Final Team Reflection

Jakob Ristner, Erik Berg, Anton Hansson, Haris Cehic
Nima Hansen, Mohammad Jamal Basal, Jakob Henriksson

June 2, 2021

1 Customer Value and Scope

1. **A:**
   The chosen scope during the project has included a hybrid of functionality and design although the customer has leaned somewhat to prioritizing functionality and features over design. However, a common theme during the meetings with the customer was that they wanted us to "Do our own thing" and "Just develop the views in order" which we had to actively attempt to counter (their initial approach was more waterfall like). As the project evolved they began to understand the value of iteration and giving constant feedback which also enabled us to make better us of the agile process.

   As an example, in one of the sprints we created multiple examples for the layout of a form and the customer was asked to give feedback as well as decide on what they thought works best for them. We believe this made the customer get a sense of contribution to the developmental process instead of following the waterfall principle and just letting us do all the work.

   B:
   In an ideal world, the customer would be in this mindset of iteration and feedback earlier in the process. Since we have many views in the web application that look and work the same, the customer having this mindset would lead to us having a better grasp on exactly how we would build these components quicker. The prioritisation of features for our next project needs to be more inline with both what the customer wants, and what we as developers need. It would be preferable to prioritise different types of features as it makes it better for everyone if many types of fea-

tures gets iterated further.

(A → B):
One part of the problem is that, although we developed what the nurses believed to be the most valuable to them, perhaps their idea of this was somewhat skewed by the fact that they care most about what they can directly see and give feedback on, which while it is their right, sometimes they missed the overarching goal of the application. An example of this is that a major part of their trial-work is to log statistics in order to determine if their home-care strategy is economically sound. This feature of downloading data was never prioritized and ended up not being developed in time and therefore they cannot perform this statistical analysis using our application as of right now.

What we should have done instead was to develop one of the minor forms to completeness, alongside other features such as login or exporting of the data. This would let us get consistent and good feedback on the minor forms to later be able to quickly create the other, very similar ones as well as get ahead on features that were now left behind due to prioritization. Even though we created what they said gave them the most value, perhaps we could have been better at giving them more insight in what this value could be. On the other hand, having everyone working with the react forms gave the whole team a very good understanding of how to work in the development environment.

2. **A:**
The main goal for our team has been to learn how to develop a react/express web application, as well as understanding and applying the agile process. Completing the project has been something to work towards but it was never a concrete success criteria for us.

Another goal has been to work together with new people from different backgrounds with different sets of knowledge and experiences prior to this project, and then finding the best way to efficiently complete the project.

B:
In an ideal world, the projects success would be heavily determined by all the factors above as well as the completion of the application. However, this would be a much more realistic and achievable outcome if one were to work in an established team. In other words, if we as a team were to do another project right now the success criteria then would probably have a lot more to do with the completion of the project.

(A → B):
We are satisfied with our success criteria in the context of this project. There's not much to change except in case there is a new project, and then we feel like the goal would be to finish it entirely.

3. **A:**
We noticed early on in the project that it is very easy to neglect the scrum process and just do your own thing. At first we were simply reading the user stories and implementing them, only to later look at what tasks and acceptance criteria were fulfilled. This is obviously not very good practice and was mostly caused by not creating very concrete tasks. Therefore we spent a lot of time early on to address this issue which led to an increase in productivity and sense of completion.

   The biggest issue that we had concerning task breakdown during the project was when we worked mainly with design elements. We quickly realized that it is very difficult to create a concrete task breakdown for a layout implementation in contrast to a feature. This was exacerbated by our lack of well defined mock-ups for the views and forms.

   Another thing that we realised during the project was the value that our effort estimation brought us. In the beginning we did basically no effort estimations which, while not a hinder in productivity, it was harder to track how we were improving as the project went along.

   The way that we later did our estimations was by writing a number in our Discord, everyone at the same time, and then the person with the highest number and the person with the lowest number had to justify their numbers (inspired by Scrumpoker). This made sure everyone was on the same page and was a good starting point for discussions about the user stories.

   B:
   It would have been ideal if we had spent more time initially and started using our Trello smarter right away. Well-worded user stories and tasks aid the development and efficiency of the coding. It would also make it easier to perform effort estimations since all members would have a better understanding of each user story's scope as well as what needs to be done on each task.

