Final Reflection

GitHub Repository: https://github.com/FN890/DAT257-Punch-Card
Group: Punch Card
Members: Simon Andersson, Sara Arnesen, Viktor Berggren, Victoria Bjørnstad, Filip Darin Nordqvist, Rasmus Otterlind, Gustav Vadström
Date: 2021-06-04

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) We had the task of creating a booking system for a water skiing club where bookings could be stored. The prices and info for the different activities can be easily viewed and the ability to get a satisfying overview of the upcoming bookings in a calendar. The system also had to be somewhat dynamic. An example of this was the ability to add new activities that can be used and then removed if they are not needed anymore. All these features added value to the customer because there was no central source of info where one could find out if it was already booked or if it was still available for booking. A quote from an administrator was "We have to make about 11 phone calls to check that everything lines up properly for a new booking".

At the start of the project, our supervisor said that the scope of our project seemed relatively small. A booking system for the administrators of a water skiing club seemed small but with the domain description we received the project to us seemed to be just the perfect size. However, this turned out to not be the case as the weeks went on we realized that to make a satisfying non-prototype MVP we had to implement a lot of features. The pillars of importance were a database to store bookings and other info, a frontend web application capable of adding new bookings with different activities and views in the frontend to display information about created bookings; it also required login authentication so that had to be implemented as well. However, to create actual value and work in vertical slices, we had to add things in all three layers: the database, backend, and frontend. This sometimes proved difficult as the amount of work needed was disproportional between the layers. An example of this was the ability to add new bookings which was a user story that seemed simple at first but proved to be very difficult to implement correctly in the frontend.

Towards the end of the project, we had to compromise a bit more to see what features created value because we were running out of time. So in the last week for example we prioritized login authentication over the ability to edit created activities as we already could remove created activities and create new ones which work as a workaround to editing an old one. We realized that some features were more QOL (quality of life) features rather than creating value to the customer.

B) The desired outcome is as always to ensure that the customer's requests are met, by completing the corresponding user story and prioritizing the features that create the most value to the customer. An example of good prioritization could be the ability to delete activities and add new ones as this creates a lot of flexibility, provides info, and also solves the problem of editing activities at the same time.

A->B) In the future, we will be more diligent in planning what features are really necessary and provide value to the customer. We will also think about how features interact with each other. Some combinations of features might make other seemingly important features redundant when trying to create value for the customer.

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).

A) Our success criteria for this project have always been to deliver a finished product to the customer. Regarding the success of individual learning outcomes, we strive to learn as much as possible about agile development, as well as being able to put new development frameworks in our individual "toolboxes".

We have all learned a lot about the frameworks used in this project, which definitely will be useful for us in the future. The project success criteria about the customer being able to use the product have not been fully met. All valuable features of the product have been implemented, but there are bugs and some small fixes that need to be done before the customer can use the product.

The teamwork and engagement has always been great, but the time didn't let us fully finish the project in time.

B) The goal was to deliver a product that the customer could use. The team should also be satisfied with the finished product and their effort in creating it.

A->B) We planned to finish all of the user stories and implement all the features in time, but what we did not plan for was debugging, deploying, and making the finishing touches of the product. In a "next" course, we would definitely take this in mind and perform more testing of each user story.

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) Through the project we have been making user stories on Mondays during our sprint planning. When planning the stories we sat down and looked at what was missing from the complete domain and what could be the next step in accordance with the prerequisites we had in the already developed code. We then looked at what would add the most value for our customers and prioritized these for the upcoming week. During this prioritization, we looked at what was written in the domain as well as Gustav, our representative of the PO, would share his thoughts on the matter and we would all discuss and reflect on it through the perspective of the customer.
On the next sprint planning, we would look over our backlog with stories, add new ones if necessary or change the priority order if Gustav and our evaluations indicated that they should be reprioritized. When the user stories for the sprint were decided we broke each story down into specific tasks to allow parallel work on the code while lowering the risk of disturbing each other too much. Acceptance criteria were discussed during the effort

estimation of the respective user story. They were either broken down into tasks or written down on the story. The acceptance criteria were always discussed as a natural part of the estimation, but not always written down. Criterias were at first introduced in week 5 and written down if they were not obvious or differed from the DoD. The DoD was;

