School of Computer Science & Engineering

COMP9242 Advanced Operating Systems (AOS)

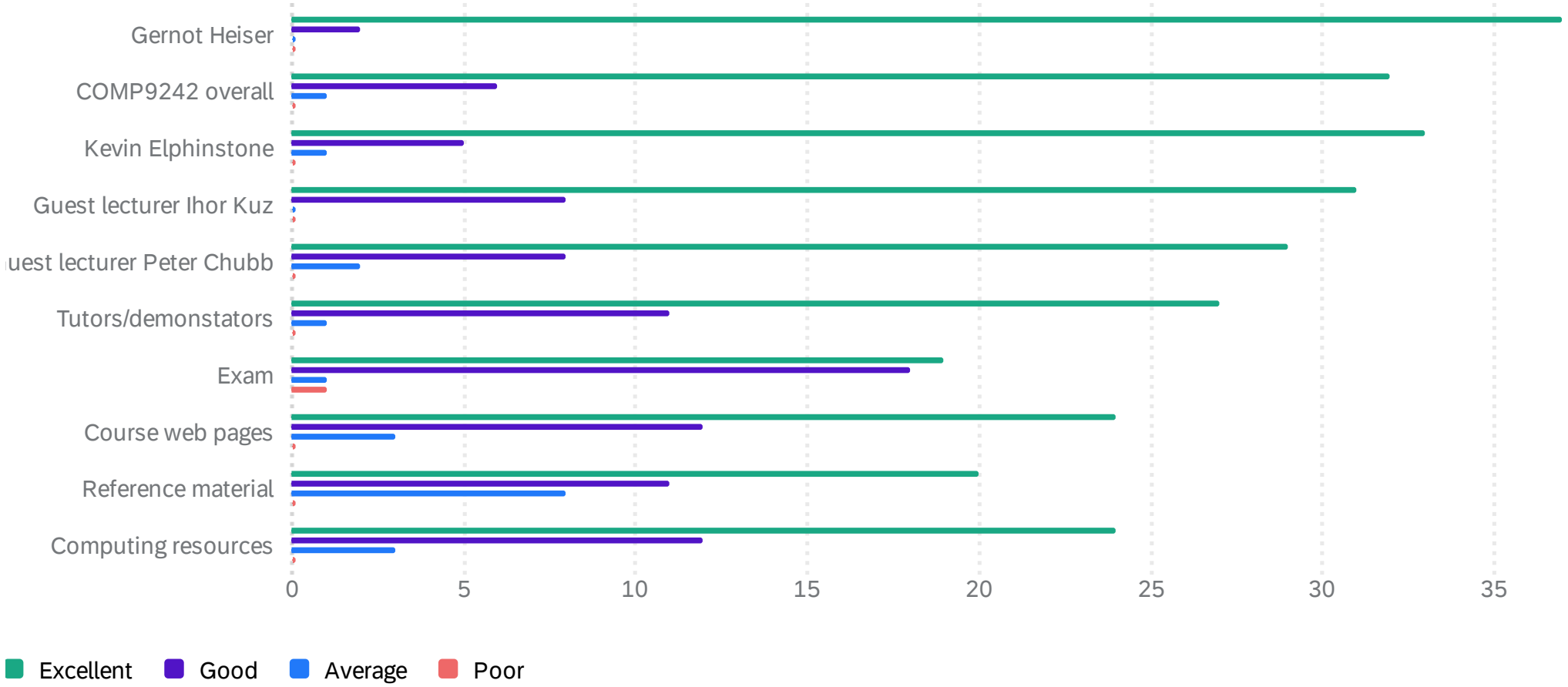
2024 T3

AOS Course Survey Result

@GernotHeiser

Copyright Notice

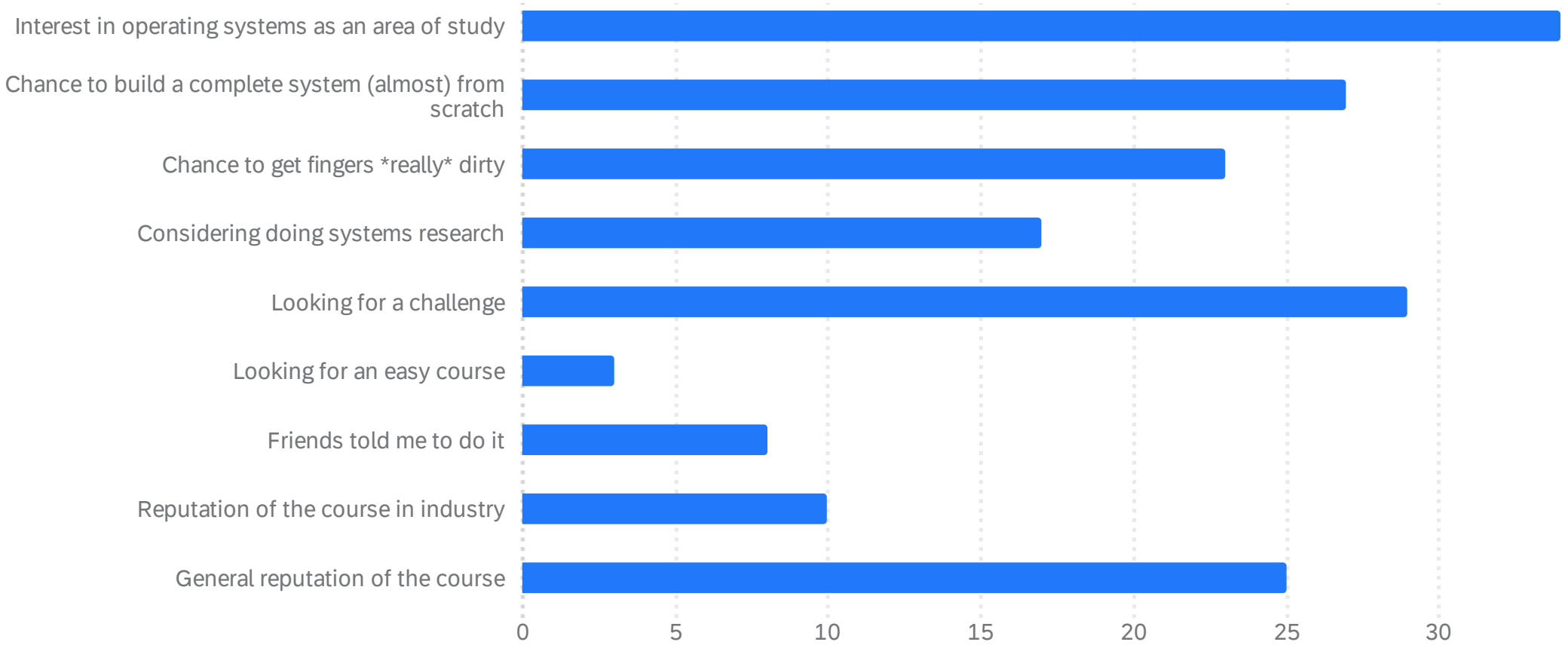
Q1: Quick evaluation



Q1: Quick evaluation

Quick evaluation single answer	Average	Minimum	Maximum	Count
Gernot Heiser	1.05	1.00	2.00	39
COMP9242 overall	1.21	1.00	3.00	39
Kevin Elphinstone	1.18	1.00	3.00	39
Guest lecturer Ihor Kuz	1.21	1.00	2.00	39
Guest lecturer Peter Chubb	1.31	1.00	3.00	39
Tutors/demonstators	1.33	1.00	3.00	39
Exam	1.59	1.00	4.00	39
Course web pages	1.46	1.00	3.00	39
Reference material	1.69	1.00	3.00	39
Computing resources	1.46	1.00	3.00	39

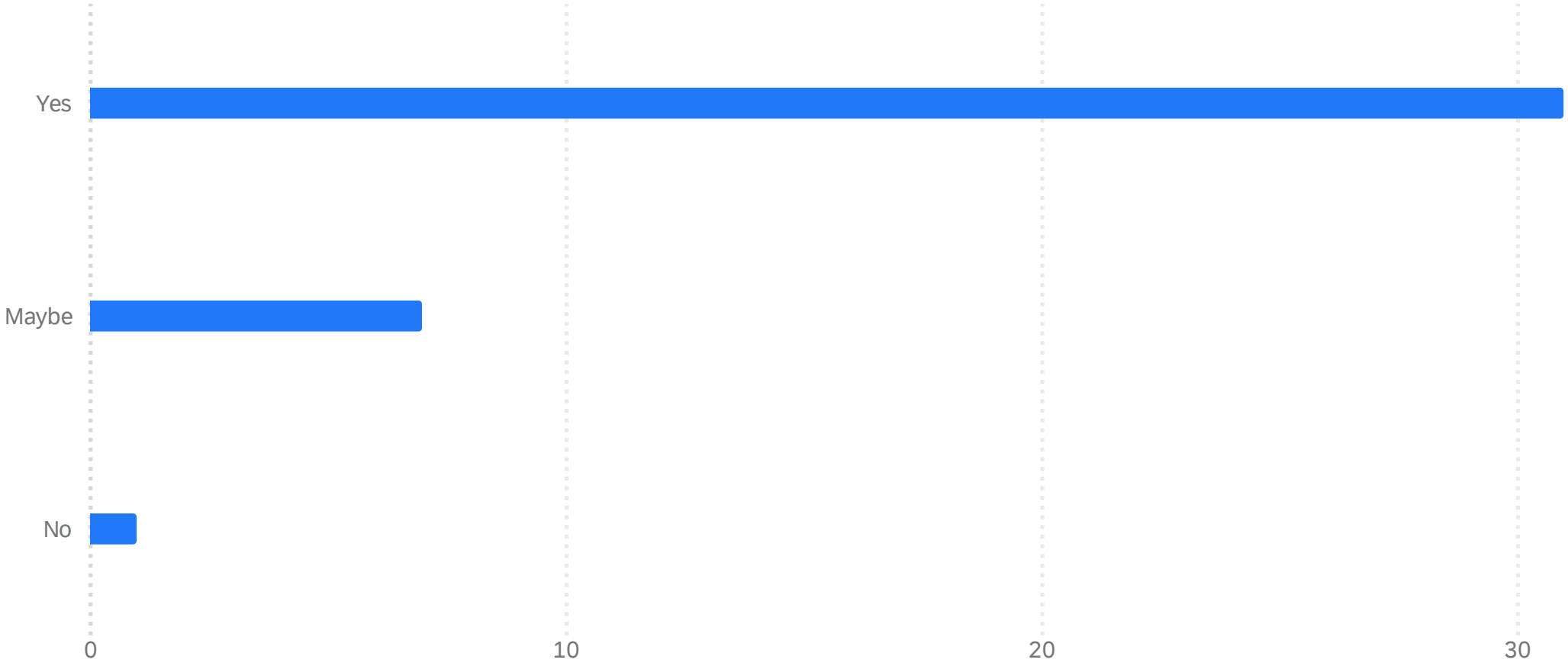
Q2: Your main reasons for taking AOS?



