Final report - Janet Jackson

Customer Value and Scope

- the chosen scope of the application under development including the priority of features and for whom you are creating value

A: The scope of the application is to create a fitness application that helps users organise their exercises and get motivated to have a more healthy life. The user can save their favourite exercises and sessions to be found more easily when the user needs them. The most important feature is to show exercises to users and give them inspiration and motivation. There is also a recipe page in the application where the user can see recipes and their nutrition. Two slightly less prioritised pages are the calculator page, where the user can calculate BMI and more, and the tips page. The tips intend to motivate and advise the user by giving tips and advice related to health and fitness. The application's personas are people who need guidance to have a more healthy life and athletes who need a place to organise their sporty lives. Additionally, we focused on creating value for the PO and the potential user by efficiently using the APIs, which are used in the main pages of exercises and food and health.

At about the midway point of the project, we managed to have a functioning (though not polished) minimum viable product, which was a product that had all the main pages in a functional stage. The rest of the weeks increased the project scope by adding requirements and wants (usually by the PO's wishes) such as login/account/user, more aesthetic design, responsive pages and design, and user-friendly design and features. These were all features we managed to add in because our scope was initially limited and only later increased incrementally by the wishes of the PO.

B: In a future project, we would like to continue starting with a smaller scope, a hopefully realistic scope, and then incrementally adding additional features, as we believe that to be a winning strategy. Additionally, hopefully, the PO in such a future project can assist in finding good value features to add, as they did for us this time.

A -> B: To reach this, we will advocate for a small MVP and push for projects that we can modularly expand with features in the above incremental fashion. Additionally, we will communicate with the receiver of value for the project and discuss the possible value. Hopefully, this might shed clear priorities and a path forward when it comes to what features should be added or not and in what order (with regards to value).

- The success criteria for the team in terms of what you want to achieve within the project (this can include the application, but also your learning outcomes, your teamwork, or your effort)

Our goal was to create a functional fitness application to attract athletes as well as beginners who want to get motivated and inspired. The product owner was satisfied with the application and that we delivered all his wants, such as filters, sign-in/sign-out, and being able to save exercise sessions. In addition to that, we had some success criteria which give value to us as a team. First of all, Learning Agile and Scrum methodology and being able to work efficiently in a team. Secondly, Gaining some soft skills and technical skills as well—some team members had no idea about how to create a web application but, in the end, could take and finish code-based user stories.

There is always space to improve when the motivation exists. So in the future, we believe we could implement more complex features such as using AI to give a personal analysis using the user's data. As well as improve the communication and the use of scrum methodology.

We could reach that by watching more advanced tutorials to gain the technical knowledge we need to implement these features. To improve Scrum, we can reach that by consuming Scrum resources such as books, courses, podcasts, and online articles or by working and using it.

- your user stories in terms of using a standard pattern, acceptance criteria, task breakdown and effort estimation and how this influenced the way you worked and created value

A: We made user stories for the sprints with acceptance criteria for each user story, and a user story is considered done if its acceptance criteria are met. The user stories were related to the sprint goal. Sometimes when the user story was big, we broke it into tasks instead, making it easier to divide the work between the group members. The task breakdown was mainly for the design purpose of the application GUI. For example, the first user story was about making the start page of the application, so we divided it into a task for making the header and another for making the home page. In the sprint planning, we estimated every user story by voting between 3,2 and 1, 3 considered difficult and 1 considered easy. User stories helped plan the sprint and divide the work between us efficiently so we could avoid conflicts.

B: In some cases the acceptance criteria for the user stories were not so detailed and did not always describe the user stories that we had perfectly. We also believe we did not use tasks so well in our application, and we used them only on the application design when the user story was very big.

A - > B: Next time, we should make more precise acceptance criteria to help us estimate the user stories better and to know when the user story is considered done

because our DoD depends on the acceptance criteria. We could improve our usage of tasks by making tasks for every user story to define the technical steps we need to do so the developers know exactly what they need to do because the user stories are written from the user perspective.

- your acceptance tests, such as how they were performed, with whom, and which value they provided for you and the other stakeholders

A: During every sprint, we checked the results together during the team review and decided if they were good enough to show the PO our performance. We checked that the acceptance criteria were fulfilled in every user story and that everything worked correctly. We also checked the code quality to see if it was sound and well documented. Then we showed the PO our performance for the sprint, and he gave us his opinions about the result and told us if he wanted to change something in the product. The feedback from the PO was helpful to us because it gave us a better understanding of his needs and how we could fulfil them in the best way in the coming sprints.