- The whole application is runnable, meaning that your feature is compiling with the rest of the code. This affects our way of working by forcing us to focus on the project as a whole and not allowing us to take the easier way out for the specific task or story.
- All tests complete successfully, meaning that the code should be tested and debugged.
- Code is documented. This criterion was supposed to facilitate further development and for others to read and understand the code. We were often laying behind with the documentation which led to some difficulties with understanding each other's code, but after some time certain people got comfortable with specific areas (frontend/backend/database) and worked more in respective areas which made the obstacle with poor documentation less prominent.
- Gustav is happy, which should imply that the PO is happy as Gustav is the representative.
- Peer-reviewed, to guarantee quality.

B) In a future project, we would continue with our Monday sprint plannings as they were effective and standardized, and made it clear what we should work with during the week. All user stories should have clear acceptance criteria and they should be followed, as well as the DoD. The criteria should be created so that they ensure that the features are sufficiently developed, but not over-processed so that time could have been spent on creating other valuable features.

A->B) Especially concerning the criteria of documentation, we could have been better. Our solution with laying behind was occasionally time-consuming, although it was a result of us trying to maximize the value we could create during our time. For the next time, we should document successively as features are done as this will spare us time in the end and make the code understandable for others and in turn ease the communication of what the code does. Next time we should also include the customer in deciding acceptance criteria so that we are sure that our stories create value to the customer and that we set the bar correctly.

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

A) Team members would test features during meetings, in order to receive guidance and assistance from one another. We would then also access if the features met the DoD and our expectations which were discussed during the sprint planning. It provided value to us through transparency on how the team worked as well as we all got the chance to discuss if the feature could be modified to give higher quality. In this way, we could all feel confident that the product held a level we could present to the customer. The discussion of acceptance criteria, when we estimated the workload of each story, also offered value as it made it clearer what was expected from a certain story and tasks that we took on, and we did not have to spend time dwelling on what we should include in our task when we started it.

B) The ideal situation would be to have the customer present both during the process of developing acceptance criteria as well as during the assessment of whether the user stories met the criteria. This would provide value to the customer by confirming that expectations were met and that the features were made as agreed upon. By letting the customer participate in the development and review of acceptance criteria we thereby guarantee that the features have value to the customer.

A->B) A measure that shows to be repeated in this report is that we should agree with the customer that they shall be present at certain times to validate our work. This time it worked out well because of Gustav's outstanding insight and personal incentive to want the project to go well and give the customer the best possible product. In the next project, this might not be the case. We should also write down the acceptance criteria that we discuss because they were a bit vague and if they were not as obvious they might have been hard to remember.

The three KPIs you use for monitoring your progress and how you use them to improve your process.

A) Our three KPIs were stress, burndown, and a subjective estimate of how much of the value we promised to give the customer and what we actually gave. Stress was on a scale of 1-5 where 5 is the highest, burndown was measured in percent where 25% was the lowest and 50% the highest, and the subjective time estimate was on a scale from 1-10 where 5 was exactly what we promised and >5 is above our promise and <5 is below our promise. The KPIs were measured and discussed every Friday during our sprint review meeting. We then had a discussion about the results from the KPI measurements to help us with how we should plan future sprints. Our KPI measurements didn’t start in the first few weeks as it was forgotten.

The stress index showed stress in general and not only stress about this course, so it showed whether we could increase, decrease or stay at the current pace depending on our stress index. A lot of stress usually came from other courses and the bachelor’s thesis. This helped us with the sprint-planning where there was a lot of stress from other sources we could then do fewer and smaller user stories. The burndown index was very useful for us as it showed us how well we estimated the user stories. Underestimating user stories was something we did in the first weeks of the project, this index helped to tell us that and in future estimation, we had more of a discussion of what the user story included to better estimate the time taken to complete it. The final index helped to tell us if we spent too much time perfecting different aspects of the program even though it was not necessarily needed. For future improvements, it helped us with the discussion of what is included in a user story and its’ acceptance criteria.

