Individual Instructor Report Spring 2026 Version B for C S 373 - SOFTWARE ENGINEERING (53380) (Glenn Downing)

Project Title: **Course Evaluations Spring 2026**

Courses Audience: **58**
Responses Received: **58**
Response Ratio: **100 %**

Subject Details
merged_subject_id
merged_subject_display_name

Report Comments

Guide to the Interpretation of Course Evaluations at UT Austin

The goal of course evaluation process at UT Austin is to drive teaching excellence and to support continuous improvement in teaching and learning experiences. Course evaluations provide snapshots of student perspectives on their course-level learning experiences. Most experts on teaching evaluation advise that no individual method gives the complete picture of an instructor's teaching effectiveness, multiple and diverse measures, on multiple occasions, are advised to give a full picture of the teaching effectiveness of a particular instructor. Moreover, other factors, such as size of class, level of the class, and content of the course, can cause small variations in the ratings. Therefore, student perspectives for a particular instructor or course should be interpreted as a snapshot, and not as providing complete information on the teaching effectiveness of that instructor. For additional details, including the scales and how the Mean scores are calculated, please review the Report Guide at the end of this document or, UT Austin's Viewing Course Evaluation Results webpage.

Creation Date: **Friday, May 15, 2026**

Core Questions

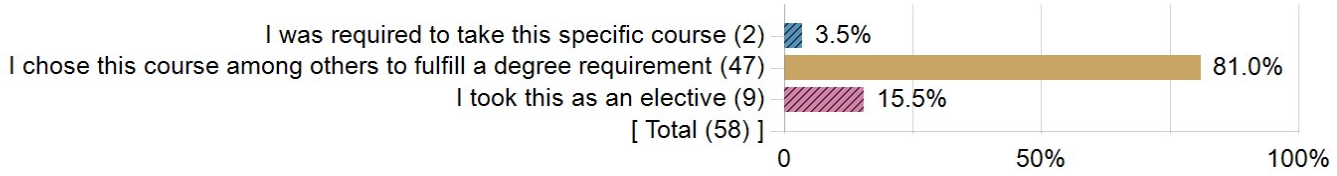
	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree	Responded	Mean
During this course, I gained a deeper understanding of the subject matter.	39.7%	44.8%	8.6%	3.4%	3.4%	58	4.14
The course was well organized.	41.4%	41.4%	8.6%	8.6%	0.0%	58	4.16
The instructor clearly explained the course objectives and expectations.	46.6%	41.4%	5.2%	6.9%	0.0%	58	4.28
The instructor fostered an inclusive learning environment.	44.8%	31.0%	13.8%	5.2%	5.2%	58	4.05
The instructor effectively explained the concepts and subject matter in this course.	41.4%	39.7%	15.5%	3.4%	0.0%	58	4.19
The instructional techniques kept me engaged in learning.	43.1%	39.7%	6.9%	5.2%	5.2%	58	4.10
The instructor checked for student understanding of the concepts presented in the course.	50.0%	39.7%	10.3%	0.0%	0.0%	58	4.40

Overall Questions

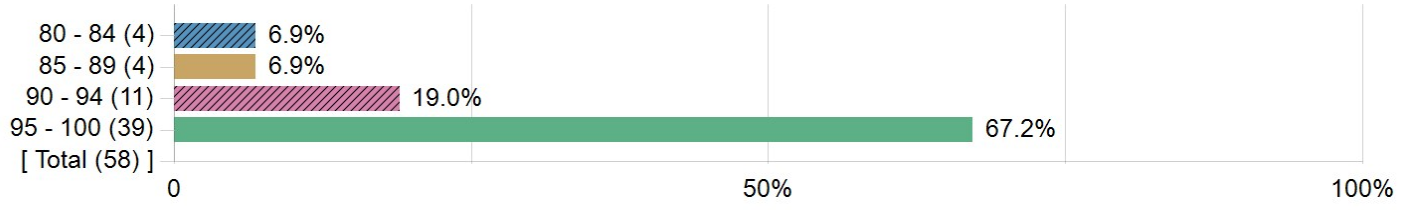
	Excellent	Very Good	Satisfactory	Unsatisfactory	Very Unsatisfactory	Responded	Mean
Overall, this instructor was	41.4%	32.8%	15.5%	6.9%	3.4%	58	4.02
Overall, this course was	29.3%	32.8%	29.3%	6.9%	1.7%	58	3.81

Other Course Questions

Why did you take this course?



