Did the non-code tasks affect your implementation strategy or behaviour in subsequent deliverable... (r0_NonCode)

It did a little bit, but not to a significant extent. It mainly required that I uphold a standard that complied with the non code artifact description. (r10_NonCode)

Yes, it did. It forced us to develop user stories and decide what the front-end of the project was going to look like. (r2_NonCode)

not for c1, but it did for c2 and c3. for c1, the non-code was just something i wanted to get done, but i used the non-codes to guide me and help me build for the rest of the checkpoints. the user stories specially were helpful (r3_NonCode)

They honestly didn't change much. We discussed in class that testing metrics can be faked. I had a similar feeling towards the non-code tasks, while they did present a small opportunity to reflect on the work done during the checkpoint, it is fairly easy to 'just get it over with' (r4_NonCode)

No not really. I didn’t find the non-code tasks (like the scrum meetings) to be structured enough to impact how I implemented the project. (r5_NonCode)

No, we did non-code tasks at the end because the coding tasks take so long, and then afterwards I was too focused on the next deliverable to look back. (r6_NonCode)

Making user stories made me think about how and what to implement for the deliverable, so the process of implementation itself became easier. (r7_NonCode)

Yes, the non-code tasks helped me become a better developer in terms of how I plan out my project beforehand. Before it was just mindless coding, but the non-code tasks allowed me to appreciate the advantage of taking it slow and planning things out. (r8_NonCode)

Definitely, the non-coding bits like planning and documenting made me approach my coding in a more organized way, which helped a lot down the line. (r9_NonCode)

Yes, it helped me start more clearly with user stories and such. (r1_NonCode)

Yes, for example the C2 non-code tasks made me consider some of the smells i introduced in C1 (some of the functions are getting long) and I was forced to consider refactoring mechanisms (r11_NonCode)

I think we did apply some of the start/stop/ continue items of the first non-code artifact as they were good ideas, but otherwise most non-code tasks felt pretty contained (or looked backwards rather than forward) to their checkpoint. (r12_NonCode)

Not really, honestly, mostly because my partner and I struggled with the checkpoints so our strategy was always to just try to get some working code - unfortunately the non code artifacts came as an after thought (r13_NonCode)

I don't know if this would be a good idea going forward, but maybe have separate deadlines for the non-code artifacts? Like to be honest, I understand it's the student's responsibility to balance the work and stuff, but I felt like me, and any friends I talked to were so stressed about the actual coding part that we didn't even consider the non-coding artifacts until like the last day. When in reality doing the non-code artifacts would've probably helped guide the code part. So maybe have them due like a week sooner or something, because I do see the value in them, but I don't think there's a large incentive to actually do them first when everything is due at the same time and it's only worth 20%. Like I imagine a lot of students wanted to get the coding part done first. (r14_NonCode)

Not really, the non-code artifacts felt easy and more like an afterthought if anything. We tended to write the non-code artifacts after we had delegated tasks and had a clear design already. (r15_NonCode)

Not really. I usually did the non-code artifacts at the very end of the checkpoint (once all of the code was implemented and passing tests). The only exception was in C3 when I implemented the user stories before any coding. I felt as though I had a good grasp of how I wanted to implement the different checkpoints without the need to complete the non-code artifacts first. (r16_NonCode)

yup thinking about designs (r17_NonCode)

Yes, I would implement code blocks or put up PRs in relation to user stories effectively splitting up work (r18_NonCode)

not really (r19_NonCode)

Helped me connect my mental image of what the project would look like, and made implementing it more clear and helped me break down the tasks more. (r20_NonCode)

Not really, the user stories didn't have much effect since everything needed to be implemented anyways. Same with the refactoring, the code was bound to have mistakes and I was going to need to refactor anyways. (r21_NonCode)

They didn't because they felt a little bit contrived, again, they were introduced and taught too late for it to be a natural inclusion in our code/ development process. (r22_NonCode)

No, the instructions around them were not clear neither was TA facilitation of them and they were not useful to development. (r23_NonCode)

Reminded to always to continue good software practices even if the technical debt may not be significant at this point, but since it was required for the non-code tasks (r24_NonCode)

Yes, the code tasks make me think about implementation strategies to ensure that our code was maintainable and contained no code smells. (r25_NonCode)

No. Mostly just massaged some work to fit the non-code task description. (r26_NonCode)

I don't think it affects. (r27_NonCode)

I don't think they influenced my implementation too much. Realistically, I made an extra PR to refactor because that was an extra non-code artifact. I probably was going to do this refactor anyways in order to add the new functionality. (r28_NonCode)

No, it felt like it was just extra work on the side for making a PR nice or writing user stories. So most of the time, we would just do it after we implement our code artifacts. (r29_NonCode)

Not really, I feel like there wasn't any follow up or consequence in not following what was said in the previous non-code task. An example is the C1 Collaboration submission, I never heard about what my partner said about me and they never heard what I said about them if anything. (r30_NonCode)

Slightly, not really to be honest. (r31_NonCode)

Not particularly? i don't see why it would. I look at non code tasks as a reflection kind of thing and lets me look back at the part of the project we completed and go damn, i made that, and then think about what I learnt etc. It doesn't particularly make me change anything in subsequent deliveries. Why would it. (r32_NonCode)

