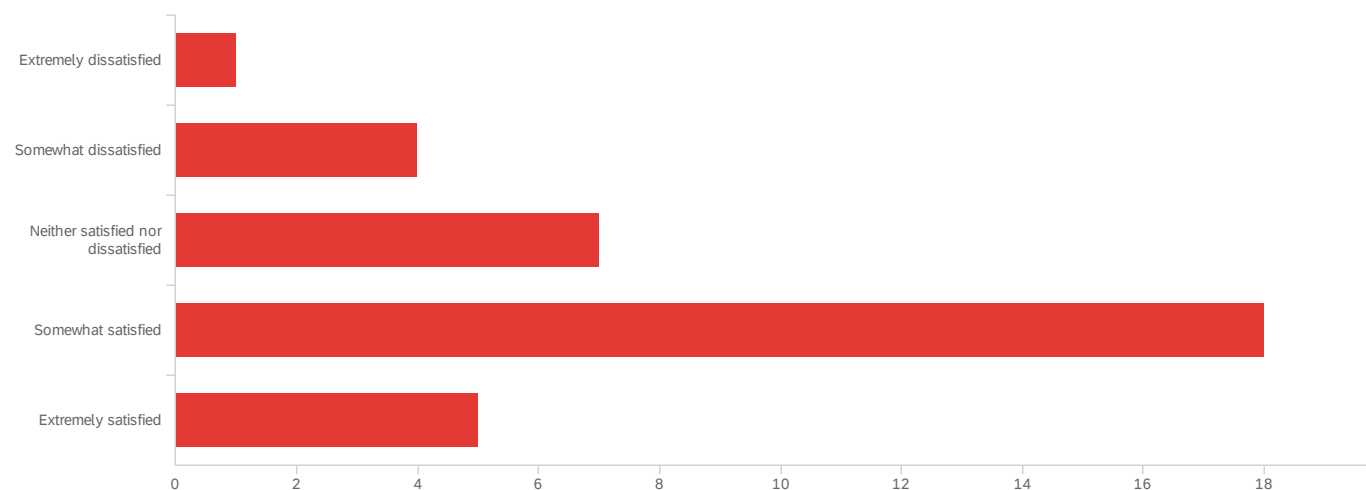


Default Report

SENG302 Testing workshops feedback

February 15, 2022 3:21 PM MST

Q1 - How satisfied were you with testing workshops in seng302?

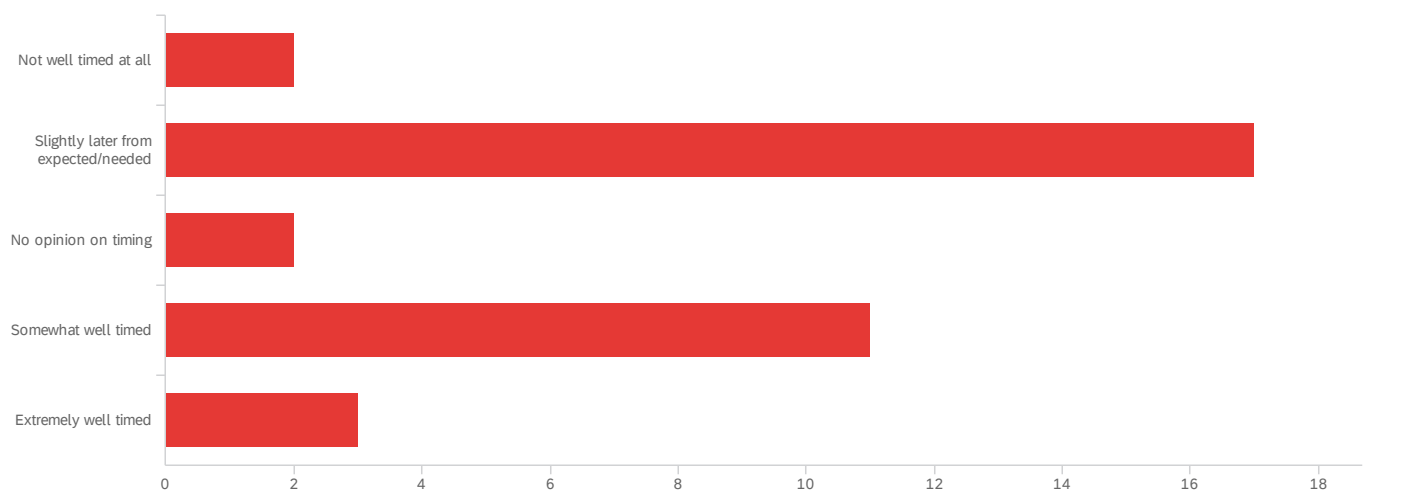


#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	How satisfied were you with testing workshops in seng302?	1.00	5.00	3.63	0.96	0.92	35

#	Field	Choice Count
1	Extremely dissatisfied	2.86% 1
2	Somewhat dissatisfied	11.43% 4
3	Neither satisfied nor dissatisfied	20.00% 7
4	Somewhat satisfied	51.43% 18
5	Extremely satisfied	14.29% 5
		35

Showing rows 1 - 6 of 6

Q2 - What do you think about timing of testing workshops (first half of term 2) ?



#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	What do you think about timing of testing workshops (first half of term 2) ?	1.00	5.00	2.89	1.17	1.36	35

#	Field	Choice Count
1	Not well timed at all	5.71% 2
2	Slightly later from expected/needed	48.57% 17
3	No opinion on timing	5.71% 2
4	Somewhat well timed	31.43% 11
5	Extremely well timed	8.57% 3
		35

Showing rows 1 - 6 of 6

Q3 - Please provide reasons for your chosen answer in the previous question.

Please provide reasons for your chosen answer in the previous question.

halfway through the project - already expected to have been testing our code

Our team had low Jest test coverage. This was because we didn't start writing Jest tests until after the workshop. This meant files that were created before the Jest workshop had low test coverage. Our team was unable to go back and write Jest tests for all older files (we did try). I believe having the Jest workshop earlier would have been beneficial for our team, as we would be able to write Jest tests as we went, instead of having to go back to older files if we had time.

Our team had a range of experiences with testing so it would have been better to have had the lab much earlier as by the time we had the lab most of our team were up to date with most testing processes.

It felt very disorganised and I didn't feel like much was learned that 301 didn't cover

Felt like a lot of the workshops covered stuff we already knew from writing tests. Maybe if they were done earlier it could help teams start off on a good footing.


Would have been helpful to have the testing workshops earlier as we didn't understand what we were doing in terms of backend testing and ending up having to rewrite a lot of our tests

The testing workshops were very helpful indeed, they got us comfortable with new technologies such as Cucumber and Jest. However, their timing was way later than I hoped it would be. Especially in the context of my team as our first few sprints, we basically neglected testing as in the first sprint we did not know we are supposed to test the frontend, the 2nd one we generally "didn't care" and jumped right into development, and thirdly when we got to the workshops, a lot of us had this "well it is too late to go back" mentality which held us back. Not entirely blaming it on the timing as much as I am blaming my team's culture of resistance to change.

The content was good but could possibly be improved by an introduction a bit earlier on as it felt quite bad having to write tests for already written code, . . . Though could obviously overload people having too much too early

The labs are useful because not only do we get to experience how to test certain things, but we can go back on the lab later in the year when we need it more. I always thought they came too late and I hold the opinion that all the labs should be done as quick as possible and then make it easier for students to access whatever they learnt in the lab later in the year.

The workshop was at a time when I was starting to think about improving my testing. I found Mathew's explanation about mocking and how to unit test really helpful.

Everyone in our team  seems to enjoy and is able to follow the material pretty well.

The testing workshops were great to help build on my understanding of testing and how to effectively test. I think they would have been slightly more helpful if they were done earlier in the year.

It would've been nice to learn how to use mocking and Jest earlier on, especially how to configure them. But it was early enough that we were able to pick it up

I think the content of the workshops was mostly adequate, however more of a focus on how to mock things in Vue with Jest would be ideal as that was something our team struggled with a lot. Specifically mocking Vue router and mounting child components with props would be ideal (although I understand that Vue router was a dependency that not everybody used). The timing is also a hard question to answer. Our team felt that we should have started all forms of testing far earlier, but then again dumping all of that on us at the very beginning would probably have done more harm than good. I think the Jest workshop could have been earlier though.

Pretty much just covered basics which we already knew.

Please provide reasons for your chosen answer in the previous question.

The TDD and ATDD workshops were well timed and our team were able to maintain a good test coverage for the backend. However, it would have been better to have the Jest workshop earlier as our team had low test coverage and some components were not tested as we didnt have time to go back and test older methods etc.

Most teams had some team members with knowledge of JUnit and Cucumber from SENG201/SENG202, but few teams had members with knowledge of Jest. I think it would have made more sense to have the Jest workshop first as it would have filled a significant gap in many teams' skills.