B: The product owner was not very active and could not attend every team review, so someone in the group had to show him the results after the meeting and then we received the feedback late. Sometimes, this makes it difficult to know if we are doing well and if we are on the right path.

We did not have great testing for the code. We also did not use any testing tools except Postman test for the login. That is something that could be improved in the following projects because it is crucial to check that the code works correctly and is not messy but also well documented.

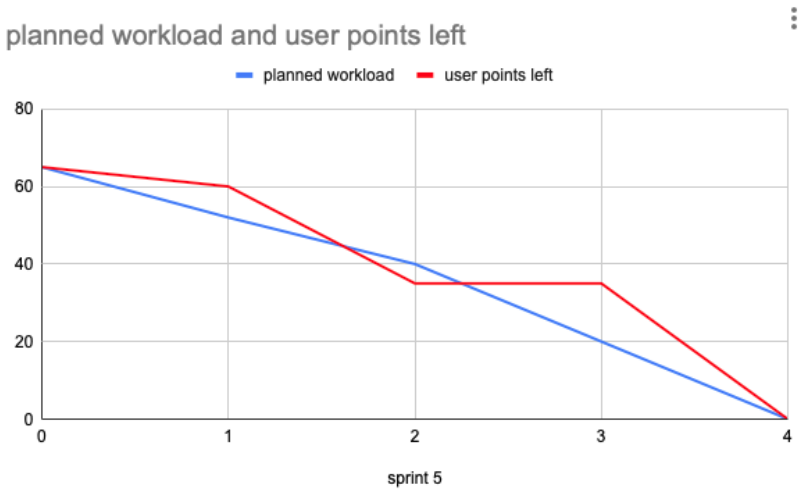
A - > B: for the PO in the next projects, we could require them to be present at the sprint reviews before we start with the project, or we can come to an agreement with them to have another meeting that fits with them better if they cannot be with us in the sprint review on our determined meeting days (which was Friday for us). This would help because everything must be clear before the project, so we avoid misunderstandings.

For the testing, we can use testing tools for React if we do a react project. There are many testing tools for React like jest or React-testing-library and a lot more that we can use.

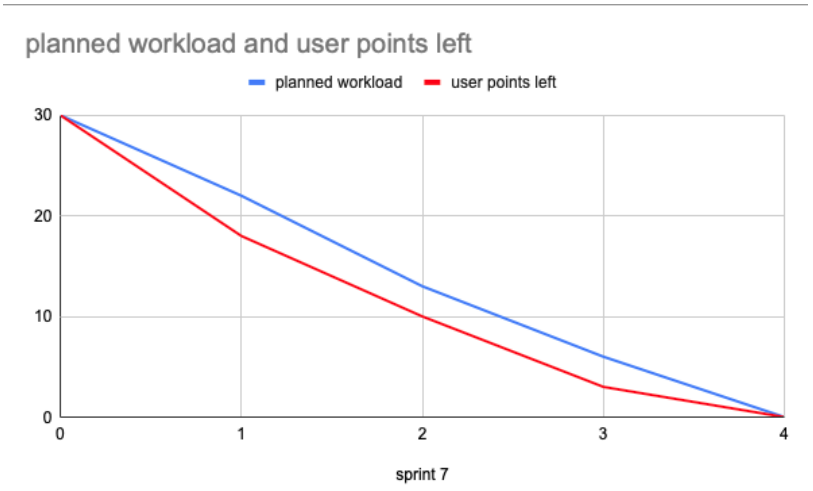
- the three KPIs you use for monitoring your progress and how you use them to improve your process

A:

The three KPIs used during the project were average time spent (and how much of that time was efficient work), the average stress level, the velocity of the sprint, and lastly, a burndown chart. The burndown chart we introduced after sprint 4. We found that all of these KPIs added value in tracking progress. Especially insightful was to see how interrelated the KPIs were. When the sprint's velocity was higher, the stress level also increased, as did time spent. Not only could we compare them with each other within the same sprint but also between separate sprints. Having the burndown sprint was good since it gave us a clear oversight of the sprint in terms of planned workload and the actual workload, which could help us with planning upcoming sprints.