Not really, I did the dataset half and I feel like the user stories weren't super helpful at least for my part. As for the refactoring, I understood the importance of having well maintained code, so I would have refactored anyways without having to do that one non code artifact. (r33_NonCode)

Not really. I found them to be mostly separate from the implementation and had little impact (r34_NonCode)

Not really I usually did non-code after the implementation (r36_NonCode)

It was not that helpful. I feel like the partner and I agreed that we just did that part to get some grade, but we didn't think we were doing it in the sense of what teaching team intended to do. (r35_NonCode)

No. Those were done last second usually. (r37_NonCode)

The non-code tasks didn't drastically alter my implementation strategy in subsequent deliverables, but they did contribute to ensuring smooth progress to some extent. While my primary focus remained on coding and technical aspects, engaging in non-code tasks such as project planning, documentation, and communication helped maintain an organized workflow and facilitated effective collaboration within the team. Although these tasks didn't directly influence my coding approach, they played a supportive role in ensuring project success by providing structure and clarity throughout the development process. (r38_NonCode)

Not really, but if we are forced to finish our non code tasks then it might be consequential to the project deliverables. (r39_NonCode)

It did, because I would not have written user stories in advance or prs with as much detail as I had (r40_NonCode)

In my specific case yes, as they made me reflect on the quality of how I was implementing perfom query and based on that how to do things. I tried a lot of refactoring a lot of high level strategies and it really improved the way I was handling bugs (r41_NonCode)

No. We mostly finished code tasks before looking at non-code other than user stories which we did before coding. This did not affect subsequent deliverables. (r42_NonCode)

Not particularly, it felt like more of a hurdle i had to do (r43_NonCode)

Not really. Just for C2, since we knew that in our no code artifact we must explain how we used a design choice or refactored, it affected how I did my refactoring for C2. (r44_NonCode)

in C2, yes because i had to complete a refactoring as part of the non-code portion\i think writing user stories also helped to break down the task (r45_NonCode)

no. our group did not make good use of user stories (r46_NonCode)

Not really. Most of the time, we finished the coding tasks before writing up the non-code tasks. However, I would say we still broke the tasks into simple, independent parts inside our heads rather than jumping straight into implementation. (r47_NonCode)

Not really. We implemented the spec each time and it worked, didn't really think in depth about implementation strategy. (r48_NonCode)

Yes, creating issues / stories before implementing code did affect my strategy. In general, when I am working on something by myself in the future, I think it is a great idea to spend more time planning / thinking, as creating issues / user stories really helped me understand and think much more in depth before coding. (r49_NonCode)

Yes so how, for example for the non-code tasks to demonstrate some of the refactoring from previous checkpoint. I will then look at how can I refactor the most instead of allowing multiple code smells (r50_NonCode)

Yes. For C1, the non-code deliverables made us consider how to improve our PRs (we only made about 2 large PRs). As a result, we made smaller PRs in subsequent deliverables, which made the PR process less cumbersome. (r51_NonCode)

'- Yes, knowing I needed to refactor prompted me to. I also structured the features I implemented around those user stories.\- They were a good reminder of the principles we learned in class. (r52_NonCode)

Not particularly. I completed non-code artefacts early to get them out of the way, and did not especially revisit them... the way that the TAs told us the user stories must be presented (very narrow and specific) made them of limited use for applying to the code writing practice. (r53_NonCode)

They did not really affect my implementation because most of the time I will write them after having the full implementation done. (r54_NonCode)

Yes, it made me think more critically about how this project (and something like it) might exist in the real world, and I found myself always reflecting on how to make it more usable (r55_NonCode)

Writing user stories affected the order I did things, since writing clear implementation instructions and thinking about it in advance helped me plan out my road map for each checkpoint more efficiently. (r56_NonCode)

I don't think having to write user stories ever really affected the way that I approached the work or the order I did anything in, because it was always so interconnected that separating things into stories felt a bit arbitrary. It also didn't really affect how or when I put PR's up for the same reason. I almost feel like user stories would have made the most sense for C1, as opposed to C2 or C3. In C1, the division of tasks was more clear (e.g. for the dataset person, you need to do add, list, and remove and that can be divided up easily). But in C2 and C3 we were adding onto existing code and it was always easier and made more sense to just do it in one go. User stories as a tool for understanding and defining the requirements also would have been more clear for C1, when we were still trying to wrap our heads around the spec. (r57_NonCode)

No. I mostly forgot about them. (r58_NonCode)

Not really. While I understood the reasoning for asking for them, I felt that they often were very general and almost forced. This being said while I still did use some of techniques and things referenced in them (like creating GitHub issues), I felt like doing them to the standards of the non-code stuff was often unnecessary. (r59_NonCode)

I did not find the non-code tasks helpful in any way. (r60_NonCode)

Non-code tasks, such as design and documentation, significantly influenced my development approach. The process of outlining the system architecture helped me make well-structured, modular design choices within the code. (r61_NonCode)

Not really. I didn't really perceive dependencies that resulted in the non-code tasks affecting implementation. Rather it seemed like I could do the code task first and then do the non-code tasks in a retrospective manner. (r62_NonCode)