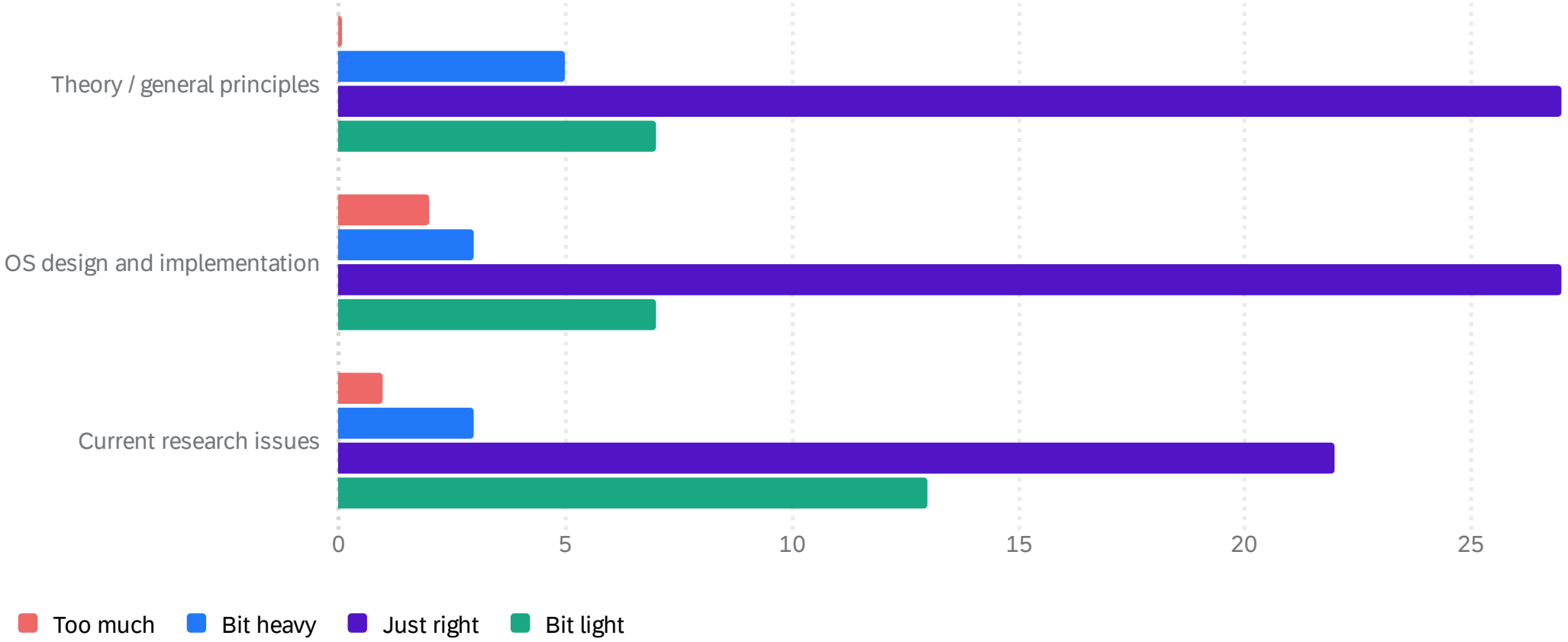
Q2: Your main reasons for taking AOS?

Other factors not mentioned above
Waleed told me to do it
Made the decision on a whim when I dropped one of the two courses I was taking in T2 meaning I needed to pick something in T3
Friend willing to do it with me
Pressure from Gernot :) Definitely glad I didn't bail out, it was fun :)

Q3: Would you recommend this course?



Q4: Content Balance



Q5: The best things about AOS? (1/5)

What were the best things about this course?
Everyone involved with the course has been a pleasure. Very interesting content. Fun project. Actually getting a challenge in a COMP course.
The project, despite being a major time sink, sitting down with your partner and brainstorming and/or coding solutions to complex problems is very rewarding.
Really enjoyed the multicore lectures, was very interesting. Also enjoyed the VM lecs.
All the implementation details!
M0 is good for diving in with an easy to implement task, when getting used to the sel4 system. The project is very enjoyable and follows through a good structure. The lectures were consistently good.
Was fun and exciting to have the freedom of designing and writing large subsystems from scratch. Having daily unallocated labs was good and supported a fast iterative design and implementation cycle; demonstrators were generally very good (shout-out to Julia!); lectures were interesting, providing some appreciated context on operating system design principles and some cool lore
Coverage of the state-of-the-art microkernel design.

Q5: The best things about AOS? (2/5)

What were the best things about this course?
The open-ended nature of the project. It was fun to learn about up how OS features were implemented in existing kernels and then adapt them to SOS.
The project for sure.
I liked that the milestones allowed you to take a stab at implementing one piece of an OS at a time. The milestones for the proceeding weeks were excellent as a reflective learning experience as you often would find out whether you made the right trade-off.
Lectures were fast paced and genuinely interesting and any theory that was directly relevant to the project felt like it appeared at an appropriate time. The project as a whole was great insight into challenges with OS development and large scale, low level software projects in general.
The project (obviously), but also the exam (surprisingly). The skills I gained pepping for the exam were not course specific, and the knowledge I learned in the practices papers I read was interesting and relevant.
One whole project towards the end, building up learning knowledge.
Feeling rewarded every time we got a new thing working. It's a hard project but it makes it feel so much better to complete.

Q5: The best things about AOS? (3/5)

What were the best things about this course?
the depth and granularity of knowleadge taught, the quality of the lectures and the hands on assignment
Implementing my own operating system was very enjoyable
The project and the design decisions that came with it
the final solution we got to build
- Self-led project and freedom to design - No weekly quizzes/unrelated weekly submissions
Stuff about caching, performance, SMP and multicore
The project and the discussions during the lab!
Literally everything
Working with seL4.
Project and the marking system. The relaxed late penalty and multiple retrys made it easy to focus on developing the project and learning while doing it
Being able to create our own designs was really fun and rewarding to do.

Q5: The best things about AOS? (4/5)

What were the best things about this course?
learning some seL4 history
Building an OS
The content was amazing! I loved learning about Operating Systems, computer architecture, and kernels. Gernot and Kevin were awesome and the project was very fun to do :)
The community within the course. Talking to course staff and other students about systems problems and concepts.
Bit of a double-edged sword, but the amount of freedom given to students in this course is (in my experience) unparalleled.
I really liked the operating systems theory that was covered. My favourite lecture for reference was the security lecture, I felt like it maintained a really good balance of theory and practical content.
Mostly that it was about systems, i love systems
Very interesting and engaging project, and highly enjoyed getting to design and build up the system over the term. Staff were also clearly passionate and very helpful.
Very fun project and really interesting content.

Q5: The best things about AOS? (5/5)

What were the best things about this course?
The project
Lectures, I like the content, it is fun and interesting.
Even the exam, although stressful it's really cool that we can understand and criticize current research in OS.
The lectures were informative and interesting and covered a variety of topics useful in systems and elsewhere.
Getting to work on a non-trivial project and making consequential design decisions
Building an OS is difficult but having a microkernel as well as a SOS already set-up makes it much easier to get into.
Having a lot of tutors available each week to discuss.

Q6: The worst things about AOS? (1/8)

What were the worst things about this course?
Wish there were more than 10 weeks in a trimester. I feel like you can do reasonably well in the course without ever going to lectures. It would be nice to find a way to make those assessable.
The course timetable/website doesn't mention when open labs are scheduled until quite late, this forced myself and my partner to be forced to have things ready earlier in the week because we had already been allocated labs/titles to teach later in the week. So if things weren't ready we'd either have to wait and lose marks or transfer our hours.
Milestone 4 (the filesystem milestone) was a little dry, largely it felt like we were implementing wrapper functions, the masochism stuff related to it felt like it would be rather interesting though, sadly we did not get time for it
Too little time…
The documentation of the odroids is very poor. The debugging is not ideal, and not convenient to use. The thread/event lecture might be nice to be given earlier as my team already decided and started working on the relevant milestone before the lecture was delivered.

Q6: The worst things about AOS? (2/8)

What were the worst things about this course?
Completely ruined my sleep schedule
Studying for the exam, going through the past papers was often mind-numbing
Divergence between the lecture, project and final exam.
I think we could have had a bit more preparation of reviewing papers before the final exam.
While the milestones were effective as a learning experience, by providing feedback week to week as to whether your design choices would hold up, they were also the main struggles in the latter weeks. I found the nfs file system implementation particularly challenging and washed over into the latter milestones. The success of a student in this course hinges particularly strongly on whether the choices made in the earlier weeks are, ultimately, appropriate for process management. I wish I had identified this earlier on so that I could have included this in my decision-making process for the design.
Understandable since large C project using an unfamiliar microkernel, but the assignment did sometimes veer away from design and problem solving and more into debugging hell. Expected, but still annoying.
Parter dropped out during the middle of the term, making the project harder for a single member.

Q6: The worst things about AOS? (3/8)

What were the worst things about this course?
Limitations imposed by trimesters - wish I had had more time to spend on this course instead of cramming it into 10 weeks.
lectures and assignment content deviated after the first fiew weeks, assignment felt less theory. not much accademic reading done in lectures ether
Using some features of sel4 without properly understanding them lead to headaches in the future. It would be awesome if demo 0 was more indepth in covering the sel4 features we will use
The lectures on seL4 did very little to prepare me for its use in my own operating system. Also would've liked to learn about the parts of the operating system that were done for us.
A 24 hour exam during the first week of my internship
how hard it is but that is i guess what makes it a good course
- Extreme project workload across 10 weeks - No option to solo the project, even at own risk - Less freedom around demos because of tutor availability (maybe an automated system could alleviate this so students would have more freedom to demo whenever)
Requires a lot of work

Q6: The worst things about AOS? (4/8)

What were the worst things about this course?
Debugging
Some specs were a bit vague
The workload.
Milestone marking seemed inconsistent between tutors. Some had extra expectations beyond the marking criteria (which is good from a learning standpoint) but a bit fustrating to argue past when trying to mark off a milestone.
Not enough time :(
too much workload to do with other courses, eat up a lot of time
The labs. Really all they were was for demonstrating milestones. Would have really liked more time allocated, as well as more tutors as most of the lab was sitting waiting to get marked. Would have been great to have more lab time and tutors, to discuss ideas with and really talk about OS design concepts, trade offs, considerations etc, instead of solely marking.
I think the project specs could be done better, especially the bonus sections. A few things were pretty unclear and the only way to realise what the spec means or expects you to assume is to do lots of asking.

