How did the bucket grading system affect your implementation strategy? If it did not, you may spe... (r0_BucketStrat)

Didn’t really affect my implementation strategy. (r1_BucketStrat)

N.A (r2_BucketStrat)

The bucket grading system didn't really change my approach. I just focused on meeting the criteria for 'Proficient' as a baseline. (r3_BucketStrat)

I just implemented it according to the spec so N/A (r4_BucketStrat)

I would say the impact of the Bucket System was to take a broader approach and not get caught up in very specific use cases, rather getting the entirety of the spec worked on earlier. (r5_BucketStrat)

It didn't affect my implementation strategy, but it affected my mental health that's for sure. (r6_BucketStrat)

I think it helped me save time to work on other things for other courses. If bucket grading wasn't used, I likely would've spent much more time trying to debug a couple tests to get 100%. (r7_BucketStrat)

We used it to develop incrementally, by focusing on one bucket at a time. (r8_BucketStrat)

For c2, bucket grading made us desperate to just get enough functionality to hit the 80% bucket, for all other checkpoints it had no effect. (r9_BucketStrat)

Less focused on the details. Stopped after getting proficient. (r10_BucketStrat)

Would get to proficient, not aim for perfection. (r11_BucketStrat)

Really it just focused on getting atleast an MVP up before really honing in on the spec or doing refactorings. (r12_BucketStrat)

If it was percent based then we would have needed to spend more time on implementation since Proficient != 100%. So as long as we got Proficient we stopped there. (r13_BucketStrat)

Not very much/ none at all. (r14_BucketStrat)

It did not, we just wanted to get proficient and see all tests pass irresspective. (r15_BucketStrat)

Nice to be able to get 100% with one or two tests still failed. (r16_BucketStrat)

It did not affect anything. Because the Autobot was sometimes so vague, there was no strategy we could implement at all. (r17_BucketStrat)

it only affected my implementation when i was in the top bucket because i interpreted it as done. otherwise, no affect because i aimed for the top bucket (r18_BucketStrat)

it made us want to get proficient because otherwise we would be stuck at 80%. If we had gotten, say, 90-95% we'd have been satisfied and not gotten the last few percentages but 20% is too much to give up. (r19_BucketStrat)

It did not really affect our implementation strategy. (r20_BucketStrat)

We implemented until we reached all points in each bucket, and then we stopped. (r21_BucketStrat)

It was sometimes frustrating when we were just short of the next bucket. Ultimately, I think it prompted us to push code that would get us to the next bucket even if it's not the most principled/disciplined (r22_BucketStrat)

It did not affect our implementation since we had the goal of getting the highest bucket by completing all the tasks. (r23_BucketStrat)

I think it mostly just meant I didn't strive for absolute perfection, which I think was much healthier and more reasonable. I always still went for proficient, but once I got to proficient I stopped. (r24_BucketStrat)

it was the only clear cut way to measure our progress. ie 'according to bucket grading, we have a 60' which was helpful, and a little bit annoying at certain points of the project. (r25_BucketStrat)

It did not. N/A (r26_BucketStrat)

Created goals to aim for without needing absolute perfection. (r27_BucketStrat)

It helped us not stress about the “last 5%” if we had a small bug and saved us a lot of time (r28_BucketStrat)

I think we definitely had a lot more leeway with bugfixing with the bucket grading system - if we were already at proficient, we could put of fixing some of the bugs, but if we were almost at proficient, then we could try to fix some of the residual bugs in our system to try to enter the bucket. (r29_BucketStrat)

Our implementation almost always hits proficient once we're confident that everything is working. So the bucket grading didn't affect our development too much. (r30_BucketStrat)

Spotting areas of the code to focus on. (r31_BucketStrat)

It was nice to see what I might need to work on next, otherwise, I didn't really use it much to build my implementation strategy. (r32_BucketStrat)

The bucket grading didn't affect my strategy as I was targeting full marks. (r33_BucketStrat)

I was less stressed. (r34_BucketStrat)

It allowed my group to get an idea on how work should be divided (r35_BucketStrat)

it did, as once we got into a good enough bucket we stopped working on it. (r36_BucketStrat)

It did not really impact how we would implement the project. I would just say it was a bit frustrating to work on a big chunk of the project but still being in the same bucket as the gap between each bucket are not linear. (r37_BucketStrat)

Didn't affect it, we always aimed for 100%. (r38_BucketStrat)