Not really, mainly because they appear intentional to some degree, and as the specification is quite straightforward to grasp, we don't necessarily require a meticulously crafted user story to commence our tasks. (r63_NonCode)

The programming tasks took significantly more time, so I unfortunately tended to neglect the non code tasks. (r64_NonCode)

Breaking stuff into user stories allowed me to focus on small things first. I also thought of refactoring beforehand (r65_NonCode)

Yes. User stories were an incredibly useful tool in organizing and distributing tasks with my teammate. (r66_NonCode)

It made me think a bit more about the project before implementing (r67_NonCode)

Sometimes. I think some of us hand in the non-code part too late to be able to have it impact our implementation process. (r68_NonCode)

Not particularly, we just agreed to commit more often and plan some things out beforehand. I

don't think it really changed how we implemented our project, since we still went at a similar pace, and there were always things that you couldn't really plan for. (r69_NonCode)

Many did not because they were written or changed after completing the project checkpoint. (r70_NonCode)

Refactoring (r71_NonCode)

Not really, we mainly looked at the artifacts after completing the code component. So by that point, the implementation was already set in stone. (r72_NonCode)

The user stories that we were required to write helped when dividing the work equally in subsequent deliverables. The reflections on refactoring and collaboration also helped me reflect on what could be improved for subsequent deliverables. (r73_NonCode)

Yes, I think it gave us more clear guidance and allowed us to be more deliberate in the strategies and features we added when it came to deliverables that were more open ended like C3 where we could choose the frontend we wanted to implement. (r74_NonCode)

sometimes, as it was mostly a reflection, we can sometimes be positive about the coming future deliverables. however, life isn't always as expected and we're not always able to match our standard or have our implementation match what we stated we would do. (r75_NonCode)

Yes, especially the user story non-code task. I completed the user story for C2 first before starting to implement the new features. Thinking everything through and planning out the attack provided a lot of guidance to my development. (r76_NonCode)

No, because generally I worked on code portions before non-code, so I would end up just putting in arbitrary responses to the non-code portion. (r77_NonCode)

Not really. (r78_NonCode)

Yes, personally having to write up about implementation strategies such as building user stories, has always been helpful for me. (r79_NonCode)

Not especially, the code I wrote didn't end up getting changed and I feel that I already know how to write code that inherently follows the principles we were taught in 310. (r80_NonCode)

I didn't particularly like the non-coding activities. I appreciate they are important to practice, but I felt like I was BSing them. (r81_NonCode)

They did not as the main focus was finishing the project so we often did the non-code tasks after we finished the coding tasks (r82_NonCode)

No, it just seem liek another assignment, and it doesn't really invoke reflection (r83_NonCode)

No, not really. I found that because of the time crunch we did not really rely on non-code tasks to guide our coding. A lot of the time they seemed like an after thought. (r84_NonCode)

No, I would only use the criteria for the checkpoint. I did this as it was faster. (r85_NonCode)

yes. Every noncode helps us revise our code and make improvement (r87_NonCode)

Not really. I spent more time coding to make sure that autograde tests passed and usually started non-code tasks after I was happy with my autograde. (r88_NonCode)
It did. For many projects and assignments that I've done in school, it was very much fine to just submit and get the grade without consequences for the future which is what I am used to for a lot of my schoolwork and assignments in my academic career. However, in this case, the non-code tasks forced me to think about why and how I am implementing and coding my project rather

than just going through it brute force. (r86_NonCode)

In the final checkpoint, definitely! They guided my frontend implementation. (r89_NonCode)

Not too much being honest. (r90_NonCode)

Not really because I did them after (maybe if it was required to do them before I would have been more willing to do them). (r91_NonCode)

No. I didn't think the project was complex enough that the non-code collaborative components were really necessary. My partner and I mostly split the work 50-50, so didn't really need to collaborate much. (r92_NonCode)

Yes, it did. For example, for the room parser, I first wrote the User Story and, while writing it, thought of the implementation of having one RoomParser class, one HtmlParser class and a GeoLoc class. (r93_NonCode)

Yes they did make it easier when I had a concrete idea of what to implement. (r94_NonCode)

Nope, they were always after thoughts. (r95_NonCode)

I don't think so, because we already had an idea of our implementation strategy before the non-code tasks. (r96_NonCode)

Not really, but it was helpful to practice making user stories. (r97_NonCode)

no, cause I mostly finish the coding part then started the non-coding part (r98_NonCode)

I don't think so. It was good to think about refactoring, but if I start a new project today, it'd still be messy in the beginning and require a refactoring along the way. (r99_NonCode)

The user stories helped implementation as it gave us a criteria of what our program needed to include. (r100_NonCode)

By writing out the user story, we have a better idea about the big picture of the project and clear task division. Other than that, it's like the task needs to be finished to gain marks. (r101_NonCode)

Writing user stories made me think more about how to decompose tasks.\Other than that, no. (r102_NonCode)

No, I didn't find the non-code tasks to be too helpful in general (r103_NonCode)

Not really. I find the non-code tasks to be a bit of nuisance although they are easy free marks so I am happy to do them. (r104_NonCode)

