Feedback Provided by Kate Morris

The Task

We were tasked to install an old version of Ruby and carry out a set of simple tasks that increased in complexity. We had to film our screens and the system timed our progress. Once we started we were fed one challenge at a time and had to deploy each as they were completed. If the deployment was successful we were given the next challenge. If the deployment was unsuccessful then penalties were applied in the form of time penalties. If a break was needed then the system could be paused and then restarted.

This Feedback

I enjoyed parts of the challenge and didn't enjoy others. Some of the reasons I didn't enjoy this were personal and perhaps have highlighted, correctly, areas I need to work on. I have tried to ignore these so that my feedback is constructive. The areas I need to work on can be covered elsewhere.

The Purpose of this Challenge

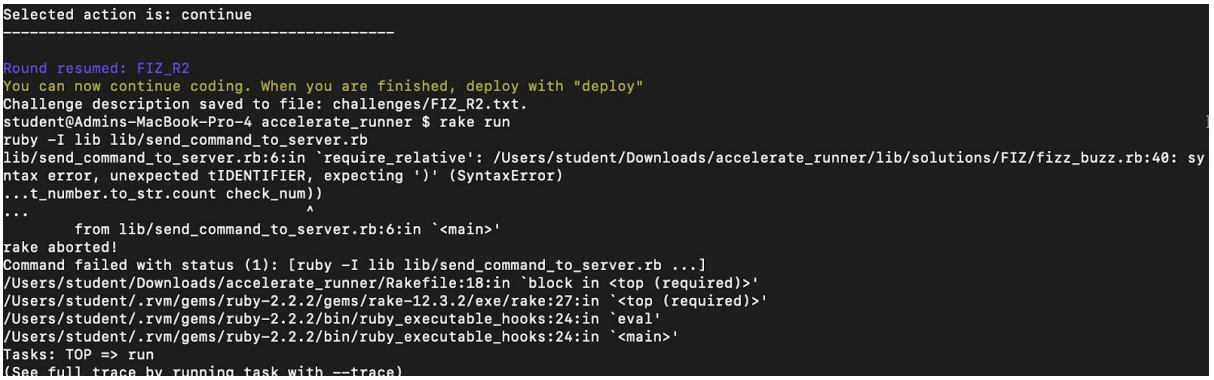
I understood that this was a 'test' of our ability to follow process. With that in mind I decided to update the README with each requirement in turn and give a rough idea of how I was going to tackle the problem. A comprehensive test suite would be required and after the initial challenge I also decided to run a code linter. I refactored once the tests passed and on the last challenge particularly, spent some time renaming functions so that the code was easier to read. Obviously all of these take time but result in better test coverage, cleaner code and a better overall product.

Set-Up

I initially had problems installing Ruby 2.2.2, as did others. Once this was completed I was very aware that, even though this section was not being timed, I was behind the others. I question why this downgrade was required and as the implication of the process was that it was a race, perhaps if we all started at the same position that would have been better.

Bugs

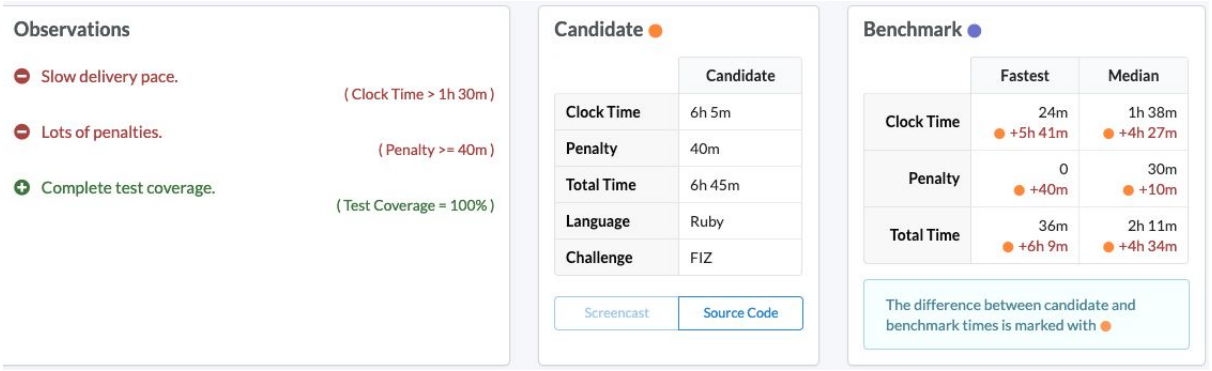
I encountered a bug in the system. From the logging screen I think that it was about 16:48 that I started the final challenge. Around 17:20 I attempted to pause the system so that I could take a break but rake run gave the following error:



