CS 3240-002 **Advanced Software Development** - Fall 2019

ENGR (18474)

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

Respondents: 70 / Enrollment: 133

Summary: CS 3240-002 Advanced Software Development - Fall 2019 (18474)

Overall Course Rating

CS-3240-002 Mean 4.18
CS-3240-002 Std Dev 0.95
CS-3240-002 Response Count 347

SEAS, 3000-level courses Mean 4.07
SEAS, 3000-level courses Std Dev 1.02
SEAS, 3000-level courses Response Count 12930

Overall Instructor Rating

INSTRUCTOR: Sherriff, Mark
Mean 4.62
Std Dev 0.74
Response Count 140

SEAS, 3000-level courses Mean 4.49
SEAS, 3000-level courses Std Dev 0.78
SEAS, 3000-level courses Response Count 6585

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

1. How accurate is this statement for you: The XP system was easy to understand.

~
Question Type: Likert
~
contributed by Sherriff, Mark (mss2x)

Results for CS-3240-002, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
70	3.81	1.18	24 (34.29%)	25 (35.71%)	8 (11.43%)	10 (14.29%)	3 (4.29%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
70	3.81	1.18	24 (34.29%)	25 (35.71%)	8 (11.43%)	10 (14.29%)	3 (4.29%)

2. How accurate is this statement for you: I prefer the XP system to "normal" percentage-based grading.

~
Question Type: Likert
~
contributed by Sherriff, Mark (mss2x)

Results for CS-3240-002, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
70	2.99	1.44	16 (22.86%)	9 (12.86%)	17 (24.29%)	14 (20.00%)	14 (20.00%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
70	2.99	1.44	16 (22.86%)	9 (12.86%)	17 (24.29%)	14 (20.00%)	14 (20.00%)

3. Any comments regarding the XP system or Dashboard tool?

~
Question Type: Short Answer
~
contributed by Sherriff, Mark (mss2x)

Results for CS-3240-002, Sherriff, Mark

Total	Individual Answers
54	*See below for Individual Results*

I personally didn't feel any difference, because I felt like I automatically converted from XP to percentage in my head anyway.

Very indifferent to each style of grading. In the end they're exactly the same thing just displayed slightly differently.

At the end of the day, it's just normal percentage grading, but harder to understand. Also harder to gauge how you're doing until the end.

XP System is very tedious and furstrating to keep up with. It gives no clear indication on how you stand in the class. Any other class implementing this system, I would have fine with as learning is more important than grades, but not for this class.

It was self explanatory and pretty easy to understand the grade distribution if you looked at the excel sheet

It feels the same overall, and is kind of confusing at times because for most of the semester I don't know what grade I have

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
	Dashboard isn't bad in that it requires fewer clicks that Collab gradebook does, but XP isn't any better than the typical grading system. If it were a true XP system, we should be able to gain XP if we spend more time and effort besides doing the required assignments or activities. This is just a different representation of the exact same grading scheme, but it fails to serve its intended purpose, which is positive reinforcement.
	I think it's much easier to calculate the points I've been deducted than what I've already earned. With the XP system, it's especially hard to understand how well I'm doing in the mid of the semester since the grades are not complete.
	I liked that you could earn back points on the XP system, but it was really stressful not knowing your grade and sitting at a below Failing grade for most of the semester.
	I liked it, many don't, and most of them do not have a legit reason. First of all. I like the positive incentive idea, and I think it works great, both theoretically and practically. Secondly, when people disagree with that, 90% of them are just not used to a new idea. They resist change, which is acceptable, but it's not good when they relate other anger for the course to the "unfairness" of the XP system. IF you were to ask anyone, what is unfair about it, I bet they can't even list one good reason
	I understand your intention behind the XP system, and the logic behind trying to shift the class to be more focused on additive rewarding over punishing for things going wrong. But with this XP system I still simply found myself calculating how many XP points I was missing from the total available so far. So I basically just had to go through more steps to figure out what I had missed, which completely defeats the purpose of the system. And I know many of my peers did the same thing. I think that having a more standard grading scale that would allow people to see how they are doing in the course without having to jump through hoops would be much better.
	very cool
	i liked the normal system more to be honest
	Hard to understand at first, once I got it, it was fun to use and monitor.
	Sometimes it was hard to get a feel for exactly what grade I would be able to get. Having all the assignments on the dashboard was helpful (even future ones with 0's).
	I liked the visuals of the XP system, but it would also be nice to see the normal percentage-based grading, too. Knowing how much XP to the next level didn't really help me visualize how I was doing in the course so far. Perhaps have the progress bar with current points / total points and markers for where A, B, C, D are at. This visualization would help me more than the current one with the bar going up only until the next level such as "1500 exp until the next level".
	Have gradescope be more clear by not including the roles that you did not complete in the group.
	It was confusing and hard to tell how well you were doing. It also seemed to have a lot of errors and seemed more work than was worth it.
	Cool idea, but it seems like more work for professors and doesn't really help students.
	You never know what you have until the end, it is easier to track with regular grading. However, it does feel like the xp system makes it easier to get a good grade, which I like.
	The only thing I did not like about the XP system is that it was hard to track your grade over the course of the semester. You didn't have an idea of what your grade was until the last week.
	The hardest part about it was how much XP is still up in the air by the end of the semester, so it's hard to see that your grade is below an F, even though with all the grades it in could be an A. Maybe some guidance on the projected amount of XP could be valuable.
	N/A
	You might want to separate out "completed" and "not completed" items so that it's easier to see how much of the grade was still to come. Also, maybe have some indication of how you are doing relative to how many points you *could* have earned up to this point. Just comparing to the final grade is nerve-wracking at times because you don't know how well you should be doing.
	It kind of made it hard to figure out where I stood throughout the semester and made it so I wasn't really sure what my grade was until the final grades were put in. I understand the sentiment but it feels like it's just the exact same system as traditional courses only with the added element of students being in the dark about their grades until the end of the semester.
	I think the XP system wouldâve been more helpful if we could have more of an idea where we fell prior to the final project grading, because it wasnât very hard to tell where Iâd end up before the very end of the semester.
	The Dashboard was nice, but the XP system was not useful. I don't think the idea of "earning" points as opposed to "loosing" points helped me feel better about the course, and nor did it have the advantage the XP systems have in video games (the ability to choose what to do to earn XP). We were never given a choice between two things to do to earn XP.
	I liked it. I liked the grading philosophy which accompanied it as well.