B) In future projects, we would still like to have and measure KPIs but we would like to change what KPIs we measured. Our last KPI, the subjective estimate of how much of the promised value was fulfilled, rarely made a change and could also be connected to the burndown index. Additionally, we would like to measure KPI directly from the start to be able to measure how the indexes evolved as progress on the course and project was made. Furthermore, we would like to discuss the KPIs more as our discussions were a little brief and we could gain more from them if we talked more about the measured values, and what they meant. Additionally, changing the way we presented our KPIs to a graph instead of to better show connections between KPIs is also something we would do.

A->B) To reach the desired state we would, after everyone had responded to the form and we can show the values, have at least a 10-minute meeting just focused on discussing the values from the KPIs. At the start of the project, we would also have a discussion about what KPIs to measure to be able to find KPIs that would benefit us the most with regards to the project being worked on and the general state of us. We could then better make connections with the value a specific KPI got and how the week went and then from that be able to implement improvements better and earlier. We would also change how we presented our KPIs so instead of showing them in staples and circle diagrams we would show them in a graph.

Social Contract and Effort

The rules that define how you work together as a team, how it influenced your work, and how it evolved during the project

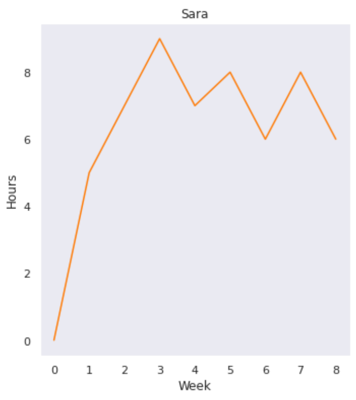
A) The Social Contract has only been changed one time during the project process. However, the rules that were established the first week of the project, were very thorough and reflected all of the group members' opinions and thoughts. As the collaboration in the group has worked seamlessly, there is no surprise that the Social Contract has not been altered along the way. We took into account that team members might not behave as agreed, so the contract included agreements on consequences that would be given in case of misbehaviour. Luckily we never got to use this.

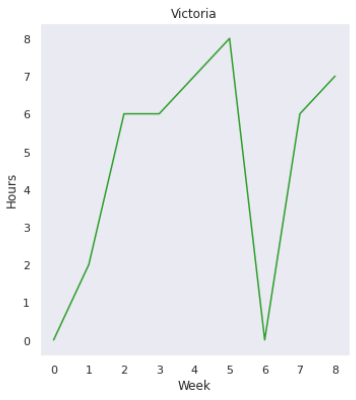
B) The desired outcome when regarding the Social Contract is to have guidelines that the team members adhere to, and that are reasonable and facilitate an effective workflow. It should also lay a basis for a positive development environment and avoiding conflicts.

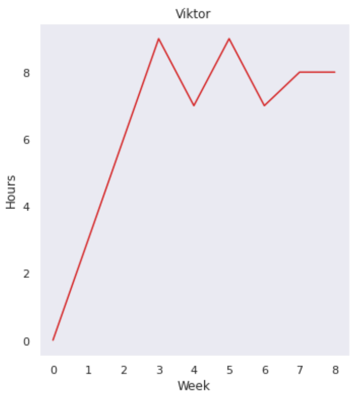
A->B) Our contract worked well for this project and we believe that we could go forward with the same strategy next time as well.

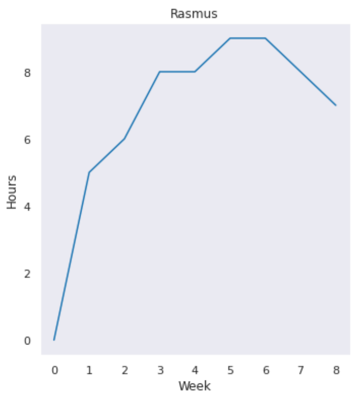
The time you have spent on the course and how it relates to what you delivered (so keep track of your hours so you can describe the current situation).