It would've been more effective to learn testing sooner rather than later. I think an effective time should be before the due date for sprint 1. The testing workshop should be when the GIT workshop was as GIT should not be a formalized workshop and should rather be an informal workshop (i.e. non-compulsory) for those not familiar with GIT.

The jest one came a bit late as our team had already figured it out but other teams had done none and we had already had feedback saying we need more tests before we had learned it properly

For me personally it wasn't until term 2 that I really invested any significant effort into the tests in our codebase. If the workshop had been any early I don't think I would have utilized them to any significant extend and if they were any later it would likely have been too late to actually make any difference.

Would be nice if it was done earlier (but I understand why it wasn't given we were in backend/frontend teams at the start of the course)

I appreciated the fact that they were held earlier in the year rather than later. However, since testing is a fairly important part of the development, I think that it could have been held slightly earlier - perhaps in the final weeks of term one? This would give us a few weeks in the holiday to play with the testing suite if we wanted to.

Would have been better to have these workshops as soon as possible so our projects weren't untested

The timing was well as it provided us without enough time in future sprints to implement what we had learnt.

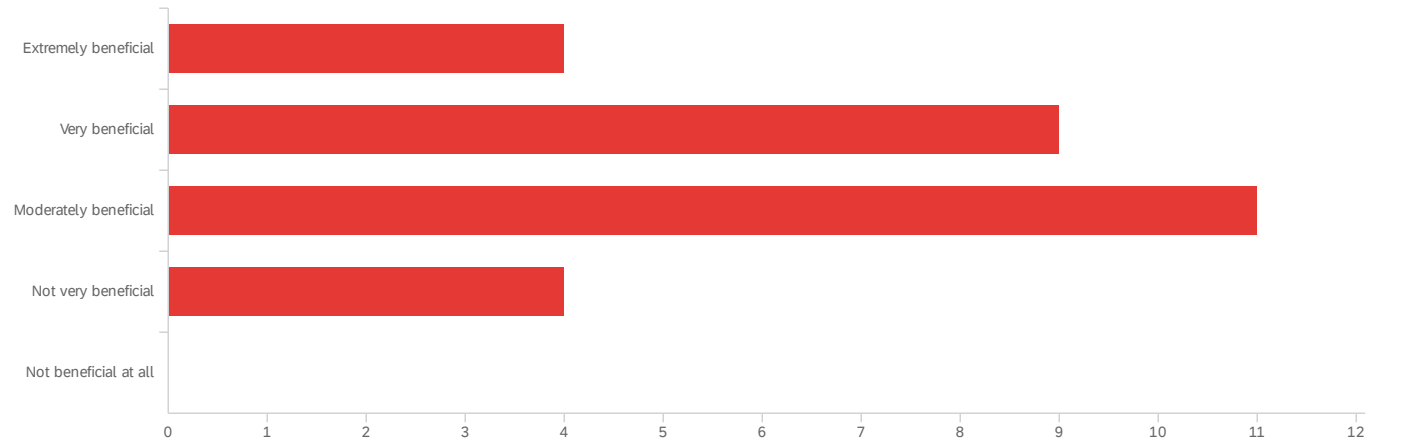
Could the workshops be on other days in the timeslots for 302?

The Jest workshop in particular was far too late than needed, because by that point there were a significant amount of frontend components that required Jest testing and it was a continuous struggle of playing catch-up from that point onward. Additionally, the workshops didn't contain useful information around common issues that everyone frequently ran into, and some content, for example the TDD workshop with Mockito, differed from the documentation for Spring Boot testing that my team had found earlier.

It was already sprint 2, but the information was very relevant for sprint 1.

It would have been more helpful to have the testing workshops as soon as possible, as it's beneficial to start testing as early on in the project as possible. Having the testing workshops later may create situations where there are large parts of code early in the project that aren't tested.

Q4 - This year Testing strategies workshop has covered Unit testing, Mocking and Acceptance testing. Did you find it beneficial to cover them all in one workshop?



#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	This year Testing strategies workshop has covered Unit testing, Mocking and Acceptance testing. Did you find it beneficial to cover them all in one workshop?	1.00	8.00	4.68	2.82	7.93	28

#	Field	Choice Count
1	Extremely beneficial	14.29% 4
2	Very beneficial	32.14% 9
7	Moderately beneficial	39.29% 11
8	Not very beneficial	14.29% 4
9	Not beneficial at all	0.00% 0
		28

Q5 - Please provide reasons for your chosen answer in the previous question.

Please provide reasons for your chosen answer in the previous question.

I struggled with mocking during the start of the project. It could have potentially been more beneficial if more time was spent on it. Otherwise, having the testing strategies all covered in one workshop was good. I was able to see the similarities and differences between the testing strategies.

The resources were sufficient to learn all three variations so I doubt spreading out the content into different labs would be necessary.

It felt like the workshop touched too little on all of them and I was still very confused about mocking and acceptance testing

I feel they should have been broken into categories so the students could think clearly about the difference between "Unit testing", "Mocking", and "Acceptance testing".

Yeah I think it's fine to have it covered in the same lab. As long as it covers everything that the students might need I see no issue.

Helped outline the difference between them all

We already had jest test seetup but the workshop show us the best way to write the test and a better understanding of the setup and tearDown and introduced mocking.

These were all aspects that were very useful for writing good and effective tests and helped my group immensely in testing.

I found the labs very informative and was able to apply the skills afterwards.

I think these are all necessary things to cover and alone they might not be enough for their own full workshop so combining them was good.

I found it more helpful to get a brief introduction to all the concepts early, rather than to have a more thorough workshop on individual topics later on. Getting an introduction gave my team the foundation to do more research on our own and get a better understanding of the concepts.

The workshops helped our team improve our backend testing as we were doing slow integration testing that would recreate our application with every single test.

Mocking is such a big part and probably needs more focus

If they had been covered in seperate workshops it would have taken significant time out of our time to work on implementation so I'm glad that there wasn't more time dedicated to it.

It was good if they were all covered but given the brief nature of it (as they were all in one workshop), my learnings of each specific method were limited. Offering more workshops specifically tailored to each one would have been good.

It was good to see how they all fit together, and it was good to hold them together to save time and allow for more time in development, but a small improvement could have potentially been made by separating out acceptance testing and unit testing from each other.

Might have been too much content in one workshop, couldn't practice it all

I think just introducing us to them was enough, I don't think in future people need a full on workshop for these. Personally I zoned out during the workshop and learnt it myself when I got home.

There is no need to split it into separate workshops.

Please provide reasons for your chosen answer in the previous question.

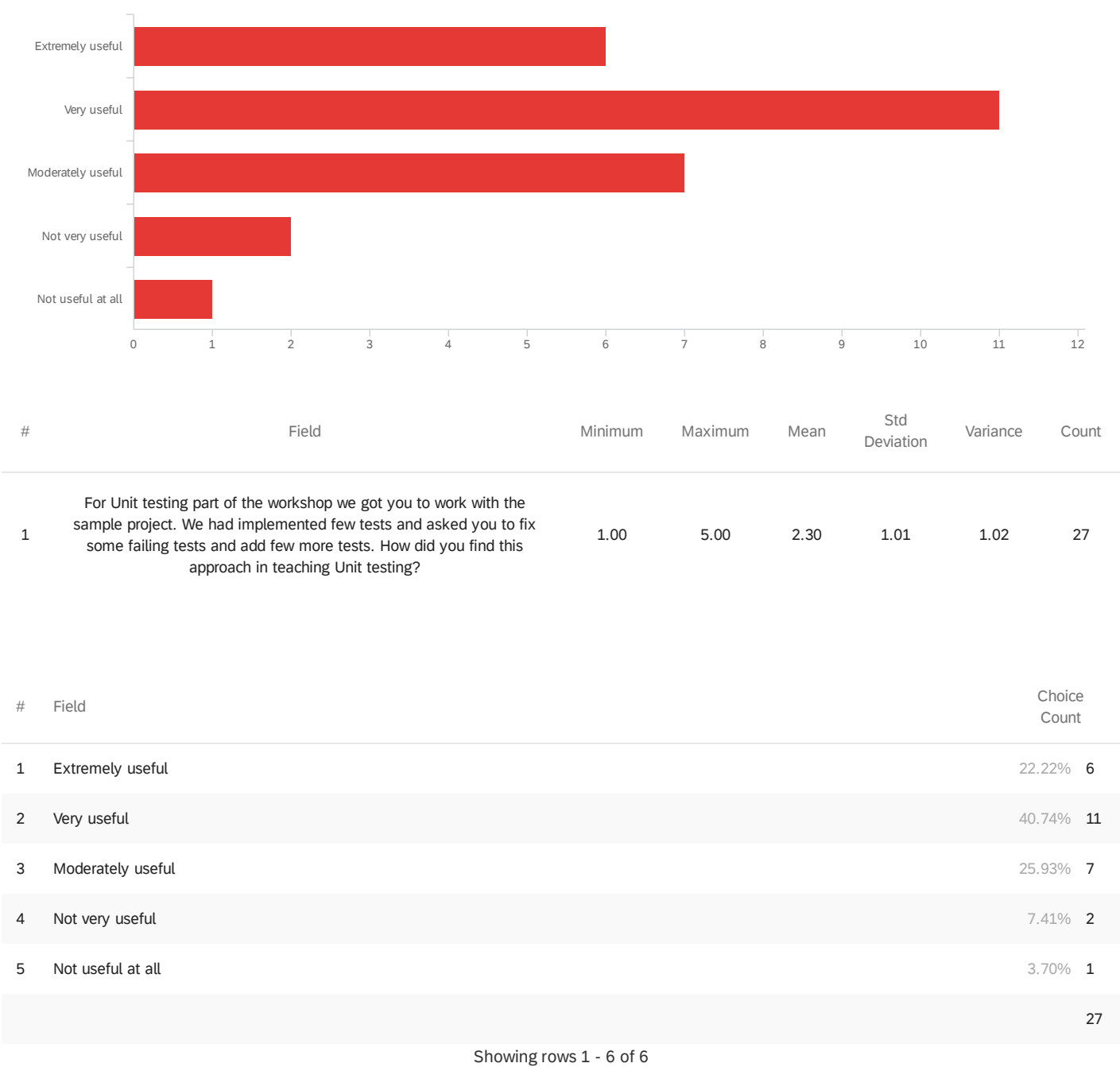
I feel Unit testing shouldn't be in the testing workshop, besides from a footnote as we're expected to know that we should be unit testing?