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
	The only thing I really found frustrating about the XP system was that grades would take a while to be uploaded from Gradescope to the Dashboard. Overall I largely prefered the XP system, however, if would've been nice if the XP table included in the syllabus was also available on the Dashboard so I didn't have to flip back and forth between the Dashboard and the syllabus.
	It was fun to visually see my grade.
	My only real issue with XP is that it implies some level of grinding that would let you earn minimal xp for some repetitive work.
	Gamification of grades doesn't really do anything for me. Just have our grades available quickly so we can calculate our averages.
	The whole "We don't know what will be an A or A- yet" spiel makes zero sense. You should always make it clear and obvious what the standard is, especially for a goal that everyone is trying to achieve.
	It was kind of weird to see if I was on the right track for the grade I wanted or not by using the XP system.
	It's a really cool concept, but for high-strung UVA students, working all semester and looking at your dashboard to see you have a D/F because of how the XP worked made it pretty difficult. Maybe do hit points or something? I like the game aspect but the system probably produced more work for the staff and more stress for students than necessary.
	I donât feel like it added anything. Earning 8000/10000 XP is clearly the same as an 80% on an assignment
	Other than human error, I actually really liked the system. TA's seemed pretty unclear specifically with sprint grading, where sometimes Travis being slow would stop us from fixing errors in time, but then we would get 0 credit for a while until we addressed it in lab.
	I really like systems that clearly lay out weighting of individual assignments rather than just displaying total averages. It allows you to see total subtractions from a 100 so you always know the max grade you can get. I think things are a bit clearer in the system that Archimedes uses (see COA1 and COA2 systems) where it was a grade earning system like exp but didn't have the oddness of the level of the student and exp needed to level up. Like most MMOs end game tends to be the only part that matters so having intermediate levels doesn't really do all that much as it doesn't pertain to the final grade (which is what everyone is concerned about). I know there have been gamified elements in the game design class with Prof. Floryan involving currencies and consumables. I think this would be the only way to make the exp system meaningful (but then you deviate too far from what the Software Development course is aiming for and step on the toes of game design). In its current state it is just a glorified point counter that masks what's going on.
	I thought it was difficult to factor in the weight of grades, especially since the XP range wasn't clear if it was set at the start of the semester. It also made some things seem trivial - for example, "I lost 2,000 points on a quiz." Which to anyone else that sounds like a ton, but when it is out of 200,000 it is not much. Hard to gauge and for people who don't enjoy the level up video-game-like system, its not the most stress free when you're grade is in the "D" range for a while simply because of grades not happening yet. Its really hard to know where you stand in the class. Averages help a lot more with that.
	While the XP system is innovative and interesting, it is unconventional and makes me a bit anxious. I find myself converting to the normal percentage-based grading at various times throughout the semester to see where I stand, and it would be easier for me if I could just see what percentage grade I had. I do like the idea of positive reinforcement of "earning points" rather than "losing points", but seeing myself with low XP earlier on in the semester made me feel uneasy. The dashboard tool is easy to use, and I like how we can see feedback and comments.
	I didn't like that it didn't update regularly. It's been weeks since my grade on gradescope was updated but it hasn't updated on the dashboard.
	None.
	I think it was a cool/creative idea but it created a lot of confusion as to where I was at in the class (grade-wise).
	It just seems like a convoluted way of assigning points and percentages to assignments
	Love the schedule being available to us. XP was rather hard to follow and comprehend when compared to traditional grading guidelines. Maybe move the entire dashboard and grading system to its own website that is linked from collab?
	If you have a conversion table like "2000 XP ~ 1 point", it will be fine with any options.
	In my opinion, the XP system makes sense but makes it difficult to understand how you're actually performing, which makes it useless. I ended up tracking my grades externally, which just added a layer of inconvenience (then again, I think the collab gradebook is also bad, so...).
	It is hard to tell where you stand at any point in the semester based on XP and Max possible alone
	I really liked the Dashboard - made it easy to see my assignments and progress.

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~

the only negative i have is that it's hard to tell what your grade in the class is thus far.

I wish certain categories (Guided Practice, Lab Attendance) didn't have a point cap

I enjoyed this very much after getting used to it. Would recommend again

I think it's cool and new and makes students have fun with it. It's not that difficult to understand.

It was kind of fun, but definitely made me sweat a little not knowing what the threshold was for an A

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

~

Question Type: Likert

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-002, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
70	4.27	0.87	32 (45.71%)	30 (42.86%)	4 (5.71%)	3 (4.29%)	1 (1.43%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
70	4.27	0.87	32 (45.71%)	30 (42.86%)	4 (5.71%)	3 (4.29%)	1 (1.43%)

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

~

Question Type: Likert

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-002, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
70	4.19	0.87	27 (38.57%)	35 (50.00%)	3 (4.29%)	4 (5.71%)	1 (1.43%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
70	4.19	0.87	27 (38.57%)	35 (50.00%)	3 (4.29%)	4 (5.71%)	1 (1.43%)

6. 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-3240-002, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
69	4.39	0.83	39 (56.52%)	20 (28.99%)	9 (13.04%)	0 (0.00%)	1 (1.45%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
69	4.39	0.83	39 (56.52%)	20 (28.99%)	9 (13.04%)	0 (0.00%)	1 (1.45%)

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

~

Question Type: Short Answer

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-002, Sherriff, Mark

Total	Individual Answers
61	*See below for Individual Results*

For me lecture wasn't all that interesting as I had taken SDE the semester prior (and my attendance wasn't great as a result). That being said, I think my favorite lecture was ethics. It's not something I really think about a lot, but is crucial to being an effective and conscious programmer.

software security! tbh also just loved how the "too close to home" stories from industry too

I found the architecture and design unit to be the most interesting as it introduced me to so many concepts in software development.

My favorite was actually learning about all that goes into software engineering, especially different architectures and ethical principles.

Learning about Agile seemed super relevant to the real world

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
	scrum v. plan-driven, most applicable to real-life. I liked the MVC material Requirements Engineering because I liked learning about how to actually write requirements (functional, non-functional, and constraints). I also liked learning about the different elicitation techniques. I enjoyed learning more about agile vs. plan-driven because I learned about agile at my internship over the summer and it was useful to dive into it a bit more I liked the one where professor sherrif talked about the mortgage problem and maintenance. Real world example. The first lecture because I realized I would get to spend the rest of the semester taking an awesome class with an even better professor :) My favorite topic in this course was requirements elicitation and going out to get real answers from students. It was fun to see what other students found as a nice requirement to have for the project that we didn't initially think of. Anytime we talked about real-world applications I thought that was fun and I enjoyed those activities. It was also good to be aware of the code of ethics. Probably MVC and architectural patterns, it definitely helped me understand how a lot of frameworks I've used in the past work, and gives me a much better conceptual understanding on what needs to happen to add a feature into a webapp. The software development process was helpful since it showed the way the real industries work as SW engineers. none They were all great, I like learning about these big ideas, I do think they should be taught in a requried class, at some point in the major Maintenance because covers all phases Ethics. I found the Code of Ethics interesting and very applicable Hmmmm. I think that security is important to cover for students so I'll say that that was my favorite that students who didn't know about those things got exposed to them. All the "non-sexy" lectures, like code maintainence, ethics, etc. They provide a different perspective on CS that you don't normally get until it hits you in the face in the workforce. Code Smells I really liked the unit on system architecture and learning about different frameworks. Security MVC and Rest because they seemed useful and new, whereas everything else we covered felt kind of obvious I like most of the lectures. MVC and modularity felt most relevant to industry and OOP actually learning WHAT software engineering is First day of class was the best as we had little idea of what we were in store for. After that, the lecture were very bad as it didn't connect with the project which was the bulk of the class. It feels as though we were taking two different classes with the different lecture and lab structure. Getting to see our application used by another person was super cool. Seeing the application running on a laptop outside of your team made the experience feel a lot more real. lectures were all interesting but none stood out in particular Design patterns because I already knew most of the rest, and the patterns help me program better. I liked learning about Models, Design Patterns, and Management because these are useful when talking to other software engineerings about. In addition, I liked learning about the history behind them. It cleared up a lot of confusion between all the different concepts and terms. Code smells. It was entertaining and really useful. I thought ethics was the most fun to learn about