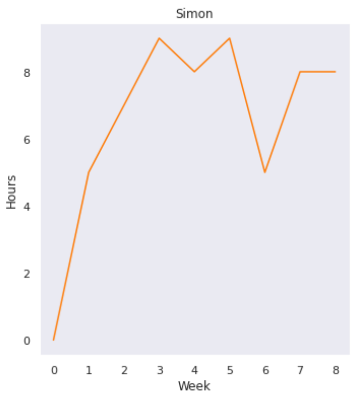
A) This is the current amount of hours spent on the project per week per individual. There is also a graphical depiction below of the total time per week spent by the group as a whole. The total time spent on the project, in terms of coding, making reflections and learning how to code, is so far **384 hours**. Considering the initial competences on current topics we find our deliveries to be good with regard to the imposed time constraint we had. Initially we aimed to code 5 hours a week, which would give 280 hours of solely coding. This is kind of accurate with the total spent time on the project, as approximately 3 hours per week per individual were dedicated towards the individual reflections.

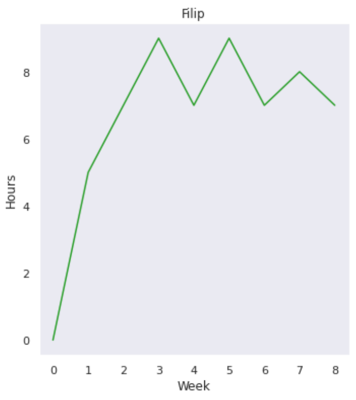

Sara
Hours
Week
0 2 4 6 8
0 1 2 3 4 5 6 7 8

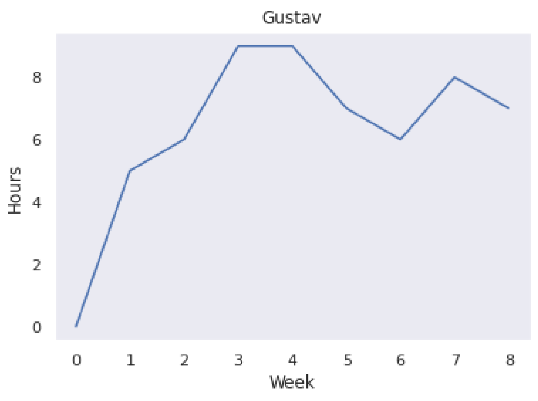

Victoria
Hours
Week
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8

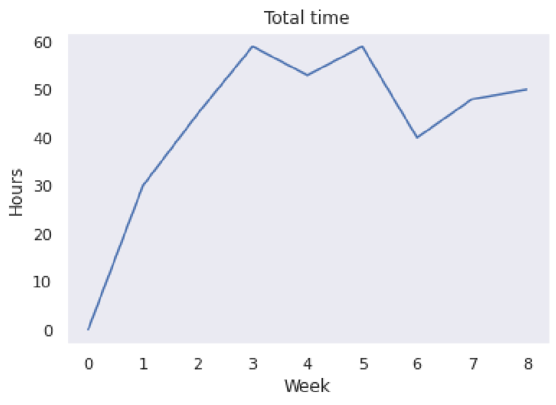

Viktor
Hours
Week
0 2 4 6 8
0 1 2 3 4 5 6 7 8

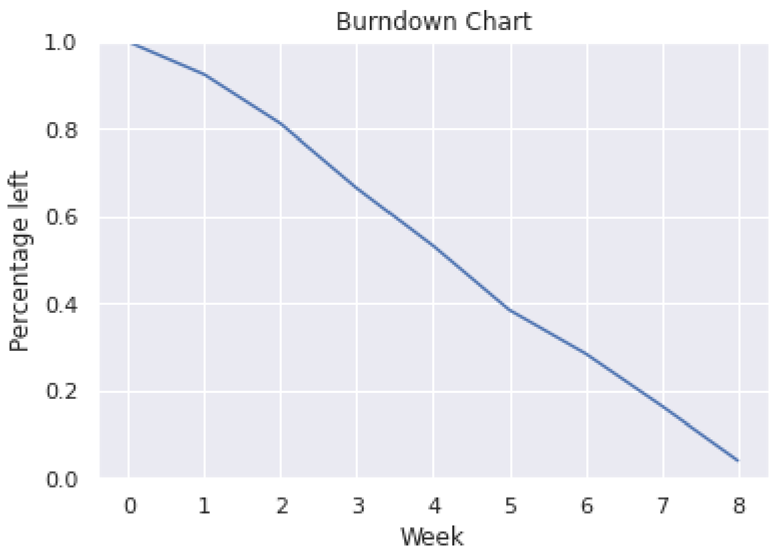

Rasmus
Hours
Week
0 2 4 6 8
0 1 2 3 4 5 6 7 8


Simon
Hours
Week
0 2 4 6 8
0 1 2 3 4 5 6 7 8


Filip
Hours
Week
0 2 4 6 8
0 1 2 3 4 5 6 7 8


Gustav
Hours
Week
0 2 4 6 8
0 1 2 3 4 5 6 7 8


Total time
Hours
Week
0 10 20 30 40 50 60
0 1 2 3 4 5 6 7 8


Burndown Chart
Percentage left
Week
0.0 0.2 0.4 0.6 0.8 1.0
0 1 2 3 4 5 6 7 8

B) The desired outcome was to deliver a finished product. However, as our supervisor said in the beginning of the project, it would probably not be reachable. We also wanted to spend approximately 5 hours a week of coding to reach this outcome, as it would not be feasible otherwise. For the next project, it would probably be better to narrow down the scope to the most essential features, and ensure that these do not have any bugs etc. Thus, the customer could have received a product they actually could use.

A->B) To reach the desired outcome for the next project, the group believes it is important to have a clearer communication in the start about the objective with the project. Additionally, each individual should estimate the time they are willing to spend on the project, and the team should have created a detailed plan over what ought to be done. By doing this it will be easier to estimate the required time to complete the product.

Design decisions and product structure

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

A) Our choice of API and program architecture helped us very well with creating customer value. Our API and design decisions simplified the completion of User Stories. as it enabled us to use good coding templates to create our code. For the backend, we had Spring and different Spring dependencies that helped, simplified, and in turn, sped up the creation of a lot of backend elements. For the frontend, we used React with two major libraries, PrimeFlex and PrimeReact. React worked well and helped create customer value as it simplified creating the frontend. PrimeReact helped a lot with making user input such as text boxes and buttons look nice which created customer value as it increased the websites' ability to show values of bookings well. PrimeFlex was for structuring the website, it helped with positioning elements and components on the site. We do now regret using it as a lot of time was spent on trying to place and position components correctly that we believe could've been done a lot easier and faster with other libraries.