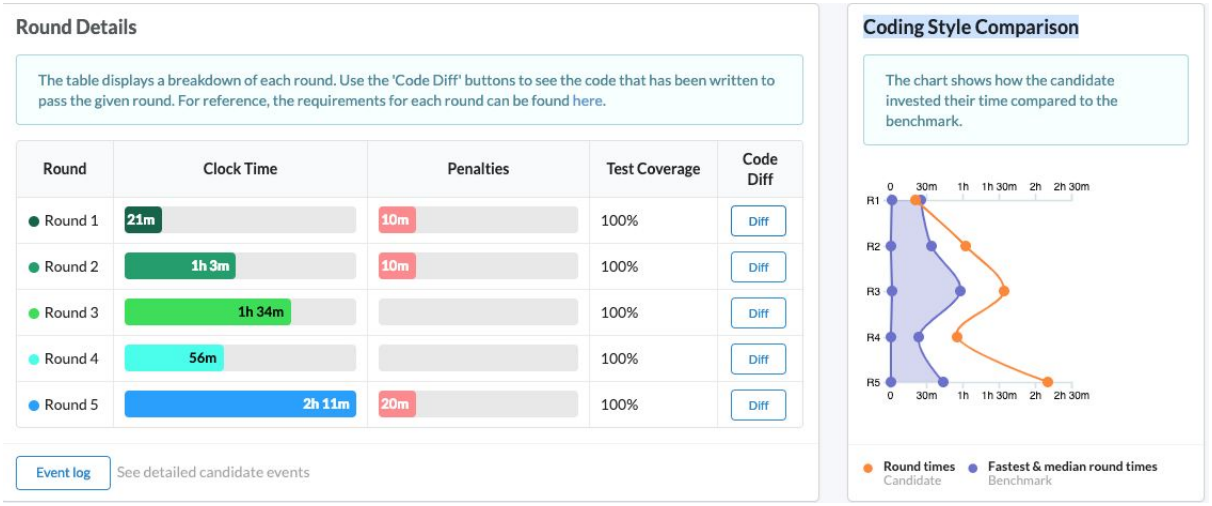
I was unable to pause the system or fix the bug. So for some time it continued to run the clock while I had a break. This is a bug, it needs to be fixed. Fairly straight forward.

Emphasis

I understood that the challenge was to cover process rather than speed. Although speed is important, initially quick badly written code is ultimately more expensive, in every way. I completed the challenge first thing this morning and with a fresh pair of eyes was able to refactor the code. I was quite happy with my solution. Until I got the report. The report seems to be entirely based on time taken.



Above is the first section of the report, 7 of the metrics in these combined tables are time related. Assuming that the penalties are due to not meeting the specification, 4 are due to that and 1 refers to test coverage. The quality of the code does not appear to be measured.

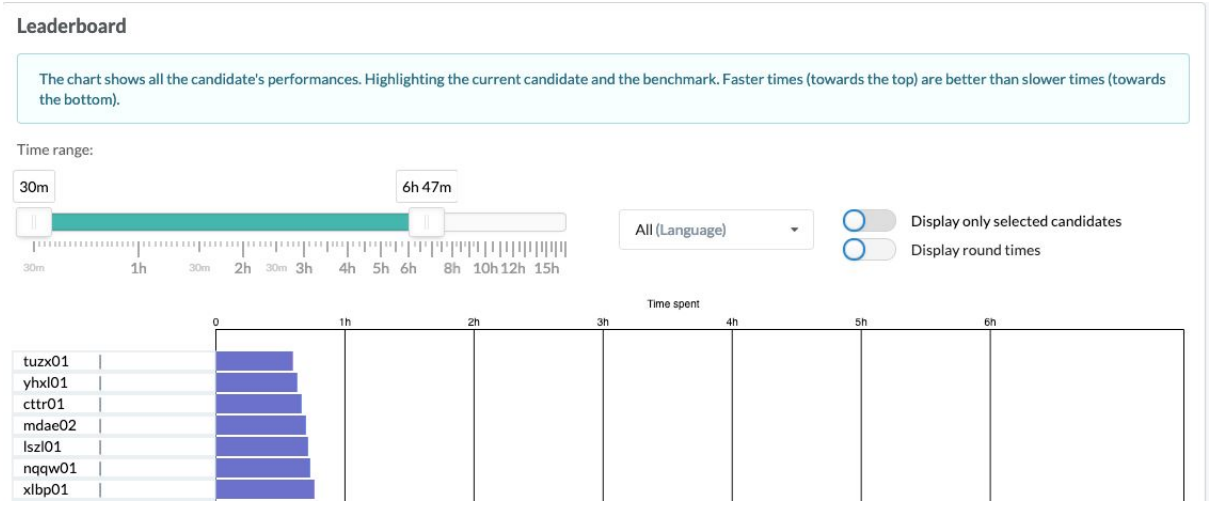


The next sections of the report, above, give a colourful pictorial breakdown of timelines in a clearly understandable table. It also provides a cryptic Coding Style Comparison chart, also, about timing. So, what is it telling us? How is ‘investment’ calculated and by investing more, was that a good thing, or not? Is the object of the exercise to be as fast as possible? Are the penalties for less than 100% test coverage proportionate? I think that this could be clearer.

Competition

The emphasis on Makers from the beginning is that we, the cohort, are not in competition with each other. The publishing of a table of

relative speeds, called the Leaderboard, implies the opposite.



This Leaderboard is based almost entirely on time with penalties having a slight impact. Clean code and test coverage may have an impact but this is not mentioned in this table. We now have a clearly defined pecking order as to who is the best and who is not. Overall, this has had a negative impact on the cohort and may not actually be representative. I would suggest that a relative table is not helpful unless it is relative against one's own performance. That would be useful.

Overview

Overall, although this may not be apparent, I enjoyed the coding. I found the timing issue stressful as it seemed like an exam environment. A much more recent experience for the graduates. In a real world environment I would hope to not be under that kind of pressure without dialogue with the client and/or manager.
Is this to prepare us for interviews? In which case perhaps some sort of warning would help.
Or is this to prepare us for coding in hostile environments? The changing requirements I could understand, and enjoyed, it sharpened my mind as to the naming of things so that reuse was much clearer. But the timing element and the resulting hierarchy, I'm not so keen on.