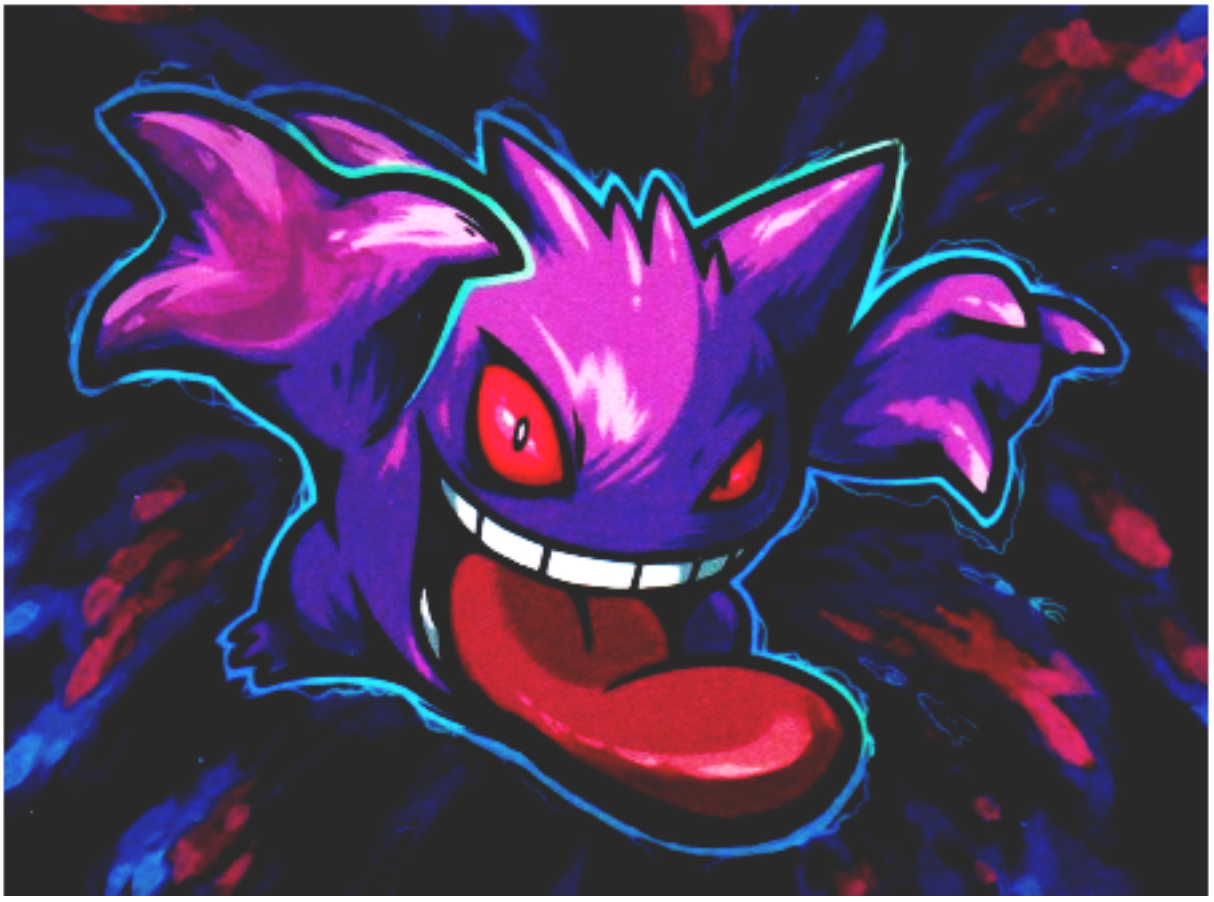
FINAL REFLECTION



DAT256

Grupp Gengar:

Jesper Adolfsson
Daniel Aryan
Hampus Ekberg
August Sölveland
Anthony Tao
Henrik Tao

Chapter 1: Customer Value and Scope

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

A: We started looking into potential project ideas and decided on an initial project scope relatively soon after the project was started and the group formed. After a group discussion we decided upon developing a platform that connects people who, for lack of time, disability or other reasons, do not properly recycle their Pant[1] with people who are willing to collect Pant to earn money and contribute to a more sustainable society. In order to make the platform easily accessible we also decided to make it as a mobile application. Given the limited amount of time and resources we had to develop our platform we decided to only develop the platform as an android application and not IOS.

The initial idea included functionality for communication between the two parties, functionality for payment between the two parties(which would allow for us, the owners, to make a cut) as well as a support function which would be necessary to keep this ecosystem up and running. After extensive discussions, the group agreed that the time necessary to implement these functionalities would not be proportionate to additional value added for our customers. Hence, they were prioritized lower than other features such as integrating google maps to the application, creating a user rating system to ensure the quality of transactions and ensuring that the interface allowed for a good user experience.

In the beginning of the project we had a lacking knowledge of android development and how it differed from the programming we were used to. Therefore we didn't really know how to assess how difficult it would be to realise our initial project scope. This led to some relatively optimistic assumptions being made about some parts of our scope. In the initial meetings we basically just tried to come up with as many functionalities as possible that we wanted to include in our project scope, with a rather vague idea of the difficulty and effort required to implement them. We did however prioritise these functionalities based on estimated customer value and decided on a set of base-functionalities that would create the most customer value.

As a result our project scope has been a bit fluid and unclear throughout most of the project, as we had not decided what was to be included in our MVP[2], but instead started working on the product without more than a rough view of where we wanted to end up. While we don't necessarily believe that it is wrong to be working under these kinds of fluid conditions in a project as open as this, we also believe that this way of working requires a more extensive and structured review/planning-process than what we have had (more on this below). During one of our early review sessions, however, we did sit down and really reevaluated our scope by trying to redefine what we wanted our app to be and what we wanted it to look like. By doing this we gained a clearer picture of where we wanted to end up. Unfortunately, we did not reevaluate our scope to this degree systematically. Doing so would probably have helped us gain an even clearer picture of what we needed to do and might have resulted in an even better end-product. In the final couple of sprints we finally dealt with the problems relating to our planning- and review process (more on this below), which also led us to look over our planned scope once again. In the end we decided to narrow down our scope as we had a limited timeframe and

[1] Refers to soda,beer or other type of liquid containers that can be recycled (by giving to stores in Sweden) for money
[2] Minimal viable product

as we wanted to focus on developing and improving the most crucial aspects of our planned scope that created the most customer value.

B: In future projects we will implement a more structured and formalised process for setting and reevaluating/updating our project scope. We will also relate every part of our project scope more closely to our business model and will make sure that every addition to the project scope really creates customer value. These are our ideas for how to implement these changes:

A -> B: First of all, in cases such as this when we are dealing with uncertainties when going into a new project (in this case uncertainties pertaining to android development), we will hold initial Spike-meetings in order to deal with these uncertainties straight away. This is important as we would like to have the best possible starting point as we decide our initial project scope. When these uncertainties are dealt with we are in a better position to make better estimates and formulate a more sound initial project scope.