Q6: The worst things about AOS? (5/8)

What were the worst things about this course?
Nothing really, other than maybe just the lack of time to be able to accomplish everything I wanted to do from the courses. But there's nothing really one can do other than trimester->semester.
The exam was incredibly time consuming which is pretty demoralising after having spent an entire term working strenuously on the project. However, I do understand why that is since given the nature and volume of the content, a traditional exam would be difficult.
This is not like the worst thing about the course, but I would like to see more current research being covered. The stuff that was covered, that is LionsOS and Pancake were pretty cursory. I really wanted some time protection stuff!!! Was sad it was not covered.
Probably milestone marking, I kept getting stressed about showstoppers and how hard it is to get marked when its busy. I wish all the showstoppers were on the website instead of just in person during marking, it doesnt have to catch people out because its just a milestone checkup, tell us what you want!
I think the uncertainty and lack of communication around demos post-w10 was stressful, but I should have trusted that it would eventually get sorted out.
Hard to catch up once your behind, lots of late penalties

Q6: The worst things about AOS? (6/8)

What were the worst things about this course?
The project is difficult and challenging in a good way. However, the reputation of the course as very hard isn't from the work being extremely difficult rather the quantity being unreasonable. Each weeks project is equivalent to a bi-termly assignment from other comp elective. Due dates on Friday are particularly painful as in practice it doesn't give you the weekend to work through things, and demos are not available until after the weekend. If showstopped, in practice it would give you only a few days during the week until you can complete the next milestone without penalty. The gruelling schedule and quantity of work makes the work which is very challenging and interesting instead painful. The exam format is also extremely painful. A 24 hour exam encourages students to get very little sleep if they don't want to perform comparatively worse to their peers who are willing to sacrifice their sleep. Given there are two papers anyway, two 8 hour exams would be much less painful since you are forced to leave some time for resting. For example 2pm to 10pm and 10am to 6pm would be less harmful to students health. Lastly, the title of 'Advanced Operating Systems' seems misleading, as the course really revolves around seL4 rather than advanced operating systems techniques in general. In particular an exam question involved critiquing a new paper which claimed to make major improvements over seL4, which seems to introduce a bias towards responses critical of the paper given the lecturers' affiliation with seL4. The course would be much better as a dive into operating systems research in general if there was greater focus on techniques other than those used by seL4 (and on seL4 itself), as it felt the course was really just a primer into operating systems research on seL4 in particular rather than anything else. This isn't particularly a huge issue, but I feel like this should be better advertised in the course name/outline as it's not really apparent until after you start the course.

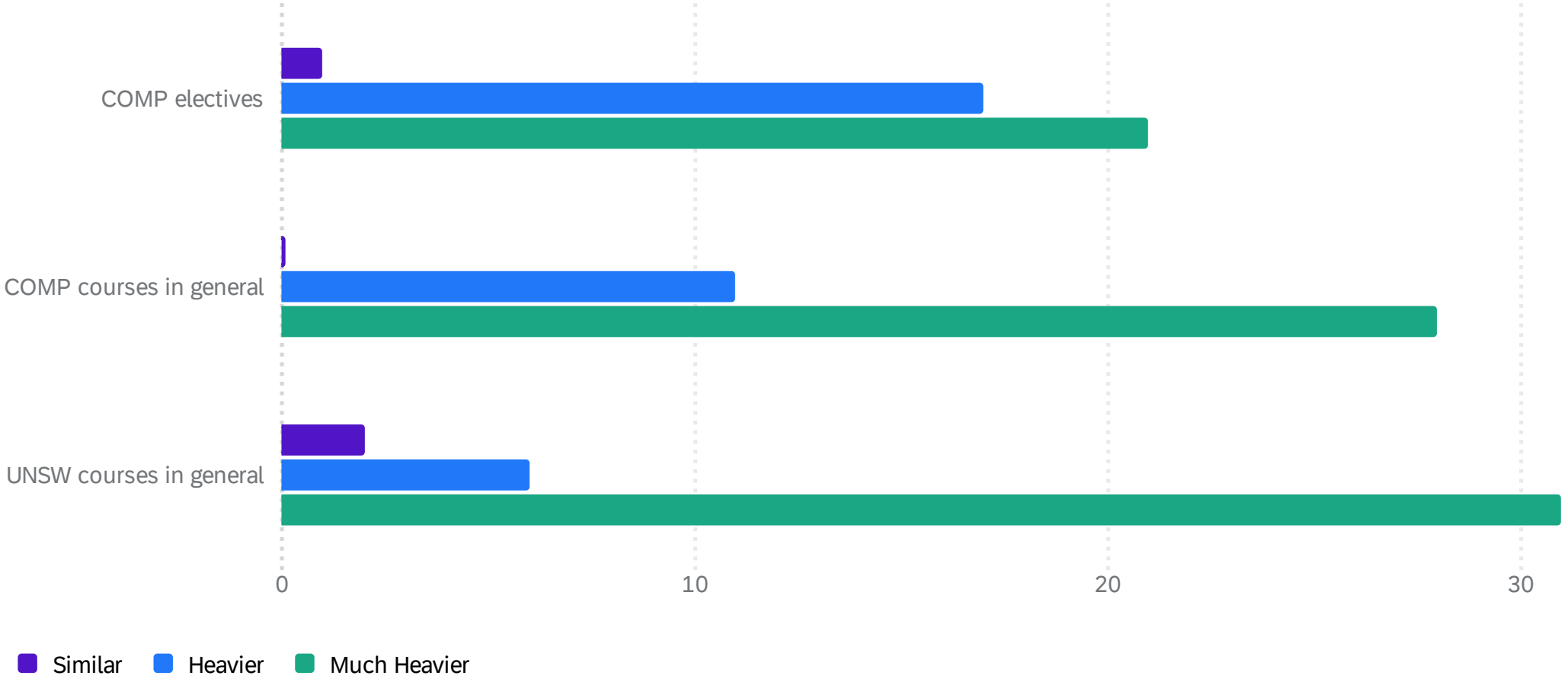
Q6: The worst things about AOS? (7/8)

What were the worst things about this course?
Workload/time management vs rest of life - it was very difficult to dedicate as much time to the course as I wanted while trying to maintain full time study requirements and working (even at reduced hours). If I could've taken the course on its own while staying a full time student, I would have. Felt less engaged with lectures on current research issues, and would've appreciated more lecture content that related to (or could be compared to) the main project, or otherwise going deeper on the theory and design topics
Sometimes I wished the labs / consultations were just slightly longer, especially during the start of the term. I think there were too many students at the start so it was quite difficult to get help, discuss the project and get questions answered, since tutors were busy marking milestones. Technical difficulties for the lectures
It's a bit frustrating to be struggling with an assignment and then attend lectures that don't really help. It can feel like two separate courses
Unfortunately my partner was much more experienced and skilled than me whilst having subpar communication which meant I took less out of this course.

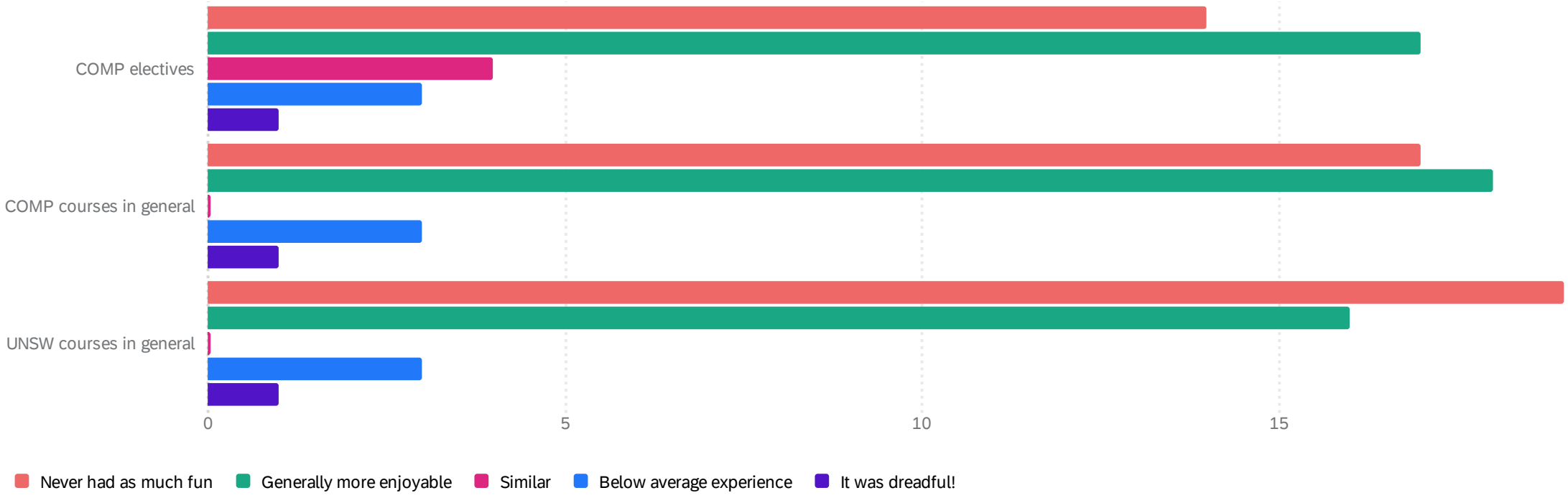
Q6: The worst things about AOS? (8/8)

What were the worst things about this course?
Worse sounds bad, but I've always been looking for someone who would analyse how an ideal system would be designed ie. we pretend we get to design the hardware how we want, what abstractions would we give ourselves to provide a good operating systems? I'm always annoyed that we play this game of cat and mouse with the hardware people, each second guessing what the other does. I'm sure there are reasons why it's beneficial in practice to have this divide, but from a pure technical point of view, it's just plain bad treating systems we rely on mostly as whiteboxes we don't (barely) get to touch.