B) For our next project the desired outcome is to have made design decisions that help create customer value by simplifying and speeding up the coding process. We would continue to use Spring and React as they worked very well. One thing we would change is PrimeFlex, we would not use PrimeFlex as it took too long time for the value it created. We are unsure about what we would use but just using CSS or bootstrap has come up as an option.

A->B) To help us reach what we want to do in our next project we're going to discuss this more. Discussion, experimentation, and exploration here are key to finding what would help us create the most value for the customer. These discussions will start with one larger discussion about what to use from start and continue discussing in the first few sprints while the project is at a size that a change of a design decision would be more worth it in the long run than the time it takes to implement it in the short run.

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

A) The technical documentation used for this project was an ER-diagram to be able to create the database with the desired properties. Furthermore, we received a domain description from the customer Alingsås Vattenskidklubb which we used in order to create the ER-diagram, but also to design some of the features in the web application. One could say that the domain description from the customer laid the foundation for the project.

B) For the next project, the group believes it would be useful to utilize more technical documentation. For people that are not involved in the project, it would be difficult to understand why the web application was structured as it is.

A->B) To reach the desired outcome for the next problem, the group believes it would be beneficial to have a conversation about this early on in the project. As this was not a part of the discussion during this project, the idea of having more documentation got neglected from the start. Thus, it was hard to implement it later on.

How you use and update your documentation throughout the sprints.

A) Throughout the sprints, we mostly used Trello to update and use our documentation. We also wrote our individual reflections and team reflections every week to keep track of what tasks we did that week and what we had accomplished as a team.

At the start of the project, we put most of our focus on coding and didn't comment much on the code. Something that led to us not writing so much documentation was because we worked on different parts of the project. But at a later stage, we added some documentation because most of the coding was complete and we felt that it's more important to document complete code than incomplete code. Many parts of the code are still undocumented and we can now at the end of the project say that this was a mistake and left the code with a huge technical debt.