This leads us to our second idea, a more extensive planning period where we decide on our initial project scope and build a clearer picture of what our MVP is to include. At the risk of sounding too waterfall-y, we have come to believe that the initial project scope is of utmost importance throughout the entire project. This initial project scope will, to some degree, set the bar for the rest of the project. An unrealistic initial scope will require much effort to set right later on, and a scope not properly founded in customer value might create confusion later on. Therefore it is important that the initial scope is properly planned in the beginning and that it to some degree reflects the project as a whole. However, that does not mean that the scope should remain the same throughout the entire project. As seen in the agile iron triangle below, the project scope can really only be estimated in agile projects.

The initial project scope will undoubtedly have to subjected to certain changes, as it is not possible to perfectly plan the entire project from the start. During the course of the project mistakes will be made, requirements will change, our understanding of the effort needed and of what the customer values will evolve etc.. Therefore the project scope will need to be fluid and changeable, which leads us to our third and final idea.

Lastly, in future projects we will implement a formal structure for reevaluating our project scope during sprint-review meetings. The product owner and their thoughts will be closely integrated into these discussions, either through the scrum master or by letting the product owner attend these meetings in person. These discussions will closely relate to customer value as well as what conditions on- or assumptions about the project that has changed since the last review meeting.

2. **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: At the start of the project we mainly focused on how we could create a unique and valuable product. Before deciding on a specific idea our success criteria was mainly that we wanted to create an application that fits inside the mobility-sustainability borders and solves some issues that people have today. When we decided on making Pantad the criteria became more specific. One of our criterias was that we wanted to make sure that the application included everything required to post an ad and getting a response. It was also important that the people who were interested in responding to these ads got all the relevant information such as location, distance, value and who posted the ad. Creating something that could generate money was also important. A lot of energy went into discussing how to monetize this tool. At first we thought that it would be a good idea to act as the middleman and decide the cut for both the collector and the owner. We then realised that it would be much easier to either take a small cut for posting an ad (like blocket.se) or generate revenue by working with companies that have can recycling centers at their stores.

As the project developed we started to both realise and accept that this course wasn't about the final product. It was about the process, how to work together as a team, how to continuously deliver value and iteratively improving this. Then our success criteria started to shift more towards us and less towards our product.
We wanted to make sure that everyone knew what they were going to do at the start of each sprint, that we had alternatives if something wasn't estimated correctly, that our tasks were independent and so on. We started to really improve on our process and the result was more and better deliveries. So by putting our process at the center of attention, our initial success criterias relating, to the product, were automatically fulfilled.

B: In future projects, we will make sure to include process-oriented success criterias from the start. We saw a lot of improvement when we shifted our focus from being entirely on the end product toward being more process-oriented. By focusing more on the process and what the individual team-members were doing more progress was made, which had positive effects on the product. Therefore we will focus on setting up success-criteria that ensures that all of the team members deliver their share and that they maintain a sustainable pace. However, that does not mean that no success criteria relating to the product will be set up. Some success criteria relating to the product may still be formulated in order to create a clear picture of what we're aiming for.

A->B: In the initial stages of the next project, in connection to the extended planning period described above, we will spend more time discussing and setting up success-criteria relating to the team and our accomplishments. Exactly how these criteria will be formulated is something that the team will decide together, but they should track how the team member as well as the project as a whole is progressing in concrete terms. We will also discuss and set up more abstract product-oriented success criteria through discussions with the product owner, where they describe roughly what it is that they expect from the end-product.

3. 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: When formatting the user stories the group followed a standard pattern which is: "As an *A I want to B, (in order) to C." An example of a user story would be: "As a donor I want to get a push-notice whenever an ad is claimed by a recycler so that I know, without having to open the application, that my ad has been claimed."* The team estimated and assigned effort points to each user story relative to each stories' complexity and time consumption. The effort points then gave the team a direct overview on how to plan each sprint regarding which and how many user stories to select in order to match the team's velocity in that particular week. Apart from effort points, the user stories were specified with acceptance criteria and definitions of done. The user stories were then further divided into smaller tasks in order to clarify who does what, to reduce the risk of misunderstandings between the team members and to simplify the process of moving cards on the scrum board.

The team wrote the user stories together after discussions about what core features the application should have and what value it gives to the target customer. However, we realised early that our user stories had to be broken down because they were too broad. Some user stories were more like epics, which caused uncertainty when starting to work with those stories. Slicing the cake vertically was difficult in the beginning which caused several user stories to be codependent and resulted in some members having to halt their process until other members finished their user story.

Effort estimation on the user stories was a difficult task for the team. In the beginning of the project it was not accurate which was one of the reasons why the team failed to deliver in the first sprint, but towards the end we believe that we managed to estimate the user-stories good enough to match the team's velocity.

B: For future projects we now understand the importance of properly vertically slicing the cake to clearly separate the user-stories and to prevent user-stories from getting tangled together, so that each story can be completed independently of each other, which will improve the efficiency of the team tremendously. The experience we gained as a team throughout this project will definitely benefit us in future projects as we have learned how to properly formulate and break down user stories.
We will also look back and reevaluate our user-stories more often (discussed in more detail in question 13).

A->B: When creating user-stories it is essential to think about the underlying structure of the feature related to that user-story in order to avoid user stories getting tangled together. In our case, for example, a user story about creating a rating system for users was formulated and estimated even before a story about user profiles had been implemented/thoroughly considered, which led to some problems. Therefore, when user stories are being constructed, we have to try to briefly think about how the user stories relate to the finished product and if the stories really are independent, to prevent codependency. When the user stories are then divided into smaller tasks it simplifies the process of estimating the user story. The smaller the tasks the team manage to break down, the easier the estimation process becomes. Additionally, as the estimation process will get easier with time, it is crucial not to spend excessive amount of time trying to figure out the perfect estimation. Instead, it is more important to reflect and learn from the previous inaccurate estimations. It will take some time to develop a sense of how to estimate well. To make the estimation process easier, apart from task

breakdown, acceptance criteria and definition of done, formulating a description and the purpose of the task would be something worth considering. Additionally, trying agile estimating and planning techniques such as Planning Poker, may further improve the team's structure and accuracy when estimating stories.