It caused me to create more thorough local tests before I started coding (r39_BucketStrat)

I still do not understand that system. TAs and instructors always said different things about it. Like if you are Proficient, you might get 80 or 100. Some TAs said it's an automatic 100% but instructors said no? (r40_BucketStrat)

The leniency provided by bucket grading let me take a step back and say "good enough", instead of stressing over a percent here or there. That being said, feeling like you're right on the edge between two buckets is frustrating. (r41_BucketStrat)

Didn't stress about everything being perfect, instead focused on core functionality / the spec requirements and implemented them the best we could. If we got Proficient and all our tests passed, then we stopped (even if some smoke tests didn't pass), otherwise we kept working on the suggested areas until we reached Proficient (r42_BucketStrat)

Not really. I guess if I get "proficient" bucket then I am done regardless of if there are still bugs. (r43_BucketStrat)

N/A. My team's goal was to get 100 do having buckets while motivating did not really make a difference. (r44_BucketStrat)

The bucket grading system definitely allowed us to complete MVPs without stressing about making every part of the implementation perfect. I think the fact that the grading was so non-punishing was essential for a large project where lots can go wrong. (r45_BucketStrat)

I mean I always shoot for 100%. I guess giving us 100% at proficient when there can be some tests that failed means I didn't go for a perfect implementation that handled every single edge case properly. Given the tight timelines for the project I think its fine, but it definitely motivates me to not write perfect coverage unit tests. (r46_BucketStrat)

There was more flexibility on implementing our project (r47_BucketStrat)

Didn't - I would have tried for 100% even without it (r48_BucketStrat)

If I did not get the check for that function/ criterion I would look focus on that area. (r49_BucketStrat)

I implement to pass the grading system.. So if it doesnt satisfy the system I will change my implementation (r50_BucketStrat)

less specific focus on a grade number but sometimes difficult to gauge progress in the project (r51_BucketStrat)

I think it probably saved us a lot of time trying to get every single test to pass instead of just focusing on the overall program - which is good, even if we got proficient at every checkpoint. (r52_BucketStrat)

We gave up earlier on project since making the jump from 80 to 100 took too much time (r53_BucketStrat)

We tried to make it work well enough for proficient and so we did not focus a significant amount of our time on very obscure edge cases, which helped the development process (r54_BucketStrat)

The goal was mostly just to get to proficient, rather than a perfect score. (r55_BucketStrat)

It did not. We were aiming to get as many tests to pass as possible regardless of the grading scheme. (r56_BucketStrat)

It often demotivate us as spending a large portion of (limited) time when we were not sure we would get proficient was frustrating. (r57_BucketStrat)

Did not affects implementation strategy, did make us stop implementation if proficient was achieved, even if not all tests were passing (r58_BucketStrat)

We would focus on getting the mark first before considering refactoring or other performance issues. (r59_BucketStrat)

Yes, it made it a lot less stressful because I knew I didn't need to get perfect to get 100% (r60_BucketStrat)

I would usually not try to perfect my implementation if it was already proficient. (r61_BucketStrat)

the bucket grading system probably affected our implementation strategy solely because it alleviated the pressure of needing to pass every single test. the proficient bucket allowed for our implementation to potentially not be as complete as it should be, but we stopped working on it as soon as we hit proficient. (r62_BucketStrat)

It did not affect my implementation strategy, my entire strategy was to follow the spec and hope I pass all the tests afterwards. (r63_BucketStrat)

It made sure that we went all the way to 100% instead of giving up at 90% or something, but idk how much that really changed implementation. It would still give us proficient even if there were still a few bugs. (r64_BucketStrat)

It did not really affect my implementation strategy. (r65_BucketStrat)

It did not affect us a lot as we mainly aim to implement the full functionalities. It did offer us some flexibility in the level of coverage and completion (especially in c0 and c1), which I appreciate. (r66_BucketStrat)

Just try to get to the highest bucket and place code design in lower priority (r67_BucketStrat)

It definitely did, as my implementation was not perfect - but I appreciate having that bucket grading system that rewards those that have reached the learning objectives of the project (because at a certain point of trying to get a perfect score, time is wasted looking for nit-picky edge cases and not necessary learning anything) (r68_BucketStrat)

N/A. We relied very little on the bucket grading until all smoke tests passed, and by the time they passed, it always returned "proficient." (r69_BucketStrat)