Not really. I used to do the non-code part at the end after complement my code part for that checkpoint so I used to create the user story etc. after I was done implementing them which helped me to ensure that my story aligned with what the code did instead of having a half implemented story. (r105_NonCode)

It did not really affect our strategy, but it did help somewhat to plan things out (r106_NonCode)

Unfortunately not particularly. The user stories and refactoring artifacts were good at teaching us those two concepts well, but the reflections didn't change much in my opinion. Perhaps this was because we were mostly concerned with getting the code artifact to proficient to get a grade, rather than making our code extensible. (r107_NonCode)

Not at all. It is very hard to justify spending hours of my time writing unit tests / typing out 20 different user stories and making github issues to track everything and making a cool little sprint board and all that fancy stuff, when in reality It's a school project (one that I wouldn't even put on my resume tbh since every UBC student has it). And it is an absolute time crunch, and given that we have

so many other responsibilities during each checkpoint (school and outside of school), the best way that I found to finish, is to dive straight into implementation, and use the auto grader as your test suite. Granted I wrote a few basic unit tests to figure out if my code was even doing the right thing, but other than that I relied heavily on my c0 e2e tests that we were required to make. For me atleast, the non code artifacts and the sprint meetings were not very beneficial to my learning given my experience working in a scrum team for a year now, it is kinda just a watered down version of the things I already do. (r108_NonCode)

No, we usually did the non-code after the code so it really depends which you do first (r109_NonCode)

Yes. I wrote the README.md carefully because of non-code tasks. Otherwise I won't do this. (r110_NonCode)

No, because we didn't have to go out of our way to do them (r111_NonCode)

No. They were just extra work that at best was just redescribing what I was going to do anyways (r112_NonCode)

The non code tasks were often done near the end, closer to the deadline rather than in the beginning so they didn't affect my implementation strategy that much. (r113_NonCode)

Not really. I feel non-code is a lot about reflections and applying (user story), but my strategy/ behaviour is mildy affected by it (r114_NonCode)

can make me more thoughtful about my work and collaboration strategies and code quality (r115_NonCode)
no, I mostly saw it as something else that I had to complete (r116_NonCode)

Yes, we had more of a plan going into subsequent deliverables being encouraged to write user stories before implementation. This made engineering tasks much easier to figure out. (r117_NonCode)

Not too much. I think the non-code tasks should be separate from the code tasks and be due before starting the coding for each checkpoint (r118_NonCode)

A little bit, helped me plan what to do better for the next deliverable, particulary project planning (r119_NonCode)

I think creating the issues / user stories shaped how I was going to tackle the upcoming segments and how I scheduled it (r120_NonCode)

The non-code tasks did affect the implementation strategy since they asked about code smells and tech debt, which made my partner and I think more about it later during the project. (r121_NonCode)

The requirement to have examples of collaboration in PRs discouraged pair programming and encouraged leaving imperfections in PRs for the other partner to comment on. (r122_NonCode)

Not really, they were mostly an afterthought for me and my partner. (r123_NonCode)

We often did the non code artifacts right before the deadline so it did not affect our implementation strategy. Ultimately, our focus was on the code which was worth much more so we didn't pay attention to the non-code artifacts until we had already done everything we could for the project. (r124_NonCode)

I did not find the non-code artifacts helpful for subsequent deliverables, probably because they didn't feel that important and useful. (r125_NonCode)
not at all (r126_NonCode)

I found using github issues helpful for tracking action items and breaking deliverables into smaller tasks. (r128_NonCode)

no it didnt because it was easier to complete the code before doing non code tasks (r129_NonCode)

Going into C2 knowing we would need to refactor did affect how we changed the code in order to get a proficient grade for the non-code. (r130_NonCode)

It did. A primary example of this were the user stories that were required. Having these user stories structured the way I approach implementing features especially with the engineering tasks. (r131_NonCode)

non-code task gives me a clearer view of what I will do for the rest of the coding part. (r132_NonCode)

Non-coding tasks did not have a significant impact on our development. My partner and I worked very closely with each other throughout the project though, which may not be the case for every team. (r133_NonCode)

No too much, did not seem necessary to change strategy as the teams were only made of 2 people (r134_NonCode)

no... i dont see how necessary it is for a group of 2... i think its only helpful when it involves big project with group of many people (r135_NonCode)

Yes, my coordination with my project partner made it much easier to implement our project than expected because it was easy to communicate blockers and timing our PRs so that we could complete what was needed in a timely manner. (r136_NonCode)

to a small extent, had to do things on purpose for non-code (r127_NonCode)

The user stories are useful guidance for completing C3 features. (r137_NonCode)

Yeah, the non code part helps me to figure out what to do especially in checkpoint three. The user story. (r139_NonCode)

No - was too focused on "completing" tasks to bother with agile development (r140_NonCode)

It did not. I didn’t find them to contribute to my processes. (r141_NonCode)

Not really. we usually put the coding first, then the non code part. (r142_NonCode)

I don't really think the reflection affected my implementation too much, although I say I was going to do "these things" on my refection, I was often in a time crunch, unable to do any of them (r143_NonCode)