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

A: In the initial stages of the project, we had not decided on formal acceptance tests for our implementations. Part of it was due to not knowing exactly what API:s and implementations were required, thus we did not know what tests were needed either. Another part was not fully understanding how an android application functioned. A suggestion was using property testing. However, we later realised that our application mainly consisted of simple graphical functionality for controls (e.g. changing a view with a button), API integrations from third parties such as google and firebase, and helper methods with generally low complexity and functionality. Therefore property testing did not fit in for most of our implementations. Although we did test and make sure every new implementation did not cause any compile errors or runtime errors, we did not have any formally written acceptance tests. This meant that most of the implementations were left without any formal acceptance tests for the first sprints, and the definition of done criteria for those user-stories were not completely fulfilled.

Because we lacked ideas on how to properly test the graphical implementations, we decided to consult our supervisor, who suggested to use an interaction protocol, where we document the main flow and exception flow, step by step, of every new implementation as formal ways to test every functionality. The interaction protocol can be found here. The protocol was valuable for us in that we could efficiently and easily run through the protocol of both new and old functionality (in case the old ones were affected by new ones), and check for bugs and errors, saving time and effort. With a proper way to test, we were also more comfortable with our definition of done.

In the later stages of the project, we used pull-requests which allowed team members to easily review and test each others code. This way, we could also identify errors which occurred in applications and devices different from the creator which further helped reduce the amount of errors.

Another type of informal testing we had was having a user (our supervisor) run through our application. We had not thought of it as a test, but this way, the potential user could give feedback on how he expected it to work and what held value for him. After every sprint, in the monday meetings, we showed how the current state of our application was, and the user could give his opinion on what he thought was missing and what was a valuable feature or not. We documented these tests and thereafter discussed whether some changes were necessary and whether we had time for them.

B: For future reference, we think it will be advantageous to initiate the project with some research around what kind of implementations our application mostly needs, for example complex algorithms or simple graphical functionality, factors that could help us determine what tests are needed (as mentioned in question 1). Then follow up by discussing what some optimal ways of testing could be, instead of immediately implementing without a proper definition of done. If we really don't have any idea about what kind of implementations are necessary, we could first start implementing and then also have a user-story dedicated to finding ways of testing these implementations, in order to be able to finish user-stories in the first sprints. Generally, we should create more user-stories for the development team, e.g. to make sure the whole team understands how the application works as a whole, in order for us to better identify and understand errors. Pull requests should also be used right at the start, as they proved to be a useful tool for testing and ensuring the quality of each others code.

Through this project, we also realised that external testing such as demo testing with feedback from a potential user helped us gain a better idea of what held value and not. This should obviously be a more intentional and planned testing, with more users to test than just one, and with a specific flow to follow.

A->B: As mentioned above, we will research about ways of testing, preferably during the initial planning stages of the project. This can also be done either before the sprint starts, or after as team-related user-stories, in order to get a better idea of what tests to use and what that gives value. To improve our external quality (includes usability, robustness and reliability of the application), we need to make better use of user based testing, which we need to plan for. Some improvements include having more users to test. In our application, we had two different roles for the users, recycler and donor. It would be optimal to have more users and also specifically select users with the “real life” roles of donor and recycler in order to test both roles for optimal result. We could have questions ready for the user to answer, or maybe have the user freely give feedback about the application (depends on the type of testing).

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

A: Initially, three KPI:s were decided upon:

- A “Feelings table” where each member of the team evaluated their levels of Motivation, Productiveness, Stress and Team communication every week.
- Test Coverage, where a high amount would lead to easier detection of issues created by changes. It would also serve as a further guarantee that the algorithms implemented were correct.
- Code Quality, through using the tool Sonarqube, which would increase insight into code smells and possible bugs, and hence help the team improve the overall quality of the code.

When choosing these KPI:s the group did not have a clear picture of what we were hoping to accomplish by using them, or if they were suitable for our project. This led to a lower prioritization of working with the KPI:s than to, for example, just getting to work on the code. At around the second sprint, the group decided that the test coverage KPI would be hard to implement, seeing as much of the code at that point was generated through working with Android Studio. In deciding not to use it further, we did not replace it with a better suiting KPI right away. We also failed to begin working effectively with the Sonarqube tool during the first couple of weeks.

In the second half of our project, we decided to introduce a Success Rate KPI which would assist us in tracking what percentage of User Stories were actually completed within said sprint. This KPI also proved to be of underwhelming use, and so, it was also discarded at the end of the project. To summarize, the group only managed to use two out of four KPI:s to an extent where they are relevant to be presented. These are the Code Quality and feelings table.

Code quality

Figure 5.1 below shows us the last three code quality analyses done to the code. Bugs, vulnerabilities and code smells were the three aspects of the code analyzed. As you notice, the amount of flaws on the code increase, which is due to the increasing volume of code over the course of the project. However, even though analyses of the code were made, looking at the figure and how the team referred to the analysis during sprints, it did not bring the team nor the project any significant value. To shortly address the bugs, three bugs categorized as “major” by Sonarqube were nullpointerexceptions and a condition which always evaluates to false. The last two bugs were categorized “blocker” and “minor” which were about closing resources and checking values returned from a stream read. All of the vulnerabilities were categorized as “minor”.

Figure 5.1. Code Quality

The reasoning about why code quality as a KPI did not bring any significant value for the project nor the team had to do with the complexity of the application but also mainly the focus of the course. As the product was not the main focus of the course, the group decided on a lower level of complexity of the application. The decision was made considering the main purpose of the course and the agile iron triangle with limited time and fixed cost for the project. Due to a lower level of complexity and a smaller project (compared with real projects at IT-companies) we did not see the value for the team nor customer to implement an application that is modular, code that is free from bugs, vulnerabilities or code smells, etc. As long as the functionality work as expected and deliver the customer value that we expected to deliver then we view it as success. That is why code quality was not rewarding for us in this particular project, which was intended for us to learn about agile development, not software development. However, we definitely recognize the benefits of having code quality as a KPI. In a more complex and comprehensive project the KPI gives us an in-depth understanding of the code, provide guidance to improve the code based on the analysis and then prevent undesirable events happening in the program.

