*** PROVISIONAL REPORT ***

UNIVERSITY OF TEXAS AT AUSTIN — COURSE-INSTRUCTOR SURVEY — Spring 2015 DEPARTMENT COPY
Downing, Glenn P — C S373 — 51995 — SOFTWARE ENGINEERING — Enrollment = 38
B100 BASIC — Surveys Returned = 36

	NUMBER CHOOSING EACH RESPONSE					NO. REPLIES THIS ITEM	AVG.
	Str Disag	Disagree	Neutral	Agree	Str Agree		
1 COURSE OBJECTIVES DEFINED-EXPLAINED	0	0	3	8	25	36	4.6
2 INSTRUCTOR PREPARED	0	1	0	4	31	36	4.8
3 COMMUNICATED INFORMATION EFFECTIVELY	0	2	0	10	24	36	4.6
4 STUDENTS ENCOURAGED-ACTIVE ROLE	0	1	1	6	28	36	4.7
5 INSTRUCTOR AVAILABILITY	0	0	1	11	24	36	4.6
6 COURSE WELL-ORGANIZED	0	0	2	11	23	36	4.6
7 STUDENT FREEDOM OF EXPRESSION	0	0	5	7	24	36	4.5
8 HELPFUL COURSE MATERIALS	0	2	4	7	23	36	4.4
9 STUDENT PERCEPTION OF AMOUNT LEARNED	1	1	2	8	24	36	4.5
	Vry Unsat	Unsat	Satisfact	Very Good	Excellent		
10 OVERALL INSTRUCTOR RATING	0	1	5	4	26	36	4.5
11 OVERALL COURSE RATING	0	0	6	7	23	36	4.5
	Excessive	High	Right	Light	Insuff		
12 STUDENT RATING OF COURSE WORKLOAD	0	14	20	0	2	36	
	Less 2.00	2.00-2.49	2.50-2.99	3.00-3.49	3.50-4.00		
13 OVERALL UT GRADE POINT AVERAGE	0	2	7	14	13	36	
	__A__	__B__	__C__	__D__	__F		
14 PROBABLE COURSE GRADE	12	14	8	1	0	35	

For the computation of averages, values were assigned on a 5-point scale so that the most favorable response was assigned a value of 5 and the least favorable response was assigned a value of 1.

COMMENTS:
Total Number of Comments: 22

---

1. Thank you!

---

2. I liked the team mechanic and working in a large group. However I thought that having us submit a set number of issues and encouraging a certain number of commits etc. was not the best way to encourage the development of a good team workflow. I think a better way might be to force people to have a development branch or follow a specific procedure for branching or merging, basically imposing a certain good but not perfect workflow, which would be more work for the groups maybe, idk. Just my two cents, I enjoyed the class, even though I feel kind've screwed over on the first project for forgetting to push documentation P.

---

3. Apiary was pretty easy to do, but almost an afterthought. Github, unittest, coverage, RESTful APIs, Flask, and Bootstrap are useful. More clear instructions about when to do the Git SHA for project one could be helpful to those getting started tonot lose 20 percent of their project grade for a simple misunderstanding. The Google forms could have been standardized order of names, number of members, etc Maybe a 3rd test, because it's hard to know what to study for. Most of the speakers were good, although they were preaching a lot of things I have heard many times before independent projects, internships, "passion"

---

4. I thought that being called on by roll would be more annoying than it was, but you do a good job walking people through the problems you ask them to solve. The quizzes, however, were exactly as annoying as I expected them to be. It frustrated me when you would have us read some number of chapters of a book and not quiz us on them at all. Extreme Programming was also one of the most cringeworthy textbooks I've read in my life. I felt really ill-prepared for the first test as well, because I feltlike in class we had been shown more of how to use things that we were asked to actually implement on the test, which was a huge difference.

---

5. The idea of requiring 10 issues on the repo especially such a high number is ridiculous. One simply doesn't have that many issues in some of the projects. This then encourages bad practices, such as purposefully creating issues in your code and committing them just to fulfill the requirement. I understand wanting us to learn to use github, but creating 5 of our own issues should be enough to do that. I also did not enjoy the last 3 assignments the website . It was tedious and not the leastbit rewarding. Lastly, the style of lecture where people are called on to answer questions that they couldn't answer with only this class and its prerequisite classes is ridiculous. It serves no purpose, wastes a lot of time, and lowers morale.

---

6. Overall this course was great and I'm glad I decided to take another Downing class. I learned more tools and languages that I was unfamiliar with. Each speaker this year was excellent, teaching us about tools that they use, showing me what to learn outside of classes. I liked the approach this class had with the group project, where it was less structured, relying on us to put in the effort to figure it out on our own. Only problem with the group project is that of course people divide up the work, which means that you don't get to learn about some sections of the project, maybe would be nice to have more time and rotate jobs, front end sql stuff etc., this way everyone get chance to learn about everything. Thanks for a great semester!

---

7. Quizzes should not contain material that hasn't been taught yet

---

8. First of all, I love your teaching style, and I think the depth that you go into the languages is brilliant and more engaging than most courses I've taken. However, at times, I was confused as to why this course was called Software Engineering. It seemed to be more of a course on Python and SQL, though I did enjoy the design patterns portion and felt it to be the most useful and applicable part of the class. I would encourage you to continue diving deeply into the in-and-outs of the languages, while also teaching us how to use this knowledge in real world applications, which to me seems most helpful. I've also taken your OOP course and felt the same way. But overall, you've been my favorite professor and you're doing a great job. - Eric

---

9. I'm a huge fan of Downing's class, so I don't have many complaints. I learned to use many tools in this class that will prove useful to me in the real world, something that other classes lack. I also made unforgettable friendships in this class. Calling students out one by one and leading them through the course material is a great way on making the students engage in class. The only thing that bothers me is the quizzes it's like they're made for us to not pass. Other than that, this is probably one my most favorite classes I've taken in my time at UTCS! Thanks!