Overall, approximately what percentage of the course meetings did you attend or complete (online, in person, or asynchronously)?



Statistics	Value
Mean	94.48

Comment Questions

Identify aspects of the course that were the most effective in helping your learning.

Comment
The lectures where we went over every important concept.
The cold–calling ensure everyone was paying attention.
Cold calling, quizzes, ed lessons
I liked how he would instruct and we would have quizzes the next day to reinforce the content learned. I feel like usually the quizzes, and his explanations over the answers really helped make sure I actually understand the content.
He explained the concepts in a way that showed where they came from, how they work under the hood, and why they are needed. We learned Python concepts that are not easy to find explained clearly together in one place.
The Ed Lessons were helpful, along with the quizzes at the beginning of each class.
The projects and no devices in class rule were what really helped me stay engaged in learning. As well as the high points of the class which were very intriguing to learn
The projects were the most effective in helping my learning.
Cold Calling was good throughout the course, though a bit stressful and overdone at times.
Cold calling helped keep me engaged
Cold calling was very effective in keeping me locked in. I was taught a lot of Python syntax that was a bit more niche. This can be helpful in interviews, and the assigned Kattis problems are also helpful in that way.
Learning the concept, then reinforcing our knowledge with practice problems.
I liked it when we went over concrete things that would be useful in our careers and on the project, like Docker.
I really liked that quizzes and exercises since they constantly tested our knowledge and made it we had to incrementally apply what we learned.
The cold calls and quizzes, made it so we had to keep up with the content at all times.
N/A
I liked the collaborative nature of the class
Lectures were interesting, and I learned quite a bit from them. I also learned a lot from figuring out how to do the full–stack projects.
collaboration. that helped with quiz and exercise grades catme peer feedback. my group wised up after the first phase, really ramping up our meetings. met 4–6 times a week, totalling at least 6 hours. very helpful, no team conflicts. tests being a review of quizzes funny phases "another high point of this class! and you aren't lookin' excited!" <– I will forever remember this (in a good way)
quizzes, problems, lectures
The lecturing style of Professor Downing helped me stay focused in class, and as a result, allowed me to process content easier.
Beginning of lecture quizzes
Aspects of the course that helped me learn was the group project, posted python notes, and morning quizzes
Prof. Downing's lectures and daily quizzes.
The most useful thing I took away from this course was the emphasis on Test–Driven Development. It was practical and I've already applied it outside of class. The individual HackerRank project was also a highlight, as well as being able to use Docker during the later projects.
I think the quizzes were especially helpful
Daily quizzes, frequent exercises
He is a very good teacher and helps the class know the details about software engineering.
I really enjoyed all the exercises explaining python concepts, SQL, and design patterns.
The daily quizzes and exercises were the most effective in keeping me up with the material, especially with the collaborative nature of the activities. I also liked the random group assignments as they simulated real–world software engineering teams, and I was fortunate enough to get a good team.
Quizzs for forcing to review
The exercises were the most effective in helping my learning. Also, the style of lectures where the professor takes notes for you helped me to concentrate on the material during the lesson instead of losing focus to take notes myself.
Live code demos