Feelings table

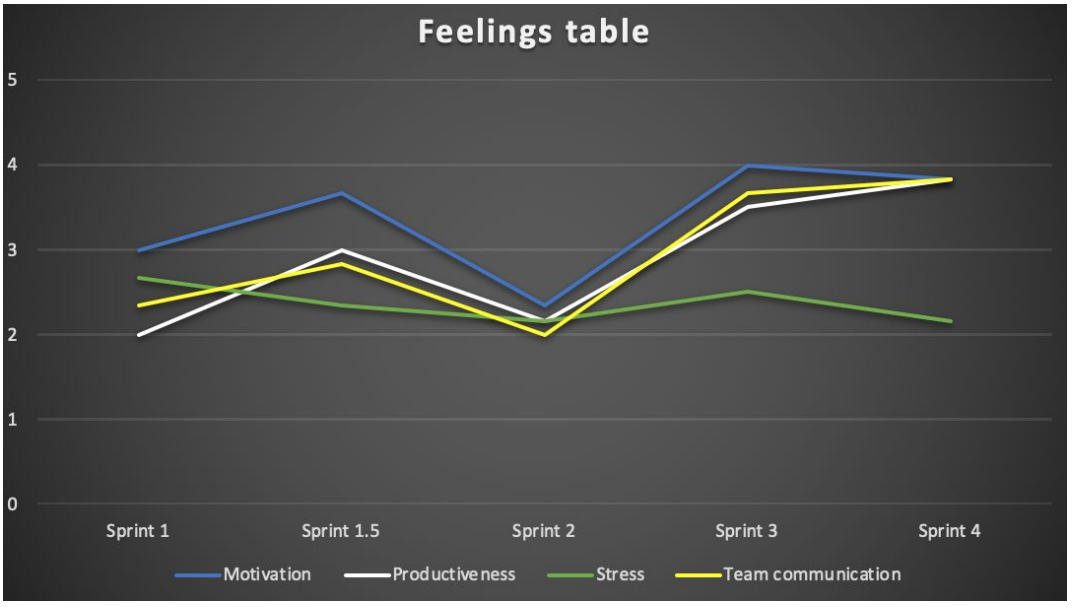


Figure 5.2. Feelings Table

Above is an overview of how the Motivation, Productiveness, Stress and Team communications developed over the course of the project. In each sprint retrospective, all team members scored the different categories between 1-5. This diagram was created through calculating an average of the team score for each sprint.

Traversing through the graphs, from Sprint 1 to Sprint 4, a correlation between the graphs for Motivation, Productiveness and Team communication can be distinguished. During the initial stages of the project, the group was eager to start working on the project but had to spend much time on getting accustomed to tools and clarifying what we were hoping to accomplish with our application. This is reflected in a lower sense of Motivation, Productiveness, Team communication and the highest level of Stress throughout the entire project. As we finally began producing code, the three correlating categories rose in union, and stress levels decreased. At this point in time, however, three consecutive weeks were cut short due to holidays, and as a result the group got slightly disconnected. This can also be noted during Sprint 2 in the graph, where the levels of Motivation, Productiveness and Team communication reached a low point. After this, the group realised we had lost pace and decided it was time to start focusing on the problem at hand, which resulted in a reduced Stress level, while all other categories simultaneously reached their highest points. This pattern continued until the end of the project.

The Feelings table has successfully been useful for tracking how well the project was progressing, based on how each team member felt in different periods of the project. Another interesting learning point that can be distinguished from this table is that a team which communicates effectively is more motivated, has a higher sense of productivity while at the same time feels less stressed. This is worth remembering for future projects.

B: The use of KPI:s has not been successful, seeing as we have only ended up with two out of four which were used continuously and actually resulted in useful information at the end of the project. An optimal scenario would have been that the group realized right away that KPI:s can be powerful tools for gaining control over the project. In the early stages of our project, we had issues with delivering on time, and using KPI:s to increase pressure on delivering continuously during sprints could have improved this. In other words, many of the issues our group struggled with during the course of the project could have been worked out if we used proper KPI:s, effectively.

In one of the last meetings with the supervisors a thorough discussion was held about KPI:s and how they could've helped us establish a better process. An example of a proposed KPI from this meeting was the tracking of when merge requests occurred during the sprints(if they were concentrated to the last days or evenly distributed). This KPI would've most probably worked as an incentive for the group to work and deliver continuously over each sprint.

A->B: To reach a point where working with KPI:s is a natural part of the project a discussion about useful KPI:s should be held in the project's initial stages. In preparing for this, the project group needs knowledge of which KPI:s are available and how they should be used. A proposed strategy would therefore be to dedicate a meeting to this purpose when starting a new project, and during this meeting, lay out a plan for which KPI:s might be suitable for the project in question.
One thing we did do well, and that we would try to implement in future projects, is the choice of KPI:s that cover completely different parts of the project. In this project, the two KPI:s that stuck until

the end, i.e the Feelings table and the Sonarqube tool, had completely different usages. The Feelings table was used to examine the team members' attitude towards the project, which is always of utmost importance in project management whereas the Sonarqube tool was used to examine the quality of our code, over time.

Chapter 2: Social Contract and Effort

6. 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

A: A social contract was drafted during the first group meeting. This contract included how meetings were to be conducted, rules for how the group should handle tasks, how failure to deliver what was promised should be handled as well as the general objective for the project. The initial version of the social contract was reworked about a week into the project, adding a couple of details to further specify how we wanted to work so that nothing was left to chance, and this later version has since been left intact for the remainder of the project. The final version of the social contract can be found by following the link above.

From an early stage of the project it was clear that we had a great group dynamic, and even though we had laid a framework for how potential issues and disagreements were to be handled this never had to be enforced. We can, however, see why this could be useful for other constellations of people, for example in future projects. All members have frequently approached fellow group members with an attitude of understanding, for example when parts of the group has had to study for exams or when the bachelor's thesis was due to be turned in.

A majority of the rules have however been enforced throughout the project, perhaps with an exception to the documentation of meetings. This has been treated as unnecessarily bureaucratic in many cases. Therefore, in the tradeoff between documenting everything and spending time on other tasks that have been seen as more productive use of our time, the latter has often been chosen. Seeing as we have struggled with some issues linked to a lack of structure in the work process, however, one can argue that we should have prioritized differently. Although, we did ultimately manage to create the structure needed to work in a focused manner as the project was approaching its end.

To summarize, we see how social contracts can be of utmost importance to many projects. For our group, most rules became a natural part of how we worked on a day to day basis, with the exception of documentation of meetings. We would say the biggest effect of having a social contract was that the process of drafting it was a great way of bringing the group together for discussions on how we wanted the cooperation between group members to work in a more formal and detailed way.

B: Regarding social contracts there are few things we would have liked to do differently in this particular group setting. If anything, perhaps we should have revisited the document more often to remind ourselves that we should document more while holding our meetings and therefore perhaps minimize the risk of forgetting what we decide upon. This has not, however, been a big issue for the group. In larger projects, or projects where the group dynamics calls for a more formalised and bureaucratic approach, a more extensive social contract will be written that covers more ground than ours did.

A->B: To get even better at working with the social contract, we should have reminded ourselves of its contents on a more regular basis, for reasons stated above. The more extensive social contract

proposed for larger projects, which calls for more structure, could include stricter rules about how the work is to be conducted. For example, this social contract could specify consequences more concretely, and KPI:s could be introduced to track to what extent the team members follow the rules of the social contract.