   The user stories could have been better worded with more concrete tasks to them.

Another thing that we felt was lacking this project was the acceptance criteria of the user stories. They were often not detailed enough or so vague that it was hard to know if they were completed or not. For the future we would need to create better acceptance criteria. This also affected our KPI:s (see 1.5).

(A → B):
One thing we could have done prior to the first sprint is doing research on what makes good user stories and finding examples that relate directly in some way to what we are developing. This could help us with better acceptance criteria and tasks.

We usually created user stories and tasks during our Monday meetings and then divided into coding-pairs. To ensure better user stories, maybe this should have been done on Fridays after our sprint reflections when it was still recent. Which also would have resulted in more time to reformulate and possibly make better effort estimations. On the other hand, most members are usually tired at the end of the week which might result in less well-worded user stories and tasks.

We would also have much better acceptance criteria if we did not just try to write them ourselves but instead try to communicate with the customer, translating their non-technical explanations in a better way to something we can build on. In fact, we should have integrated our user story creation into our meetings with the customer.

To resolve the issue of task breakdown in a design context there is mainly one solution we would like to implement now that we know that tasks related to design are not as helpful in future development. Having a dialogue with the customer and iterating over mock-ups early on to solidify a design before implementation is key.

4. **A:**
In the beginning of the project we believed that code reviewing could be performed naturally during pair programming, however after a while we realised that this was not enough. Thus we begun performing tests and code reviewing prior to merging our code in the last 3 weeks of development, even though it was placed as a criteria in our definition of done much earlier. We performed these tests on Thursday at the end of the sprint as we usually were mostly done with our user stories at that point.

We performed the testing with each other by rotating our group members. One member from each group stayed and presented the code and features

while another joined from a different group to test/review.

After we started testing we also created a testing/code review document where we placed a few key points to discuss during the tests and documented our findings. We usually found a few bugs but code reviewing never really yielded much of value.

B:
We should have had a structure regarding how testing should be performed from the first sprint. A standardized testing-template ensures that the team members test the code uniformly which minimizes variation in the testing. Testing creates value for us developers but if done correctly it would also generate value for our customer and other stakeholders since it boosts quality and spreads knowledge among group members.

In another project we would have wished for the code review aspect to play a larger role and have more impact, especially considering the possibility of handing this project over to some IT department in the future. The person that performs the review should also know good code practises.

(A → B):
Since the code reviews created less value for us than we would have wished for, we would like to solve this by doing more research regarding code conventions as well as documentation standards.

Structuring the testing/code review document together, already from the beginning of the project, and then letting this be a living document that changes as the project evolves.

A way to ensure the best code reviewing could be to have role specifically for reviewing which then should be someone who has a good grasp on best code practises and how to write structured code.

We would like for the testing have more value for other stakeholders in the future. This could be achieved by learning from the code reviews and making sure that there is enough time each sprint to fix possible errors/bugs and if needed restructuring.

5. **A:**
   We first integrated KPI:s approximately halfway through the project and it felt as more of an afterthought from our point. The KPI:s we used were: Amount of tasks completed, percentage of tasks completed, number

of uncompleted user stories, and team member satisfaction.

Amongst them there is one we felt was least useful; number of uncompleted user stories that sprint. This was a very bad indicator of performance since the reason a user story was left uncompleted varied a lot. Sometimes it was a minor bug or a poorly written acceptance criteria and sometimes it was because only half of tasks were written in the first place.

B:
For a future project we would need to begin the project with creating meaningful KPI:s that is useful in tracking the progress and efficiency of the team.

(A → B):
In order to come up with relevant and meaningful KPI:s we should do more research beforehand, discuss with our supervisor and perhaps even discuss with the PO. Altogether this builds a foundation for the conception of better KPI:s which we could use actively to improve our efficiency as a team and as developers. It would also help if we set our KPI:s as early on as possible in the project.