A lot of people had covered unit testing previously in the SENG201 and SENG202 courses, so spending less time on unit testing was good and covering mocking and acceptance (cucumber) testing was beneficial. However, there should have been more time allocated to cover cucumber and mocking more in depth as there were a lot of issues with mocking cucumber in particular as it very heavily differed from mocking in the other (JUnit) integration tests.

They're useful techniques that are quite easy for students to miss when looking at the documentation of the technologies in the stack. Resulting in a very difficult testing experience.

Several workshops would have given more time to ask questions about testing.

Q6 - For Unit testing part of the workshop we got you to work with the sample project. We had implemented few tests and asked you to fix some failing tests and add few more tests. How did you find this approach in teaching Unit testing?



Q7 - Please provide reasons for your chosen answer in the previous question.

Please provide reasons for your chosen answer in the previous question.

I can't remember doing this. I think I potentially had issues with the setup. However, I do believe it is useful to have the opportunity to put into practice what I had learned. I feel the main issue with being given tasks to work on during the workshops is that some tasks were not given enough time and some were given too much.

I find learning is easier by doing the topic so the git project was a good idea in my opinion.

This was great. Worked very well for me

This approach is great for getting us acquainted with how we should test in our codebase.

It was an interesting approach therefore made it a bit easier to sit down and do. I see no problem with this approach.

I don't really learn things unless I do them myself. This was good as I was able to try it myself and ask for help when it didn't work.

Practical > Theory

Having hands on practice with testing helps to understand what does and does not work when writing tests.

It's good to get hands on and see some examples. Fixing some examples made this even better.

I think it was a useful exercise.

It's been so long since the workshop that I honestly don't remember sorry.

This was an effective way to learn unit testing.

Not actually understanding what we were testing. Better off getting us to write one on a basic feature on our project

I was very unfamiliar with tests in the first place and having example tests to work on was very good for me. I like having templates to work off of.

The sample tests provided gave some insight into how the tests could be formatted. However, there was a slight pressure to conform to the style of the pre-written unit tests, which may have impacted the quality of what was written. I did like that we had to fix failing tests though

Allows us to see what is wrong with some tests and fix them.

Same answer as before, I just zoned out, wasn't interested but learnt it all at home which was very helpful

It is easier to learn and understand Junit in a hands-on approach.

I find it was a good jumping off point for a lot of students and also aided in learning how to problem solve when it comes to tests. However, I find that there was a lot of focus on the tests themselves rather than the testing mindset, which would've been beneficial. Learning how to write the tests and fix them is one thing, but learning the analysis and how to assess whether you have good coverage is something that will always be more beneficial as it can be applied to any testing framework. Another useful thing would be to teach students how to catch false positives in tests rather than just fixing failing ones, as those are often problems that go unnoticed in the real world and can hide substantial issues with your projects.

I did not find that the practical experience of fixing the tests drilled into the rationale for how to write good tests.

Please provide reasons for your chosen answer in the previous question.

It was useful to see examples for how units tests were implemented. However, unit tests that were more closely based on Spring Boot would have been more beneficial.

Q8 - Was there anything missing from Unit testing part of testing workshop?

Was there anything missing from Unit testing part of testing workshop?

Not that I can recall

no

No.

Not that I can think of.

Creating a competition between teams along with rewards to induce motivation

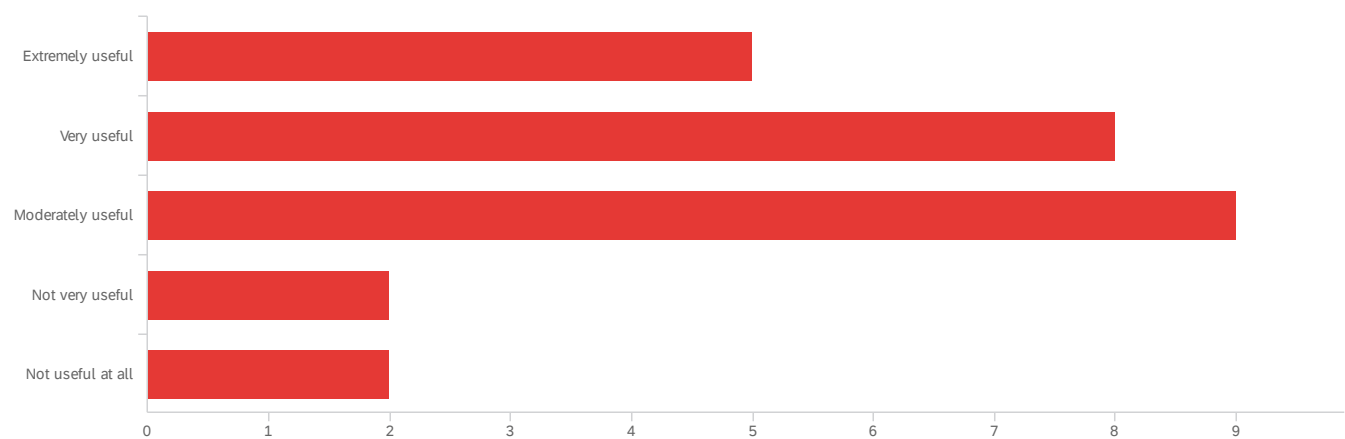
Parentising tests

Not that I can remember no, just going in more depth would have been good.

Not that I can recall

I think there should've been a heavier focus on test analysis to teach students the skills to know when there is sufficient coverage and how to analyse the test cases that are required.

Q9 - For the Mocking part of the Testing strategies workshop we have showed you how to implement mocking on the sample project. How useful did you find this approach in teaching Mocking?



#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	For the Mocking part of the Testing strategies workshop we have showed you how to implement mocking on the sample project. How useful did you find this approach in teaching Mocking?	1.00	5.00	2.54	1.12	1.25	26

#	Field	Choice Count
1	Extremely useful	19.23% 5
2	Very useful	30.77% 8
3	Moderately useful	34.62% 9
4	Not very useful	7.69% 2
5	Not useful at all	7.69% 2
		26

Q10 - Please provide reasons for your chosen answer in the previous question.

Please provide reasons for your chosen answer in the previous question.

Demystified a lot of the concept of mocking, and why it is beneficial to mock to test a specific component. This practical approach taught how to mock components very quickly and simply.

Again, I don't recall doing this, but it would be useful to put into practice what I had learned.

Was useful to carry out tests in a realistic context.

It was a good idea but was far too confusing to understand

Some useful information was there, but I feel going in depth with that would have helped the students better. In my earlier sprints, I struggled and put too much time in getting mocking in Cucumber and Jest to work.

Mocking was always something a bit difficult to do in our project, and I can't help but feel like the mocking we did as part of the workshop covered the very basics, but those basics could have been found with a single google search on some stack overflow post. Mocking can get complicated depending on what you're trying to mock so I think it'll be hard to put all that into a single lab.

Actually doing it myself helps me learn

Mocking is essentially useful in the project / in the future. and it might take us way longer to find that out and we might be lazy to refactor it if its too late.