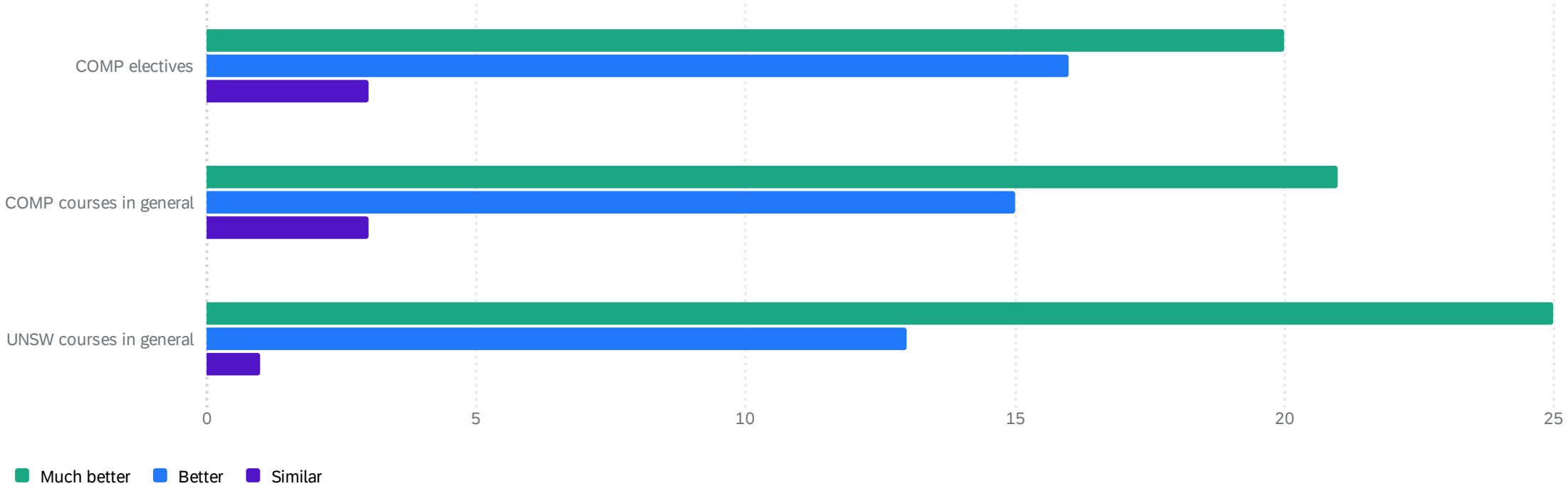
Q7: How does the workload compare?



Q8: How does your experience compare?



Q9: How does quality/value compare?



Q10: Required background (1/4)

What background knowledge do you think you were missing that would have helped you in this course? Is a distinction grade in COMP3231/9201 a suitable preparation? Is it too harsh?
It's perfect, I don't think any other comp course would help
Honestly I think the standard OS course was sufficient. It would obviously be nice if one comes in with some experience with asynchronous programming but the lecture on OS execution models does a great job introducing that when covering continuations
a think a distinction isn't high enough of a bar potentially
I don't think I was missing any background knowledge per se, although having done more work on computer architecture or concurrency would have been nice. That being said, those topics receive good coverage in this course, so I wouldn't consider making taking those courses a prerequisite. COMP3231 DN is an OK proxy for preparation although students largely self-select for this course. Maybe to gain better insights on this point you should exit interview the students who dropped out before census date
Missing some hardware prerequisite as a lot of the hardware level information such as chip layout was extremely difficult to follow due to lack of a starting point
The project is challenging for me due to lack of time. I think a DN grade in base OS is suitable for the lecture and final exam, but not sufficient for the project.

Q10: Required background (2/4)

What background knowledge do you think you were missing that would have helped you in this course? Is a distinction grade in COMP3231/9201 a suitable preparation? Is it too harsh?
I think perhaps too harsh, I got just under distinction for the first os course but HD'd this course and found it a lot more enjoyable. At the same time I recognise that this course is quite a lot of work and definitely challenging so I understand why the distinction grade is there.
I think the distinction grade is suitable and would recommend it remain. I potentially would even push it to 80. However, I do understand there is a trade-off with wanting more students and more quality students. I found that most of my difficulties were in understanding the inner workings of seL4. For example, the abstraction used for the hardware TLB and virtual memory mappings. This is likely an unsolvable problem, or one with me, but it was often difficult to validate whether a seL4 API was used correctly.
It's probably fair, though some people without the added context that EOS provided might of had a harder time.
honestly maybe like a 2hour recap video from 3231 would be useful before taking AOS, a problem i had is knowing how to apply some 3231 concepts so maybe a bit more direction could be good
It's perfect
I think a distinction grade in COMP3231 is a suitable preparation.

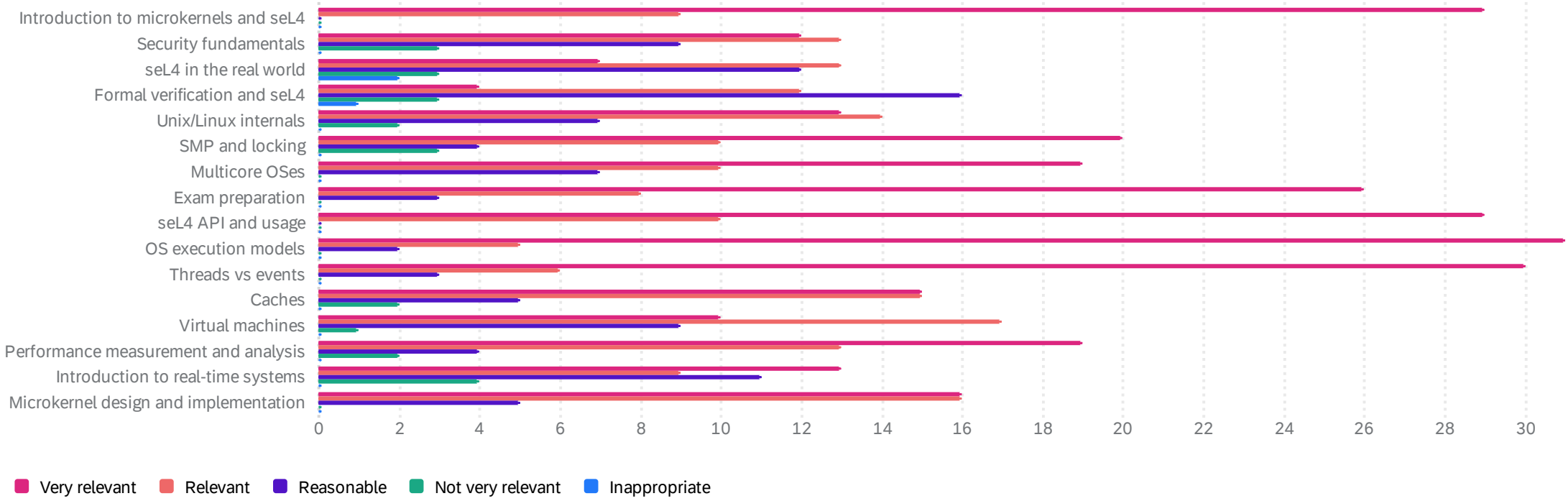
Q10: Required background (3/4)

What background knowledge do you think you were missing that would have helped you in this course? Is a distinction grade in COMP3231/9201 a suitable preparation? Is it too harsh?
Distinction seems appropiate, I wouldn't lower it. Personally I felt like I was struggling a bit with implementing in C, particularly with the file structure / build process. 1521/2521 don't go into detail with larger build process. Eventually I did learn by googling around. Perhaps linking some online resourses could help?
I believe that it is entirely possible for non-distinction students to complete AoS. Alternatively, I think it is also completely reasonable to have distinction as a pre-req as it really sets the tone for the expected difficulty, despite what others may or may not have heard about this course. Also, knowing that distinction in OS is a pre-req is also reassuring in terms of the people you discuss the project with, and the teammate you will have to find, as it is a very involved project.
Honestly fine as is. The only possible thing to benefit would maybe be a maturity requirement (similar to comp4161) but at the same time I recognise a lot of 2nd year students do the course and do very well on it
I think OS distinction is enough. I don't think I had *missing* background knowledge, but did find that non-OS background knowledge (eg. computer architecture, security) helped with certain topics.

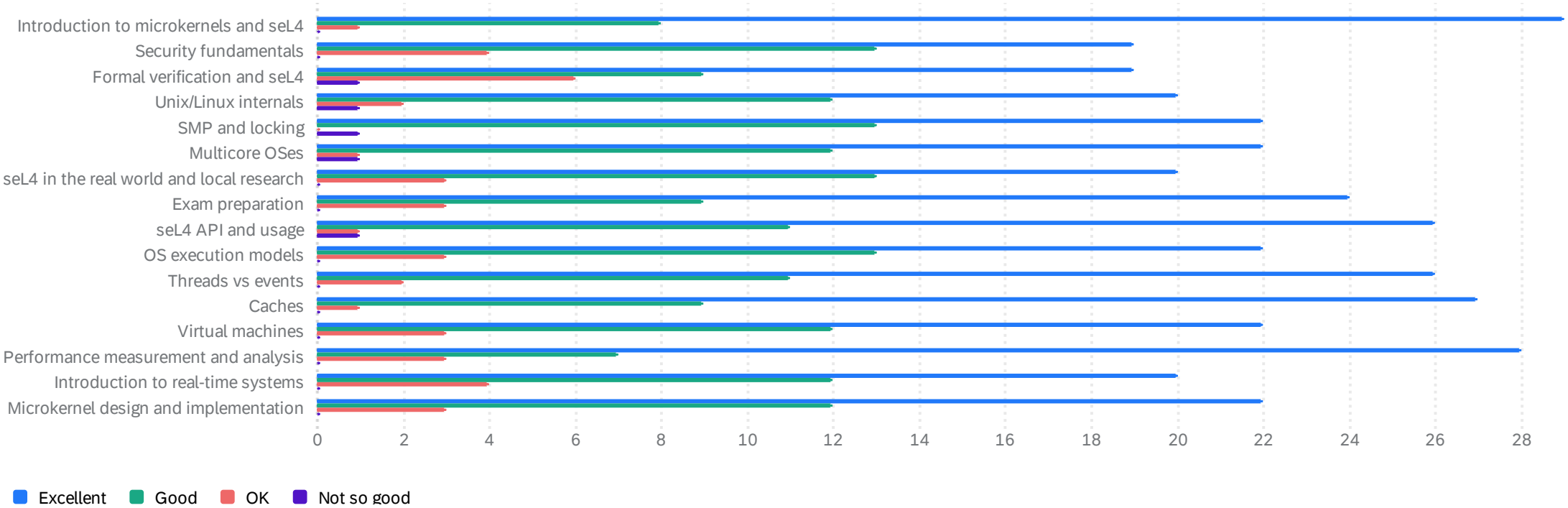
Q10: Required background (4/4)

What background knowledge do you think you were missing that would have helped you in this course? Is a distinction grade in COMP3231/9201 a suitable preparation? Is it too harsh?
COMP3231 + general interest + the content that was recapped was definitely enough
Yes I believe distinction grade is suitable, my only background in OS comes from COMP3231 and it is enough to complete AOS, without limitation.
It seems reasonable. A guide detailing which components are relevant to be revised before starting the course could be helpful.
I think having OS and AOS in consecutive terms is a good improvement.