Figur 1. Burdown chart sprint 5



Figur 2. Burndown chart sprint 7

The above charts show the burndown chart for sprints 5 and 7. Sprint 5 was a rather intense sprint that became heavy towards the end, while sprint 7 was a calmer one which is visible from the charts. Using charts in this way was informative for us as a team and helped us track progress.

B: The KPI burndown chart and velocity were not used initially, which would have been preferable. Having a clearer view of the project's progress and not just of the sprint might be useful in a future project.

A->B: It would help to be more critical at the beginning about which KPIs were to be used. Additionally, it could help to reflect on whether it adds value in terms of progress and performance in a better way. Using some KPIs such as velocity and burndown chart from the beginning will make it easy to compare progress across the project and not only focus on the sprint. Furthermore, to have a clearer view of the project's progress, a chart or table can be used that incorporates all KPIs and from all previous sprints.

Social Contract and Effort

- your social contract, i.e., the rules that define how you work together as a team, how it influenced your work, and how it evolved during the project (this means, of course, you should create one in the first week and continuously update it when the need arrives)

The original social contract we created at the beginning of the course was relatively standard. It focused on topics such as: clear and often communication and our mutual values with work (equality, focus on learning, democratic, fairness, etc.). For a few weeks, the contract remained unchanged—primarily, it remained the same because we never saw a need. There were no social or conduct issues. We never even had any need to bring up or look up the contract at all. Then, once we were mid-project, we decided we needed to add a section of our Monday meeting to look it over and possibly revise the contract. We did this despite us having not found any need for change, but simply because we wanted the contract to be fresh in all of our minds and to see if any undisclosed social issues or solutions might have appeared for anyone in the group. The Monday meeting came up with only one new clause—prioritise Wednesday meetings. This, for the simple reason being that we failed to prioritise it previously and felt some more mid-week communication could have solved some minor confusion in an earlier sprint. We immediately noticed that our clarity of communication and work improved when everyone appeared and was decently present at this meeting. So the change remained and became a priority in the future sprints (and even after the sprints too!)

In a future project, where a social contract or similar system exists—perhaps even if it is simply a semi-unspoken code of conduct—we want to take with us the reminder that often, deliberate, efficient communication (preferably in the middle of the sprint) is key. Attendance and keeping everyone up to date on their tasks and troubles is a

lesson we needed to be reminded about that we will carry with us in future projects and employment. Furthermore, to make sure this happens, that we have learnt from all this, we will push for communication, mid-week or mid-sprint meetings, and possibly a mid-project quick lookup of the social contract (as that turned out to be surprisingly efficient even when nothing was noticeably faulty in the first place).

Beyond the above, one of our most used clauses was the clause to help each other—which we often did by pair programming. We found this to be, in most cases, the most efficient work method, and therefore, in a future project, we would all still pair programme—which would be achieved by advocating *for* pair programming.

- the time you have spent on the course and how it relates to what you delivered (so keep track of your hours so you can describe the current situation)