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
	They were all alright, definitely enjoyed sherriff's enthusiasm/humor. It helped keep the most boring lectures interesting. Security because security it my favorite part of CS Probably code smells and maintenance because it helped to identify ways to refactor/improve code overall which is very useful! Good question, ethics probably, it's always fun to discuss what happens when faced with a moral dilemma. Security was interesting. Different code patterns because I didn't know about them initially Ethics Modularity because it was an interesting way to think about Development. I enjoyed the Agile/Plan Driven lectures. Those concepts really helped me understand a lot of the lingo that was thrown around at my internship. Code smells, because I like refactorings. I really enjoyed the lecture in which we dived into the various framework options as this expanded my knowledge of tools I may be using in my future career and was an interactive/engaging class. Design decomposition, it was very interesting to take a function or a class system and break it down into its individual components. The brief time spent on REST. Design decomposition I liked Professionalism and Ethics because it focuses on not the code. It's something you don't get taught otherwise. Development tools; it was interesting to see what all tools are available for us to use and see why we chose to use certain things for the class and project. I also loved the methodologies lecture. Ethics. Needs to be taught more in CS Ethics because I am kinda a geek for that stuff. I liked the requirements topic. I was requirements manager, I enjoy designing systems, and I enjoy that end of engineering more than I enjoy learning about systems' architecture. Also, I liked the lesson about tech debt and code smells because I thought it was very relevant, helpful, and something no other class has taught. My favorite topic was the ethics session since that is not something that I had not heard about prior to this course and a topic I believe is worthwhile knowing. Agile methodologies Maybe testing just because we do not learn too much about it. testing I thoroughly enjoyed the Software Development Methodologies lecture. The Polar Chart guided practice was easy to understand and helped me enjoy the topic also. Licenses was good to learn about. I have always seen these licenses but didn't have a good understanding of what they were. Software development techniques was also interesting. Code of Ethics. Those are the most difficult problems that arise and are the most thought provoking
8. 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-3240-002, Sherriff, Mark Total: 60 Individual Answers: *See below for Individual Results*
	I think the architecture lectures will be most useful in planning future projects "Code will haunt you"

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
	Source control management scrum v. plan-driven management I think many offer equal utility for the future Design Patterns Verification/validation I liked the Django Assessment from the beginning of the course. It helped me understand how to work with Django and be a solid contributing member to our project. Code maintenance and design patterns. I think the entirety of the software development process unit will be a major facet of my career going forward. Agile/Plan Driven Lectures MVC and GitHub Design patterns and code smells--A lot of the examples I felt that I could really relate to, and helped me put words to knowledge I already sort of knew. Architectural Patterns (REST API) professionalism Agile development Risk management. Maintenance maintenance The MVC Material Development lifecycle I think the topic I will find the most useful in the future is maintenance since like it is stated in lecture, that is the longest portion of the software cycle. Topics of the software development cycle; however, this can be learned in the matter of 10 mintues on the internet, no need to have three different lectures on the same topic. I think the MVC lesson was highly important because it clarified a lot of the 'magic' that happens behind web apps. Speaking from internship experience, architectural patterns. Security Code smells or design patterns, they apply to almost any role in software development. The lecture about GIT is the most helpful since I have a deeper understanding of how git works and learn how to use github for sharing codes. Frameworks and design patterns both seem really helpful. Code smells literally every single lecture. Looking at the different workflow philosophies and ethics as I will encounter those while on the job. the software process lectures seem like they will be useful in a real world setting. Maybe the ones about development methods All of it to be honest. MVC Testing Testing

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
	The overall softwared development process Modularity because its the most universally applicable for all types of coding. The methodologies lecture; it's the kind of practical that's hard to get any other way than in a job or internship and I appreciated being able to have that knowledge going into a practical setting. Maintenance + security I think i will probably find Maintenance the most useful, because it rally does make up the majority of nearly every single software development process... This class puts students in an environment that is similar to a real-world job and teaches us about how a team operates throughout the development of the team's software. All of it (?) Definitely the lectures corresponding to ethics. While this is a topic that is often brushed over, it is important to know about copyrights, the moral code that all software engineers should follow, and how ethics plays into development in general. Agile vs. Plan-Driven. Most other places don't teach this. Keep this up code smells/ security MVC and Rest Software Process: we went over the phases of development which were really useful Requirements Professional Issues, because I didn't know what GPL was. earlier lectures on the general software development process, methodologies, requirements elicitation Most of them since a lot of them tie into the same overall topic. MVC Scrum and Plan Driven methodologies are words I hear all the time in job interviews. The most useful lecture to me was the lecture on Code Smells. It was cool to see specific patterns to look out for that have known fixes that aren't necessarily language specific. requirements and overall differences between plan driven and agile The ethics lecture was definitely the most useful and easiest to relate to. Development strategies, agile vs plan driven.
9. 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-3240-002, Sherriff, Mark Total: 54 Individual Answers: *See below for Individual Results*
	Ones that talked about dated technologies Can't think of one I do not think there was a lecture I could say was not useful at all. N/a None in my opinion Security felt like an afterthought. It was interesting, but needed more time to actually be useful. Design Patterns... it generally covered rules of coding I pretty much already knew. The lecture on design patterns seemed too long for the content discussed. It is nit-picky, but V&V seemed to drag on for a while even though they are simple topics (I could have also felt this way due to already learning it in SDE earlier). V&V is not not useful (meaning it is useful) but dedicating two days to it seemed excessive.

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
	MVC stuff was presented super confusingly, especially as a new SE student.
	Ethics, it's taught in so many courses
	Verification and validation, the entire topic feels like it could just be summarized in like four sentences (not meant to be snarky, just saying that the detail that we went into feels unnecessary).
	frameworks
	EULA stuff
	Code smells could have been a small worksheet or something not a 75 minute class
	N/A
	N/A
	N/A
	N/A
	Professionalism - a lot of the information was glossed over.
	random small things
	Risk management
	Most of the lectures taught. Reading off powerpoint about topics students can learn in a few mintues rather than helping learn the tools we used for the project was very furstruating and not helpful.
	Hmmmmmm.....I can't remember.....oh....eh idk
	Basically everything
	Not much!
	None. Some might say mvc here, but it's an important topic to understand present day plus helps us new kids related to older folks that used this early in career. I say this as knowing about mvcs would have helped me communicate with older devs better this past summer instead of me learning this stuff on the fly
	Verification/Validation lecture seems a little long
	After talking with other students it seems many of them did not understand branch management, or least to the extent that they bought into its usefulness.
	Risk Management, because it is too obvious.
	MVC and REST: hard to understand
	The design patterns lecture could have been better placed in the sequence; it wasn't as helpful coming at the very end of the class.
	t e s t i n g
	I don't really care about ethics, but I know you are required to teach it.
	MVC and REST seem like good concepts, but don't feel entirely applicable aside from being buzzwords
	Software security, while very important, is in an awkward time in the semester where it's hard to fully appreciate the concepts.
	None
	None, I think
	I know testing is a useful subject matter, but I feel like the Travis testing wasn't all that useful and just a tedious matter.
	Ethics...Most of that was pretty common sense.
	I think that some topics had too much "fluff" and wasn't as important as other topics.
	For me, the more back-end data modeling lectures is just not what I ever plan on doing in my life. But, I think for many others it was important.
	Requirements

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~

Requirements

n/a

n/a

idk

None.