Q11: Topics relevance/appropriateness



Q12: Quality of the lectures



Q13: Most useful material (1/5)

Which material do you think will be most useful to you in the future?
Threads vs events and Execution models for just general design of applications. Starting at a low-latency team in trading so, performance analysis, SMP, caches
Unix/Linux internals, multicore OSes, SMP and Locking and maybe Virtual machine and caches
Multicore and SMP
just the general design and problem solving skills i gained
Performance measurement.
Performance measurement and analysis was a killer lecture, definitely will be applying things learned there whenever I get the chance. Unix/Linux internals decently useful to know whenever things go wrong ("why is the system locking up again?"). Threads vs events + OS execution models (closely coupled) were also great and likely will be broadly useful in designing and working with software where performance matters.
OS execution models
Performance measurement and analysis, microkernel design and implementation, formal verification and multicore.
I think the more general lectures such as VMs, security fundamentals, etc.

Q13: Most useful material (2/5)

Which material do you think will be most useful to you in the future?
While having skipped a lot of content (sorry!) due to time constraints, I found the threads vs events and virtual machines lectures very interesting. It was nice to hear about the history behind these concepts, how they've evolved, and their strengths and weaknesses from Gernot and Kevin. These two topics are particularly useful as they shed light on the systems we certainly will operate within, or even build, as Software engineers. I couldn't imagine a better way to dive into these topics.
The performance measurement and analysis / benchmarking content and security since those seem to be the most relevent across more than just OS development. Caching as well for understanding potential ways to improve software performance on different hardware.
threads and multicore
Nothing in particular - I find I benefit from having deeper understandings of anything and I'm not looking to go into systems as a career path, but at the same time, I don't think my time was wasted on any of the covered content.
most if not all lectures afer and including Virtual machines
More intro lab work on sel4
Performance measurement and analysis as it is fairly common practice in engineering trading systems

Q13: Most useful material (3/5)

Which material do you think will be most useful to you in the future?
Caches and SMP
idk
Performance measurement and analysis + SMP, locking and multicore stuff.
Caches, execution models, performance
Benchmarking crimes!
Caches
Most of it; the Unix/Linux internals one seems the least likely.
Caches, OS Execution Models, benchmarking advice
I think the performance measurement and analysis was a really interesting lecture, and would apply to many topics outside of AOS too.
VMs
Experience in designing and building a system
I think material regarding caches, threads vs events, virtual machines and unix/linux internals will be most useful :)

Q13: Most useful material (4/5)

Which material do you think will be most useful to you in the future?
Probably nothing just because software/systems engineering is not the pathway I am pursuing in the future.
If needed in the future, it is reassuring I can always come back to my project and/or the lecture notes. Particularly: Multicore OS's, Locking, Threads, Performance analysis. These topics will ALWAYS be important.
Linux Internals, Security Fundamentals, Caching, Microkernel design, Performance analysis!!!!, OS-execution models
Hard to say, im only a student. I like concurrency and hardware issues though
General theory and concepts (eg. caches, smp, multicore) - topics that are applicable to helping me understand things I encounter in the wild, not just seL4
Almost everything, especially memory/cache concerns and mixed criticality concerns.
All the material on threads. Performance measurement and analysis lecture. Virtual machines
Caches, Virtual machines, Introduction to real-time systems.

Q13: Most useful material (5/5)

Which material do you think will be most useful to you in the future?
Maybe understanding caches, performance analysis, and virtualisation seem like widely applicable skills.
Performance measurement and analysis
- Knowing the keyword "real time system" to learn more, I didn't know that was a whole field before. - Knowing about formal methods - Performance lecture - thread vs. events (very nice to have two decks arguing for the other)

Q14: Missing material (1/2)

Which material, not presently in the course, would you have liked to be covered? When commenting here, please also comment below, as we cannot add stuff without removing others.
Not sure what completely new content could be added but the linux internals part was quite brief.
Would be really really cool if there would have been a lecture covering more modern concerns in computer architecutre and how they affect OS design (in a similar vein to how complications of out of order execution affects how one thinks about concurrent programming)
Peripheral devices
current (non-microkernel) operating system research
I would have liked more context on the abstractions between the sel4 APIs and the arm architecture. For example, I struggled to decipher how the seL4 APIs managed the arm page tables. Often I would be trying to decipher functionality via trial and error - rarely would I finish up my testing knowing exactly what I did with 100% confidence. Perhaps the more actionable
unikernels
Microkit ;-)
More design and discussion around OS design and alternatives.
Time protection, more security content

Q14: Missing material (2/2)

Which material, not presently in the course, would you have liked to be covered? When commenting here, please also comment below, as we cannot add stuff without removing others.
Would adore a dive into current monolithic OSes, like the structure of windows and macOS, perhaps mobile devices or even embedded OSes and how they differ to desktops. Would certainly attend these lectures next year even as a AOS alumni
Cover a variety of the latest research in operating systems.
Distributed systems
SMP and concurrency in the hardware could be of higher quality I think (but it was already covered quite a bit, which is good). It's a very hard topic to learn about by just googling, so to me it's a very important part of the course, in particular - weak memory (ref. Viktor Vafeiadis's work from MPI) - Mothy's work I realise this is very hard + a lot of work, but the question didn't require that the suggestion be easy to implement.

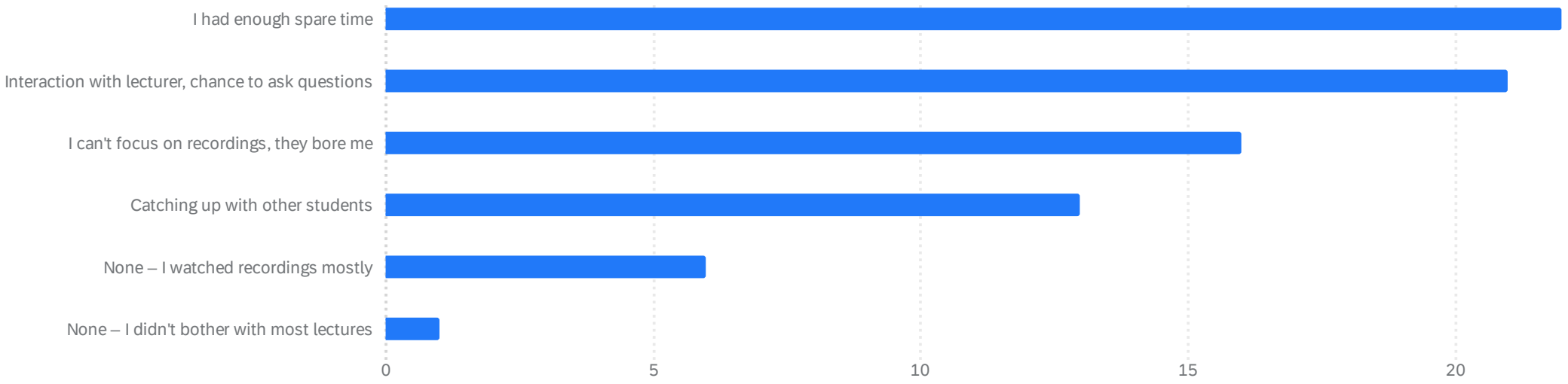
Q15: Material to scale back/exclude (1/2)

Which of the current material would you like to see scaled back or excluded?
Probably the verification portion, I honestly can’t remember a single concrete thing from those lectures either from it going over my head or disinterest
Formal verification
security fundamentals
as much as I loved the multicore content, if anything had to be cut probably this. a lot of the SMP material focussed on historical primaries, and this material could probably be scaled back somewhat as well
seL4 verification and seL4 in the real world could be bonus lectures or recordings
(See content allocation and "worst things" responses)
Less focus on seL4 beyond in the early weeks for the project.
Some of the dicussion about the performance trade offs in SMP locking weren't very interesting to me (though the key observation that locking too often for too little is worse is definitely worse emphasising

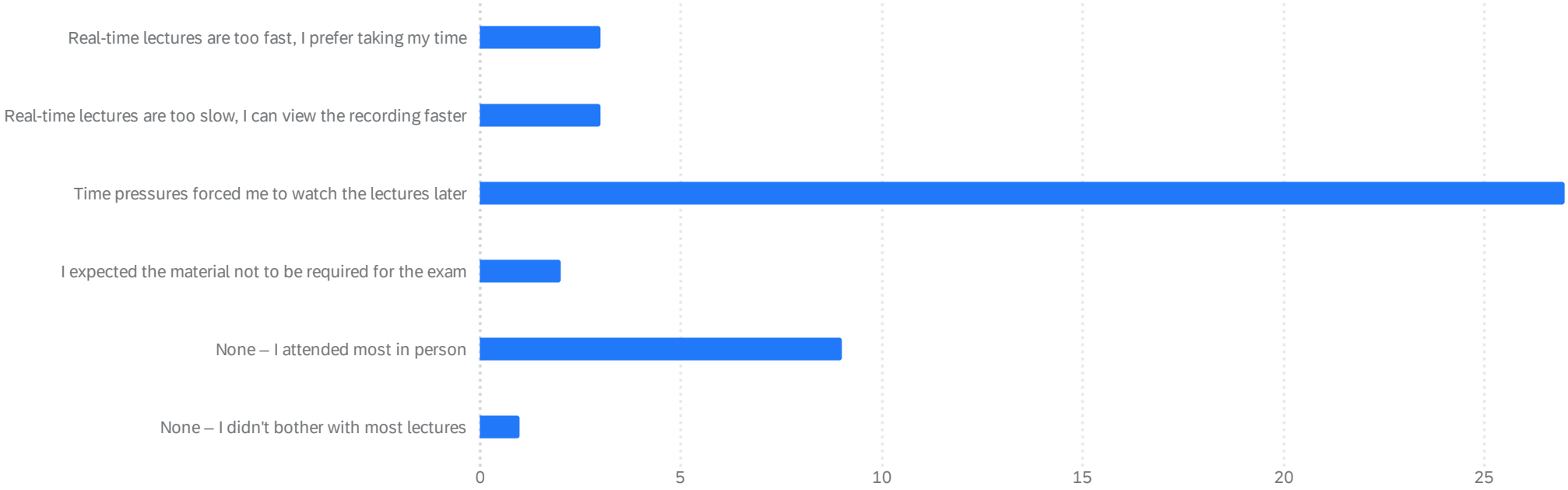
Q15: Material to scale back/exclude (2/2)

Which of the current material would you like to see scaled back or excluded?
I think the presentation of the seL4 abstraction can be done quicker and a bit better. Even knowing already how the abstractions worked, I had a hard time following that lecture. The key to me is to give _just_ the intuition, not the details: threads, CSPace, VSpace, notification words, endpoints. For the last two, I think it would be really helpful to have the diagram of the state machines. (I'm of course biased, and implementing this advice could very well make the course worse I don't know).