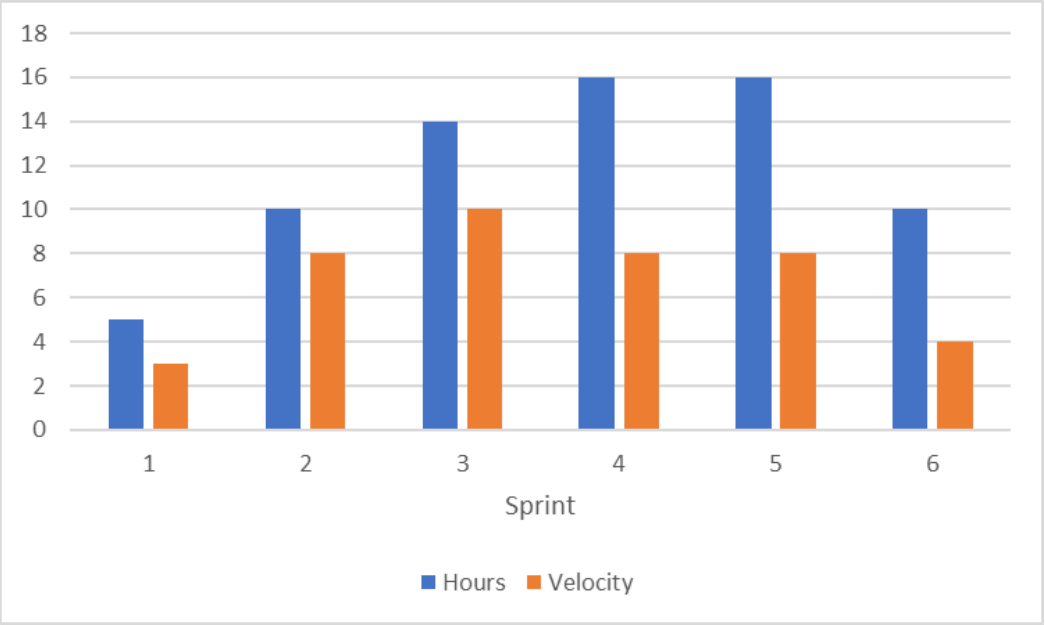


Figure 3. The hours are based on our 1-10 system but multiplied with two. (the KPI of 1-10 worked like this:10 being 20 hours and more of efficient work—the % lowered based on the work time available in the week — with regards to holidays or sickness)

The diagram above is the average hours every team member worked each sprint and the velocity we had that sprint. The hours being the blue staple and the velocity marked as orange. The velocity is measured by adding the weight of all the user stories done that week. The hours are measured in efficient work. Every Friday we had a sprint review and everybody gave a number 1-10 on how many efficient hours they had worked that week which were one of our KPI's. One being almost nothing

and ten is 20 hours or more. Every sprint planning, we voted on the weight of a user story, one being the easiest and three being the hardest. The velocity per hour was higher at the beginning, with an average of 0.7 velocity per hour compared to the 0.466 in the last three sprints. We had a total average of 0.585 ratio between the velocity per hour. In the beginning, the user stories were easier to complete and more overall designed compared to the latter sprints where the database (mongoDB), which no one had worked with earlier. So there were many bugs that we did not know how to solve, and much googling was required. By comparing weeks 1 and 6 with week 4 or 5 we can see that the more hours we worked, the higher velocity we had. And any other result would be questionable.

In the future, we could create more accurate results by completing smaller user stories each sprint instead of having a user story with a weight of 3. Thereby completing multiple user stories each sprint and getting a more accurate result. Alternatively changing the system for measuring the weight of a user story where the point system goes from 1 - 10. It was pretty hard to draw the line between a user story being of weight 1 or 2. The velocity could also have been increased by using a different database that people were proficient with. But we agreed that using a new (mongoDB) was better from a learning perspective.

This could be done by breaking down the user stories into even smaller user stories. Each time we voted on a user story and the weight was over 2 try to break it down into 1 point user stories. Or each sprint planning everyone voted 1-10 of a user story to get a more accurate estimation. Postgresql or firebase database could've been used as a database that some team members were proficient in to increase productivity.

Design decisions and product structure

- how your design decisions (e.g., choice of APIs, architecture patterns, behaviour) support customer value

In this project, we searched for different APIs before developing the application. Then we chose the APIs that had the most value, both for us as developers and the customers. The architecture of the application first consisted of just a frontend react application. Later a backend had to be added to support the authentication of users and storing data because the project owner said he wanted it. This meant we had to refactor and rewrite some code. We chose React and Typescript because some of us had used them a little bit before and knew they were nice to work in, but we wanted to understand them better. Typescript was chosen instead of Javascript to give us type safety, which can come in handy when many people are working on the same code—to get compile errors instead of run time errors. MongoDB was chosen as the database mainly because none of us had worked in it before and wanted to

learn it. It also creates value for the customer by storing data. We also used two different frameworks for the design, both Chakra UI and Bootstrap, because we had developers that were proficient in one or the other.

In a future project, it would be good to have a clear view of the architecture from the beginning and create it early. This would allow us to have a clear picture of what the customer sees as the end product so that we can make design decisions accordingly early on in the project to support that, by creating some sort of architectural diagram, for example, so that everyone can understand how the application works early on and update it as features are added. We should also only use one design framework instead of two, as this increases the number of dependencies, resulting in more waiting time for the user.

To achieve this, we should have talked more to the PO, both in the beginning to get a clearer view of what he wants and sees as the end product, but over time as well. We should have talked more as a team about some design decisions, such as why we used two design frameworks, instead of deciding on one and sticking with it. We talked about it, but both sides thought their framework was better, and no one conceded.