Mocking has been one of the most important parts of our testing suite. Learning to correctly mock made test writing a huge amount easier.

Implementing mocking on sample projects made it easy to understand how it fits in

Mocking is certainly something we needed to learn and this workshop was useful

It helped me learn the basics of mocking so I could do more research on it with my team.

The workshops helped our team improve our backend testing as we were doing slow integration testing that would recreate our application with every single test.

The complications we had with mocking could not be applied to the example project

I didn't understand how to do mock testing and because we were doing it in a team a couple of our team members basically did all of the work for this part while the rest of the group slacked. In this regard it wasn't useful to me but I can't honestly say that that's due to any fault of the workshop itself.

I felt like I didn't learn much from it

Laying the basis for the mocking in a context not dissimilar to our own was good to see, and helped to fill in some gaps with our understanding of it

Gave us knowledge on how to do it for our projects

Same again, I think just introducing/mentioning it is enough for us to work on it in our own time

Please provide reasons for your chosen answer in the previous question.

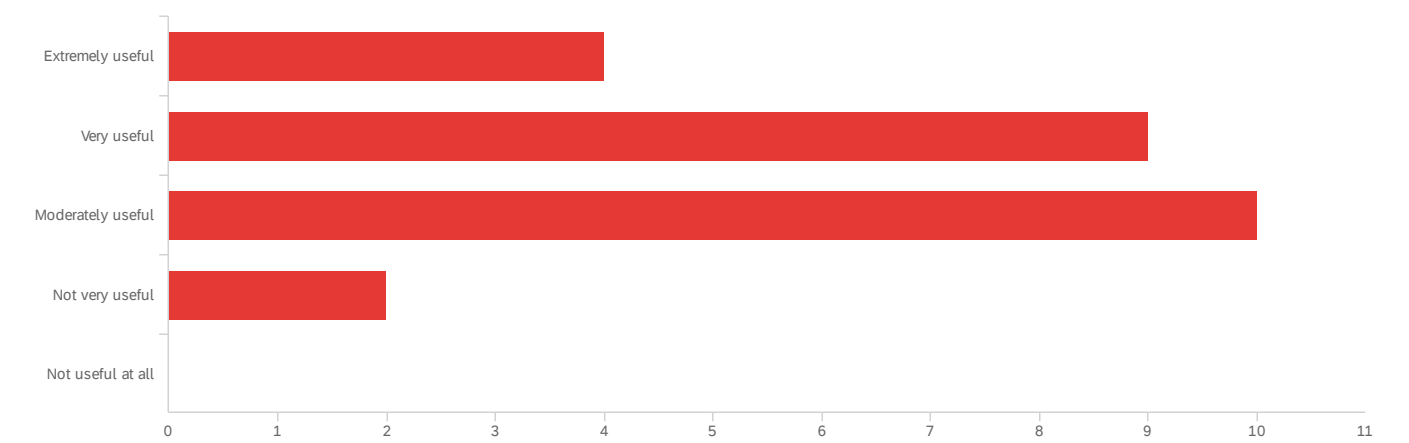
I do not recall the mocking part of the workshop.

The documentation differed from the Spring Boot documentation we had found previously, and we needed more coverage on how to mock within Cucumber tests as that differed again.

The example did seem a bit contrived, but I think it was broadly effective.

It was useful to see how mocking was used in real code, but an example of how to mock Spring Boot components (e.g. services) would have been more applicable.

Q11 - For the Mocking part of the Testing strategies workshop we have also walked you through a sample mocking scenario in the context of this year's project. How useful did you find this exercise?



#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	For the Mocking part of the Testing strategies workshop we have also walked you through a sample mocking scenario in the context of this year's project. How useful did you find this exercise?	1.00	4.00	2.40	0.85	0.72	25

#	Field	Choice Count
1	Extremely useful	16.00% 4
2	Very useful	36.00% 9
3	Moderately useful	40.00% 10
4	Not very useful	8.00% 2
5	Not useful at all	0.00% 0
		25

Q12 - Please provide reasons for your chosen answer in the previous question.

Please provide reasons for your chosen answer in the previous question.

Same reason as mentioned before: mocking was a mystery for us who never heard of it, and it helped at least some members to understand it, implement it in the project and other team members can follow that example when doing their own testing.

This was a good teaching strategy as it showed what thought process you should go through when mocking database calls.

Was useful to carry out tests in a realistic context.

Same as above

Pretty much the same reason as before, it was very basic and not something we needed a separate lab for.

I can't remember this part

Seeing an example and then apply it in our own project is the best way I can think of.

Again, having hands on experience with mocking helped to improve my understanding of how it works.

It was good to have a realistic example to go through.

I don't remember this part of the workshop so I can't really tell you how useful it was.

It was an effective and engaging way of testing.

If I had paid attention during that part of the workshop then I imagine it would have been useful. I find that samples are good for learning.

Didn't find it very helpful and left having to learn most of it myself

Seeing the way that mocking could be implemented in our actual scenario was very helpful to see how it would all fit together.

relevant knowledge on how to implement it on our project

Same again, I think just introducing/mentioning it is enough for us to work on it in our own time

I do not recall the mocking part of the workshop.

It's good to get practice, but at that stage my team had already learned how to perform mocking for our integration tests.

I would've liked it if it was a longer segment, but having a practical example is helpful for getting ideas for how to use mocking.

The sample mocking scenario showed some pointers towards how to mock in Spring Boot, but it wasn't fleshed out enough to be useful.

Q13 - Was there anything that you were missing from Mocking part of testing workshop?

Was there anything that you were missing from Mocking part of testing works...

While the mocking workshop was very helpful for some team members, it took a while after the workshop for some other members to grasp the concept of mocking and its purpose. Because of this, more time could possibly be spent explaining exactly what mocking aims to achieve in testing, and the common mistakes people make when mocking (such as mocking the components that one intends to test, which then causes confusion when said component and its methods don't function as expected)

I would have found some more examples useful.

no

Detailed scenarios

Possibly something like trying to mock a static class such as the java class Files. Or other more interesting mocks such as trying to mock multipart data.

Cant actually remember whats in the workshop but seems covered enough for me.

I think it would have been useful to go a bit more in-depth with how to mock database functionality, as this is one area that I somewhat struggled with.

Not that I recall.

Creating a competition between teams along with rewards to induce motivation.

More in depth on it so I could understand it better

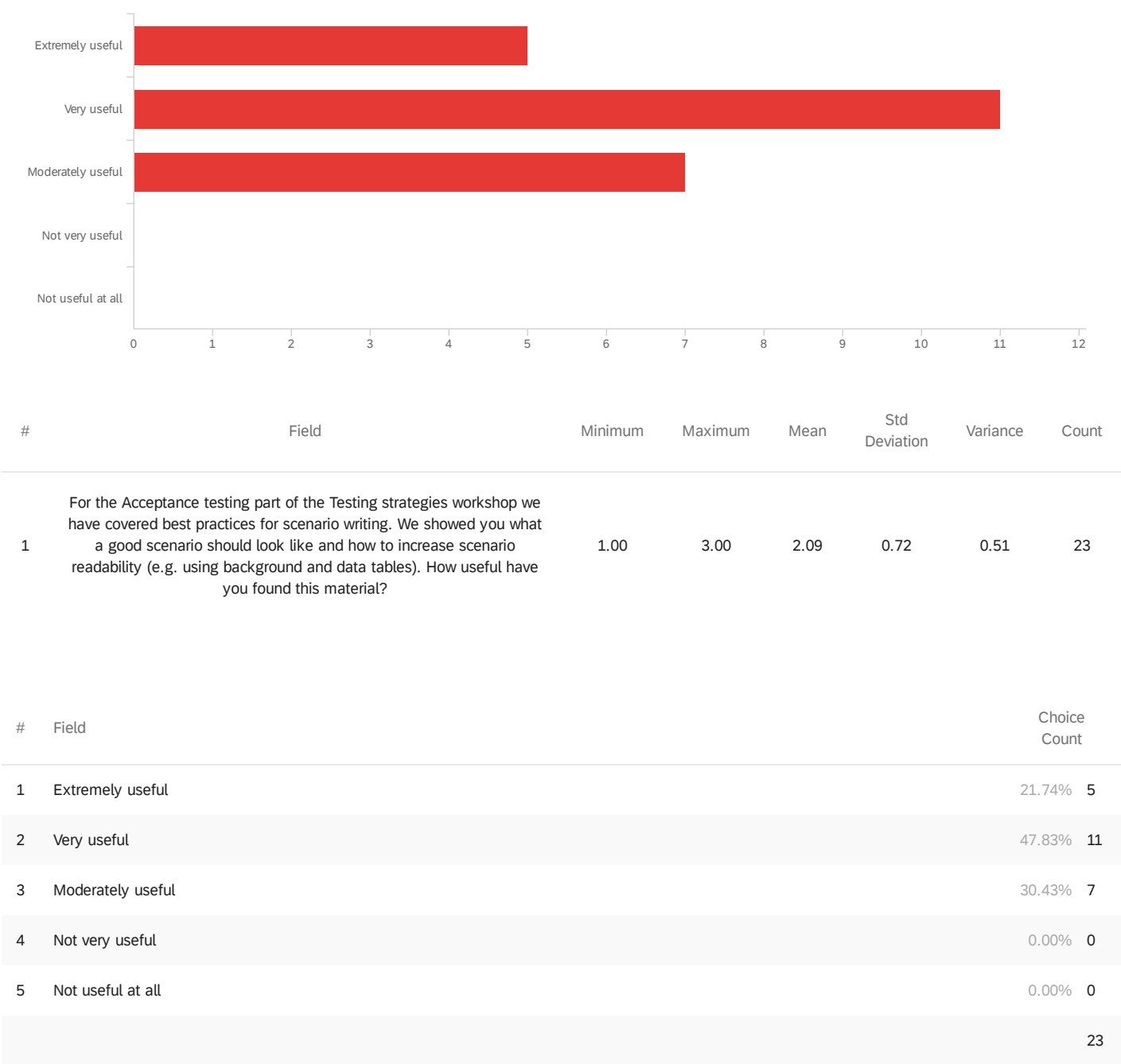
Not that I can recall - maybe the difference between the .do and the .then notations?