---

10. Friends always told me to take at least one Downing class before I graduate, and I am so glad I did. The notion of being randomly called on in class daunted me at first, but it really kept me on my toes. The level of scrutiny you gave to every line of code was also EXTREMELY helpful. Not having done any Python before, I found the Collatz and Netflix projects as good exercises in not only familiarizing myself with the language but also with some of the tools. The group project, with all of its ups and downs, was overall one of the best experiences of my entire CS career. The guest speakers were all awesome, but I especially liked the Twitter dude. Saved the best for last. Anyhow, loved your class. Would take OOP if I had another semester.

---

11. What made this course excellent is its similarity to real-world programming. Professor Downing challenges us with software engineering concepts, writing documentation, marketing ourselves, guest speakers, rates of change in tech, current issues in tech through the articles, version control, working with a team on a 3-phase project, and learning new tools. What could be improved about this course post all of the code written in class, a lot of material to learn so it would be really helpful if there was a video recording transcript of the lesson being taught each day, and practice tests so we know what Professor Downing emphasizes to be the MOST IMPORTANT things to know rather than all the topics we'e learned. TY!

---

12. Great class, great professor

---

13. It was a really great class! The content was relevant and varied enough to cover a broad spectrum of different topics. The talks were very insightful and eye opening in regards to what software engineering looks like outside of college. My favorite part about this class has got to be the projects. The projects were very applicable to real life problems and were a great deal of fun to complete. Keep it up!

---

14. I 777 859 855 838 t 776 874 838 784 873 769 771 w 869 788 875 780 844 768 a 873 784 781 843 874 785 780 s 786 772 872 830 859 874 772 873 838 870 785 836 788 849 s 779 844829 868 875 877 uc 768 785 774 783 870 873 h 849 859 788 877 878 768 784 844 a 868 770 777 838 769 843 848 g 873 776 869 855 879 r 876 768 859 785 830 836 870 e 834 770 781 786 776 829 a 773 859 855 787 869 t c 780 781 855 829 780 834 782 l 829 859 785 831 a 830 s 848 781 878 784 772 s 775 782 . 776 770 784 794 831 877 871 859 879 878 848 794 I 872

---

15. git - love it GitHub - love it unittest - good testing experience coverage - neat tool, never used any kind of coverage before pydoc - neat tool Collatz - good easy intro into python and course requirements Netflix - liked it, interestingproblem Apiary - neither love nor hate RESTful APIs - good experience for real world Rackspace - good experience as well Flask - enjoyed using this framework. straight forward to set up and use, no hassle Bootstrap - fun to use Speakers Rackspace - enjoyed it Square - enjoyed it MutualMobile - enjoyed it Cloudera - wasn't present that day

---

16. I really liked the final project. It was cool working a larger project in a group I feel like this project taught me more than any other class assignment about what a work environment is like. Also, I now know how to make a website!

---

17. I thought the course was excellent. I would suggest spending more time on refactoring, and going slower through certain points. It's a lot of information, and sometimes I think Prof. Downing gets excited, talks too quickly, and accelerates throughthe material. I would not assume that everyone knows certain points or nuances. Sometimes it takes going slower and going over things a few times, like with math. I like going line by line in the code. It is very effective. I would do even more with good coding examples. Overall, a great job.

---

18. This may be beyond the scope of the class, but I would've appreciated going through some of the tools used for the projects in class. Most of them I'd never used before, so the learning curve was quite steep, especially considering that we ALSO had to stay on top of the material taught during lectures. My group always spent WAY longer on each project phase than we estimated, which could've been avoided with just a little more help.

---

19. I would love to see more conversation about Javascript and when to use it within projects and how a front end and back end should properly work together. It seemed like a common issue with teams. I also love using real tools like git, GitHub, TravisCI, Apiary, etc. Not sure how much time we should have spent on setting up Rackspace but it felt like too long. Wondering if a PaaS is better for the class like Heroku or Google App Engine. I didn't like the style of the Exam 1, while most of the class had practical things, it was not. Rewriting existing functions is educational, but not great at testing practical knowledge, which is what the class is made up of.

---

20. I would definitely recommend changing writing your own extra credit for assignment 1 Collatz . I seems to have been removed from SPOJ index for a reason you can only find it archived . Most successful times are just reverse engineering the answers rather than actually optimizing algorithmic performance! Overall, I think this class was fantastic. Well-organized, consistent, informative, guest speakers provided a nice logical break. Honestly, the only real speed bump I noticed is that the second group of people who drop the class after the group that drops first week all drop right around project group formation. Not sure what the solution to this is, if there is one, but the timing seems to coincide.

---

21. I would've liked the course to have focused even more on software design, testing, and techniques rather than Python, given that the course is titled Software Engineering. Consider re-writing the quizzes to maximize their clarity. Quite a few quizzes were confusing to me or led me to misread them, causing me to lose points on topics that I actually understood. Please use more descriptive names in code when possible for your examples. Names such as 'uf' and 'bf' are fine, but random single letter variable or class names should be named to something better. Research and review the assignment specs more before giving them out. Collatz extra credit and Flask models are good examples of problematic specs that were not planned well.

---

22. I honestly couldn't stand the way Downing treat students like we're in Highschool. Required attendance with serious grade repercussions and required participation in class show a distinct lack of trust in students. Additionally, the way projects aregraded is idiotic, the bonus points from tests can only serve to reduce the overall grade of the class I would be surprised if you ran grades with and without those points and this weren't the case . Oh, and grader feedback was completely worthless, we were given virtually no constructive feedback "didn't like the search page", what does that even mean

---