7. **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: Throughout the whole project, the team has had trouble taking exact records of the hours spent each sprint as we should have done, which was caused by several reasons. The main problem was the spontaneous start-up of the project, which lead to many complications in both the technical aspect and aspect of agile development. Coupled with the break and holidays in the middle of the project, the team's momentum was lost and we lost contact with each other. Additionally, due to the re-examinations, some members had to shift their main focus away from the project. At the time, the team focused mostly on how to move forward with the project and unintentionally neglected the task of recording the exact hours spent each week.

Although we did not record the exact hours, each team member knew approximately how many hours were spent working in the sprints. During the first sprints where we were confused about how to work and did not completely understand how the application worked, the motivation was low and less than half of the recommended time was spent on the project. This is noticeable as we had no deliveries on the first sprint, and less than optimal on the half sprint thereafter. If we were to put an average number of hours spent per person, the team agreed on that between 8-10 hours would be a relatively accurate interval for the first sprints. This improved considerably after the break, as we began solving our issues one by one. What had the most impact was that our sprint startup meetings improved from sprint 3. When we assigned tasks to the team, with improved effort estimation, everyone could better plan their sprint and had a clear idea of what to work on. This lead to motivation and overall work effort increasing. From there on, we estimate the time spent to be around 16-20 hours. These last sprints, most of the user-stories assigned were finished.

B: The goal is to spend the recommended hours from the beginning, and as efficient as possible. We want to record the time spent in relation to delivery in order to improve this. A smooth sprint without any major obstacles, minimal confusion and a clear sight of the overall deliveries, as well as each members' objectives in each sprint would be an ideal situation. At the same time, since we can not expect everything to be smooth sailing, we want to compensate by having better effort estimations by continuously optimizing the effort points and regulating our velocity, depending on each members' schedule and capability after every sprint, to achieve a sustainable pace for everyone throughout the project.

A->B: The team has realised that sprint planning is a big part of achieving success in a project, and recording and planning the time spent is part of that. The problem we had with not recording the time spent could be solved through having a desktop reminder while working, alternatively create a check-list of points that needs to be done, and check it off every daily scrum meeting so that nothing important is left out. The point is to make sure to not forget things of importance when there are many issues to mind.

To be able to spend the recommended hours efficiently and deliver well proportionally to the hours spent, in future projects, we need to identify records where much time was spent, but little was

delivered. We can then analyze what caused it, and discuss a solution. For example, if a team member is not delivering every sprint due to not spending enough time, or if someone is spending too much time to finish the deliveries relative the standard, we need to discuss how to solve it in order to ensure a sustainable pace for everyone. If for some reason one or more team members cannot spend the time required, we would have to change the velocity and lower the scope of the project. Obviously, the optimal situation would be to identify these cases before they happen. If any member cannot dedicate enough time for the project, he or she should state so before the sprint. However, this is not always the case and sometimes, the velocity has to be optimized after seeing some progress.

Chapter 3: Design decisions and product structure

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

A: A lot of time during the final couple of sprints was dedicated towards refining the design in order to make the application as intuitive and easy-to-use as possible. Things such as swipe-functionality and more relevant icons were added during these sprints in order to make it as easy as possible to navigate through the app. We also made the buttons easier to spot and made sure that relevant information was more clearly displayed up front (such as whether an ad had been claimed or not). All of these things were done with customer value in mind. However the assessment of what that makes an app intuitive and easy-to-use was an entirely subjective assessment made exclusively by the team without any external input (Except for small comments from our supervisor).

During the creation of the application we decided to rely heavily on certain APIs, such as Google maps and identity API. These API:s were mainly incorporated mostly for practical reasons, as it was easier and faster to use something fully tested and ready than creating something new from scratch. However, these API:s were also chosen because of their contribution to the customer value. For example, we believe that a majority of our users already will be familiar with Google maps and Google authentication, and that they therefore will feel a sense of familiarity and security when using the map and login method. By using API:s that our users are familiar with, we make it easier for them to understand and use our application.

The usage of firebase also provided customer value since it made it possible to implement features such as push notices. This would have been a lot of work otherwise since it would require us to have our own server and database. Since the user gets a push notice when important events occur the responsibility of continuously paying attention to our application is removed. Our users do not have to check if their ads have been claimed constantly for example, since they will get a notice as soon as that happens.

B: There are a few small changes that would make the application even more intuitive which we didn't have time to implement. An example is a button which links an ad in the pickup window to the map. If a user looks at the pickup window and finds an ad which interests them, they currently do not have a simple way to get a map view of the corresponding location. By implementing this feature both the map and the pickup window provides more value, especially if the user doesn't recognize the address or if there's a lot of ads in the vicinity. In future projects we want to get started on these design aspects earlier in the project, as they contribute much to the overall customer value.

In future projects we would like to take in more external input when designing our product. In this project we tried to make the design support customer value as much as possible, but we only relied on our own experience and created something that we think is intuitive and easy-to-use. There is a possibility that others wouldn't agree or that there are flaws in our design that we did not see.

A->B: In order to retain more external input regarding our design we would like to have more discussions with our product owner (or in this case, our supervisor) about the design. We will also look up examples of good design for whatever format we are creating in. Another thing that would be useful is doing some research/reading about human-computer-interaction to gain a more theoretical perspective. Lastly we would like to have beta-testers whose purpose is to give us feedback about our product. For smaller projects, these beta-testers could simply be friends or friends-of-friends. For larger projects, where the stakes are higher, a more formal group of beta-testers made up by people within our target group would be optimal.

In order to get started on the design aspects earlier on, we will create/prioritise user stories for this earlier on, and in some cases we will incorporate the design into the acceptance criteria for certain tasks, and not just the the functionality itself.

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

A: The only technical documentation we had during this project was comments in our code and an interaction diagram which we didn't really keep up to date. We talked about creating a proper UML diagrams and even generated one at the start. Unfortunately the generated one didn't create much value other than spawning discussions about how the first version of our application was built. Since we decided to use the test project, which was created by one of our members during the first week, we never really started to discuss the project structure from the beginning. Instead the structure was a bit inconsistent and it became the responsibility of the person assigned a task to make sure that everything worked.

This became a problem since it was hard for everyone, especially in the beginning, to understand both how our application worked and where to place new functionality. Even though we knew of this problem quite early on, it wasn't long before it became a pretty time consuming task to create a diagram which showed our current project structure. It also became quite pointless since our project wasn't founded with a clear structural idea and android development was new to all of us. We therefore decided to keep developing without changing too much and instead be sure to help out the other members by explaining the parts that we were responsible for. Since we are a small team of only six members, and we don't have to worry about having a million lines of code, this lack of documentation was still manageable. Although we probably would have saved a lot of time by introducing these early on and make sure to update them.