2 Social Contract and Effort

1. **A:**
   The social contract has been mostly static during the project and we never really paid much attention to it. The one edit we made to it during the project was adding the KPI:s. We agreed on the structure and content of the social contract during the first two weeks of the project and then we never really looked at it again. It came in to use once when the group felt that one of our team members was performing below the team's expectation. Instead of following the social contract by first warning the member and then having a discussion during a meeting, we decided to contact Håkan to see what he had input on how to handle the situation. When a meeting was booked with Håkan the person was notified during a sprint meeting. Furthermore, we didn't follow the point where it said that if oneself feel like they're not contributing then they should bring it up themselves. However the situation did resolve itself after our meetings with Håkan and our own discussions after these meetings.

The way we tried to prevent such problems from the start was by doing pair programming. We were two different groups that knew each other prior to the course and one that did not know anyone in the group before. Therefor it was decided that one IT student and one I student should work together and the odd one out got to choose a group to work with. We tried to scramble these groups every week whilst still keeping the same premise of not programming in groups where we already knew each other.

B:
In a future project we would like to come back to the social contract more often, updating it as time goes by or if a situation arises where we realize the social contract does not work. We also believe that things such as testing and code reviewing should have been a more central part of the social contract.

We would not include or reword the point *"Om man själv känner att man inte gjort så mycket så ta gärna upp det själv och be om hjälp."* since it is very difficult to bring up one's own faults. This point works better when people know each other in a group but does not in the situation we found ourselves in. However, it is a good point if everyone is comfortable with lifting their problems with the group and therefore it should probably be added to the social contract somewhat later in the project, if at all.

(A → B):
When a situation arises such as the one we encountered where we did not follow the social contract, we should realize that our fear of creating conflict is not a fault of our own, rather a problem in the social contract as it stands. In this case we should go back and change what was previously written, solving the issue in a better way.

When we bring new processes into the project such as KPI:s and code reviewing it would be best to decide not only the process as we did, but also how we should handle this in terms of the social contract.

In future projects, we should take into account the group dynamics and how well members know each other beforehand. This affects both how the social contract should be formulated and which points are key. The contract should also be dynamic and be able to change as the group dynamics change.

2. **A:**
In the project we never logged our hours, therefore we never had a concrete understanding of how much time everyone was putting into the project. However we did get an estimation of how much time each member was

willing to put into the project each week by what effort estimation they gave. Because of external factors such as other courses to study, the relative effort of every individual fluctuated heavily. This also lead to the value delivered varying from sprint to sprint.

B:
If we were to log our hours, then we would have a better grasp on the relative effort of our team members. We would also be able to relate the time spent to the value delivered to the customer and track our efficiency as a team and how it evolves during the project.

(A → B):
If we were to start a new project in the same team, we believe it to be valuable to log hours. Perhaps we should have started doing this somewhere during the project when we became an established team. However if we were to start a new project with a new team, logging hours can easily turn into a method of finding faults in other people where they do not exist thus leading to unnecessary friction in the team. Therefore we believe logging hours is more applicable in established teams.

Another thing that could be done is to evaluate how the effort estimations worked in the end of the week. By doing this we could get more accurate estimations and better feel for time spent with regards to what was delivered that sprint.

3 Design decisions and product structure

1. **A:**
   We created a web application instead of what the customer originally wanted which was a desktop application. This allows the application to be cross-platform without much hassle.

   Our mission was to find a middle ground between our PO:s direct translation of paper-based forms to digital ones and what was possible/appropriate from our standpoint. The digitalization of the forms our customer work with every day meant that a lot of new features could be added such as input validation and ranges. We tried to communicate, without too many technical terms, what was doable and what information we needed from them in order to create a valuable application.

Another design decision that we made was the use of a postgreSQL database. At the start our customer wanted the data saved in an excel file, but we realised that this was because they wanted to easily view the data and did not care how it was stored. Therefore we opted for using a database for storage and then the ability for the users to download the data to an excel file at any time they wanted.

B:
Although we used hashing to securely store the passwords, an oversight was writing the backend as a http server. Instead of this, we should have created a https server. This would have simplified many of the security needs the customer had. Especially considering their wish to store personal information such as birth dates which turned out to be a security problem to such an extent that it could not be fulfilled in the time frame we were given.

(A → B):
Besides writing a https server instead of http and researching this prior to starting the project we are satisfied with our design decisions during the project.

2. **A:**
This was something that we actively ignored for long parts of the project. We didn't really have time to document our code so much and instead focused much more on learning about React and working with the iterations every sprint. It was always on our todo list to document but we never really started doing any big documentation of the codebase. At our height of documentation we commented some complicated functions in the code.