Somewhat, although there wasn't a massive emphasis on them besides just completing them on the form (that is, we are not graded on the quality of fulfilling those non-code tasks) (r144_NonCode)

Yes, me and my partner communicated our code to eachother after the first checkpoint and receiving feedback about our pull request comments. (r145_NonCode)

For the last part, when we wrote the user stories first, we had a better mental picture of how to implement C3. (r146_NonCode)

Not really, I felt like I could complete the non-code tasks after the code tasks without much of a downside. For example, I found it easier to complete each checkpoint in one go rather than think about user stories. (r147_NonCode)

perhaps the refactoring requirement. though, I was going to refactor anyway. otherwise no. I often found myself refining pr descriptions and user stories after they are actually used (r148_NonCode)

It helped to divide up the work (r149_NonCode)

not really - we admittedly left the non code artifacts until last minute even when the code artifacts depended on them because we wanted to make sure our code artifact (which made up most of the grade) was done first. the non code artifacts were more of an afterthought and it was more effort trying to conform our non code artifact responses based on the code that we had already written and merged to the main branch. (r150_NonCode)

No. As there are a lot of requirements in specs already, it took a lot of time to implement those and most of the time you don't really have time or any energy so I was mostly looking at the finished part of the checkpoint and based on that created answers for the non-code (r151_NonCode)

Non-code tasks influenced the code structure and checkpoint timeline strategy, leading to a more organized approach in subsequent phases. (r152_NonCode)

No, I hated the non-code tasks and they were an after thought. (r153_NonCode)

They somewhat had an effect on our implementation strategioes for subsequent deliverables. I believe the non-code tasks for C1 (which asked for improvements in terms of collaboration) inspired us to use a GitHub project board for C2 which helped our development process as C2 had a lot of moving parts. (r154_NonCode)

Minorly. The first non-code task, requiring us to reflect on how we worked as a team seemed the most worthwhile (although circumstantially, our teamwork was already pretty solid, so not much changed as a result; still a helpful exercise). In contrast, although the user stories sort of worked for the frontend, they felt very contrived for the back end, requiring us to chunk up work in a way that seemed unnatural given the problem. Finding specific code smells matching the list given in lecture and then solving them using exactly one of the refactor techniques given felt very contrived, especially considering the scope of the project. (r155_NonCode)

No. TBH this project was so massive and overwhelming that I often didn't have time to even consider the non-code artefacts until AFTER I received a proficient in the code artefacts. So I typically did non-code artefacts the night before or day of checkpoint due dates. (r156_NonCode)

Not at all, I felt that non-code artifacts were additional work with no actual value. Personally I did them the last day before each part of the project was due. (r157_NonCode)

The non code tasks affected my implementations stategy. Making user stories beforehand made the implementation easier as I had a clear guideline of what to do (r158_NonCode)

Not much. Non-code tasks in later checkpoints focus on the current checkpoint, which I used to complete them after the coding part is completed. In the first checkpoint we discussed how the project can be improved later, but we are going to make those changes anyway with or without those non-code tasks. (r159_NonCode)

yes, made it fast (r160_NonCode)

Yes. The non-code part forces me to think deeper about what function I want to implement and how to implement, it also forces me to think the definition of done, how to determine if the function has already been implemented. (r161_NonCode)

yes The non-code task could lead the development of code part. (r162_NonCode)

Not really. The non-code tasks were more seen as extra stuff to do at the end. I didn’t really take the non-code tasks to heart. (r163_NonCode)

The user stories offered great help for pre-coding planning of our tasks; they contributed to task distribution and decomposition of each feature, which made progress communication clearer and more logical. (r164_NonCode)

Not really. (r165_NonCode)

Yes,\For refactoring, I specifically paid more attention when coding to make sure it fit in the refactor requirement. (r166_NonCode)

Not really, because we only need to pass the tests of the GitHub bot (r167_NonCode)

No. The non-code tasks did not affect the implementation strategy at all, and it was more of an afterthought. It did lose me some grades when I thought I could submit my partner's issues as mine (I misunderstood the rubric), which was a bit infuriating. (r168_NonCode)

Not really. I thought based on the specification it was already quite easy to see where to get started so I didn't feel like I needed to make tickets beforehand. (r169_NonCode)

Not really, as I felt the non-code tasks were more-so based on producing quality code but due to the time constraint and the nature of the autobot, I felt more inclined to just get my code done before worrying if it was good code or not (which made it harder to make it better designed code later on) (r170_NonCode)

The non-code submission for each checkpoint allowed us to enhance collaboration and task management. User stories allowed us to break down our tasks into manageable bits, while reflecting on teamwork allowed us to have better communication for the following checkpoint. (r171_NonCode)

Definitely, they had us think about design decisions more before we made them. They also helped us keep our feature implementation prs more organized. (r172_NonCode)

Not really, because we typically finish the code task and then make up the non-code part... since the autobot limit to give feedback 2 times per day so it pushes us to focus more on the code task (r173_NonCode)

Yes. After writing the user stories in c2, I understood what I needed to accomplish when I read through the spec for c3, allowing me to specify my implementation on those goals. For example, after reading the spec and writing the user story for c3, I know that I need to focus on having an element that prompts users to add a new dataset. This makes the process of understanding the spec faster for me. (r174_NonCode)