It did not, we looked to implement the spec and then try grading our work. (r70_BucketStrat)

it make us struggle a lot to be honest :(especially when the we got 5/6 checkmark done but only got developing, but my friend got 4/6 they reach proficient. (r71_BucketStrat)

Yeah, I felt like i could move on to another issue once i had achieved a developing or proficient grade (r72_BucketStrat)

Happy with it - saved from very low ROI at the end. (r73_BucketStrat)

We would stop working once we get proficient. However, this results in some bugs that are carried on to later stages. Eg. I found our code only evaluates the first 2 clauses of an AND operation, when we are working on frontend! (r74_BucketStrat)

Rather than shooting for perfection, we shot for just enough. (r75_BucketStrat)

Not really. Just made it more ambiguous about what my code/strategy was lacking. (r76_BucketStrat)

Definitely affected it a lot. For C2, there was no time for us to get all the smoke tests, but we already got to Proficient, so we decided to take a rest instead of pulling an all nighter. (r77_BucketStrat)

only programmed until we got 100%, and then stopped (r78_BucketStrat)

We tried our best to maximise our bucket grade. (r79_BucketStrat)

Bucket grading made me less stressed about having a perfect implementation (if one feature is buggy, we could still get full marks). When we were struggling to get into the developing bucket for c2, we tried to maximize as many of the smoke tests instead of focusing on the implementing the project as a whole (r80_BucketStrat)

No, because I was aiming for full marks anyways. (r81_BucketStrat)

We didn't strive for literal perfection, I liked it because if we got 99% right, it would still count as proficient. Saved lots of debugging time. (r82_BucketStrat)

N/A. Always the goal has been to get 100% so I don't think this impacted anything. (r83_BucketStrat)

No. N/A (r84_BucketStrat)

Yes it did, I implemented what it told me to (r85_BucketStrat)

This only affected our debugging strategy. Without it, we would've been flying blind. (r86_BucketStrat)

most time it's not related to my implementation strategy (r87_BucketStrat)

It kinda made it less attractive to put more effort to get a higher grade, one we reached around the 80% grading. (r88_BucketStrat)

It did not. N/A (r89_BucketStrat)

we had to spend most of our time writing more tests reather than actually doing the work (r90_BucketStrat)

I really liked it. I don't think a project created will ever really be perfect, so I liked that there was some lenience in the amount of things you can get wrong. It feels like you're able to make progress that way while not getting wrapped in tiny details that might not matter. (r91_BucketStrat)

Not really (r92_BucketStrat)

N?A (r93_BucketStrat)

It would often lead us to cutting corner and aiming for bottom of the buckets (r94_BucketStrat)

No except for early stopping due to the lowered bar caused by a loose definition of "proficient" (r95_BucketStrat)

We mostly focused on big picutre things first, then worked on details second. (r96_BucketStrat)

It made us focus more on the core functionality of the app first, and getting it as correct as possible. Once we had a strong foundation, we moved on to handling rarer edge cases or smaller, non-core features. (r97_BucketStrat)

It did not, I was still aiming get 100 anyway (r98_BucketStrat)

It helped me prioritize tasks (r99_BucketStrat)

N?A (r100_BucketStrat)

We had one or two missing "points" in the smoke tests, but still achieved proficient. We figured this was ok to keep as is when we saw this, so we didn't spend too much time agonizing over the code and trying to make it 100% perfect, which saved us time. (r101_BucketStrat)

We divided our tasks into things that were graded by each bucekt adnd put more focus on buckets that have more value. (r102_BucketStrat)

gave us goals to work towards. e.g wouldnt have to define an arbitrary cutoff like 94% (r103_BucketStrat)

It did not. (r104_BucketStrat)

since. bucket grading tell which implementation is the most wrong , I prioritize that first over the rest (r105_BucketStrat)

buckets are a cool implementation on how dev might look in industry. tried to maximize buckets by doing the easiest things first. (r107_BucketStrat)

It helped us by giving us something to work towards; we would look forward to increasing our percentage in the bucket. The small incremental feedback provided by bucket grading made approaching tasks feel less daunting. (r108_BucketStrat)

My group was unable to figure out how to get persist to disk working the way the spec seemed to require (seemingly due to misunderstanding the spec), so the bucket grading allowed us to not worry about this one aspect of our code when everything else worked (r109_BucketStrat)