Comment
Quiz
I enjoyed the hands on learning of the in–class exercises that we did. I also enjoyed the collaborative nature of them, allowing other students to contribute ideas.
I think the collaboration aspect of the class and talking with my peers has allowed me to learn the most. Especially on projects, sometimes I haven't had experience with something that maybe some of my group mates have, so being able to talk things through with them has really helped.
Assigned readings
Cold calling was helpful in staying focussed.
The posted notes were super helpful in reinforcing the content. Furthermore, the fact that the quizzes were collaborative helped me review the information from each lecture and solidify lecture concepts.
i liked the kattis problems because it proved a direct way to show what you learned in class i also liked the exercises in class because of the same reasons the course content was sometimes interesting like learning SQL was something unexpected and welcome the concept of building a full–stack website that utilizes APIs and data sources is a really important one, especially considering everyone will work in that field
Cold calling
The exercises, especially the Java ones were helpful in understanding object oriented design. It also helped with my Python and SQL skills.
The projects helped me learn the most.
The exercises that we do in class helped me gain a better understanding of the topics since I actually had to use what we learned.
The projects were helpful.
The most effective aspect of the course for my learning was the project, because it gave me a chance to actually apply everything we were learning instead of just understanding it in theory, and having something tangible to show at the end made the concepts feel more meaningful while also helping me learn new skills along the way. The collaborative quizzes were also very helpful since working with others made it easier to understand difficult concepts, and hearing different perspectives reinforced my learning while making the experience more engaging and less stressful.
Project requirements had us practice and implement many typical software development responsibilities
The quizzes and readings all helped me to improve my understanding of the course content.

Identify the aspect of the course that you found most challenging, why you found it was challenging, and suggest one thing that could be done to help future students meet that challenge more effectively.

Comment
The daily quizzes were the most challenging as it is hard to go to every lecture.
The daily quizzes were difficult but being able to work on them together helped! Sometimes a little more time was needed on the daily quizzes.
I found the project spec s to be kind of vague at times.
I think cold calling can be stressful at times. I also think it would be helpful to get a class or two going over topics of phase 2 of our project either aws, or about frontend.
We were randomly assigned to groups of five, which made the class much harder. Not everyone contributed equally, and some group members lacked motivation or consideration for others. I think the group projects should either be removed, or groups should be limited to a maximum of three students and students should be allowed to choose their own group members.
The final two projects were extremely time consuming, but not because of complexity, and instead of the number of required steps that felt useless and in personal experience not useful in industry.
The quizzes were sometimes challenging. Make sure to look over what the prof talked about last class.
I can't think of anything to be honest.
The projects and getting everyone together in a short period of time
Early project stages were a bit difficult to get the hang of, additional resources around what we're supposed to be doing / how to do it would've been helpful
I feel like what we learn during class isn't really software engineering, it's the projects that are. I wish we spent more time covering software engineering tools like cloud hosting, database hosting, etc., rather than having the students figure it out when completing projects.