Covering how mocking differs when using Cucumber versus regular JUnit.

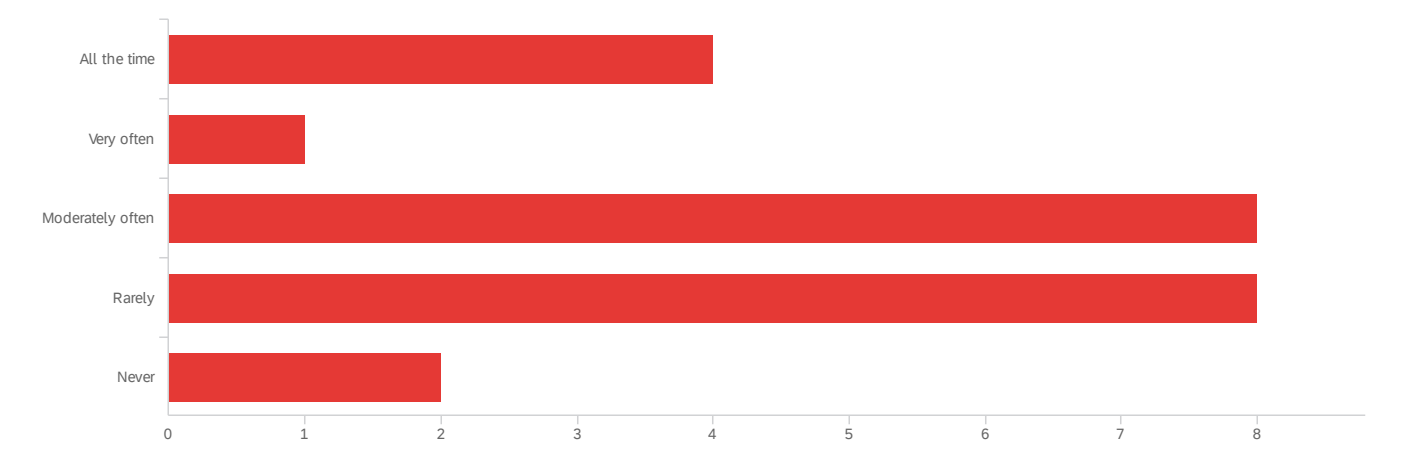
Maybe other kinds of matchers? Using a series of matchers?

Q14 - For the Acceptance testing part of the Testing strategies workshop we have covered best practices for scenario writing. We showed you what a good scenario should look like and how to increase scenario readability (e.g. using background and data tables).

How useful have you found this material?



Q15 - Have you been using material related to best practices of scenario writing while working on the project?

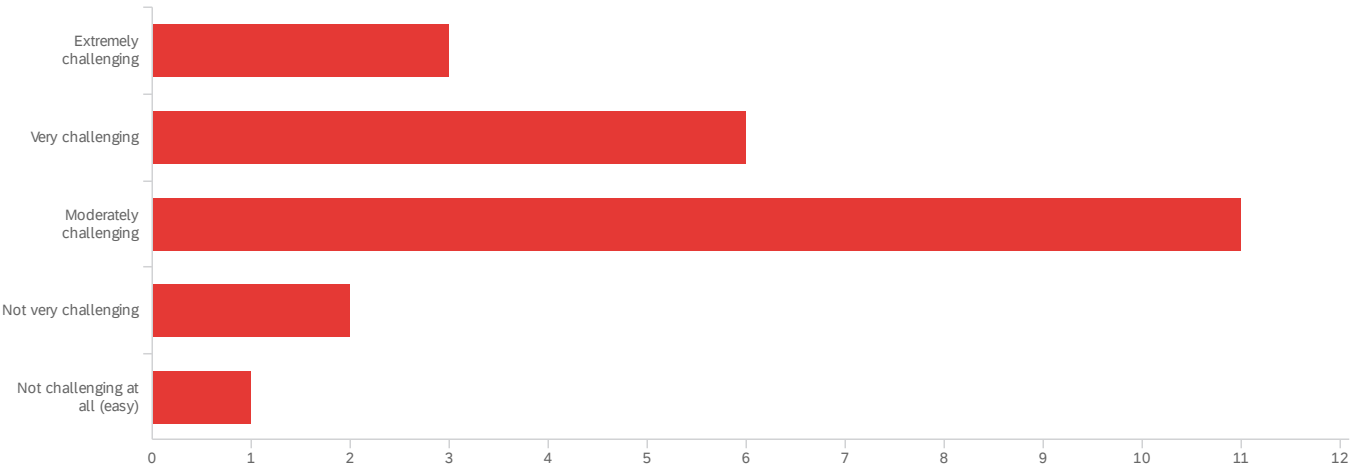


#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	Have you been using material related to best practices of scenario writing while working on the project?	1.00	5.00	3.13	1.19	1.42	23

#	Field	Choice Count
1	All the time	17.39% 4
2	Very often	4.35% 1
3	Moderately often	34.78% 8
4	Rarely	34.78% 8
5	Never	8.70% 2
		23

Showing rows 1 - 6 of 6

Q16 - As a preparation activity for Acceptance testing workshop we have asked you to set up Cucumber in your own projects so we can have time to work on Acceptance testing tasks during the workshop. How challenging have you find setting up Cucumber in your own projects?



#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	As a preparation activity for Acceptance testing workshop we have asked you to set up Cucumber in your own projects so we can have time to work on Acceptance testing tasks during the workshop. How challenging have you find setting up Cucumber in your own projects?	1.00	5.00	2.65	0.96	0.92	23

#	Field	Choice Count
1	Extremely challenging	13.04% 3
2	Very challenging	26.09% 6
3	Moderately challenging	47.83% 11
4	Not very challenging	8.70% 2
5	Not challenging at all (easy)	4.35% 1
		23

Q17 - Please provide reasons for your chosen answer in the previous question.

Please provide reasons for your chosen answer in the previous question.

Setting up cucumber came with many cryptic error messages, regarding setting up a specific spring configuration file and a cucumber runner file with the correct annotations and parent class in order for cucumber to run tests properly. The error messages related to this setup process were very cryptic and the setup was largely trial and error, and was very time consuming in sprint 2, when that time could have been better spent working on the sprint commitment.

I did not set up Cucumber for the team. I do recall other members having issues setting up Cucumber. I believe they had to ask our Scrum master for help.

There was little incentive to setup cucumber independently since only one person needed to get the ball rolling so I didn't bother to look into the process deeply enough.

We couldn't set it up for a very long time

Thank you to whoever on my team set up cucumber. I just had to write the tests, not set it up

Had to deal with the random little bugs that pop up when setting up a new technology in a project

Setting up thing or dealing with configuration is always tricky but thats also the fun part.

Having learnt to set up cucumber in SENG301 made it a bit easier. We just had to learn to adjust the setup to work with Spring Boot.

The initial setup was not very challenging for our project and we were able to get it going before the workshop

I knew some of the basics of Cucumber from SENG202, but I wasn't very confident with it.

Team members were already familiar with cucumber testing.

In that specific case the Cucumber tests were all set up by one guy in our team so the rest of the team got very little experience from that.