Design pattern. seemed redundant

I think all lecture topics were useful in some way

Django, while a framework with potential, is too difficult to just learn in just one day. It was unfair to make this a matter of 10000 XP with the practice assessment. Either teach us how to use Django, or make it worth much less.

Too much of the grade is left to the last week of school.

I think all slides cover everything.

design patterns - learning about frameworks was enough

10. 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-3240-002, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
69	4.04	1.01	22 (31.88%)	15 (21.74%)	10 (14.49%)	5 (7.25%)	0 (0.00%)	17 (24.64%)

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)
69	4.04	1.01	22 (31.88%)	15 (21.74%)	10 (14.49%)	5 (7.25%)	0 (0.00%)	17 (24.64%)

11. 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-3240-002, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
68	4.06	0.93	20 (29.41%)	16 (23.53%)	11 (16.18%)	3 (4.41%)	0 (0.00%)	18 (26.47%)

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)
68	4.06	0.93	20 (29.41%)	16 (23.53%)	11 (16.18%)	3 (4.41%)	0 (0.00%)	18 (26.47%)

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

~

Question Type: Multiple Choice

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-002, 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)
70	0 (0.00%)	1 (1.43%)	25 (35.71%)	15 (21.43%)	6 (8.57%)	23 (32.86%)

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)
70	0 (0.00%)	1 (1.43%)	25 (35.71%)	15 (21.43%)	6 (8.57%)	23 (32.86%)

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~

13. 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-3240-002, Sherriff, Mark

Total	Individual Answers
40	*See below for Individual Results*

Maybe coming together as a class to decide on slightly more detailed bare bones feature set for each project that all teams must implement. This could be done after requirements are collected.

keep it like that

It would be cool to be shown / demo the super good projects, to spark ideas and maybe evoke competitive spirits.

Enforce more Django tutorials to even the playing field at the start.

5 PEOPLE IS WAY TOO MANY FOR THIS. GO TO THE INTERNET SCALE APPLICATIONS MODEL AND DO 3 PEOPLE. I took ISA and 3240 at the same time this semester and I felt like I learned everything I needed for 3240 in ISA and then was able to apply my django knowledge in 3240 lol, which is paradoxical considering it's supposed to be a prereq

no

no

The thing I found most frustrating was that there was a person in my group who took the class because her friend was a TA. However, she had no desire to code, and has no desire in the real world to ever code again. She was not in the E school, so it was not a required class. She wanted to go into the business side of tech. For someone like that to be in a class where the majority of the grade is group-work heavy is extremely frustrating. We had minimal functionality in our app because it was really 4 people doing the work, with always the thought that she would be doing something during the week. So work was delegated to her, she took on tasks herself, and not much came out of it. We had to scramble the night before to finish. I'm not sure how this could be fixed on the administrative side except make the class have permission requests for those not in the E school. I know a lot of people who won't apply just because they don't want to bother with a permission, and so it will filter out the people who don't care.

More clearly explain the use of postgres database earlier on. Even now, I am still not 100% sure of its benefits, but I was convinced that it is more beneficial to switch to postgres. However, I was the only one on my team convinced enough to switch over. So, I had to research how to do that, and ran into some issues along the way but eventually got it to work. Then I had to help my other team members install postgres locally. This had already been a few weeks into the project, and I think it would have been much easier if everyone on the team saw the value of postgres earlier on. Also recommend integrating github with slack to notify any pushes to master.

Podcasts are great but is there any way to incorporate video/fully record lectures? I understand there may not be a whole lot of visual stimulus going on in the class, but it would be helpful for a handful of students to match an image to the voice.

1. Django Practice Assessment: It was a good refresher for me as I had previous experience in Django, but this single assignment takes too long and accounts for a huge portion(5%) of the entire grade. Suggestion 1: Divide the Django Practice Assessment into two different assignments. For example, watching the YouTube videos, accepting the GitHub Classroom Assignment, and completing the Written Assessment can be Part 1, and going over the official Django tutorial, adding a feature, integrating Bootstrap, and deploying to Heroku can be Part 2. 2. I was very uncomfortable with the half-professional-half-amateur/arbitrary/haphazard setup. One one hand, we join a completely new organization, meeting at least once a week for a stand-up meeting, learning from scratch, and using services/frameworks/tools like Slack, Django, GitHub, TravisCI, and Heroku. On the other hand, all members are developers (i.e. no project managers or marketers or designers), and yet we use Google Sheets for project management. Suggestion 2: Use either Trello or GitHub Project for project management. Students should be able to use Markdown and familiar with Kanban. Suggestion 3: Use a professional issue tracker. Since learning Jira or YouTrack for this course is too much, I think GitHub issue is the most suitable. 3. If you do want to keep the XP system, change the project grading scheme. Suggestion 4: Instead of grading the project entirely holistically, assign points for different features (e.g. small points for adding a favicon, changing the /admin URL, or forcing HTTPS; more points for connecting to Google Maps API or automatically sending emails). This way, positive reinforcement is made possible, students can better estimate what score they will get, and the lectures and the project become more cohesive. 4. Refactoring is a very important part of the software development cycle, but this course doesn't give due credit for refactoring in the course of the project. Suggestion 5: Release earlier and add a refactoring sprint. If the project has become more modular and maintainable, TAs will give a pass. 5. The 6 (or so) team evaluations were very tedious because it was not automated. Towards the end of the semester, I created a short Python script to pre-fill some fields (e.g. sprint, team name, computing IDs) and shared the URLs with the team. Suggestion 6: Pre-populate team evals so as to save 200+ students' time.

Nope, project felt really rewarding

I think having lecture recordings will help students follow along with the slides and gain that extra insight that just the audio podcasts dont provide.

