REAL voting system because we are used to that, but it turns out that wasn't even really the scope of the project... the scope seemed more like a polling site. It just didn't match the expected scope. The course itself was fine. I enjoyed it more than I likely would have under many circumstances. Sure, it's hard to teach and definitively not fun, but not much you can do about that. Sorry if it sounds like I'm railing too hard or in bad ways - I just found the project to be implemented almost exactly wrong. Its a strange class, but I think the long term project saved it from being really boring. That being said, I actually wish it hadn't been the voting polls thing.