B: We have all struggled with trying to understand new changes to the project and figuring out how to structure the tasks we were assigned. This is something we would try to prevent a lot more for a future project. The discussion about how the project should be documented in terms of coverage and rules needs to be held during the initial planning phase. If we managed to do this properly we wouldn't have the same need to spend time on explaining code verbally. We would also make sure to continuously update a UML-diagram which would show the overarching structure and the dependencies between classes. This would make implementing new features a lot easier since you wouldn't have to go out of your way in order to understand the sequence in which classes are connected to each other.

A->B: In order to have a good documentation that gives value to the developer you have to start early on, preferably during the initial planning phase discussed in question 1. Creating a domain model is not only a good decision in terms of having a structure to present for someone who isn't a part of the development team, but also because it gives the team more of a structural consensus. This can then be translated into a design model which is less vague and can support the developers, especially at the early stages of development. The consequence of such a diagram is hopefully that A: discussions concerning how features are implemented will be reduced and B: the first features are implemented much quicker and in a proper manner.

10. How you use and update your documentation throughout the sprints

A: Some of the text under the previous question could just as well been placed here.
We decided early on that we should comment the code properly, although this wasn't really followed up or discussed later on. Even though we have quite a bit of comments it isn't consistent and we haven't done it in a javadoc manner. This results in both scarcity and low value. We also tried to improve on this when we introduced the interaction diagram. Although this didn't have enough coverage and we failed to keep updating it.

We decided that we should create a meeting protocol in order to preserve important decisions and thoughts. At the same time we decided that it would be good to have a daily scrum meeting to better our communication and remind members of what needs to be done. Since this was a short and informal meeting that had a lack of discussion we didn't document these. But since we talked (almost) every day we didn't have as much of a need to keep formal meetings anymore. Therefore we didn't have as much to document as before when we used to have two formal meetings each week. The result became that every sprint startup meeting became documented but nothing else in terms of meetings. In full this was a slight improvement despite the fact that the solution didn't produce an optimal result.

The protocols that we wrote helped a lot with dividing tasks and prioritizing. It also came to use during the last sprint when it became harder to follow the agile structure because of the task's nature. Creating user stories for improving the GUI or fixing bugs can be counterproductive and result in a waste of time. When we needed a way to keep track of the things that didn't really fit on our Trello board we could instead turn to this documentation. If we would have started to do this earlier we wouldn't only have reduced information loss, there would also be more data supporting our decision making and memory.

B: For a future project we would set up clear rules for updating and creating documentation. The impact of having descriptions of both code and important decisions is huge and saves a lot of time. Code shouldn't be pushed to our repository without comments because it's a lot harder for the team to understand and sometimes forces the creator to explain it several times. Meetings should also be well documented from the start and the protocol from the last meeting should be reviewed at the start of a new one.

A->B: In order to actually improve on this we don't only need to set clear rules, we also have to take the time required into consideration. These rules should be decided during the initial planning and continuously updated as the project progresses (during retrospective sessions). Each user story should have extra velocity added relative to the time it's estimated that documentation will require. This scales with both the size and nature of the user story. Another solution is to use KPI:s which favor documentation.

11. How you ensure code quality and enforce coding standards

A: At the beginning of the project we felt confident that we would be able to ensure code quality and enforce coding standards by simply reviewing each other's work whenever it was pushed to the repository or after a task was completed. What ended up happening with this mindset was that each person just merged their own changes and pushed directly to master, and then no one else really felt responsibility to review the changes and it went unreviewed. The result of this was somewhat lacking code quality and different naming conventions for which made it less readable and harder to work on the project. In sprint 1.5 we created a user-story for standardizing the naming convention in the code, which helped to solve some of the readability problems.

We liked the idea of reviewing each others code but realised that the current structure we had didn't really allow it. In sprint 3, to fix this, we started working with pull requests instead of directly merging your own code. This meant that whenever someone was finished with a task, that person would make a pull request, at which point someone else went in and reviewed and tested the code, and if it was satisfactory, merged it into master. The structure along with the efficiency of it allowed peer review into the process and it helped us better maintain code quality.

Another change we implemented in sprint 3 was to use SonarQube. SonarQube is a program that helps you by identifying various issues, code smells and vulnerabilities in your code. Since we were late to introduce it and because of the lacking coding standards, there were many minor code smells and issues, which we decided not to focus on in order to still be able to reach our project scope. The most critical bugs were fixed, however.

B: In the beginning of this project we had a somewhat naive approach to ensuring code quality, but the changes we implemented in the later stages had a positive impact on our ability to identify issues with the code quality and in turn solve them. In future projects we would like to continue making use of pull requests, SonarQube or similar tools and have coding standards which everyone follows.

A->B: First of all, pull requests will be utilised from the beginning in future projects in order to ensure quality and to ensure that the entire group follows the same standardised patterns (such as naming conventions). By letting other teammates review code commits we will also ensure that more than one person in the group will be familiar with the changes made, which will help the group maintain an understanding of the project structure. It may also be beneficial to document coding standards (e.g. how a model-type class or a method in general should be named to increase readability), in case they are forgotten or a new member joins the team.

Secondly, we think that some kind of automated quality test such as SonarQube might be a useful tool for identifying potential problems in future projects. Therefore we will likely continue to utilize this kind of tool in future projects. However, even though some bugs were identified and fixed thanks to the tool, we did not properly go through and fix all of the problems identified by it. In future projects we will therefore integrate this tool more closely into the sprint review-process and will make sure to make time for fixing the problems identified by it (for example by writing a user-story for this and breaking it down into tasks during sprint planning).

Chapter 4: Application of Scrum

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

A: We didn't decide upon any specific roles at the start because we were unfamiliar with each others strengths and weaknesses. Our expectation was that as the project went on it would become easier to assign roles and evident which specific areas required someone to be in charge. This was the case for some parts, i.e. meetings where we started appointing a chairman and a secretary. The role of chairman wasn't really utilized because of the fact that our personalities are different. Some are more prone to talk and therefore having a chairman position that rotates would require that the meetings were planned beforehand. This position therefore became floating and its function was partly filled by some members naturally. We made sure that at the start of each meeting, that came after the second sprint, a secretary was appointed. This role worked out well and resulted in more documentation of what was said and decided upon.

The team have not assigned nor used the position of scrum master in the project, which was most likely due to everyone not being entirely comfortable with the role. The members in the team were not overly familiar with the framework and thus felt like they were not up to the task. We have had members who were more dedicated to reminding the team about the different agile practices, but how each practice worked was not properly explained. Although everyone worked with the scrum practices such as creating the scrum board with user-stories, the team felt like more knowledge about scrum and it's practices would be beneficial. Some uncertainties, such as the difference between sprint review and retrospective, are what could have been solved with a scrum-master. The role was neglected however, partly because we had many other issues to focus on, and also because no one felt like picking up the task. Perhaps forcing the task upon someone would have solved the problem, but we feel like that would be unfair when everyone was behind in terms of coding.