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
	If there is a common issue across all projects that people are having a hard time solving (images, this semester), please address it
	coming up with more useful apps?
	I didn't feel like the content warranted me to go back and listen to an entire lecture, reading the slides was enough, plus I attended every lecture.
	Give more guidence with the project. The structure of the class is very bad to say the least.
	Allow more project ideas. All three this semester was almost equivalent in the backend it was super boring. As far as I can see there is no reason not to allow more projects. The TAs gain no useful context for helping with issues when the projects are limited because each team makes very different design decisions. It's not like many requirements were given for each project either. As a student I see no benefit to limiting it to 3 projects.
	I struggled a lot with this project and finding how I can contribute to the team, but I don't know whether that was a problem with me or with the system. My only suggestion would be to connect the lecture and the lab more; it felt like they were two different classes sometimes, and I enjoyed lecture far more than the project/lab. Some more teaching specific to the project details could remove a lot of frustration and wasted time.
	Have extra office hours available during the Django practice assessment. I felt super lost during that and then after not understanding the assessment had to jump right into the project and felt super overwhelmed at first. I wish more help had been offered for that like most courses do when first setting up software and what not for the class.
	More Travis help would be great. It is an area the internet is pretty useless on a lot of the times, and the TA's usually only know what they did in their own semester.
	Keep doing the same thing!
	Every lecture was not very information dense. Couldâve been given in 45 minutes not 75.
	Nope
	None
	PLEASE READ Make the requirements more clear.... I feel like the feedback we got from the TA was good and then we got our grade back it did not reflect the effort we put in throughout the semester. Make the TAs give us better feedback or have project checkpoints where the professors step in and make suggestions halfway through so that we actually know what we need to work on.
	Cut 50,000 points of the project into 2. 25000 halfway during the semester as a checkpoint and then 25000 at the end
	The podcasts were hard to follow without knowing what slides were up or what was showing on the screens. As someone who is very visual it was harder to follow.
	It might be helpful to have a better idea up front of the timeline; what do we need to have done by the next sprint to be on track? I know part of that is part of the process, but for the benefit of the students' being able to plan better and budget time, it might be helpful to have more concrete expectations. It also might be good incentive for members to perform better to let them see a summary of their teammates' reviews from sprint to sprint; that way they know if they are expected to step it up.
	N/A
	No
	I'm not sure why Monday and Wednesday do not have OH, but I think to probably expand OH to everyday.
	I'm not sure, it would be nice if there was more accountability for terrible group members. Them getting bad grades at the end of the semester doesn't help me build an entire application and get a good grade at the end of the semester. Honestly, should someone even pass the class if they don't contribute to the project at all?
	Travis is sometimes hard to work with
	Take evaluations more seriously and check the number of commits from team members.
	It would be helpful to students to get an idea of how long a specific assignment/task will take (even though it varies). I also think, since many students get stuck and frustrated at several points, it would be useful to have resources/help available in some form other than office hours. I can remember many late nights where I felt overwhelmed/helpless.
	Tell people views are controllers in Django on the first day of class. Show actual examples of tests which pass on travis, this way more time can be devoted to writing helpful tests instead of spending hours just trying to get an assertEquals(1, 1) test to pass.
	More help/tutorials for Travis

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~

Let people repeat ideas from the past! It's not like you can copy someone else's codebase entirely, I mean think about how hard it is to get someone else's stack overflow to work with your code and that's a design pattern! Someone else's code from last semseter is ulktimately going to be no help

I know this is different/longshot, but if the lectures were posted podcast style on a spotify/etc, I would totally listen to lectures through that. Again, I'm super in the minority as someone who listens to 1+ hrs of podcasts each day, but this would totally get me to listen even if not attending lecture.

14. 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-3240-002, Sherriff, Mark

Total	0-2 (NA)	3-5 (NA)	6-8 (NA)	9-12 (NA)	13-16 (NA)	17 or more (NA)
70	1 (1.43%)	34 (48.57%)	22 (31.43%)	6 (8.57%)	6 (8.57%)	1 (1.43%)

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)
70	1 (1.43%)	34 (48.57%)	22 (31.43%)	6 (8.57%)	6 (8.57%)	1 (1.43%)

15. How would you rate the availability of TAs?

~

Question Type: Likert

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-002, Sherriff, Mark

Total	Mean	Std Dev	Excellent (4)	Good (3)	Average (2)	Weak (1)	Very Poor (0)
70	2.84	0.77	13 (18.57%)	36 (51.43%)	18 (25.71%)	3 (4.29%)	0 (0.00%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Excellent (4)	Good (3)	Average (2)	Weak (1)	Very Poor (0)
70	2.84	0.77	13 (18.57%)	36 (51.43%)	18 (25.71%)	3 (4.29%)	0 (0.00%)

16. How would you rate the helpfulness of the TAs?

~

Question Type: Likert

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-002, Sherriff, Mark

Total	Mean	Std Dev	Excellent (4)	Good (3)	Average (2)	Weak (1)	Very Poor (0)
69	2.61	0.99	14 (20.29%)	25 (36.23%)	19 (27.54%)	11 (15.94%)	0 (0.00%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Excellent (4)	Good (3)	Average (2)	Weak (1)	Very Poor (0)
69	2.61	0.99	14 (20.29%)	25 (36.23%)	19 (27.54%)	11 (15.94%)	0 (0.00%)

17. How often did you make use of the TA office hours?

~

Question Type: Multiple Choice

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-002, Sherriff, Mark

Total	Every week (NA)	Every other week (NA)	Once per assignment (NA)	Rarely (NA)	Never (NA)
70	2 (2.86%)	8 (11.43%)	8 (11.43%)	20 (28.57%)	32 (45.71%)

Results for SEAS, 3000-level courses

Total	Every week (NA)	Every other week (NA)	Once per assignment (NA)	Rarely (NA)	Never (NA)
70	2 (2.86%)	8 (11.43%)	8 (11.43%)	20 (28.57%)	32 (45.71%)

18. Any specific comments about the TAs you would like to share?

~

Question Type: Short Answer

~

contributed by Sherriff, Mark (mss2x)

Results for CS-3240-002, Sherriff, Mark

Total	Individual Answers
34	*See below for Individual Results*

Helpful!

no

N/a

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
	Extremely encouraging and gave useful feedback
	Too much emphasis on styling styling styling styling styling styling styling I hate styling If I am able to combine CSS elements and bootstrap copy paste code correctly in a certain combination I have not learned nearly as much as implementing a feature in the application
	sam really helped
	They were nice but their feedback should've been more specific. They were also unaware of the map requirement until the week before the project was due.
	TAs in lab 103 were really helpful about providing design feedback.
	I loved our project TA Annie! She was super nice and helpful.
	Nope
	Nope
	Nope
	Nope
	None
	They should be able to predict the final project grade more accurately. They said we'll get full credit, but we didn't.
	Annie was an awesome TA!
	Jazlene G. was a great TA! She always tried to help out when possible, unlike the other TA's for my section.
	The TAs, while nice, were effectively useless in fixing our errors in lab. I don't blame them, but there's not much they can do if your Heroku crashes or Github merge fails.
	N/A
	N/A
	I think it would be more beneficial to have TA office hours more spread out through the week, but keep the large chunk on Sundays, as labs are on Mondays
	Well, I liked them in lab, didn't interact with them much outside of it.
	No
	n/a
	n/a
	Jaz was great, during my lab time I felt that the other TAs had no knowledge
	I think they are helpful. They put a lot of effort to solve my issues. If there was anything that they did not know, they would not try to give a wrong answer to that, but instead suggest some approaches to do that.
	I think the TA OH should be spread out.
	One of my TA's was kind of rude, but the other two were lit (lab 103)
	Tell them to not tell kids to just "Google stuff" because chances are people tried that already. HELP them Google things or refer them to another TA who might know better. Some TAs upset me with the way they deflected students and didn't want to help.
	I wished they were more helpful in understanding guidelines. We asked multiple times about specific guidelines and they told us we had a finished app and anything else would be extras. We were given the impression it would be best to spend our time perfecting the UI then adding any additional features; however, we ended up getting off for this at the end. That is frustrating.
	They gave good feedback at sprint checks but were very slow when it came to checking off towards the end of the semester to the point where we had to stay late to be checked.
	They didn't seem like they wanted to help much.
	some of them really did not know what they were doing and were just there for a paycheck which was ridiculously frustrating

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~

19. 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-3240-002, Sherriff, Mark	
Total	Individual Answers
37	*See below for Individual Results*

Could have gone into more detail about design patterns and how they could apply to the project/future work

Would have liked a lecture on technical communication with non-technical managers/public/execs.