Not really, we largely handled the code first and then the non-code last. (r175_NonCode)

Non code tasks helped me remember the importance of documentation as i went along so I could look back and remember what i did and what still needed to be done. It was very helpful to make github issues, have explanations in my commit messages and good code reviews. (r176_NonCode)

The non-code tasks did not affect how we approached the project in any way shape or form. We usually do them last after the coding tasks because they provide no value in our team's workflow. It may be because we were both people who like to dive right in. (r177_NonCode)

I found myself often completing non-code tasks after being done with the code, with little carry over between c0, 1, 2 (r178_NonCode)

Not really, I thought a bit about opportunities for refactoring while developing my code but it did not have a major impact (r179_NonCode)

Not really (r180_NonCode)

Kind of. We build the frontend based on the non-code user stories we wrote. (r181_NonCode)

It did not. Mainly because the tasks were simple enough that there was no real need to break it down or use the implementation strategies, the solutions were the same as the implementation strategies in a sense but I felt that it wasn’t really using them in the ways that were demonstrated in class. Perhaps if we were given an old repo and

told to maintain/refactor it, it would have forced us to use some of the strategies/patterns more. (r182_NonCode)

Sometimes, usually non-code tasks were done after implementation. (r183_NonCode)

The non-code tasks did not help my implementation strategy much because I usually note down the steps I take to complete these tasks without user stories. It felt that the system was not complex enough to necessitate user stories, but it was good practice nonetheless. (r184_NonCode)

I think if I was better at time management, it would have more influence. In that, when my time management was better and I had more time to think about what I was doing, the non-code tasks helped structure my behaviour and action plan for implementing new things. It didn't help when I was cramped for time because in my head, non-code tasks came second and I just wanted to get this coding done. (r185_NonCode)

No, not really. In an ideal world, with plenty of time, I could have tried to apply some of the patterns of principles which we studied, but I just didn't have the time and concern to do any of that stuff with other courses. Ideally, one would have at least attempted, and implemented partially those patterns. (r186_NonCode)

yes, I wrote more thoughtful PRs (r187_NonCode)

Yes, it motivated us to do more refactoring, and also give more meaningful reviews for each other's code (r188_NonCode)

Yes, it makes me to think about how to improve my code and leading me to further improvements (r189_NonCode)

It did when comes to the user story. It is a great way to set a path to start the project. (r190_NonCode)

not at all (r191_NonCode)

The non-code tasks were great for helping me reflect and see where I would like the project to go. (r192_NonCode)

No. They felt like a formality I just did because I had to, and didn't really match the working style of me and my partner (we usually paired or worked things out in real time over a call rather via github comments or discussions) (r193_NonCode)

No, I usually do the non-code after I finished the coding part. (r194_NonCode)

The non-code tasks kept us on track to have a well designed project, which helped each phase of the project to be smoother. (r195_NonCode)

It did - the user stories in particular come to mind; the requirement to submit our user stories ensured that I thoroughly read the user stories and considered possible engineering tasks/ implementation that could work, as well as a rough timeline or order in tackling different features. (r196_NonCode)

Yes, the non-code tasks affected my behaviour in the checkpoints. For C1, I focused on giving good feedback on my partners' PRs and also putting lots of effort and time into ensuring that my PRs were high-quality. However, this approach led me to making less but bigger PRs. For C2 and C3, the user stories affected how I implemented and pushed my code. I did my best to follow the user stories created by my partners and I and made PRs when I completed the user stories, rather than pushing when I felt that I had completed a significant portion of the checkpoint. (r197_NonCode)

No, because on a project of this scale, I didn't feel the need to use user stories. (r198_NonCode)

Usually they did not because my team had communication issues. Therefore I resulted in trying to meet the deadline. (r199_NonCode)

Yes and they allowed me to reflect on my design decisions and see what I liked and disliked about my process (r200_NonCode)

Not really tbh. We just wrote code like how we normally would. Knowing those acronyms and principles didn't change how we went about writing classes/functions too much. (r201_NonCode)

I am someone who has past Product Management experience and so I found a lot of the middle of the course quite repetitive to already learned concepts. I enjoyed the checkins with the TA and them prompting us with questions on how we would improve - this helped our teamwork a lot. (r202_NonCode)

No, I was too worried about getting marks in the coding portion to meaningfully do the non-code parts. (r203_NonCode)

yes. I think it makes our developing more structural. (r204_NonCode)

I usually finished the non-code after I finished the code part. They didn’t seem to feel effective in guiding my development. (r205_NonCode)

Not really to be honest. My plans would have stayed the same without them. (r206_NonCode)

yea, user stories I made mare more like a goal for me and my partner to fulfill (r207_NonCode)

For c3, our none code task helped implement our UI and with the engineering process (r208_NonCode)

Not really. I feel like I just worked on the code part of the checkpoint, and I wouldn't even read the non-code submissions until I was finished. When I read the non-code part and realized what I needed to do, all I had to do was reflect on my code and PRs. (r209_NonCode)

No. I had everything planned at the start (r210_NonCode)

not really (r211_NonCode)