- which technical documentation you use and why (e.g. use cases, interaction diagrams, class diagrams, domain models or component diagrams, text documents)

We use text documents with diagrams such as sequence diagrams explaining how data flows from the user to the backend and back to the user. We also have a SAD diagram explaining the architecture. They also have text explaining some things, such as the .env file and its content, and in reality, it probably shouldn't be shared through a document. We also use postman as a form of documentation as all of our API calls can be seen there and the format that is required to get a response.

In a future project, all documentation should be compiled in one place, so it's easier to find. An architectural diagram should be constructed before starting to code to see if the architecture is suitable for our needs. Comments should be used more extensively for functions and components that require it. There should also be some kind of diagram explaining the database and how it's built, so you don't have to look at the database itself to figure it out. The technical documentation should create value for us as developers to create features easier and faster which in turn creates more value for the customer. Everyone should also take responsibility to make sure there exists technical documentation for the feature that one has created, instead of having to delegate people to do it.

To reach that level of documentation, we should emphasise it more in our DoD and also remind each other at meetings. Someone could have a role each week to

ensure that the documentation lives up to our standards, maybe the ScrumMaster or someone else. Or we could all agree to take responsibility for creating all necessary technical documentation so that anyone can easily join our team and start working to create value.

- how you use and update your documentation throughout the sprints

We have the DoD,“code style” document and a document called “technical documentation”. We’ve taken notes on most of the meetings, at least the planning and retrospective meetings, but not all standup meetings, as they have been pretty short. At first, we had no diagrams or graphics to explain how the application worked. The first diagram we created to explain the application was a use case diagram to easily understand what the application does—to show how the user interacts with the application. But it doesn't say anything about the architecture of the program. So another diagram was also implemented, a system architecture diagram (SAD), with three abstraction levels, the data-level, the logical-level and the GUI. With this, you get a good view of the program and the application’s backend.

In a future project, everyone should know where they can find the documentation they need, so they don’t have to ask. We could also benefit from having more discussions and negotiations about our documentation so that everyone agrees to it. For example, the Chakra UI vs bootstrap in the code could've been avoided if we had decided on one library to use. Changes to documentation should be made whenever necessary, and it’d be good to update the other team members when doing updates to the documents. So no one misses any new requirements that may have been added to e.g. the code style document, such as a changed colour theme.

To reach this, we should’ve reviewed the documents more carefully and more considerate each time to get the most out of them. A library could’ve been decided in the beginning by a vote, so all the components have the same “base styling”. These documents were all in google docs, so the other members didn’t notice when someone changed something if they forgot to say it at the meeting. It would’ve been better to push these to the VCS (Github), so everyone could see when someone changed something.

- how you ensure code quality and enforce coding standards

To ensure our code quality we created a “code style” document before we started coding, so everyone was on the same page about what coding standards we wanted. This document also had a guide on pushing code to the VCS (Github) the correct way. It says you need to add a reviewer to your pull request and cannot merge your own changes into the main branch. The code gets controlled by another team member, and they accept the commit or send it back with requests of changes that need to be addressed before merging the pull request.

In a future project, we should use automated tests and automatically build the project on Github when a merge has happened. We should use Ci/Cd to our advantage and gain all the benefits from it. Also, we had multiple “git merge conflicts”, and as this was something that wasn’t discussed it made it so that when everyone was done with their user stories for the week and merged, there was always someone who needed to resolve a conflict.

To achieve this, the Ci/Cd should’ve been set up at the beginning of the project to avoid many git related issues. The merge conflicts could have been avoided if everyone did smaller commits and everyone pulled from the main branch before they started working on new things. We should also continue with the code review as it’s good to both give and receive feedback on the code.

Application of Scrum

- the roles you have used within the team and their impact on your work

A: When the project started, all group members had different backgrounds and experiences relating to development and programming. Some had coded in different languages and tried different frameworks. But, the chosen framework for this project was React and Typescript was new for many of us.