Regarding product owner, we have not assigned the role to anyone, the reason why is explained in point 14 where we reflect upon the role.

B: In future projects, having more defined roles would definitely help us structurize the project in terms of responsibilities. We plan on introducing a scrum-master during subsequent projects. As we are a bit more familiar with the framework now (scrum), most team members feel secure enough to fill this role at some point. Also, before the sprints start, we could update everyone on the scrum practices. We also realise the need to have someone who makes sure that meetings follow a set structure and who, besides doing development work, puts their focus towards the process and towards coordinating the team. In several other questions we have discussed the need for more structure in our process, and we feel like there needs to be a scrum master to enforce this.

A->B:
In question 1 we proposed introducing a more extensive initial planning period where the team takes some time to set up the project and to set up a process-structure. During these initial stages a lot of ground has to be covered, as we discuss in multiple other questions. One of the things that will have to be covered during these meetings is deciding on a scrum master. According to some internet-based

sources the scrum master is supposed to be a sociable “people’s person” who is also technologically able and who is familiar with the scrum process. Therefore these are attributes we will look for when electing our scrum master. This person will then be responsible for managing the meetings and maintaining their structure. The scrum master will not necessarily be required to deliver as much code as the rest of the team, but will instead put more time into planning, reevaluating as well as coordinating the team.

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

A: The team used scrum in this project, which is a framework to improve the teamwork. Most of the agile practices that the team have used, if not all, are within the Scrum framework. By working agile we have come across many agile practices such as:

Sprints, Sprint planning, Scrum board, Product backlog, User stories, definition of done, Agile retrospective, Estimations, Customer/Business Value, Daily scrum meetings, Sprint review, pair programming, Agile scrum roles, burndown chart, etc.

The agile practices are not only executable steps such as having a product backlog or perform an agile retrospective but also having an agile mindset. It is about thinking iteratively and constantly referring to customer/business value.

Four of the main agile practices which we felt contributed and improved our teamwork and agile process were sprint planning, agile retrospective, scrum board and daily scrum meetings.

In the beginning of the project there wasn't really any set structure to the sprint-planning meetings. We had just decided that we should meet every monday in order to discuss the project. However, because there wasn't any set structure to these meetings they often ended up being wasted, as we didn't really do enough to plan the coming sprint. During these meetings we did sometimes try to look over our product backlog and reevaluate/re-prioritise our user-stories, but we often failed in creating new, good user-stories or properly reevaluating old ones. (This closely relates to what we wrote in question 1)

Most of these problems were amended to a certain degree during one of the final sprint planning meetings, where we had all realised that there were problems in our planning-process that needed to be attended to. During this meeting we introduced a more formalised structure where we systematically went over our scrum board and reevaluated/re-prioritised all of the user stories in the backlog through group discussions. At this point in time we also had a lot more insight into the underlying structure of the project than we had when formulating these user-stories, which was made apparent when reevaluating them.
The result of this formalised, structured sprint planning meeting was that we all gained a much clearer picture of what that needed to be done, as well as how it could be done. This had a tremendously positive effect on our overall productivity and team spirit, which reflects in the successful deliveries.

In the beginning, the team did not realise the importance of Agile retrospective and the effect it can have on the team’s ability to reflect on what has been done, what we want to aim for and how we will get to what we aim for to improve our process moving forward. At first, it felt like a waste of time when writing the weekly reflections. However, after the previously mentioned occasion we realised the importance of retrospective as a tool. It allows us to look at the problem from different perspectives and get a feeling of “lessons learned” when we move forward. In addition, retrospective resulted in a noticeable improvement of communication and awareness of the project process within the team. Through the teams weekly retrospective in team reflections we have identified problems in the process and discussed how we can overcome the problems in the next sprint.

During one retrospective, the members of the team realised that we needed better communication. Many questions and problems arose during the sprints, questions about pieces of code that other members wrote, questions about the components and structure of the application, etc. That is when all the team members agreed on having Daily scrum meetings to facilitate communication between all members to help each other solve problems and answering questions. We discussed about having Daily scrum meetings in person, which all the members felt would give us the most. However, due to colliding schedules and members living far away from each other (and campus), we decided to have digital meetings on Discord instead. The result of having daily scrum meetings was increased communication between members, everyone knew what the others were working on, what kind of problems they came across and the members who needed help with something got help. The only downside the members felt is that the digital meetings were easily forgotten, if not reminded of.

Another agile practice used was a Scrum board, more specific, we created and used a Trello board. The main issue with trello that our team had was when to move a task from being in process/testing to being done. The team had acceptance criteria and definition of done for all user stories, however, we did not have a formal process of having another member checking the features implemented before moving the user story to done and pushing the content to our master repository on github. Therefore, pull requests were introduced to formally define the last step from the story being in progress/testing to being done. When a member's pull request had been reviewed by another member and the content had been merged with the master repository, the card on the board would formally be moved to done. This solved our problem of when to move to done.

B: In future agile projects the sprint planning should not be neglected as in the beginning of the project. A good scrum planning meeting at the beginning of each sprint is fundamental to everything that will be done later on in the sprint. For example, a non-sufficient sprint planning will cause confusion for team members which will affect the team as a whole in terms of lack of motivation, lack of confidence etc. Thereafter, the quality of the process and efficiency will decrease, which will affect the deliveries for the sprint. Therefore we will introduce a formalised structure for the sprint planning meetings from the start in future projects. It might seem obvious to some, but we have had to learn the hard way that a meeting can quickly devolve when it doesn't have a set agenda. By introducing this structure, in combination with the changes to the initial planning proposed in question 1, we hope to make sure that all members of the team have a clear picture of what needs to be done during every sprint.

Regarding retrospective, in future projects, retrospective should not be viewed as a burden, but as a tool to reflect on what went well, what we need to improve and how.

In future projects, we would definitely want to start using daily scrum meetings early on to prevent lack of communication and provide assistance between team members. Those who are in need of help should be able to get help as soon as possible. If possible have the meetings face to face.

A->B: With the experience we gained during this project, we would definitely put emphasis on sprint planning in the upcoming agile projects. To succeed we believe that all members of future projects has to know the importance of a well planned sprint in order to deliver. Otherwise, one should try to share their own experience with other team members to enlighten them. In order to make sure that the sprint-planning meetings are done more efficiently than they have in this project, we will introduce a

formalised structure for these meetings. Exactly how this formalised structure will be set up is something that highly depends on the nature of the project and of the group. Therefore we cannot propose a definitive structure that will solve all problems here. What is apparent now however is that this structure needs to be decided upon relatively early by the group and that it will need to cover at least user-story creation, acceptance criteria formulation, task breakdowns and the assigning of tasks to individual team members.