I think that they didn't affect my implementation much overall, but I noticed that when we began creating tasks and linking them to pull requests, I was making pull requests to try and match the issues that I was trying to fix instead of creating them whenever I felt like it was appropriate. (r212_NonCode)

a little. (r213_NonCode)

Yes, made me focus more on my code reviewing skill for the teammates as well as the user stories (r214_NonCode)

Not really, because I feel as if the non-code artifacts closely followed the specification. (r215_NonCode)

The non-code tasks did affect me to some extent. After the c1 and c2, I found that the lack of communication in the team slowed our implementation process as we sometimes worked on the same part individually. So I started to report my progress and exchange thoughts with the team more frequently. Then we indeed becomes more efficient in the later part of the project. (r216_NonCode)

A little affect on my strategy. This part lets me to think about adopting the concepts learned in the lectures (r217_NonCode)

It did, the user stories forces you to think for a route before starting the implemetations. (r218_NonCode)

Slightly, as writing user stories would make me rethink my approach to how i was going about my implementation. (r219_NonCode)

It didn't really affect it to be honest. It was more of a "meh" thing. (r220_NonCode)

no, we usually did the non-code artifacts after being done with the code (r221_NonCode)

Not generally, I think we did our implementation first and then looked at the non code tasks (r222_NonCode)

Some affected the way in which I divided tasks into tickets. E.g. for some, I chose to break the tasks into smaller pieces to have more options for non-code tasks. (r223_NonCode)

Not particularly. (r224_NonCode)

The non-code implementation helped slightly but since there wasn't much communication between my partner and I it didn't help extremely (r225_NonCode)

No, I tend to do them last after I have a full implementation running. I just don't find it helpful and feels like formality when there's only two of us in a project group, especially since work between the two are usually very well-divided and independent of each other. (r226_NonCode)

Yes, it definitely did. After writing the user stories, we realized that our design for addDataset was very poor. A lot of changes would have to be made for the future deliverables, such as adding the rooms functionality. We therefore completely changed the design for addDataset after the first checkpoint, so that the method can be divided into separate classes later on to add functionality for Rooms. This not only made the addDataset method itself much shorter and easier to comprehend, but also made it very easy to add new functionality for C2, with only very minimal changed required. (r227_NonCode)

Not really. Non-code was mostly a barrier to finish a deliverable that ate up valuable time. (r228_NonCode)

i don't think it affected because i'm not great at organization and ended up doing the non-code tasks later (r237_NonCode)

A little, I had to decouple some functions (r236_NonCode)

It made me think about my changes from more of a user perspective. Writing user stories helped me frame the same changes in a more realistic scenario, which helped me think of new cases to handle or just have a better idea of the expected results of the change. (r229_NonCode)

No I looked at the non-code tasks after (r230_NonCode)

not really, did not find them very valuable (r231_NonCode)

Yes, as I felt like the user stories pushed us to think about how to approach a particular problem more deeply before tackling it. We stuck with this mindset of thinking before doing throughout the project and it taught us to break down our tasks into bite sized pieces. (r232_NonCode)

It helped me better document the changes to the code. (r233_NonCode)

Yes, writing user stories and posting them as issues help me have a better idea for what specific tasks I need to do to achieve a certain feature. As a result, I have an easier time brainstorming and building a blueprint, as I mentioned above, to start coding with. (r234_NonCode)

No, they did not because I already generally decomposed specifications into engineering tasks anyways. (r235_NonCode)

i don't think it affected because i'm not great at organization and ended up doing the non-code tasks later (r237_NonCode)

No - like most mortals, I approach coding problems with a code-first approach and build the test suite later. This is because I am lazy and not sufficiently motivated. \\For the same reason, I tended to do the Code tasks first and the Non-Code tasks afterwards. The Non-Code tasks could be done last-minute and so doing them first rarely made sense, from a lazy perspective. Thus they did not change my implementation or strategy very much, beyond listing more Git Issues and writing code with an eye to the future. (r238_NonCode)

Yes, especially having user stories created before starting implementation greatly influenced my subsequent deliverables. It provided a clear roadmap, guiding my implementation strategy and ensuring alignment with user needs. (r239_NonCode)

Although the consideration of the non-code tasks changed how we approached the code tasks, it did not affect much of our strategy or behaviour (r240_NonCode)

no not really (r241_NonCode)

Yes, non code part allowed us to plan for subsequent checkpoints. (r242_NonCode)

In c1, we were graded on pull requests and code reviews. The practice of writing these got me used to writing detailed descriptions for future PRs in c2 and c3, despite it not being graded, since I realized how useful they were for communicating with my partner. (r243_NonCode)

Yes, as a lot of the non-code tasks required me to consider my plan implementation, meaning I adjusted my implementation strategy to be more scalable for subsequent tasks. (r244_NonCode)

No, non-code was done near the end of the project. (r245_NonCode)

Yes, because it gave me an opportunity to reflect the code I wrote and gave me hint on optimizing them in the subsequent checkpoints. (r246_NonCode)

no. it was just a lame thing I felt obligated to do after completing my code (r247_NonCode)

No I don't believe the project was in a large enough scale for the non-code tasks to create a large enough impact on our implementation strategy or behaviour in subsequent deliverables. (r248_NonCode)