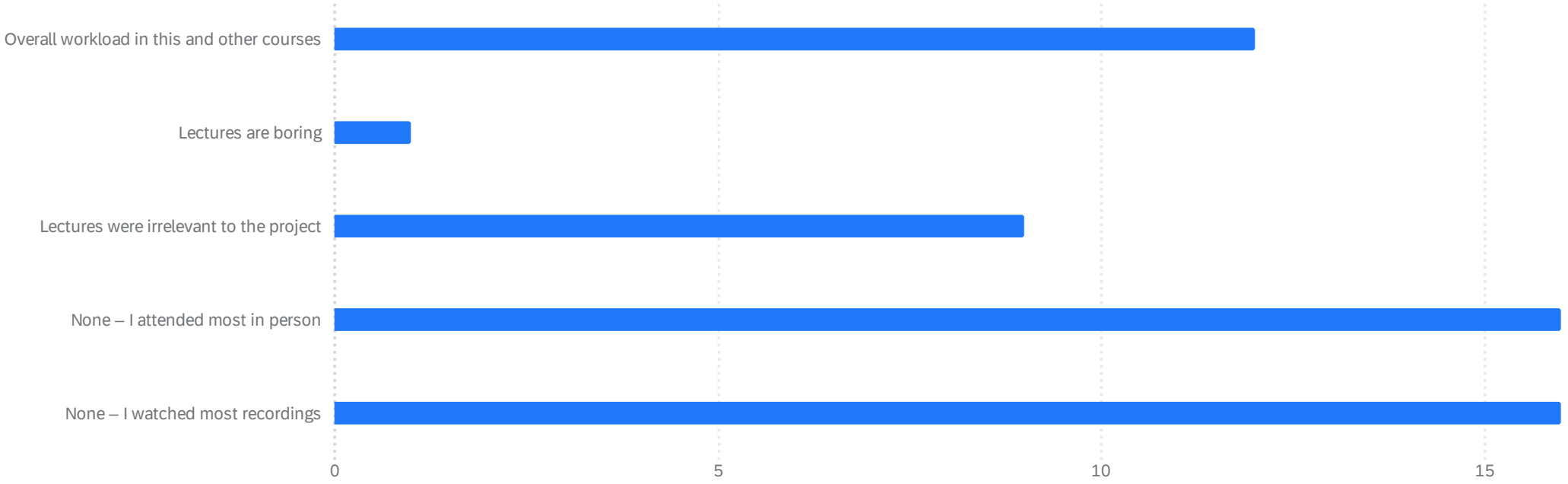
Q16: Why attend lectures in person?



Q17: Factors for watching recordings?



Q18: Reasons not to watching lectures



Q19: Other comments on lectures? (1/3)

Any other comments on the lectures, especially suggestions for improving them?
Timing mattered a lot. I attended almost every Tuesday lecture as it was straight after the consultation. Thursday was a 50/50 due to project being due Friday night and considerable time between consult and lecture, so didn't feel like staying around at uni.
While not covered in the options, the real reason I did not attend lectures was because of my commute time to uni. It takes me roughly 1h 45 mins one way to commute to uni, so it's rather hard to justify 3.5 hrs of PT time for a 2hr lecture, especially if I have nothing else on that day.
Most lectures are indeed irrelevant to the project.
I found there were definitely times I was struggling to follow along with what was being said in lectures. I took COMP3231 2 years ago, and I remembered a lot of the logic behind the concepts quite well but had forgotten a LOT of the terminology, and I think I felt too shy/embarrassed to stick my hand up and ask some questions because I couldn't remember what some things were called. That being said, I was able to (mostly) solve that problem for myself by brushing up on my notes and wasn't really struggling with that specific problem by week 3. I had my old OS notes in my notebook to lean on - maybe it'd help to just have links on the course website back to some OS slides, as a quick refresher to the students who need it?

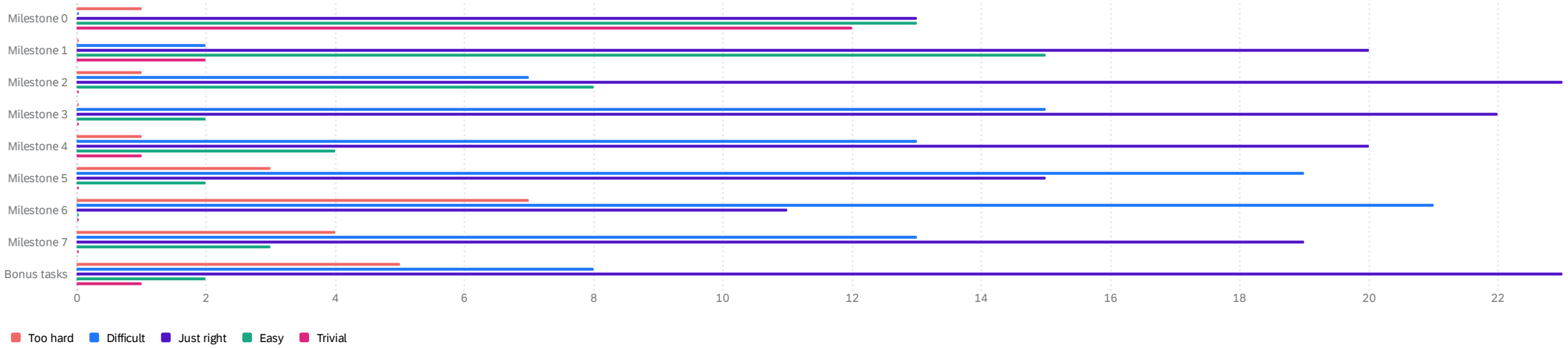
Q19: Other comments on lectures? (2/3)

Any other comments on the lectures, especially suggestions for improving them?
Would have loved to watch them live, just had a conflicts on my schedule :(
Make sure the recordings capture the whole lecture and do not have quality issues.
The main point of improvement I can think of, is making sure that the recordings are always up properly. Thankfully I wasn't affected since all the lectures during which there were issues I attended in person, but I imagine quite a lot of people were.
I know this is centrally controlled but the lectures on Thursdays were a bit of a nightmare, since we had the milestone due the next day. Falling behind in the in-person lectures while I also had the project to work on meant that I would procrastinate watching on lectures and fall further behind. Again not much that you can do on this point.
Technical issues can be devastating for recording watchers, which is why i went in person.
Really appreciated full slides being made available and early, allowed me to catch up whenever my focus slipped during in-person attendance.

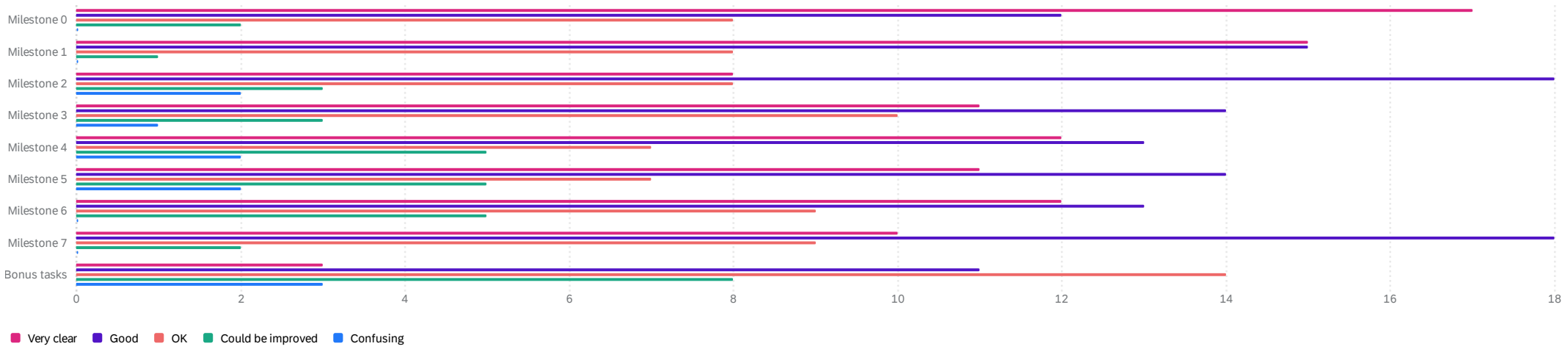
Q19: Other comments on lectures? (3/3)

Any other comments on the lectures, especially suggestions for improving them?
Lecture recording quality was poor throughout the term, particularly at the beginning. This made it challenging to watch if not attending in person. A zoom recording or other setup besides echo360 could be considered if issues persist (although it may have been due to the specific lecture room used).
I prefer Kevin's style of annotating on slides for better visualisation :)
It would be nice to have questions/high level exercises about the lectures. It'd make it a bit easier to engage. There are some questions in the lectures, but having a PDF of questions I can go through for each lecture to check that I took away the key points would be good motivation (I'd go through it before the lecture and go "oh damn I don't know how to answer that" let's attend). But honestly, I didn't attend all the lectures not because the course was poor/could be better but because I wasn't organised well enough.