The aim was to ensure that everyone got included and learned something from this project. Regardless of previous experience, we wanted to give everyone an equal chance to be able to contribute something. To reduce the gap between the group members' knowledge we used pair-programming. We made groups of two, sometimes three persons, who helped each other and did the coding together. This was proven to be a very successful way where every member got a role where they could feel included and contribute. In terms of learning, it was beneficial that the group was diversified and that the knowledge level was different. It made us learn from each other. Additionally, assigning specific roles for each group member in pairs was appreciated. It was easier to know who one could reach out to if an uncertainty occurred. In the upcoming future projects, we would like to use this approach again as it was proven to add value for us as a group and this project. To improve the pair-programming more for the future projects we can have more work-sessions. A session where different pair-groups can meet, discuss and work together.

The role of scrum master was a rotating role so that everyone got the opportunity to experience it. The scrum master made sure that the sprint was moving along smoothly and that user stories were finished according to their acceptance criteria and the DoD.

B: In a future project we would prefer to keep one person as the scrum master throughout the project. Pair-programing would also be used to ensure a good level of productivity in the project where everyone got to contribute and evolve.

A->B: To reach a state where there only would be one scrum master we would simply have a meeting at the beginning of the project and assign the most suitable person for this role. Using pair-programming would be used and the scrum master would get a feel of when and how that would be appropriate.

- the agile practices you have used and their impact on your work

A: During the project agile practices were used and impacted the work in a positive direction. In the initial stages of the project, the group was still a bit unsure of what was “allowed” to do. One such question was whether we could add additional user stories during the sprint that was not included in the sprint planning. After making sure that no group member was overwhelmed by their work and no one was in need of help, a group member who was finished with his user story could begin a new task. However, sometimes what the member did was not a written user story. This was partly due to the fact that the need arose during the sprint. But it still could have been written down. After a couple of sprints, the team started to set out clear sprint goals during the sprint planning. This was to help the direction of the sprint. Say a member, as above, was finished and wanted to do something new, it became easy to make sure this new was within the sprint goal and not something unrelated to the sprint. Also, discord was used when team members got stuck and could just ask something quick over discord sending a screen dump of said problem. Solutions were then sent back. This shorted the time of receiving help since problems were communicated quickly and immediately.

B: In future projects, we want to keep on working in a way that is flexible and not rigid. If team members are stuck and need help, help is provided. Team members are allowed to do additional work as long as no one is in need of help and the additional work is aligned with the sprint goals.

A->B: To achieve this we should in future projects keep on having clear communication that allows a flexible sprint. We would make sure that new user stories (that arise during sprints) are documented with clear acceptance criterias, hence they keep the same standard as the ones done before a sprint. We would keep on having sprint goals since velocity and estimation are good, but also, having sprint goals allows for additional tasks to add value and be relevant to the sprint. Having backup user stories is something that could be considered.

- the sprint review and how it relates to your scope and customer value (Did you have a PO, if yes, who?, if no, how did you carry out the review? Did the review result in a re-prioritisation of user stories? How did the reviews relate to your DoD? Did the feedback change your way of working?)

A:

We wanted to keep track of everyone's work in order to make sure that we were living up to the standard of our planned sprint. We also wanted to check upon our group members if they needed help or more time to complete their task. The sprint reviews took place on Fridays. First, we had an oral discussion where everyone told us about their work of the week. After the oral discussion, we logged it in the team reflection for the specific week. We did have a PO, a student at Chalmers wanting more inspiration in the gym. Overall we found it difficult to include him in our sprint reviews since his schedule did not allow for it. At one midway meeting, where the PO was present, he gave us input that helped us to rethink our design in terms of making our application more interactive/user friendly. He suggested adding a hovering effect and having a proper grid layout for the exercises and told us that being able to save data in a database would add value for him. He could not join us in our meetings every week so sometimes our group member Ali had to contact him on the weekends to brief us about the progress.

The sprint reviews made us sometimes rethink and change the plan for upcoming sprint planning. Especially at the beginning of the course, it was difficult to estimate the time a user story would take. For instance, one user story about enabling a user to add their favourite was shown to be a big one. So the feedback from the sprint review helped us to revise it and break it down into several tasks and reprioritize it for the upcoming sprint. The more reviews we had the better we could estimate upcoming user stories as we gained experience. An important factor was to vote and assess KPIs to the sprint during the review. This approach gave us a clearer picture of how the tasks were felt and where we needed to help each other. Relating to DoD, the review was an opportunity to get approval from other members before committing and merging the user stories to main.