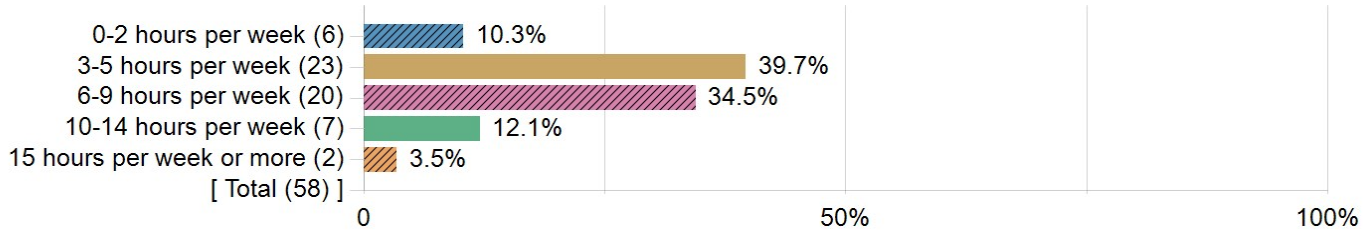
Comment
The time pressure for the quiz and problems didn't give us enough time to reason through things. I would suggest giving us more time or allowing us to continue working on it after class.
The project was most challenging because we barely learned anything in lecture that was useful to the project. You could go over more technologies that we use in the project, like React, and how hosting works. You could also make the rubric as detailed as possible and not add details in the lecture that are not covered anywhere in the document. Cold calling would be okay if you were more lenient on the responses and did not discourage students for not having the answer off the top of their heads. One of your responses, I remember you said, "Sad state of affairs that you made it here without knowing this".
I found the most challenging thing was having to do the project i.e. full stack website by learning about the technologies ourselves; though, I would say this is a good thing. However, I think guidance on how to go about a suggested way implementing these features would be appreciated.
The cold calls and quizzes, made it so we had to keep up with the content at all times.
N/A
I really did not like the papers on perusal, in the future maybe there could be less papers or more drops on them
The projects were probably the most challenging part of the course. Just make sure you have a significant portion done early.
I wish the class structure was interleaved: python, sql, then python again. Because it would have been really nice to know some of the sql syntax before the project started! Rather than figuring it out myself. Knowing obscure python tricks like tuple unpacking were utterly useless during the beginning of the class. The project instructions are "vaguely specific" in that there is a lot of tasks to complete (via issues) with very brief descriptions. e.g. make the website look "pretty", "refactor" the code, etc. The project hints were helpful, but they should be posted right when the project starts, not 1 day before it is due to be the most effective. And there is just so much expected, all at once. I did think that the in–class review of the project was helpful. Perhaps it would be better to record a more detailed video in advance, so there is more time for students to digest the content and ask questions. It also creates more class time to focus on other important content. I felt that the lecture pace was too fast. I like conciseness, but it went too fast. This is what would often happen: the professor asks if anyone has a question. I finish writing down the topic just covered, formulate a question, and the professor has already moved onto the next topic by the time I order my words into a coherent sentence.
i found the projects to be challenging, it might be useful to cover more of the required knowledge for the projects in the lectures
Sometimes the in–class quizzes were a little bit too long for the amount of time we were given.
Project guidelines were a little unclear and communication was not very efficient (in particular for phase 2 of idb)
I found the project to be most challenging because of the amount of new tools we needed to swiftly learn, and theres not much that can be done to help students besides a personalized starter guide and how everything can weave together.
Projects are split between team members so each member only does their own part so it's difficult to get hands on experience with parts that other team members are responsible for. Individual projects would have better coverage of the entire pipeline for one student. Also I feel like the lectures themselves spent too much time on Python, should talk more about software developing tools/frameworks/platforms and example usages in the industry.
For the most part, the most challenging part of this course had nothing to do with the material. It was figuring out what was actually required of us at any given time. Rubrics lived simultaneously on Canvas, the class website, and Google Docs, and they frequently contradicted each other on specific details. On top of that, Dr. Downing would sometimes announce changes in class that contradicted the published rubrics, and on at least one occasion, he pulled up the spec mid–lecture, confirmed that a student's objection was correct, and still insisted the change stood. This was a recurring source of stress across the whole class, not just for my team. The other major challenge was the classroom environment itself. Dr. Downing has a pattern of responding to student confusion with contempt. He will sometimes laugh when students get answers wrong and has told the class to "read the goddamn notes" after a bad quiz. He once remarked that it "says something sad about the state of affairs" in the CS department because a student didn't know something. He also told the class we were "an unappreciative bunch" for not reacting to the lecture material with enough enthusiasm after a "high point of the goddamn class". For me at least, it created an environment where students are afraid to ask questions, which compounds confusion and makes learning harder. It was also noticeable that during student presentations, he frequently wasn't watching. The fix for the rubric issue is straightforward: one document, locked within 48 hours of release, with any changes clearly marked as changes. The classroom environment issue is harder to fix with a policy, but it's worth naming because it meaningfully affected the learning experience for a lot of people in this class.
I think the project was the most difficult, especially the first couple of submissions and I think starting earlier could've helped with that
There wasn't much software engineering covered, it was more of a Python course. I would have loved to see projects that involve working in existing large codebases, building out additional features, fixing bugs, etc, to prepare students for a career as a Software Engineer, as these are the things that juniors tend to struggle with. Also blogs were useless.