Even though our daily scrum meetings digitally went well, having meetings physically would definitely benefit the team members even more in terms of being more available to help each other to solve problems. In addition, physical meetings are less likely to be forgotten. In this project it was not possible, however in future projects if it is within a company with co-workers or students from the same programme, we would definitely strive for having physical meetings instead.

14. 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 this project we combined our sprint review meeting and our sprint retrospective meeting. We held the meeting every Sunday through discord and started with the sprint review, and then held the sprint retrospective straight afterwards. While this was not the main cause of our problems, combining these two meetings might at times have had a negative effect on the sprint review, as this meeting was sometimes a bit rushed. While everybody shared what they had delivered over the sprint, we did not really reevaluate things properly (discussed in further detail in question 1).

We did not have an assigned product owner during this project. There were multiple discussions about whether to make a group member the product owner or not, but in the end we decided not to do this as we did not feel it was necessary. At the same time we can see the benefits of having a product owner that you can discuss things with and get feedback from. One thing that we should have done more in this project is to look back at the business model and to reevaluate what it is that really creates customer value. This is something that we should have focused more on and that a product owner would have helped with. By having a product owner who is focused on what it is that creates value, we might have gotten more valuable feedback pertaining to those aspects of the project.

B: In future projects we will separate the sprint review from the sprint retrospective in order to make sure that both meetings get the attention that they deserve and that both of these crucial aspects of the scrum process get done properly.

We will also introduce a product owner into the project (unless a product owner already exists). By interacting with the product owner more constructive feedback can be gained, particularly regarding the business and customer value aspects of the project.

A->B: Besides splitting the sprint review from the sprint retrospective (preferably these should not be held on the same day), we hope that the changes proposed in question 1 will help us reevaluate things properly during the sprint review meetings.
Regarding the product owner, it is hard to say exactly how the product owner is to be introduced at this point, as that is something that depends on the circumstances of that project. If one is provided for us (as they probably would be in a real business context), we will try to have regular conversations with them and try to integrate them into our process (this is discussed in other questions). If not, one of the team members serve as a product owner. This can be a rotating role, but we feel like it is best that there is always someone who embodies this role and who, while acting as a product owner, takes a step back and tries to view the project from a business/customer value perspective and not just as another member of the development team.

15. 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: The team has during this project used many new tools and technologies. Programming experience varies within the team, that is why some members have experience with some tools which other members do not have. Some examples of tools would be Android Studio, SonarQube, Git, GitHub and Firebase.

We would say that the best practice for learning and using new tools and technologies is by sharing existing knowledge within the team. For example, git and gitHub had been used by the members from the IT programme during previous courses, which allowed them to easily walk the other members through it, step by step, on how to use the tools. Having members sharing knowledge was the fastest and most efficient way of developing the expertise around the tools, it saved time by not having to research from sources which moreover are not necessarily guaranteed to be accurate.

As for tools that none of the members had used before, online researching was the best way of developing the expertise. By searching for the problems that we encountered or questions that we had, we easily found answers from knowledgeable programmers in for example Stackoverflow. Obviously, a sense of source criticism is always needed, we always compared answers from one user to another to see which one is most relevant. In some cases, the answer would be outdated and we had to search elsewhere for a more updated answer.

We also used official documentation for the tools and tutorials on Youtube. The advantages with the official documentation are that we could be certain that almost everything written was legitimate and updated. However, for developers with less experience in reading documentations it may be confusing and hard to understand. Youtube videos has its advantages of being able to present the information visually and audibly to enhance the comprehension for the audience. Yet again, source criticism is important when watching videos, because they are uploaded by individuals who do not necessarily have the correct knowledge.

B: The improvements we would like to make for further projects concerning practices for learning would be using new methods which put even more emphasis on effective learning and cooperative learning. A recommendation from our supervisor towards the end of the project was spike meetings. The idea of assigning time during the beginning of sprints for research to obtain knowledge and develop expertise of tools to use in the remaining part of the sprint, was definitely a great method that the team would like to use in future projects.

A->B: In order to improve the process of learning new tools and technologies we will use one of our best practices which is researching online to learn about better best practices to learn new tools and technologies.

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

A: During one of our meetings with our supervisor (Jan-Philipp), he jokingly mentioned that the group was slowly reinventing Scrum, which he was kind of right about. In the beginning of this project we did not really follow any kind of structured process. Instead we kind of just started. The initial lectures that we had in the beginning of the course served as an introduction to the Scrum process, but it was really just a short introduction and it took us a long time before we really took the insights from these lectures to heart. Over time, we have come to realise the need for a structured approach during this kind of project, and we have come to appreciate the importance of the different agile practises introduced during these lectures (discussed in more detail in the other questions). Many of the changes that we have introduced during the project has been direct proposals from our supervisor, and are therefore likely founded in some literature. However, the group itself has not really reflected about the literature properly during the project, and the changes that we have introduced have simply been natural responses to the structural failures that we have experienced during the course. Therefore our knowledge about what the literature has to say has been (until now) severely lacking. Despite this we will now try to relate some of our proposed changes to the literature:

According to *"Kniberg, H. (2015) Scrum and XP from the Trenches - 2nd Edition",* scrum planning meetings are supposed to have a set agenda and follow a set schedule. This is also what we have proposed to introduce into our planning meetings, as we have identified the need for a formalised structure in these meetings. Therefore we wholeheartedly agree with the literature on this point.

Kniberg also proposes using "Planning Poker" to estimate user-stories in order to prevent the team-members from being influenced by each other when estimating. This is also something that we have proposed as a potential change that would improve our accuracy when estimating user-stories, so we also agree with Kniberg on this point.

Another thing that Kniberg stresses is the importance of "making the team sit together", aka. having physical meetings. While we have also proposed that more physical meetings might have a positive effect on teamwork, we are not sure that it is as important as Kniberg is making it out to be, at least not in as small projects/teams such as this. As we have mentioned above the majority of our meetings has been conducted over Discord, and we feel like this has worked out relatively well for us. Of course this is something that may differ strongly from group to group, but for us these digital meetings have worked relatively well. However, Kniberg also talks about the "self organizing team" in his book, where he talks about how a scrum team that is held accountable for the product they deliver but that is free to self-organize (which includes the freedom to allow members to work from home) tends to work out just fine. We are inclined to agree with this sentiment, given that the team has the relevant knowledge required to efficiently self-organize (which we did not have in the beginning of this project, but that we hopefully do now).

B: In preparation for future projects the team wants to deepen our knowledge about the scrum process and its different components. As some of these components of scrum has been natural solutions to some of the problems that we have encountered during the course, we have no doubt that there is more to the Scrum process that we could take to heart that would further improve our way of working.

A->B: The group will do further research about the scrum process/agile, by properly reading the course literature for this course as well as reading information from other relevant courses.