More about the differences between waterfall and agile methodologies

Maybe teach us different ways (languages, databases, UI etc.) people design projects. I had no knowledge of any of these back-end things so I think it hurt me in the process of getting actual usefulness out of the project aside from the software lifecycle. Maybe make it a prerequisite to take classes like databases. I do think it would be useful to talk about the back-end.

Containerization and cloud infrastructure

ethics

1. I thought we would learn MVVM when we learned MVC. If you are going to discuss architectural patterns, you should teach at least two and compare/contrast them. 2. Maybe you could introduce six thinking hats for the requirements slide. 3. Coming up with user stories is usually not the developer's job. A good developer should be able to proactively ask implementation-specific details before it causes a problem later. For example, after reading the user story "Every user should input their computing ID when they register," a developer must be able to ask what are allowed/disallowed computing IDs. Can it contain non-ASCII characters? Can it be one character long? Is computing ID unique for every account? (i.e. If a user creates two accounts using different UVA emails, is it okay for them to have the same computing IDs?) I think "thinking from a developer's point of view" was neglected in this course. 4. All the topics you chose are important. Students just don't realize their importance because they don't relate directly to the project, and you just don't have enough time to teach everything in one semester. Overall, I think this is a decent set of lectures.

It'd be interesting to have time to see the highlights of all the projects on the last day.

N/a

I would've liked to learn more about the selling of code/webapps and freelancing.

API Development

Front End Design and UI

REST API could have been delved into a little more

We really should;ve covered a decent amount of specific software related material on the softwares we used for the course. Logistical problems took perhapos the longest time to solve in terms of our project developmen. For example, issues with Heroku, databse, and Travis CI...

probably an introduction of Django in the beginning of the semester

None

None

None

Actually talking about some of the components of the django project in lecture

REST/Architecture patterns, development tools, modularity

oooo my favorite. I understand the semester is packed with material, and we cover a lot of basics of a lot of things. Perhaps more cutting edge stuff (How cloud development works, how we containerize things, etc)

I think how to go about unit testing couldâve been covered a little more deeply, in terms of how to test things that are more front-end

N/A

N/A

N/A

n/a

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~

n/a

n/a

Like literally anything about django

Security! Also it would've been nice if we covered javascript stuff

Overall process

Would be cool to go over what processing payments looks like for software engineers. Maybe just as a bonus(ish) topic covered towards the end of the course.

More on frameworks besides Django, I would've enjoyed learning more in depth about some of the frameworks.

Other frameworks

a different framework

Probably more on MVC.

I wanna talk about AI, or maybe go more in depth into how to function as a software engineer in the workplace

20. 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-3240-002, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
70	4.27	0.85	30 (42.86%)	34 (48.57%)	3 (4.29%)	1 (1.43%)	2 (2.86%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
70	4.27	0.85	30 (42.86%)	34 (48.57%)	3 (4.29%)	1 (1.43%)	2 (2.86%)

21. The activities and assignments helped me learn the subject matter.

~

Question Type: Likert

~

contributed by Dean of the School of Engineering and Applied Science

Results for CS-3240-002

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
70	4.26	0.83	29 (41.43%)	34 (48.57%)	5 (7.14%)	0 (0.00%)	2 (2.86%)	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)
2587	4.24	0.86	1118 (43.22%)	1106 (42.75%)	212 (8.19%)	90 (3.48%)	39 (1.51%)	22 (0.85%)

22. 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-3240-002

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
70	4.13	0.96	27 (38.57%)	32 (45.71%)	7 (10.00%)	1 (1.43%)	3 (4.29%)	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	4.13	1.01	1093 (42.33%)	1030 (39.89%)	218 (8.44%)	141 (5.46%)	87 (3.37%)	13 (0.50%)

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~

23. The course materials (such as textbook, readings, or background materials) increased my learning.

~

Question Type: Likert

~

contributed by Dean of the School of Engineering and Applied Science

Results for CS-3240-002

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
68	3.68	1.11	13 (19.12%)	14 (20.59%)	14 (20.59%)	4 (5.88%)	2 (2.94%)	21 (30.88%)

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)
2586	3.80	1.11	724 (28.00%)	865 (33.45%)	439 (16.98%)	212 (8.20%)	108 (4.18%)	238 (9.20%)

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-3240-002

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
69	4.32	0.87	35 (50.72%)	25 (36.23%)	6 (8.70%)	2 (2.90%)	1 (1.45%)	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)
2586	4.07	1.04	1057 (40.87%)	939 (36.31%)	313 (12.10%)	161 (6.23%)	84 (3.25%)	32 (1.24%)

25. The instructor was well prepared for class.

~

Question Type: Likert

~

contributed by Dean of the School of Engineering and Applied Science

Results for CS-3240-002, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
70	4.75	0.50	54 (77.14%)	13 (18.57%)	2 (2.86%)	0 (0.00%)	0 (0.00%)	1 (1.43%)

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)
3293	4.39	0.86	1856 (56.36%)	974 (29.58%)	230 (6.98%)	129 (3.92%)	32 (0.97%)	72 (2.19%)

26. The grading policy was fair.

~

Question Type: Likert

~

contributed by Dean of the School of Engineering and Applied Science

Results for CS-3240-002

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
70	4.36	0.93	39 (55.71%)	23 (32.86%)	4 (5.71%)	2 (2.86%)	2 (2.86%)	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)
2589	4.10	1.01	1087 (41.99%)	954 (36.85%)	308 (11.90%)	160 (6.18%)	70 (2.70%)	10 (0.39%)

27. The instructor showed respect for students, and created a safe and supportive learning environment.

~

Question Type: Likert

~

contributed by Dean of the School of Engineering and Applied Science

Results for CS-3240-002, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)	Not Applicable (NA)
70	4.48	0.90	47 (67.14%)	12 (17.14%)	7 (10.00%)	2 (2.86%)	1 (1.43%)	1 (1.43%)

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)
3292	4.59	0.67	2159 (65.58%)	901 (27.37%)	122 (3.71%)	35 (1.06%)	18 (0.55%)	57 (1.73%)

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~

28. What aspects of the course most helped your learning?

~

Question Type: Short Answer

~

contributed by Dean of the School of Engineering and Applied Science

Results for CS-3240-002	
Total	Individual Answers
51	*See below for Individual Results*

The project for sure.

Lenient grading policies put off mental pressure.

Sherriff's lecture style made it a lot easier to stay engaged in class. Can tell he actually cares about us learning the material.

Sherriff's enthusiasm made me want to go to class

The GPs

making the app, enthusiasm of professor, having calendar of all schedules and due dates

The individual coding assignment, but also thought that a warning should have been given for how long it would take.

meeting with my team during lab

Hindered my ability to focus in school more than anything. I was passitonate about the coruse in the beginning as the idea seemed nice, but the strcuture of the class was very bad.

All helped me equally

The GPs were good practical experience of the lectures. Very helpful for quizzes.

Sherriff

The intuitive order in which topics were introduced

Doing tutorials.