I didnt actually do it, it was someone else in my team

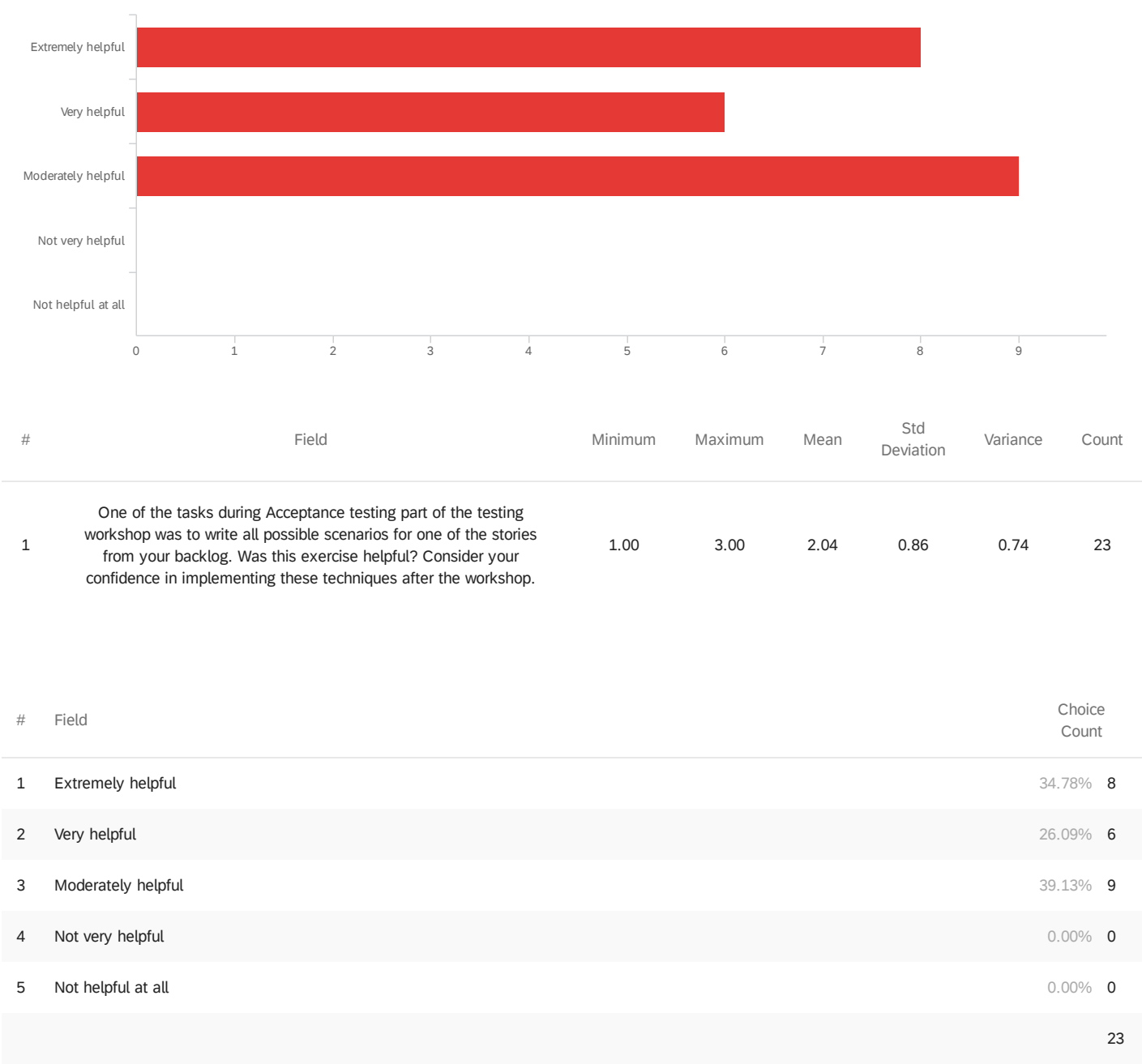
The acceptance test portion was extremely valuable as I was unsure what they were beforehand and how to set them up.

Cucumber was just difficult to setup in Spring Boot as there were a lot of configuration steps that don't have to be performed for other Java projects. I think covering the scenario writing was good to get students who hadn't used cucumber or hadn't used it well prior into the mindset of structuring them, however it should've been made more clear that this structure is good to apply to any form of integration testing, not just cucumber or acceptance testing.

I didn't work on the setup myself, so I have no idea.

Setting up Cucumber with Spring Boot was near impossible without the example given in the workshop. None of the online material worked for our setup.

Q18 - One of the tasks during Acceptance testing part of the testing workshop was to write all possible scenarios for one of the stories from your backlog. Was this exercise helpful? Consider your confidence in implementing these techniques after the workshop.



Q19 - Please provide reasons for your chosen answer in the previous question.

Please provide reasons for your chosen answer in the previous question.

The scenarios we wrote were overly broad, and did not cover a sufficient range of cases for each acceptance criteria.

Our team did not manage to finish all possible scenarios during the workshop, but it was a good starting point.

The setup of the course where there cannot be testing tasks did not lend itself very well to writing all possible scenarios for each AC. The independence of each task made it difficult to carry this out sometimes.

This would have allowed us to work on our testing if it had worked. I don't think we set it up in time

By the end of it, I felt like I was still unsure the best way to word the scenario. Got the ball rolling though

Don see other way to teach this

Writing scenarios was a good way to visualise all the things that can go right or wrong in a story.

It's good to have the test cases already layed out to start.

I think this was a good way to kickstart our cucumber testing coverage in our application and it gave us a good chance to ask questions if we got stuck

I don't recall this part of the workshop.

The way the scenarios were set up was very informative and easy to follow

Because this was something that was actually practical in regards to our project members of our team invested significantly more time and effort towards it than if it had been done purely as a learning exercise.

It gave us a template that we could use for other tests.

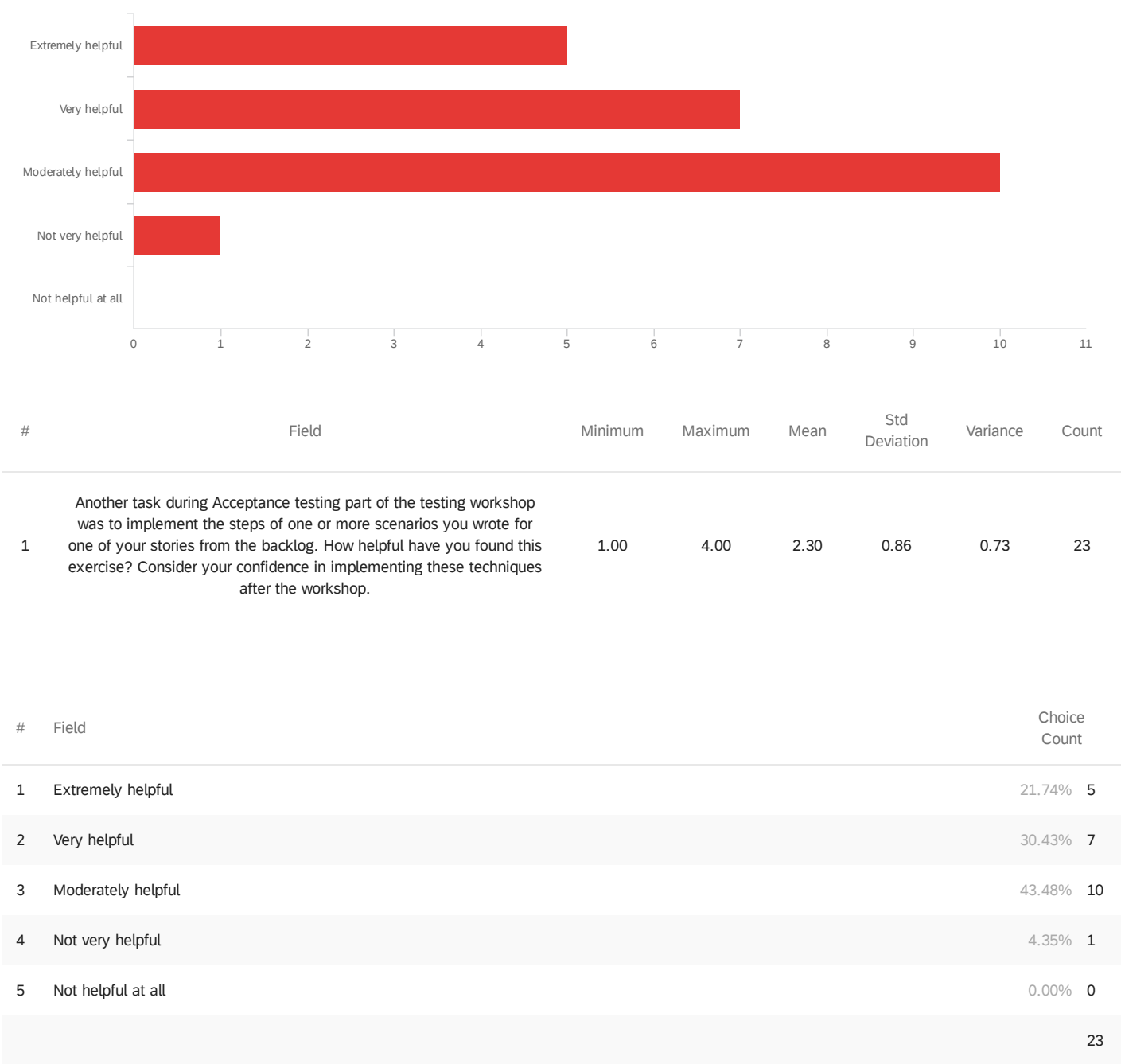
Made it easier to construct the acceptance tests.