B:
In a future project we should have used overarching diagrams such as a component diagram to update continuously during the development process and also documentation about how the product is structured and what methods does what and how the whole package interacts.

(A → B):
We believe that the best way for us to ensure good documentation would have been to have started with the documentation at the beginning, maybe even in the planning phase, so that we structured our code after the documentation and not the other way around. Another thing is to ensure that the documentation is continuously updated. In this case we could

also assign a role to a team member for them to be responsible for the updating and creation of technical documentation.

3. **A:**
   We documented far too little of our code and our application. The documentation has been minimal and our biggest flaw this project.

   B: We should obviously take documentation more seriously, especially since there's a possibility of this application being handed over to the customer for someone else to maintain. An extensive documentation also aids when we're searching for bugs and looks more professional.

   (A → B): We believe that creating a role for responsibility of documentation as well as doing research on documentation in React would be a good starting point for ensuring higher quality code documentation.

4. **A:**
   We started doing testing and code reviewing at the end of the project (the last 3 weeks). Another thing that we did was to do pair programming during the whole project which helped by creating an environment where the code was discussed daily. Since several of the minor forms looked alike, a lot of code was re-used. If the initial code was of good quality, it could be re-used and the quality would be assured.

   B:
   What we would prefer is if we were to perform our code review based on existing code conventions for React projects. These standards should also be followed to as much extent as possible when developing.

   (A → B):
   The relevant points are previously discussed in 1.4. I.e more research before the project begins.

4 Application of Scrum

1. **A:**
   During the whole process of this project we have discussed whether we should have multiple roles or not. We ended up primarily with using two roles, scrum master, and a lead designer with the focus of having the overall design role. Our motivation for this decision was that we primarily wanted all of the team members to be able to work on most parts of the project and with different teams each week. Therefore we made the decision to only have roles where we needed some more high-level decision and overseeing. Since we organized in with smaller teams we didn't feel the need for more specific organizing roles beside the scrum master.

   The impact of having these two roles have been great, first of all scrum master relived a lot of the more organizational parts of the project such as supervisor meetings, tracking our progress during sprints and having an overall view of the entire project. This enabled the rest of the team to work more efficiently with deliveries for our product owner since they could focus more on writing code. Second of all it was useful to have a lead of design during the weeks we worked on design. This was especially useful when we were working on the layout of the forms since the lead of design knew the CSS and could have more of an overview. Although all members of the group discussed design decisions, the head of design relieved some of the workload for the other teammates when working on CSS.

   B:
   Since one of the goals with the project was that everybody got to learn most parts of the project, mainly React but also CSS and Postgres. By working in different teams we think that we approached this in an effective way. If our goal had been to be as effective as possible and increase specific knowledge about each part of the project it could be more efficient to have the team members specialize on different areas such as front-end, back-end or design. Also the complexity of the project determines how much specialization a project requires. This would've required more designated roles.

   (A $\rightarrow$ B):
   If we were to be more specialized in our next project, a way to go about it could be to have teams of two people that focuses on one area and does it well. This type of process is not suited for all kinds of projects and we felt like that was the case for us. For example, if we implemented the more specialized way of working and some only did CSS/Design, there would be no tasks for them to do until half of the project is done.

2. **A:**

Sprints: Our sprints lasted for one week. In the beginning of each sprint we had a planning meeting and in the end of the week we had reviews. The sprints enabled us to have a clear focus on specific user stories each week. The nature of the sprints is also a very iterative approach that ensured we created the maximum amount of value for our customer since they could give feedback each week on what we've done and we could go back and improve it for the next sprint and so on. By the end we started having midway meetings which was extremely useful for the team since we could track our progress better.

Sprint Reviews: Made us work to finish our sprint every week, so that we had something that we could show and something that they could give us feedback on. The feedback and reflection each week helped us plan each sprint better and improve from sprint to sprint. During our review we also checked our KPI's.

Sprint plannings: The sprint planning meeting was of great importance for us in order to structure the upcoming sprint. At the meetings we developed the user stories and tasks. This enabled us to have a clear overview of what to do in the sprint and after the planning was done we could get started immediately working with the program. We also did effort estimations for the user stories that helped us better plan the sprint and from week to week we also saw an improvement of our estimations.