Guided practices

Guided practices

the hands on learning of trying to figure out how django worked and doing things myself helped my learning the most.

SHERRIFF

The GPs and the lecture itself

The project was really helpful to demonstrate practices hands on.

The group project

the project

The availability of material

Sherriff's dynamic and engaging teaching style helped grab my attention to what was most important about the topics at hand, and really made learning some of the more obtuse topics fun.

Lecture and project

group project in an agile environment

Working on the project

The hands on project was the best way to understand the concepts we were learning in lecture.

Practicing the different techniques we discussed in class within the project

project was instrumental.

The quizzes and the in class GPs alongside the entire project.

The slides were very helpful.

Sherriff made even the more dry topics engaging, which made it easy to pay attention

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
	The in class activities. The lecture contents and the actual work on the project are the most helpful. Project, Piazza, and recordings. Self learning project Professor Sherriff's lectures the coding Enthusiasm of the professor The entire project and the very first django tutorial= Being able to apply topics that I learned in class to my project Guided Practice Guided Practice The project The project The project interactive guided practices The projects It was good.
29. What changes to the course would most help your learning? ~ Question Type: Short Answer ~ *contributed by Dean of the School of Engineering and Applied Science*	Results for CS-3240-002 Total: 42 Individual Answers: *See below for Individual Results*
	Idk, even more emphasis on the project maybe (something like one lecture a week and two lab meetings) but I feel like this would probably cause the course to lose accreditation. Teaching the tools used for the project during lecture. Make some of the sprints longer to be more effective in the work we would be able to complete. make practice quizzes Teach about the actual stuff, tools such as Django, Heroku, and Travis CI that we will be using next time! Demo Projects Maybe include information on how to get started with Django beyond just the Django documentation tutorial? Maybe a condensation or guide in the beginning? More interactive lectures - maybe shorter GPs more frequently to keep it more engaging since material can be dry. Make things relevant to modern day scenarios. N/a Make lectures more information dense None really. Make the XP grading system more understandable. different type of project More incentive to come to lectures I know this is a useless point, but Travis could be removed (I know it can't deep down.) Teach more actual django techniques and coding

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
	Video recordings instead of audio podcasts - reduce group size - get rid of xp system - teach some of the core django or swe concepts in class Probably talk more about the answers of each quiz. Extra help for the Django practice. We should be allowed to use our own CI solutions. Due to the backup on Travis CI, many teams always forced deployment and ignored their tests. Even we were forced to do this occasionally, and we were an exceptionally well organized team who followed most good software practices. Allow us to use GitHub actions, or our own Travis CI servers or Jenkins. Just say you won't help if we have issues unless we use Travis CI. Not sure Resources for overcoming common computer/coding roadblocks and more time spent learning how to use IDE's/Command line and downloading libraries. Spreading Django assignment out Some of the lectures were a little out of sync with our progress on the project, so sometimes weâd learn about something after we were supposed to have implemented it in our project which was not as helpful. Connecting the project and lecture more, and making it more of a "guided exploration" rather than a "do it all yourself" type of project. Making lectures more engaging so there was more to do than just listening (such as more visuals or more interesting activities). However, that being said, I absolutely hate cold calling and having to participate when I don't want to so the whole number room thing just gave me anxiety. (Kinda just saying this to say something) Maybe connecting the actual projects themselves to lectures would be helpful. Put MVC earlier so we can have a better understanding of how Django works. Multiple smaller projects that have to interat with each other (like ISA) N/A Talk about back-end more for people who don't know n/a n/a n/a More examples of implementations of topics in lecture, integrating the lecture more with the project I wish we had spent some time learning Django. Maybe teach architecture earlier on? n/a. really good course, wish I would have had this when i was first starting CS. As someone with 6.5+ years of professional/academic software engineering, it wasn't as helpful, but really good content. Maybe a central text to reference. So we received a lot of points off on our project for not having a mapping api on our Rideshare application, but we never put it in our requirements or anything to potentially have this. I am unsure how we could go a whole semester getting 100% on the sprint check ins and our requirements doc when we were missing a requirement that would lose us so many points... kind of set us up for failure in a way in which seems unfair. more help on the coding It was good.

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~

30. 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-3240-002

Total	Less than 1 (NA)	1 - 3 (NA)	4 - 6 (NA)	7 - 9 (NA)	10 or more (NA)
69	7 (10.14%)	30 (43.48%)	23 (33.33%)	5 (7.25%)	4 (5.80%)

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	129 (4.99%)	697 (26.98%)	1060 (41.04%)	462 (17.89%)	235 (9.10%)

31. I learned a great deal in this course.

~

Question Type: Likert

~

contributed by Office of the Provost

Results for CS-3240-002

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
68	4.21	0.91	29 (42.65%)	29 (42.65%)	7 (10.29%)	1 (1.47%)	2 (2.94%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
2574	4.18	0.88	1055 (40.99%)	1097 (42.62%)	283 (10.99%)	99 (3.85%)	40 (1.55%)

32. Overall, this was a worthwhile course.

~

Question Type: Likert

~

contributed by Office of the Provost

Results for CS-3240-002

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
69	4.36	0.87	37 (53.62%)	24 (34.78%)	6 (8.70%)	0 (0.00%)	2 (2.90%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
2581	4.10	1.01	1072 (41.53%)	989 (38.32%)	309 (11.97%)	121 (4.69%)	90 (3.49%)

33. 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-3240-002, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
70	4.64	0.57	47 (67.14%)	22 (31.43%)	0 (0.00%)	1 (1.43%)	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)
3269	4.43	0.75	1819 (55.64%)	1151 (35.21%)	218 (6.67%)	62 (1.90%)	19 (0.58%)

34. 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-3240-002, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
70	4.41	0.88	42 (60.00%)	20 (28.57%)	3 (4.29%)	5 (7.14%)	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)
3280	4.40	0.81	1842 (56.16%)	1034 (31.52%)	306 (9.33%)	70 (2.13%)	28 (0.85%)

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~

35. Overall, the instructor was an effective teacher.

~

Question Type: Likert

~

contributed by Office of the Provost

Results for CS-3240-002, Sherriff, Mark

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
70	4.61	0.69	48 (68.57%)	19 (27.14%)	2 (2.86%)	0 (0.00%)	1 (1.43%)

Results for SEAS, 3000-level courses

Total	Mean	Std Dev	Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
3287	4.26	0.97	1705 (51.87%)	1015 (30.88%)	342 (10.40%)	152 (4.62%)	73 (2.22%)

36. Please make any overall comments or observations about this course:

~

Question Type: Short Answer

~

contributed by Office of the Provost

Results for CS-3240-002

Total	Individual Answers
37	*See below for Individual Results*

It was fun! I put more time into the project than necessary just because I really liked working on it. Having the project have less 'hard' requirements was nice because it meant way less stress while coding.

Team size was alright, I think 4 would be the ideal size for me. There were times where there really wouldn't be much work for all 5 of us and we would have to split tasks up in order to give everyone someone to do.