Comment
It is a little hard to set up all the database stuff and starter website stuff, so maybe spend a part of the day going over how to set up an account and get it linked.
The problems were really tricky for me. I have always been bad at Leetcode style questions and it was super frustrating trying to figure out how to solve them. Additionally, the assignments had so much weight on IDB2 compared to the other checkpoints, so I would suggest somehow splitting the work more into IDB3 since we spent so much more time on the IDB2 implementation compared to the rest of them (probably close to the total of the rest of the parts just on that one part). We also never recieved customer feedback back after any of the phases, so we don't even know if we were doing what they envisioned.
The most challenging aspect of the course is the order of the curriculum. Independently, the topics that were covered made sense to learn about, but it felt somewhat out of order and at times, out of sync with what the projects were asking for. It's not that I could never relate what we learned in class to what we were building, or that I couldn't build off what I had just learned from the previous class or the week before, but I guess the relation wasn't explicit enough to understand what the end goal of the class was. Although, there is no doubt that the topics that were covered were important to learn about.
OOP. Troublesome.
I found the projects to be a bit challenging. Communication between teams is a bit difficult, and I wish the user stories were written and given out earlier than they typically ended up being assigned by our customer teams. Additionally, learning to use some of the tools required in this project was hard, but split between all the team members was rather reasonable. There are also typically 3 different rubrics for each phase: one on Canvas, one on the project page, and one google doc rubric. Not all of these rubrics were up to date and sometimes the requirements listed on one were not listed on another. I hope that in future years, they consolidate it into one rubric that is clear in its requirements.
The Kattis Problem
I found the projects fairly challenging and also a bit disorganized. I feel that a bit more organization of the rubrics, deadlines, and project requirements would help students complete the projects better.
I think sometimes the exercises feel the most challenging just because we are pressed on time and sometimes there is a lot to do. However, the fact that they are collaborative and there is always a TA around to help really helps for the times I get stuck.
Unrealistically short time limits for some quizzes and inclass exercises. The time limits are so restrictive that for some exercises pretty much everyone resorts to sharing and copying all of their answers with each other because there is no reasonable way to complete or understand every problem yourself
The projects were extremely time consuming at times. Having to set up AWS, especially if it is your first time, and figure out some infrastructure for the database and backend is way too much for a single phase of the project. I think splitting that phase up or giving some advice on how to go about setting everything up would be better.
The project was at times quite difficult and not well organized. The project didn't follow the lectures from class, which made it feel kind of strange, as we were working on something completely different. We always got our grades for the projects quite late, which meant we didn't have time to implement feedback (as we would get our grades the day before the next phase was due).
while the idea of building a full–stack websit is great and useful, none of the course content was actually useful in doing this project, which was the main percentage of our grade Like for the project, I had to teach myself all these tools and then still go to class where i learned nothing that could help me with it if the class is going to be called "software engineering", you should not just teach python – please teach frontend and backend development
Understanding the concepts. The professor would walk through all the code and implementation to help us understand what is going on; however, the downside is that it was often difficult to get a high–level understanding of what exactly is going on when the code would get more complicated. I sometimes felt like he taught more with words than visuals.
Cold calls were nerve–wracking.
I think the material in class was irrelevant and hard to pay attention most of the times because it did not relate to actual software engineering methodologies at all.
The project felt like it wasn't really related to what we learned about in class much at all. I think either the project should be changed to be more similar to what we are learning in class, or there should be more lectures in class that teach about the things we need to know for the project. We should be taught how to use the tools we are expected to.
I hated the way the professor cold–called in class and then interrupted when the student tried to answer; it was truly demoralizing and pretty rude. If one genuinely didn't know the answer, he would proceed to pester them well into class time, which was extremely nwerve–wracking especially as someone who struggled with the material. The class was also poorly structured, with the bulk of the Ed lessons towards the end of the semester. This structuring was awful since we had 3 promised absences, no questions asked, but unfortunately the days that I couldn't make it to class were all the days of Ed Lessons. This was horrible. The professor also didn't explain the project rubric. For the first checkpoint, he made a verbal announcement in class ON THE DAY THAT THE CLASS WAS DUE THAT HE WAS CHANGING THE SUBMISSION RUBRIC – this is absolutely unacceptable, and then he proceeded to make it look like I was in the wrong for requesting that any rubric changes be made the day the assignment is released. If we as students are expected to be punctual and to–the–point in our deliverables and submissions, can we not expect