User Stories and Tasks: User stories was a very effective way to split up the process to more focused parts so that we could work on certain areas for each sprint. Doing this makes the sprints less overwhelming since you can see what needs to be done in words. Tasks was also very useful, making the user stories less overwhelming. By having both tasks and user stories we could more easily track our progress each week. Our user stories and tasks got better as we became more comfortable working in that way but also as we got a better overview of the whole project.

DoD: Our DoD included the following points: 1. Runnable code on own machine 2. Has been tested by at least someone else 3. The User story is completed 4. Code has been reviewed

Criteria 1 and 3 was not a problem to follow from the start and we had a routine of doing that. Criteria 2 and 4 improved a lot when we actually began testing and code review in a structural way. Although we mostly followed the DoD we didn't iterate it over time nor did we consciously think about it throughout the sprints.

KPI:s Our KPI:s included the following points: The KPI "Team Satisfaction" impacted our work in the way that it helped us to review each member's well-being, all members had to give an explanation as to why they were satisfied or dissatisfied with the past sprint. Based on that we could better reflect on the sprint and pinpoint some things that worked and some that didn't. Regarding the more statistic KPI:s, completion of user stories and tasks, we had a limited amount of data points and we didn't really reflect upon them or use them to track our progress based on that in a more structured way.

PO meetings: Made it easier for us to understand what they wanted to have done and therefore it was easier for us to set up our user stories tasks.

Testing: Ensured a higher code quality in our work. For the testing we used a testing protocol which was useful ensuring that we used the same standards each week. This also enabled us to track our progress better since we also documented the findings.

B:

Sprints: From the start of the project, there should have been mid meetings in order to track the progress during the week and to make sure no one is falling behind during the sprint. The optimal approach, as we see it and how we mostly did it, would be to first decide which tasks and user stories to work on in the trello. Then put the selected one's under the development column on the scrumboard and start working with them. Later when it is done, move the tasks to the testing column and repeat the process. This enables every team member to track the team's sprint progress by watching the trello online.

Sprint Reviews: Ideally we should firstly check how far everyone has gotten in order to track our sprint in terms of deliveries. After that we should check how the team morale has been during the week, if it has been a stressful or a calm week etc. Then we should reflect on how to improve for next week. An example of this was during one sprint review one of our team members who, had worked alone, let us know that his overall morale had been lower because it was boring to work alone compared to working in a group.

User Stories and Tasks: The user stories and tasks worked well in general. In an ideal world the tasks should be specific, and correct, so that the team knows what needs to be done. Also the tasks and user stories/acceptance criteria for design could be even more clear so that the design isn't based on everybody's intuition but rather on what the PO wants.

DoD: The DoD might not be applicable in some cases and an idea to handle this could be to have different DoD:s for design, features and back-end. The DoD should be iterated and modified so that it fits the specific project and criteria.

Testing: In an ideal world there should bee a testing protocol from the start of a project. There should also have been a structure of how to deal with the findings during the testing, when to solve the bugs found etc.

(A → B):
Sprints: For our next project we would like to be better at being structured with trello since it is very easy to get into the programming and forget to keep the scrumboard organized. Other than that we should have started midway meetings earlier since it was good for team morale and efficiency. We could discuss how it went and also use the time to give each other tips on how to solve the current issue. During the actual sprints we thought that pair programming was a good choice which we will continue doing for future projects.

Sprint reviews: We have been open about how we have felt in morale. We also tracked our progress by the end of each sprint and when we have not delivered as much as we wanted we have discussed the cause of it and what went wrong. We should have been better at using our reflections regarding issues in order to improve and excel. Other than that, we are satisfied with how we've conducted our sprint reviews.

DoD: We should have worked with our DoD more closely and iterated over it to adjust it dynamically based on how the project evolved. Furthermore we could've done more research on what actually makes a good DoD and used the experiences from others even more to perfect our definition.

User Stories and Tasks: To achieve B, an improvement could be to be more thorough when writing tasks and making sure that they are actually correct. A way to ensure this could be to spend a bit more in researching each problem for the task that may take the planning a bit longer to do but it will be much easier for the team to work with the tasks later.