B: We consider the sprint reviews a crucial method for upcoming projects as it reveals a lot about the current sprint and helps one to change the upcoming one if needed. In future projects, we wish to have a more active Project owner who can engage themselves more in our work. Preferably the PO would be available at a majority of the meetings, or, even better, all of the meetings. A suggestion is that he would be a part of the meeting when the sprint is done, in our case, Fridays. It would add a lot of value to the project as we could demo the system for him and show him the progress in real-time. This would reduce a lot of confusion as the communication would be clearer. Communication with the Project-owner is important for us, as we want to know if the delivered work of the sprint is meeting their expectations or not.

A->B: In order to have a more active PO, we would in a future project be more clear about what days and weeks we expected input. This would make it easier for the

project and for the PO. In terms of showing the progress and completion of the user stories, this would still be done during the sprint review.

- best practices for learning and using new tools and technologies (IDEs, version control, scrum boards etc.; do not only describe which tools you used but focus on how you developed the expertise to use them)

We have used a lot of technologies to build the application.
Github helped us to easily merge our code together every sprint and to backup a working version of the application in case something goes wrong. Visual studio code was used to write the code and Discord for communication.
Trello was used as a scrum board to organise user stories and share them between the members. In the beginning, some members had no idea how to use Visual studio code or Trello but the good communication between us made it easier for everyone to gain skills.

The best practice for learning in our field is by doing, especially when there are more expert members in the team who are willing to help. Udeme, Coursera, Youtube etc are platforms with a lot of coding tutorials so it's always a good idea to check there for solutions. Good communication is a key, sharing sources and knowledge between the members is very important to develop the team's ability. In a future project the way our scrum board was used would be improved. There weren't any direct problems with how we used trello in this project but we still see that it can be improved, especially in terms of how a user story goes from in progress to done.

In future projects it's very important to do more pair programming because it's not only good to gain skills but it's even more fun and motivates members to do more work. Having a good communication platform with many features is very important, such as screen sharing and remote controlling so the team can demo, explain and solve their problems efficiently. To achieve a better scrum board we would want to have a meeting after two sprints to discuss how the scrum board can be improved.

- relation to literature and guest lectures (how do your reflections relate to what others have to say?)

For the first two sprints, we did not answer this question at all. We did not answer because we did not know what to write—there had been no obligatory literature and guest lectures. So, we were confused and decided not to answer. However, at the third sprint reflection, once we had gotten a bit more comfortable and a little less bewildered about the project, we decided we needed to act. We needed to figure out a way to start answering this question, or we might be in trouble in the future. Therefore, we asked the supervisor about this question and how we were supposed to fix our issue about this question. Our conclusion was this: we were to read some of the non-obligatory literature on the canvas site and use this as our source, combined with our similar reflections.

In summary, our reflections on this topic compared the experiences and topics brought up by the paper[1] with our own discourse and experiences. Such as how [1, p. 31, 34, 36] brings up the course's high level of autonomy and how that was an issue for some—which was similarly an issue for us. We were, especially not initially, used to the freedom and hands-off approach to a course or project, mainly regarding the technological and scrum aspects. However, with time, we found ourselves more comfortable in the chaos and confusion. In a future project and in future work (which this course simulated well), we would like to bypass the earliest, most chaotic and panic stage, if possible. We believe that with this course and experience in our back pocket, we might be able to skip the earlier terror, if not at least realise and accept that this is a normal start to a complex project.

We also shared similar fears, as mentioned in the paper[1], about whether we are "adopting Scrum 'the right way'"[1, p.35] or not. And although we never got a proper answer to this, our increase in efficiency (both in our KPIs and in our scrum related meetings and work) could indicate a greater comfort and ease with scrum. We discussed a hope that scrum and agile practices would, in the workplace, be yet even more efficient, so not quite so much time would be spent on "not working on the product". We believe that more time and experience with scrum, and additionally trying scrum in different projects and groups, would allow us to be more efficient with the practice and let us at least *feel* like we are doing scrum "the right way".

[1] Håkan. Burden, Jan-Philipp. Steghöfer, Oskar. Hagvall Svensson, "Facilitating entrepreneurial experiences through a software engineering project course," presented at *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training, ICSE-SEET 2019*, Montreal, Canada, 2019-05-25 - 2019-05-31. [Online]. Available: https://chalmers.instructure.com/courses/18176/files?preview=2059847, Accessed on: 2022-05-25.

Github repo: https://github.com/Mange99/Janet-Jackson