B) In our next project, we would like to continue using a Scrum board like Trello. We felt that it gave the team a solid structure for planning sprints and keeping track of user stories and tasks. It also helped to know who did what so if you had any questions about a class or method you knew who to ask. Team reflections did also add value to the team since we can look back to see what we did and how we could have improved.

We would like our next project to be fully documented, not just at the end of the project but also as we develop the code. This would give the team major benefits in understanding the code and we could help each other with bugs. A documented project is also future-proof, especially if some part of the code lays dormant for a longer period.

A->B) To reach our goal in the next project we need to imply in our meetings that documenting code is a must and more strictly follow our guidelines when writing code, which should include documenting the code directly. Otherwise, we accumulate technical debt that we would have to spend more time on at a later stage.

How you ensure code quality and enforce coding standards.

A) Since the beginning of the project code quality has been an important aspect of the development. To prevent bad code we set up several guidelines that we tried to follow. Some of the guidelines that we used: follow the same coding standard, document as much code as possible, read the official documentation about the different frameworks and test the code as much as possible before pushing it to Github.

None of us had any real knowledge about the frameworks that we used or Javascript. Because of this, we had to learn everything that we needed as we completed our user stories. This was a substantial problem in the beginning which resulted in us following the guidelines less which later led to several problems that we had to deal with. This includes refactoring code, adding missing or poor documentation, and rewriting entire files due to poor code. This became less of a problem as the weeks passed by but whenever we added something new to the project we faced the same problem again but each time the problem occurred we became better at dealing with it.

B) Ideally in the next project, the code quality should not suffer and the guidelines should still be strictly followed when adding something new to the project whether it is a framework or a dependency. This will greatly reduce the technical dept.

A->B) To reach the desired goal we should try to follow the guidelines as much as possible and fix code that does not follow the guidelines before starting on something else. For example, we should try to understand the frameworks that we use before starting to code. This will reduce the number of quick fixes and lead to less technical debt.

Application of Scrum

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

A) We did not assign specific roles except for Gustav being a representative of the PO. This was decided as the PO had limited time to spend with our team, and Gustav had a close relation to the PO. As the customer was a sports club that Gustav participates in, and his dad is a board member of, Gustav had a significant amount of insight and knowledge on what would create value for the club. This insight was a basis and necessity for us to allow this substitute in the view of focusing on customer value through feedback. We operated with a fluid scrum master role where everyone took responsibility to follow the Scrum framework. However, we intended to have a rolling scrum master role so that everyone who wanted to try this responsibility would get to. There was no lack of group members who would want to take on the role, rather the opposite, but as the project went on we grew into a version where certain group members took the lead during plannings while others took a greater responsibility during reflections. This worked out quite well for our group as everyone came along well, but we do reflect upon the possibility that everyone could have slacked off and that due to no one feeling a direct responsibility the scrum framework could have suffered or disagreements could have gone out of proportions. Luckily we never experienced that. However, looking back there are several points that we have reflected upon in weekly reflections that we did not take sufficient measures to remedy. One example is how long it took for us to introduce Acceptance criteria and that when we at first started reflecting upon them, no one thought of the idea to document them.

B) According to the literature and lectures, there should be clear roles; Product Owner, Scrum Master, and the rest of the team. Ideally, we would want our product owner to be more present to avoid having a stand-in, especially from the producing team. This is because the PO would want as much value as possible and the team should compromise with the PO and agree upon the prioritization of features. When there is a mix in these roles there are no clear opposite poles and the value creation might not be as good as it could be because of laziness and lack of incentives or pushed beyond the level of sustainability because of ambition. The ideal would be to have a PO who could participate in sprint planning and/or reviews to prioritize features and approve the work frequently. In the matter of the Scrum master, we believe that the most important aspect is that the tasks of the Scrum Master are completed. For this to be guaranteed there should either be a clear delegation of the tasks in case of a divided scrum master role or simply one scrum master who is responsible for these tasks to be fulfilled. In the matter of a fluid role, there is a higher risk that certain tasks fall through because of miscommunication and unclear work delegation.