There wasn't enough time provided to be able to get through all of the scenarios, as there was a lot of waiting around during the workshop to move onto the next stage.

It was very helpful to get feedback from a tutor to make sure that we were on the right track.

This gave us a lot more confidence in knowing that our acceptance tests were in the right format expected.

Q20 - Another task during Acceptance testing part of the testing workshop was to implement the steps of one or more scenarios you wrote for one of your stories from the backlog. How helpful have you found this exercise? Consider your confidence in implementing these techniques after the workshop.



Q21 - Please provide reasons for your chosen answer in the previous question.

Please provide reasons for your chosen answer in the previous question.

Good for gaining a basic understanding of how to implement the test, but subsequent stories would often come with their own set of challenges in regards to testing. There were also a number of other problems we encountered in the project related to cucumber, such as duplicate scenarios (not reusable) and database problems.

Again our team did not manage to finish all work during the workshop, but the work that was completed was a good starting point for writing future cucumber tests.

The setup of the course where there cannot be testing tasks did not lend itself very well to writing all possible scenarios for each AC. The independence of each task made it difficult to carry this out sometimes.

Still found it very challenging to word the scenarios correctly after the workshop

Gud and GUD

It was helpful learning to implement steps for scenarios, but I struggled some of the time to get them to work with our database setup.

After the workshop we found it difficult to implement these tests as we ended up rewriting a lot of code for duplicate step definitions and not knowing if that was the best strategy

I think this was a good as gave us a good chance to ask questions if we had any issues with a step

The feedback we got from the tutors helped us understand that we had been implementing our tests poorly, as we had been using internal logic to complete the steps of the scenario instead of using the endpoints.

After the workshop I was able to write cucumber tests for our project.

Same reason as above, having this as something that is useful for our actual project rather than purely as a learning tool was a good idea.

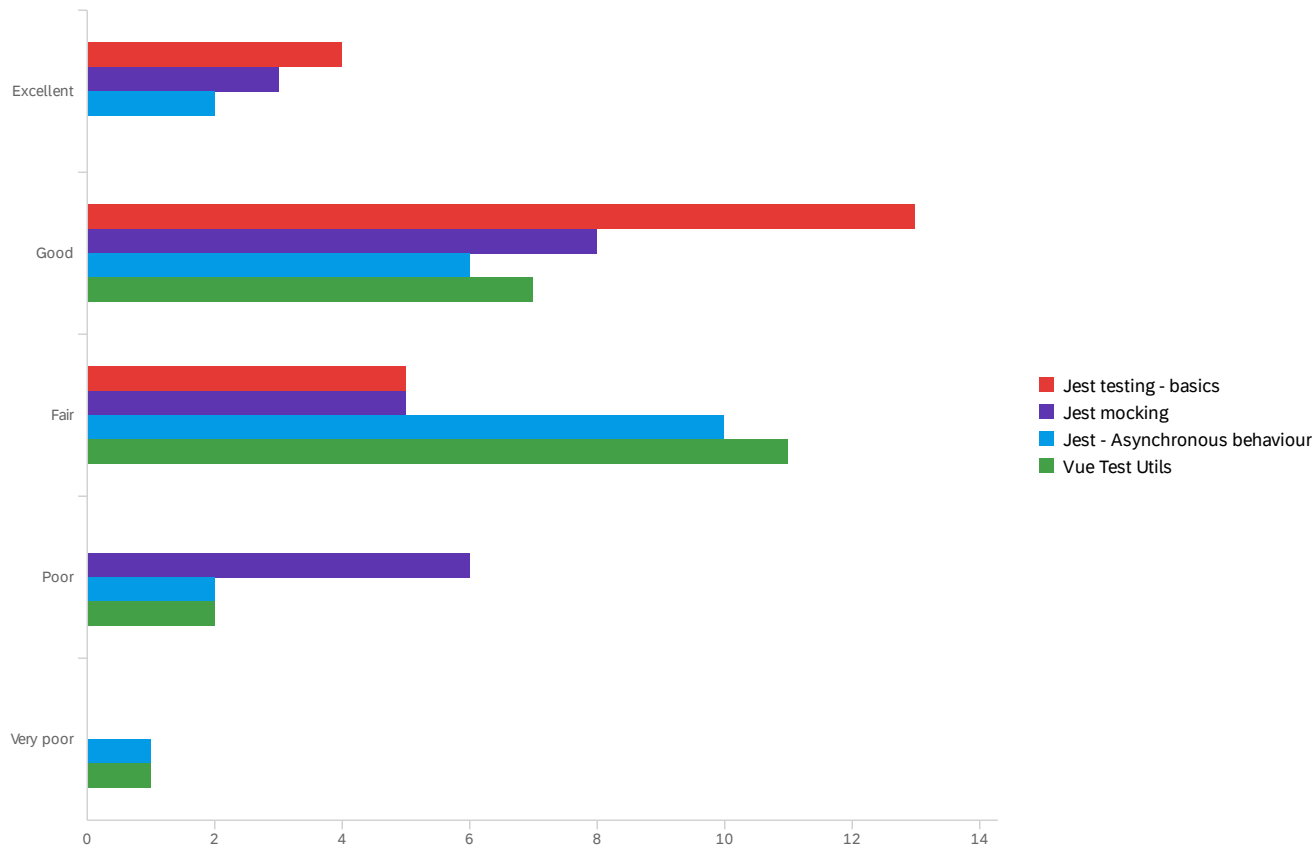
It gave us a template that we could use for other tests.

Not enough time in the workshop, as mentioned above.

I think it is quite reasonable that we can figure it out on our own, but it is nice to get it all working in 1 workshop session.

We weren't given much help on implementing the scenarios since we didn't have enough time to implement them. Most of the time we spent on writing the scenarios and getting help on fixing our Cucumber setup

Q13#1 - In Jest workshop we have covered basics of writing Jest tests, mocking, asynchronous behaviour an... - Quality of material



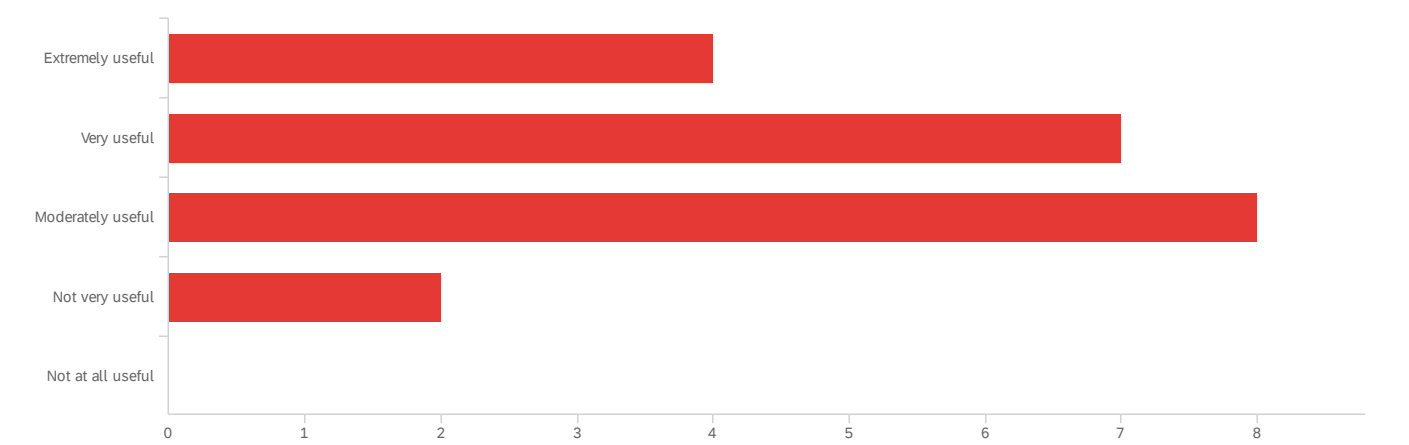
#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	Jest testing - basics	1.00	3.00	2.05	0.64	0.41	22
2	Jest mocking	1.00	4.00	2.64	1.02	1.05	22
3	Jest - Asynchronous behaviour	1.00	5.00	2.71	0.93	0.87	21
4	Vue Test Utils	2.00	5.00	2.86	0.77	0.60	21

#	Field	Excellent		Good		Fair		Poor		Very poor		Total
1	Jest testing - basics	18.18%	4	59.09%	13	22.73%	5	0.00%	0	0.00%	0	22
2	Jest mocking	13.64%	3	36.36%	8	22.73%	5	27.27%	6	0.00%	0	22
3	Jest - Asynchronous behaviour	9.52%	2	28.57%	6	47.62%	10	9.52%	2	4.76%	1	21

#	Field	Excellent		Good		Fair		Poor		Very poor		Total
4	Vue Test Utils	0.00%	0	33.33%	7	52.38%	11	9.52%	2	4.76%	1	21

Showing rows 1 - 4 of 4

Q23 - In Jest workshop we gave you the sample project with implemented various Jest tests as a reference. Your task was to write test for an existing Vue component in your project. How useful have you found this approach in teaching Jest testing?