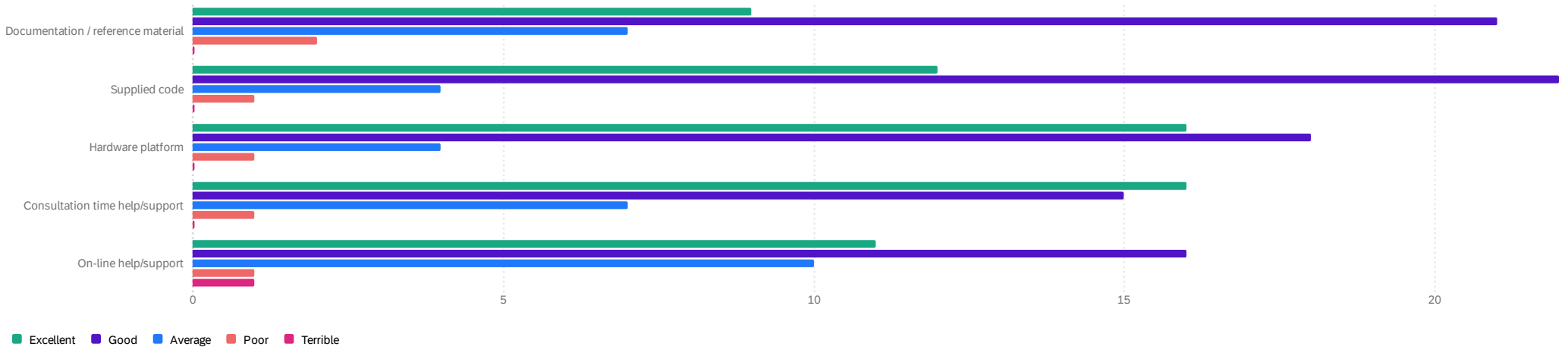
Q20: Difficulty of various project parts?



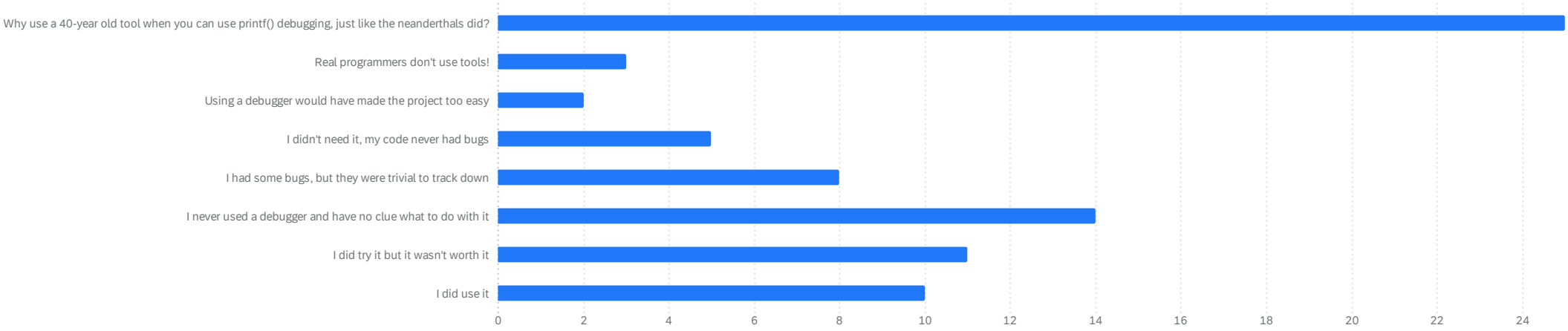
Q21? How well was the project specified?



Q22: What was the quality of...



Q23: Why didn’t you use GDB?



Q24: Comments about GDB (1/4)

Any comments about GDB?
Setup locally ended up not working so I reverted to printf and obj dump and it was fine; event driven design allowed for this and I don’t think it would be as good for multithreaded designs.
I was programming locally on my machine and GDB required my source code to be available on my CSE machine, this made for a rather annoying workflow as whenever I needed to debug I would have to push my code to gitlab, pull it on CSE and then run the debugger.
I used it a bit but often used print debugging because I have become dependent on my IDE to give me an interface for debugging, and I didn't have the time to look into setting up CLion to work with the remote GDB; though allegedly it does support remote GDB, but we are working with a special case.
It was useful to get another example of how to set up a seL4 thread.
The integration and use of gdb was a little finicky.
Set up was a bit annoying but I did get it working. It did help me save on some time initially when I wasn't familiar with the codebase.

Q24: Comments about GDB (2/4)

Any comments about GDB?
I had used GDB a little bit before but wasn't that great with it beyond basic stuff and when things got trickier I found myself falling back on what I was familiar with, which in hindsight may not have been a smart move. Also, a lot of bugs were caused by not fully understanding how to use certain seL4 calls / behaviour, so you already knew exactly where and what the problem was but not why it was happening, which GDB didn't seem to help that much with. Once again, not great with it and other's mileage may vary.
My strategy with debugging was to start with prints, and then if it became clear after 30 minutes that debugging with prints was not going to cut it, I switched to using GDB - which I did a number of times. I think I was a) intimidated by what I had heard about it bricking the Odroids if you connected too quickly (once I used GDB for the first time on the project this fear went away but it's why I didn't use it until milestone 2) and b) I was too lazy to recompile everything with GDB support every time I encountered something.
We broke GDB in our OS at some point and didn’t see it worth the time to fix it (it would have been worth it).

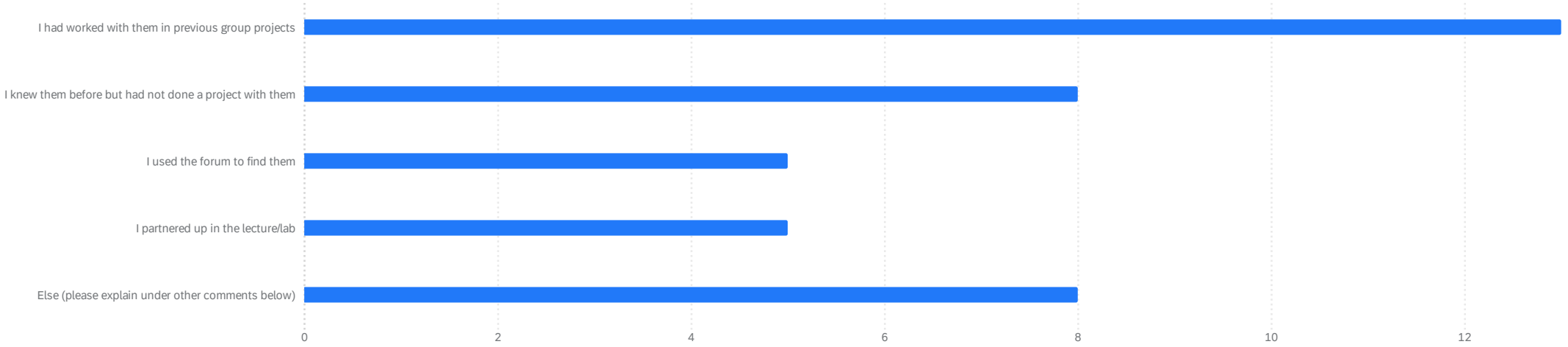
Q24: Comments about GDB (3/4)

Any comments about GDB?
This was the largest project I have done in C, and I had forgotten what little GDB I previously knew.
Didn’t know how to use it.
It worked great for setting breakpoints / printing values, but it would crash whenever I tried to print an expression that would require actually running code. My partner never used it because he never got it set up (not sure if he thought it would be too difficult or just wasn’t bothered), so I ended up doing most of the gnarly debugging.
Searching by instruction (e.g. 0x34adb3) sometimes didn't work and required a lot of scrolling. Other times, I'd use it and be left just as confused as if I hadn't used GDB.
Just that it was super super useful and thank you so much for setting it up.
Thanks for adding it in, I managed to get through with printf though.
Some bugs were too complex/hard to track down to effectively apply GDB-style debugging.
Annoying that connecting to GDB too early can brick the machine, but I eventually set up a script to only connect after it was ready so it was fine.

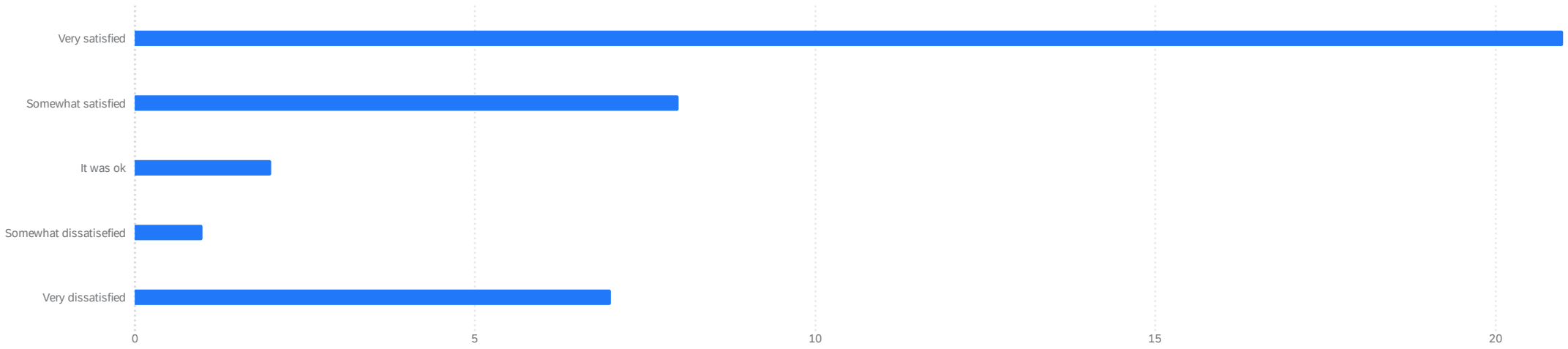
Q24: Comments about GDB (4/4)

Any comments about GDB?
I have never seen anyone use a debugger, and when I try to use one, it takes more time to thinking a little bit (or a lot) with the help of a few prints.
I think the main problem is that I don't know _when_ using debugger is good and when it's bad. The few times I used it, it was pretty inefficient so I quickly stopped trying.
Definitely do NOT do a lecture explaining how to use GDB, there are plenty of tutorials out there, that's not the problem. What I would like to see is videos of experienced engineers programming something, hitting a bug and deciding to drop to a debugger and using it. The only resource like this I know is Andreas Kling (developing a Unix-OS and a browser) on YouTube, but he does printfs... This kind of resources are really underrated IMO. There are two kinds of lectures I want to see: 1. Very theoretical lecture that give deep insights, general results, key high level observations, etc 2. Some experienced person solving a real problem and thinking out loud (ref. Gerwin's regex lectures from the Isabelle course, or Andreas Kling). Really all that takes is shoving a camera behind an engineer and asking them to think out loud. Things in between (eg. going through the tedious details of an API, working through a long tedious example) tend to be really annoying. I think everyone forgets mostly everything from a lecture apart from the key intuition anyway.