A->B) In the next project, we will ensure that the PO can be present and give iterative instructions and wishes so that there is less use of personal intuition and estimation of what will create value. However, it should be said that this project worked well as Gustav had very much insight into the PO's situation and desires. On the other hand, the PO also serves a role in adding pressure and incentive to time-efficient work. In the next project, we will be tougher on the division of responsibilities. Starting with an investigation on who wants to be a scrum master, then a discussion on whether the role should be rotating or elected to one team member. In the case of a rotating role, there should be a definite list of when a member has the responsibility, and ensure that the plan is followed.

The agile practices you have used and their impact on your work.

A) The agile practices we have been using are working in sprints, INVEST criteria for the US, task breakdown, sprint planning meeting, scrum poker, scrum board, and sprint review meeting. Working in sprints even out the workload and prevents the “student syndrome” as well as it allows us to evaluate our work iteratively and reflect on how to improve.

The INVEST criteria helped us develop User Stories that were customized for an agile workflow;

Independent- The independent criteria were especially important for us so that we could code from home without disturbing each other's code. Negotiable- we negotiated the scope of every story during the sprint planning.

Valuable - the stories were derived either from the domain given by the PO, the second meeting with the PO or from own discussions on what would create value.

Estimable - the stories were measured in burndown points for the week that was decided with scrum poker and discussion.

Small - Our stories were small and clear, which we do reflect upon to be one of the main reasons for why we did not write down any specific acceptance criteria for specific stories as the story itself, especially when segmented into tasks, was sufficiently specific.

Testable - We strived to make testable stories and this often happened quite naturally as a consequence of wanting measurable and concrete results. This was important so that we could validate our work and let other team members review the feature. However, tasks were not always testable but we considered them sufficient with testable stories.

During the sprint planning meeting, everyone participated in developing user stories. We estimated their scope by playing scrum poker and discussed the highest and lowest score that was given. We then re-voted to see if the score converged, which it usually did, and finally went with a mean value if the score was not homogeneous. We then broke down the stories into tasks and assigned them or left them open to grab during the sprint. We also made time to take the floor and mention obstacles in the upcoming sprint that made it hard for you to participate in certain meetings or pair programming time. We also updated our scrum board (Trello) to visualize the content of the sprint and ensure that we were not working on the same task without knowing. We consequently updated our scrum board and the pillars Epics, Backlog/ User Stories, Sprint Backlog/Tasks, Todo, In Progress, Testing, Tasks Done, and User Stories Completed. This made the workflow very transparent and facilitated delegation of both tasks but also on providing help to others when a member had spare time. We did a review of the sprint at the end of the week, this was usually conducted efficiently as bullet points and takedowns had been written before the meeting. During this meeting, we also went through our KPI's and had a look at the burndown chart and Scrum board to see if we had done what we had planned.

B) The ideal would be to have a dynamic investigation and development of agile work practices to see if there are new methods that could improve the workflow even more. However, we believe that our practices were quite good and efficient. the implementation could have been standardized more.

A->B) An improvement till next time could be to standardize the practices even more. We worked on this but ex. delegation of tasks was sometimes done successively through the week, and sometimes all at the beginning. There could also have been more thorough reflection on specific measures to improve to ensure that we found the most reasonable arrangement.

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) During the project, we have had a sprint review every Friday. At the review, we took a look at the Scrum board, and if there still were any tasks on "todo" or "in progress" we talked about what the situation was with these, and if the task could be done in the last few hours of the sprint or should be put back to the sprint log for next sprint. We then went through the bullet points for the weekly reflection and answered the KPI's as well as plotted the number of hours we had coded the current week. This gave insight into workload and helped the group adapt to a better work delegation over time.