Not much, I tend to do the coding part first and the non-coding part last, because we have a detailed requirement document at each stage. (r249_NonCode)

Yes. User stories allowed for both me and my partner to be more goal oriented with specific tasks and helped us be more organised overall. (r250_NonCode)

Not really, we were too focused on getting the server implementation to work that our non code artifacts did not deter us from trying to debug our server. (r251_NonCode)

They gave some targets for us to consider when actually implementing the project. (r252_NonCode)

Not really. We focused on code artifacts first, then completed non-code artifacts. (r253_NonCode)

For the most part, the non-code tasks did not significantly affect how I approached the project, as they were usually done after doing the coding parts (probably not the best practice) (r254_NonCode)

Yes, a bit. Since some refactoring and cleanup tasks were required for the non-code artifacts, I spent a few days refactoring the whole query system before starting my checkpoint 3 work. I guess I primarily did it because it looked kind of messy to me, but the task contributed to my reasons for refactoring. (r255_NonCode)

The refactoring component I remember specifically as one that forced me to consider how my code currently worked and how I could improve it. Needing to answer specific questions guided my thinking and led to some genuinely helpful changes to my code. (r256_NonCode)

yes, since we work on teams, when I write a code I try to make it more readable so that my teams can easily understand it (r257_NonCode)

Not too much because coding part is done by the autograder and none-code part is like a summary to the code passing all tests. (r258_NonCode)

The users stories definitely helped us visualize what we needed to do. I think a lot of the other non-code artifacts were not overly useful though. For the most part, everything I wrote about in the non-code artifacts were things that I was already going to do. For example, in C2, the refactoring was something I already thought to do and was really just reflecting on my actions in the non-code task. (r259_NonCode)

Yes, it helped me properly document and keep track of my work and what needs to be done. (r260_NonCode)

No, it did not. It seemed just like a checkpoint you have to do which would already have been done if you finished the project. (r261_NonCode)

no, i found that i did them after finishing the coding (r262_NonCode)

No - I felt that the non-code tasks were often not related or exist at a higher level to the implementation. (r263_NonCode)

no, i did the non code tasks after finishing the code tasks. (r264_NonCode)

not very much (r265_NonCode)

Slightly. Because for the C3 user stories I had to link a PR that implemented the user story, I had to make sure my commits to a specific branch implemented a specific story. (r267_NonCode)

Writing user stories required me to read the spec carefully and to think about the steps I will take to implement the functionality so I think it affects my implementation strategy a bit. (r268_NonCode)

User stories somewhat affected my implementation strategy by giving me a clear goal to follow, even if I did not end up always following the rigor requirements defined by my user stories. (r269_NonCode)

Yes, in C2 and C3, we were asked to write user stories before the implementation, this changed my implementation behavior since I used to start coding right away without planning. (r270_NonCode)

It made me and my teammate plan out our future deliverables better, but did not effect the current deliverable. (r271_NonCode)

Yes — they allowed me to plan how I would handle the next portion of the project rather than doing so in the middle or during that portion (r272_NonCode)

I think they were quite important, in splitting up work. Specially the user stories helped a lot in exactly splitting tasks, and if me or my teammate were unfamiliar with a user story concept, we helped each other by providing the engineering tasks for each other as a starting base, which sped up the process by a lot. (r273_NonCode)

I would generally do the non-code tasks last since I felt the webpages about each checkpoint had enough detail for implementing each checkpoint. (r274_NonCode)

A little bit, the formal planning process just seemed fairly impractical and not really worth rigorously following beyond that was necessary. (r275_NonCode)

A lot of the times I did the non-code tasks (user stories) after coding everything, so it didn't really affect my implementation strategy, because the user stories were often focusing on the end product, and therefore the intricacies of the implementation (and where most of the thinking had to be done) was not outlined in them. (r266_NonCode)

The non-code tasks didn't help me with my implementation strategy. Mainly because I was quite overwhelmed with doing the project along with assignments from other courses so I would only consider the non-code tasks during every project checkpoint due date (r276_NonCode)

Yes I think the non code tasks that forced you to think about how to implement the code were useful. Particularly the ones where you had to create user stories. (r277_NonCode)

The user stories in C3 I feel affected my implementation strategy most. C2 had some effect but not as much. (r278_NonCode)

Yes. The reflection helped me changed my approach later on (r279_NonCode)

Not really. We didn't think hard enough about the non-code tasks to realize (r280_NonCode)

The non-code tasks definitely influenced my approach to implementation and behavior in subsequent project deliverables. it helped me develop a more structured approach to coding. Knowing the project’s scope and deadlines in advance allowed me to plan my coding tasks more effectively, ensuring that I spent time on the most critical features first and managed my time efficiently. (r281_NonCode)

I feel like they are pretty good in terms of forcing students to reason and understand non-functional defects that we probably would otherwise ignored given the nature of assignments. They definitely helped me out a lot in terms of understanding them. (r282_NonCode)

Sometimes it did. If I felt I had enough time to refactor the code I have, I would do so. If I felt that I did not have enough time or the code wasn't bad enough to be refactored, I just built on top of it. (r284_NonCode)