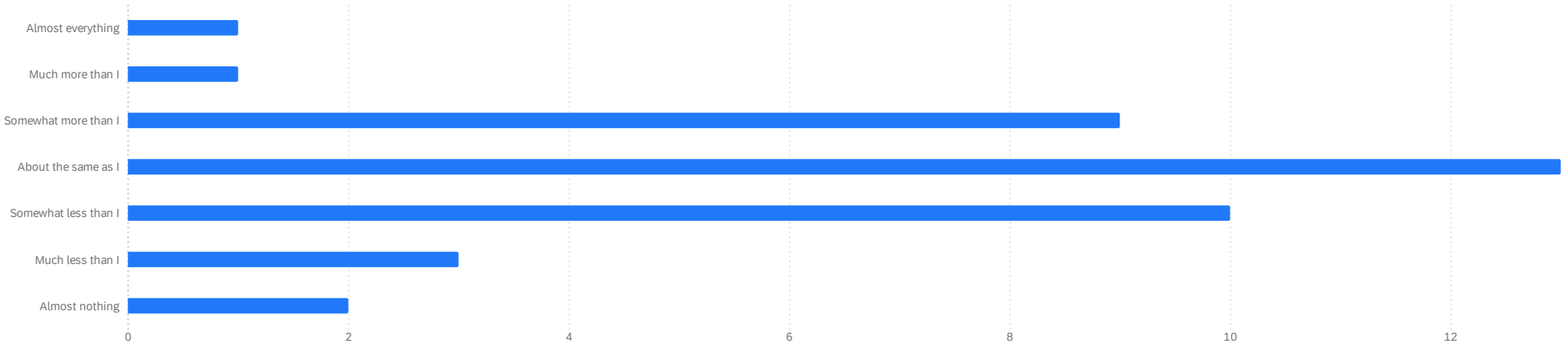
Q25: How did you find your project partner?



Q26: How satisfied were you with your partner?



Q27: Fraction of work the partner did?



Q28: Did your group split?

Q29: Comments on Partnering (1/4)

Any other comments on partner(s)? What could we do to improve things?
Been friends since high school and have done every comp project together so far. Maybe needs even more warning about how important it is to find a good partner as we only heard by word of mouth.
I asked on the CSESoc Discord channel about needing a partner, my partner said he needed one and we decided we would work effectively together and our goals were aligned.
He was in his second year and seemed unprepared for this course, this would've been fine if he made up for his lack of knowledge by investing more time.
Found them on the CSESoc Discord server.
Option to solo the project, with ample understanding of the workload involved would be nice.
Found through discord.
My partner was a long-time friend.

Q29: Comments on Partnering (2/4)

Any other comments on partner(s)? What could we do to improve things?
I found my partner via discord. I'm not too sure if there is anything that can be done by the course to improve this aspect without too much micromanagement or overhead. The challenge from a course like this is of course managing complexity. You don't want to overthink your solution, however, you also don't want to shoot yourself in the foot when you inevitably reimplement your solution down the line. I found that my partner, rarely considered the system outside of the immediate milestone. The cause of most disagreements with my partner would be around what is too complex and what isn't. It was often a struggle to collaborate as I would try to consider latter milestones in the earlier milestone implementations, while my partner would disagree with my proposed view and consider only the milestone at hand. Maybe it would be helpful for alignment between groups if it was clearer what the expected behaviour is at the end of each milestone? I understand that the perogative of the course is not to be too prescriptive on design, however, providing a list of expected functionality, or cases that must succeed, still leaves room in the design of the system from my perspective.
I was very fortunate that my partner was only taking AoS this term, and hence made a ton of manpower into the project. He was very knowledgeable, easy to communicate with and discuss issues/designs with. Couldn't have done this project without him.
Found my partner on discord a term before, we were clear about our objectives for the course.

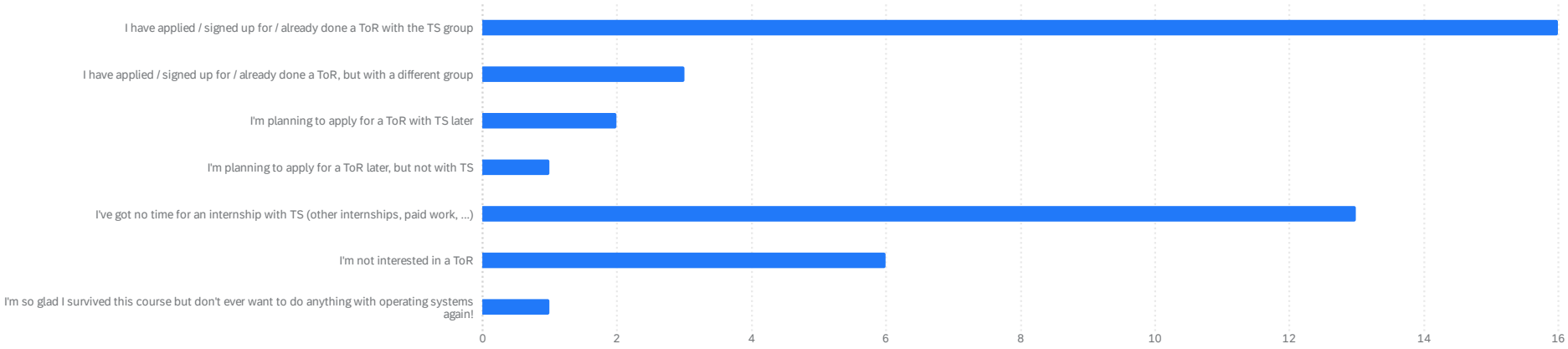
Q29: Comments on Partnering (3/4)

Any other comments on partner(s)? What could we do to improve things?
Other than this course, my partner had a heavy workload (so did I). We never discussed design and alternate approaches. I mostly started off the milestones on my own and he would come do the leftover bits in the end when most of the crucial work was done. And then expect me to explain the whole thing to him for demos. This was extremely stressful since I did make mistakes and my partner's proactive participation would have helped a lot. My other work suffered a lot as a result. My mistake was I could handle the milestones on my own as the later ones were significantly more work. I wish there was no need to work with partners.
Pair programming was a bit tricky at times... There were quite a few moments when either my partner or I would end up coding most of the solution (after discussing it, of course). I can see how that could lead to some teams having an uneven distribution of work. Additionally, there were moments where the integration was really difficult, and with the tight deadlines for each milestone, individual schedules made it way more time-consuming to integrate everyone's work than just writing it yourself, which sometimes forced one of us to "reject" the other's code. If not communicated properly, that could definitely lead to issues down the line.

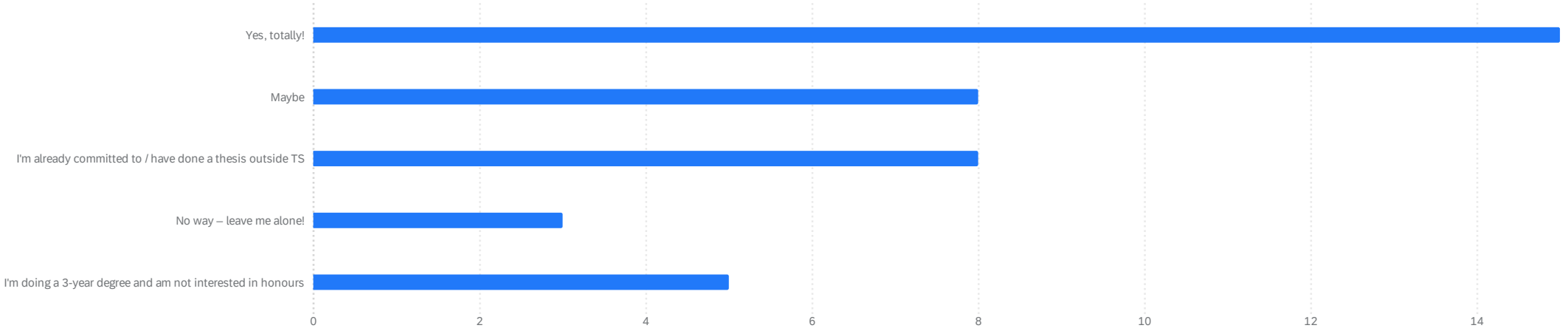
Q29: Comments on Partnering (4/4)

Any other comments on partner(s)? What could we do to improve things?
Not sure if there was anything the course staff really did wrong here given that I didn't clearly communicate the problems.
My partner had some anxieties that were not related to the course but impacted our project. We made it out though.
Have mandatory project checkins like other comp courses to see what each person did + discuss future iterations with a group. This would help add some transparency/accountability. If group members are not on track/contributing equally, this should be addressed for future milestones, with term long issues resulting in adjusted marks.
Probably helpful to make it clear on the course web page before the term starts that people should look for a partner _before_ the course starts.

Q30: Interest in Taste of Research with TS



Q31: Interest in Thesis with TS



Q32: Comments on TS ToR or thesis?

Any comments on TS internships or theses?
I'll be applying for a theses as soon as I can, just doing research on how an honours works in general at the moment.
If I had done this course sooner, I would have probably applied after finishing this course. (Un?)fortunately, this was the last course I needed to graduate, and I was under the impression ToR was for current students.
Not doing an honours at the moment but could change my mind depending on grad offers.
Looking forward to doing thesis
Not sure what TS's options are for electrical eng degrees, will investigate when my honours year comes around.
If I were a lecturer, I would ask the students to come up with their own research topics. I'm really surprised that we are expected to just shop around for topics already prepared, though that seems to be how it's done in most places.

Q33: Final comments (1/2)

Any final comments you would like to make?
Thank you for the course. I can tell that Kevin and Gernot are passionate about what they do, and it comes through.
AOS needs to pay rent for how much time it spent in my head consuming my everyday thoughts.
Thank you for buying us drinks Gernot :DDD
I should do the project and final exam better.
Thank you so much for this course - really glad I decided to stick through to the end :)
Great course honestly probs the best.
Definitely a course I will remember down the track of my life. Funnily enough made me realise I was not as passionate as systems/software engineering as I thought. Not that the course scared me away or traumatised me but more in the sense that I found myself not really that excited or interested in the project or lectures (especially compared to other students) as opposed to different fields.

Q33: Final comments (2/2)

Any final comments you would like to make?
Good course, good staff.
Really great course, really enjoyed it, thank you so much for offering it
Stdin Stdout Stderr behaviour in m6 could be better defined.
I love this course, thank you so much!
Thank you guys for running the course! I'm a bit obsessed with systems, so this course was all I ever wanted from uni.

The End