During the sprint reviews, Gustav was replacing the PO, as he had a close connection with the customer as described in "The roles you have used within the team". We would review our work during the sprint by testing and discussing thoughts about the features and whether they should be developed further or were sufficient to give value and cover the customer's demand. In these discussions, Gustav had the last word, but we were all putting ourselves in the shoes of our customers to evaluate our work. When running the code we also got an insight into whether the stories were following our DoD. The DoD's are stated in "Customer Value and Scope" under acceptance criteria.
The reviews did not change the prioritization of the US, as the prioritization had been based on the customer's requirements that were stated in the domain or during the meeting with the customer. Neither did the prioritization change after we met with the customer, however, we did add some features that were not stated in the initial domain description that we were given prior to the project. We always reflected thoroughly on the prioritization when we made it, ensuring that each sprint would bring value to the project, but we also took practical preconditions into account. Eg. it was important to consider the vertical slicing and we could not take on tasks or stories that were dependent on code that was not yet written, eg there is

not much use to edit bookings if we not yet had implemented adding bookings, however, these conditions were often correlating with what was making the most value to the customer.

B) Ideally, the customer should be present both in the review and planning. During the planning, it is necessary with a present customer to plan which features should be done, how much the team should deliver in the upcoming sprint, and what the acceptance criteria are for the sprint's user stories. This is all to ensure maximum customer value as well as it benefits an efficient development because the risk of spending time on non-value creating activities is minimized. During the review, it is important with a present customer to validate that the acceptance criteria have been met and that the stories can be marked as done. It is also important to review if the customer wants to prioritize features so that the team always can produce maximum added value. There should also be a structured review of whether the completed user stories meet the DoD.

A->B) Next time we should make an agreement with the customer at the beginning of the project on how often and when we can meet. This will ensure at least some presence, and it should be in not only our interest but also the customer's interest to meet and discuss the ongoing project to ensure that it meets the expectations and gives value to the customer. We should also go through every point in the DoD and check them off when we review completed user stories. This could have forced us to document our code more frequently.

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).

A) For our frontend, we used React and many different React libraries. For the backend, we used Spring with our database in PostgreSQL. We had no tutorial or anything to learn PostgreSQL as every member in the group had recently read a course in databases and everyone had read the same course. For React we chose React's own “get started” tutorial. For Spring and the connection between Spring and React we had chosen a long tutorial that showed integration of Spring, the connection between Spring and the database, and the connection between Spring and React. That tutorial later showed to be outdated and we then switched to go through other smaller tutorials to cover frontend, backend, and how they communicate. While doing these tutorials we also found out who preferred to work on the backend and who preferred frontend and both parts then we did more guides and tutorials for their end of the project. The biggest libraries we used in React were PrimeReact and PrimeFlex. For both PrimeReact and PrimeFlex we used their documentation to learn about them and find the things we needed for our project. This was mainly because they were both introduced in the middle of the project. For version control, we used GitHub which the majority of the group were used to so we did no tutorial here except uploading a git workflow to our README file.

B) The start of the project went well and we had no problem creating our first runnable version. There were still improvements we could make here though. The first one is to have a better way to show how GitHub worked to members who were not used to it as there was some trouble getting started with it. Another improvement to be made was to better coordinate what tutorials and guides everyone did to have a more evenly spread knowledge and not constrict the function of a certain component to the only member who found a guide about it. Lastly, for our React libraries, the documentation worked fine but a more thorough tutorial or guide for PrimeFlex and PrimeReact would've been better as they were basically the looks of the website. It would've helped a lot with making the website have a better user interface.

A->B) Firstly we will have more communication in the group about what tutorials we go through and what they show. We will also have a review of GitHub, both to help members not used to it and to get a standard of how we should use it. Lastly we will also watch tutorials and guides about all major components of the project whether or not they were a part of it the whole way or were introduced in the middle of it all.

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

A) Our way of working was not always taken straight from the book, and we found our own way on how to combine different agile methods to get an efficient work environment and flow. The different methods were taken from the lectures at the beginning of the course and even some searches on the internet gave us further insight on how the scrum framework is used in practice.

B) The ideal way would be to look at those who have succeeded in their agile work, may it be professors, other professionals, or literature, and take inspiration from this. However, as we were told in lectures and at supervisions, there is not a box to fit into and every team has it's dynamic and needs to find its version of agile work.

A->B) Next time we should be following the way of the literature from the start without modifying it directly as we did with our scrum roles. This way we will be working closer to the explored and proven effective and agile way of working and will be armed against mistakes that others have encountered.