Comment
the same of our professor? Finally the groups – OH, MY GOD. I hope to God I'm an outlier in this experience but the group I was assigned (randomly) to work on the projects were some of thelaziest, most unresponsive, uncommunicative nimrods I have ever worked with who never did their work and waited till the day the project was due to make any contribution. And then they would make a bunch of bogus commits to the GitLab to make it look like they were contributing, and naturally since it was a so–and–so "team project", the TA and graders didn't really care to check who was doing what work. Overall this class was (hopefully) an easy A, but LORD IT TOOK OFF YEARS FROM MY LIFE, and I wouldn't wish this professor or my team on even my worst enemies.
The most challenging aspect of the course for me was the cold calling. While I understand that it encourages participation, keeps you engaged, and helps test your understanding in real time, I found it to be very nerve–racking and stressful, which sometimes made it harder to think clearly and respond confidently. One thing that could help future students manage this challenge more effectively is giving a brief moment to gather thoughts or allowing quick pair discussions before being called on, so students can feel more prepared while still staying engaged.
I think the challenging part might be that we have not covered a lot of tools and techniques that we need to apply in the project during the class, so it is quite difficult to learn it as a completely fresh stuff.

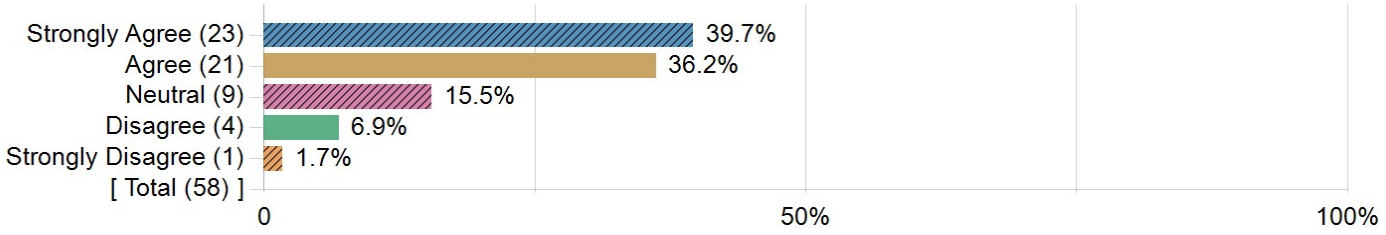
College, School, or Unit Questions

On average, approximately how many hours per week did you spend working outside of the course? Include time on homework, reading, reviewing, papers, projects, etc.



Statistics	Value
Mean	2.59

The course format (online, hybrid, face-to-face) helped me to learn.



Statistics	Value
Mean	4.05

Report Guide

Guide to the Interpretation of Course Evaluations at UT Austin
The goal of course evaluation process at UT Austin is to drive teaching excellence and to support continuous improvement in teaching and learning experiences. The two sets of scales used for core evaluation questions and the associated weights are:

Strongly Agree (5)
Agree (4)
Neutral (3)
Disagree (2)
Strongly Disagree (1)

Excellent (5)
Very Good (4)
Satisfactory (3)
Unsatisfactory (2)
Very Unsatisfactory (1)

The Mean is calculated by adding all of the weights for a single question and dividing by the number of respondents. The course workload question is not averaged.

The number of students (e.g. respondents) marking each option is reported for each of the items. These frequency distributions provide information about the level of student ratings and the spread and shape of the class distribution of responses. The distributions thus provide a picture of student perception of a course.

Course evaluations provide snapshots of student perspectives on their course-level learning experiences. Most experts on teaching evaluation advise that no individual method gives the complete picture of an instructor's teaching effectiveness; multiple and diverse measures, on multiple occasions, are advised to give a full picture of the teaching effectiveness of a particular instructor. Moreover, other factors, such as size of class, level of the class, and content of the course, can cause small variations in the ratings. Therefore, student perspectives for a particular instructor or course should be interpreted as a snapshot, and not as providing complete information on the teaching effectiveness of that instructor.