#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	In Jest workshop we gave you the sample project with implemented various Jest tests as a reference. Your task was to write test for an existing Vue component in your project. How useful have you found this approach in teaching Jest testing?	1.00	4.00	2.38	0.90	0.81	21

#	Field	Choice Count
1	Extremely useful	19.05% 4
2	Very useful	33.33% 7
3	Moderately useful	38.10% 8
4	Not very useful	9.52% 2
5	Not at all useful	0.00% 0
		21

Q24 - Please provide reasons for your chosen answer in the previous question.

Please provide reasons for your chosen answer in the previous question.

Also de-mystified a significant portion of how Jest testing works in how to set up a vue wrapper and what purpose it plays to jest testing.

Didn't manage to get vue-test-utils working in the lab which resulted in us not being able to set it up for some time. Managed to set it up around sprint 4

I did not fully understand the content of the workshop. Therefore, I struggled writing tests.

I was unable to get this part of the project working even after asking for help from tutors.

Jest is funky but useful in testing Vue component, i think the vue dev tools in browser should also be introduced or taught to other student its useful to debug the store and other state thingy.

Jest was an area that I had little to no confidence with, but the Jest testing workshop made writing Jest tests a lot easier for me.

It was useful, but was difficult to transition during the lab

This was very useful as it allowed us to get real experience in our own project and we could ask questions if anything went wrong

It was useful to use our own project because my team used TypeScript for our testing, so it gave us the chance to talk about some of the tutors about how to change the testing to make it work for us.

Our team already knew how to do jest testing.

It was good to have a sample project to work on but it would have been better if it were related to the project. Also I remember not paying much attention during all of the Jest testing so I'm not totally sure.

easy to understand how to test what the team wrote

Same again, I think just introducing/mentioning it is enough for us to work on it in our own time

Was valuable in learning how to Jest test, was unsure how to beforehand.

The workshop didn't go in-depth enough into Jest or cover the common problems with asynchronous vs synchronous testing, and it didn't cover a lot of the more in-depth and most useful mocking techniques, such as mocking Vue router and using spyOn to check if functions or router pushes had been called. The majority of our application seemed to not be able to be tested based off of the knowledge from the workshop as we were told we can't test the router when we asked, but eventually we were able to find through our own research how to test more effectively using Jest.

I don't think working on the sample project was very helpful, but writing tests for a real world component and getting feedback for it was nice.

It would have been pretty useful to see a real example, but our team had already figured out Jest at that point.

Q25 - Overall, what other improvements can we make in the future to make testing workshops more useful?

Overall, what other improvements can we make in the future to make testing...

Touch on the best practices side of testing alongside the technical - what makes high quality tests, how to avoid making redundant tests, and common errors when testing such as mocking/stubbing components incorrectly could save a lot of time and headache debugging issues related to cucumber, mocking and asynchronous calls.

Testing workshops should be done as soon as possible otherwise there's too much tech debt to go back and write tests for old code and hence percentages are low

I believe that providing some more examples (especially for mocking) would be useful.

Have them early on in the course, if topics are separated into different labs have shorter content for each lab so that teams are not as easily distracted.

Have a real time writing test case session maybe on how to debug instead of just writing?

Briefly touch on testing with a database.

Some info about configuration especially for jest would be useful

Perhaps do them earlier so we can get some help with testing right away. Otherwise they were very useful.

Have them in a venue where everyone can see/hear the person giving the presentation.

Creating a competition between teams along with rewards to induce motivation.

I think having testing done as a team activity might have been a mistake as it allowed one or two team members to do all of the work while the rest of the team could goof off.

As mentioned previously, separating the testing types out and going more in depth with them would have been better.

I think telling everyone what they should be doing and providing a pdf with examples and links for people to look at is enough. I found the workshops very boring (not because of the people running them, just the content) and didn't learn much during them. The rest of the course is structured in a learn yourself way so why cant testing be too?

More interactive

Spending more time on the testing workshops and holding them earlier, especially Jest. They need to go more in-depth than they did to be effective, especially the Jest one, and they should also focus more on the test analysis side to provide students skills that can be applied to any testing framework.

Put them earlier. How do you expect TDD if you teach testing after the fact?

Have the workshops as early as possible, and give more time to cover tricky topics (e.g. Cucumber)

Q26 - Reflect on the year as a whole. What has helped you the most in improving your testing skills (inside or outside the formal teaching activities)?

Reflect on the year as a whole. What has helped you the most in improving y...

Practice, various articles online, feedback and time constraints mean that testing has to be done efficiently and avoid redundant tests. Separation of concerns means they are individually testable using mocking. Writing down the cucumber scenarios during the planning stage gave a guideline for testing, so time is not wasted trying to think of scenarios to begin with.

I believe pair programming with other teammates has helped me most to improve my testing skills.

The lab printouts

Pair programming with someone else that tested well and looking at previous testing examples

Master of Funky Test - 

The testing workshops provided a great foundation, and then constant practice helped me improve further.

Struggling with my own code while pair programming helped the most for me

Both receiving feedback/guidance from our scrum masters each sprint about our testing and seeing my teammates implement excellent tests that I did not know were possible at the time.

Pair programming with more experienced team members.

My team did a testing session in which each member was paired with another member e.g. backend with frontend. They had to do tests for backend and frontend functionalities and the Sprint 1 backend member would teach the frontend member vice versa.

Doing testing as a pair-programming task made it less monotonous but through the year my testing progress was generally atrocious.

Pair-programming and Co-located work

Outside. There are so many different tutorials on the internet that cover so much more than you could fit in a workshop

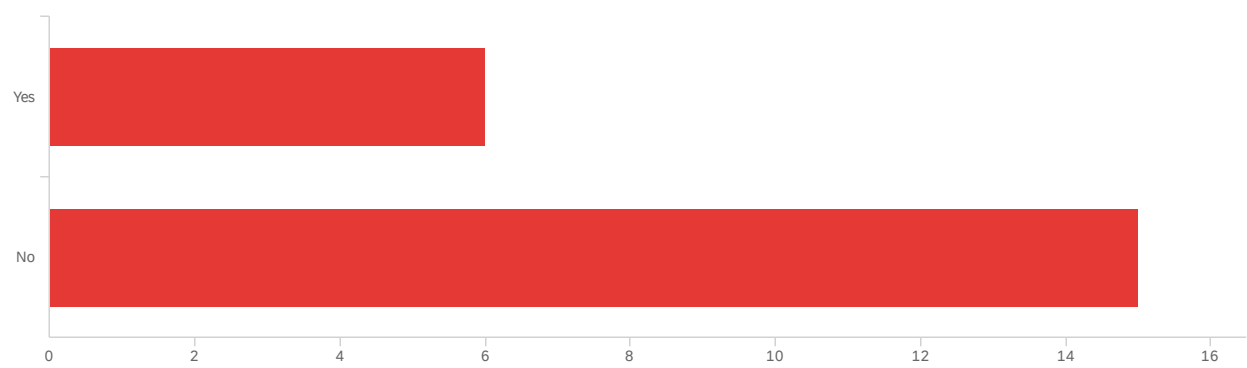
By implementing tests throughout the year and constantly improving.

Implementing the testing in the codebase.

My own research, as well as practice. My internship (and current employment) as an automation tester also helped, as did prior experience with Cucumber and Unit testing.

Writing more tests

Q27 - Would you like to receive a copy of the research results, or be notified if a focus group is planned to discuss the findings in more detail? Your contact information will not be linked to you responses.



#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	Would you like to receive a copy of the research results, or be notified if a focus group is planned to discuss the findings in more detail? Your contact information will not be linked to you responses.	1.00	2.00	1.71	0.45	0.20	21

#	Field	Choice Count
1	Yes	28.57% 6
2	No	71.43% 15
		21

Showing rows 1 - 3 of 3

End of Report