There were a lot of complains about the UI of our system. Professor Sherriff told us that as long as the UI wasn't really, really distracting/horrid, there wouldn't be points docked. However, the reason our project did not receive a 5000 was due to these nitpicky "issues" referring to the UI. The TAs made no mention to us about these issues ahead of time. If you are going to hold students to a standard, make them clearly aware of the standard. You should add slides/lecture about UI if you are going to use it as criteria for grading. Also, you should NEVER have in-clas assignments during the Fall Career Fair. I don't go to UVA for fun. I'm here because I know it will help me get a job, and I shouldn't lose points when I try to fulfill the main reason why I endure all of this busy work.

This was by far my most organized course this semester. The material was always graded in a timely fashion, which was awesome. Sherriff was an awesome professor. All other UVa professors should view this class as an example of how to teach effectively, grade efficiently, and plan thoroughly. 5 Stars 10/10 would recomend.

Great course! Very helpful and practical. Sherriff's snark 10/10. Honestly though, the lectures were very thorough and the project was engaging. A lot, but engaging. I learned a lot. Thanks!

Yeah, like I was saying before about involing Django in the course more, I think it's really intuitie if we do that. Syntax errors are perhaps the most frustrating. Google only does so much, if we can Google to become a millionaire then why are we paying UVa for this education? Even the slightest bit will help, but we should defeinitely, it should be a"must", have Django related material to teach in class. At the very least, we should have some organized sets of tools to use when we are develoing our project. Instead of us simply going off into a sea of Frenzy with Google, because surely the Django practice actually "taught us everything" there is to know in a week right? Even companies out there in the real world , gives their new exomplyees weeks, "weeks", of training... Tnank you

Professor Sherriff is an amazing lecturer and good person, but he was often very annoyed/apathetic sometimes about helping students in office hours and deflected them to TAs or seemed like he didn't really want to help. When TAs couldn't help, he kind of shrugged his shoulders and helped reluctantly.

Design decomposition wouldâve been really useful to cover earlier in the semester

Sherriff is a great lecturer, very engaging

Sherriff is great. This course needs to be advertised as one that is necessary before a CS-related internship

I enjoyed Mark Sherriff as a teacher, it definitely took me a couple lectures to get used to his energy but I found I enjoyed it. The class content was easy to teach myself and I felt like I could have learned it without the lectures, though the lectures did speed up my learning. That said, going into more challenging topics might be more worthwhile.

Overall, not the best use of my 3 credits to get the CS degree. If it were not a requirement, I would have not taken this class. Structure is very bad and didn't learn much coming out of it.

Great class and great semester thank you!

unique class(XP based) and the group project with the scrum meets seemed to represent the real workforce. A+

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
	I have had a bit of bitterness towards Sherriff since the first day of class when he asked for all the students to yell out where they interned at, just seemed a little bit elitist and uncomfortable for many students that day. It was the first day of class...
	With regards to team size, it often seemed cramped to have 5 people on a team. There often wasn't enough for all of us to do a whole lot each sprint (maybe this was because we weren't pulling enough off of our backlog, but we accomplished all of our requirements and then some). For 5 people, the work seemed somewhere in between just right and too little. I enjoyed that the lecture didn't sit there and teach us Git or Django. We are all smart enough to find this stuff online and it greatly increases the pace of things if we aren't bogged down by learning the methods in lecture. For me the class felt like the project was the main focus and then the lecture was a supplement to the project. This felt great classes lined up with our material nicely. I think it is a very well designed class and that expectations are on point. I enjoyed the quiz and final exam formats and don't have much criticism of the class. Another note would be as someone who came in with AP credit for 1110 and joined the pilot program, I never had any exposure to Python. Even so, it wasn't an issue to pick up Python and get used to the environments. I really liked the course and would highly recommend it.
	I wish there was more specifications provided from the TAs and Sherriff for the project
	Thanks for the fun semester! Also potentially groups of 5 could be managed, SCRUM master honestly just was whoever had the most free time that week, and I was a testing manager that did a lot of the Travis and Heroku work.
	Sherriff, if you are reading this-- which you mentioned in class you would be, I think you are a fantastic teacher who really cares about the students in the class. You are kind and considerate and make a great deal of effort to be accommodating for everyone. Thank you for all you do! I think that my review should be taken with a grain of salt because I had already done all of the concepts covered in this class in previous internships or self learning and so I realize that I am not the target audience of the class. But I do think that I noticed that the group members I was with who were the target audience, people with little SWE experience, really struggled to self teach some of the concepts and apply them effectively. Covering some of the things in class to tie the lecture to the lab a little bit more could potentially be good. That is all! Love ya! Your daughter is cute!
	I am going to complain that the grading in this course seems completely subjective at times. Our project page works without error. We have met all the functionalities, and went above and beyond to implement even more. We used AWS to store pictures. We have an option to customize your profile and control your items from that same page. We have an interactive map to confirm location with others. We have resizing capability to work on mobile. Yet we get points off for arbitrary details like spacing being 2pt too close, or the map not being "interactive" enough. On the other hand, other groups have the bare minimum of a website. Some even still have 404s up. Yet they have some pretty UI with glowing buttons and modern text. Apparently that is enough to satisfy the professor and earn them full credit on the project. They have A's and can sleep in on final day. Yet my group puts in hours each week yet because our background wasn't an image the professor liked, we're going to be sitting in class to retake quiz questions that got points off for having complete sentences, but wasn't put in paragraph form like he likes. If I knew all you need to do is make a cute UI, then I wouldn't have put in nearly as much effort into this class, and all for naught.
	BEST CLASS AT UVA
	This is a great course and should continue to be offered!
	Definitely the most useful CS course I've taken so far since I'm going into software engineering. A more relevant framework choice for the project would be cool.
	I was a big fan of sherriff making dry lectures more interesting with his jokes.
	When grading the final project's it's kinda of unfair to judge a project as lacking features or unfinished if only 2-3 people out of 5 are really working on it. If you are in a group where everyone is pulling their weight, obviously the final product is going to be more robust. The final project is worth a lot and even if you put a reasonable amount of work if not an above average amount of work, you could still not get a good score.
	One good improvement would be to make it clear how dense the Django tutorial can be. I expected it to take a handful of hours, and it took me at least 10-12 hours to complete. I would have learned a lot more from it if I had a frame of reference of how complex it really was.
	N/A
	Sherriff is my favorite professor at UVA! Keep up the good work and please teach more electives
	A lot of people are scared to go to Sherriff's office hours because they think he is mean. I don't but I know a lot of people are scared he will eat their head after biting it off.
	Mark Sherriff is an outstanding lecturer --one of the best I have ever had. He knows how to keep the class engaged with good humor and is clearly passionate about what he is teaching. I only have good things to say about his teaching, and I recommend his classes to my friends.
	Your group assignment basically determines your grade which I think has a lot of chance involved.
	Professors are helpful. I like how the class content does not need a textbook.
	really good course.
	A good class. I liked my group. I just wish some of the grading requirements were more clear.

~ QUESTIONS AND DETAILS ~	~ ANSWER MATRICES ~
	I was worried about getting a bad group for my project but I was pleasantly surprised by a great team, which allowed me to enjoy the project. There's a new Sherriff in town. There is clearly an uneven split of work among team members--caused by slackers' moral hazard--and no one can do anything about it. (Of course we can discuss it among us, but you can't force someone to work.) I think one way of dealing with this is increasing individual assessment (e.g. team evaluations, project document). p.s. I wonder if you can guess who I am from my writing. Thank you for the semester!