3. **A:**
Our PO was from the neonteal department at Umeå universitetssjukhus. Our contact with them has mostly been weekly meetings with occasional mail exchanges for smaller matters. During our meetings we first showed what we had done during the sprint, asked for questions, and checked if they had any feedback. The feedback was noted and on friday's (sometimes monday's) we put the feedback in the backlog in order to not forget it. As of our prioritization on fixing the feedback it always depended on

if it was a major change or not. Minor changes such as changing names on labels is an easy fix while larger changes such as a new layout could be prioritized when we had time for it. One re-prioritisation we did after a session with our supervisor was that it was better to work on functionality instead of design. After this we started focusing more on these parts of the project and neglected all design and layout. We also discussed this with the PO in the last sprint meeting and we agreed that it was a good idea.

B:
Our communication with the PO can definitely improve in many aspects, one of the major areas to improve is how we prioritized our user stories. Better prioritization of user stories form the start would lead to more focus on functionality. An example is to make sure that there's a login page and that bugs do not occur rather than great layouts for our forms. For example PO needs a login page in the final product, therefore it should have been of equal priority to the minor forms and should not have been done in the last sprints.

One should note that an improved reflection could generate more customer value. If we for example can improve our way of working, both as a group and with the customer, we can deliver more value in the same amount of time.

(A $\rightarrow$ B):
To improve and reach B we should've explained for the stakeholder more in-depth on the agile process and how we as a team can (and should) work on different parts of the project at the same time. I.e using vertical slices. The reason we believe this could be beneficial is that the PO had more input on the design and layout aspects and didn't have the experience to give feedback on the programming aspect. In this way the stakeholder could get more information and a better understanding of the project so the prioritization of user stories could be even more optimized and effective. For example, instead of just adding more forms that are similar to each one after another we could've added the "download to excel" function which is a vital, and a different, part of the application.

4. **A:**
   In the beginning of the project we could conclude that there were a lot of new tools to learn and therefor we earmarked time to learn them. Our project was implemented in React. Some of the team members was familiar with GitHub from earlier, mostly the IT-students.

The tools we used for our project was:

- Github as version control
- vsCode as editor
- React as front-end framework
- Express as back-end framework
- Trello as scrumboard
- Figma as mockup-tool

The way we developed our expertise in each tool was different since we all had different experiences to start with. For example the IT students were very comfortable using vsCode and github already so we made a #help channel in our discord server where one could ask questions. This channel was also very useful for react programming, as was Youtube and stack-overflow. Another useful thing was the setup of a guide on how to start the project, which packages to download, etc.

Another learning practice we put into use was to split up the IT students which had more experience with I-students in order to make sure that we spread our knowledge as good as possible in the team. By the end this was not necessary since everyone got a good hang of it.

For the trello the learning curve was similar for most in the group since we hadn't really made that many user stories before. Therefore our learning process was mostly through trial and error where we discussed how the week went after each sprint and looked for improvements in our tasks and user stories, making them more accurate and easy to understand.

B:
One improvement would be more documentation so that people who are new to react gets in to it more easily by reading how it works. This is even more important when we switched groups and focuses each week and had a lot of code reuse. This would enable our team to learn how our program works faster.

Another improvement we found was that sometimes when people needed help no one had time to do so in that moment, in an ideal world we would have had a better "help system" to follow.

(A → B):
By communicating and setting stricter rules about documentation of the code so that documentation took place on an ongoing basis. This would help the team members that are new to a certain part of the program understand it better and faster.

By setting specific help-times the main benefit would be that the group that needs help knows when someone can help them. This would also enable groups to better plan their time when they are in need of help, for example we sometimes just sat and waited for other team members to log in and read the help channel.

5. **A:**
We used all the slides and lectures to build a good base of Scrum and the Agile process. If we felt that we needed more knowledge we researched this using google and used other sources. As we mentioned before; more research on a few topics where the DoD is one of them, would've been beneficial and useful.

We weren't aware of any other literature provided by the course examiners.

B:
We should have read all the course literature as well and made more thorough google searches since we've realised in this report that there was a lot we could've improved just by researching it more.

(A → B):
We should've paid more attention to the canvas page in order to find the literature and read it.

We would like to note that the canvas page was unstructured and hard to follow which made it difficult for us to find all the relevant information. Just some feedback to improve next year's course :)