Pegah Amanzadeh
Elin Berthag
William Hugo
Edenia Isaac Galan
Klara Jakobsson
Elias Johansson
Alva Leufstedt

Final Team Reflection

COBOL

Table of Contents

Customer Value and Scope

The Chosen Scope and Features, for Whom are We Creating Value

A: This project started from our stakeholder’s original idea, she realized the doctors at her department lacked a way of following the patients wellbeing from one meeting to the other and thought of an app that would work as a quiz where the patients could input data regarding different parameters the doctor needs to follow. This data would be later shown to the doctor as graphs, to give them a quick overview of the patient's health status. We started our development process by asking our customer what she wanted and through discussions and suggestions from our side, we came up with a list of functionalities that we wanted for our app in the form of user stories. Our product Owner made a brief priority list and we started with the most important and most expected demanding ones, the quiz and graph functionality.

As there were many members on the team and we happened to underestimate how much time we would need to be done with this main functionality, we started even working on some less prioritized user stories early on, for example the Calendar Page. Later on, our customer told us that other things were more important than some of the user stories we had started working on, which led to a reprioritization to make sure that we would be finished with the most important features at the end of this project. This was of course not ideal, but we are still satisfied with our decision given that it had been worse to keep working in functionality that didn't create any value for our customer. We were also forced to prioritize away some of the functionality we thought of implementing in the beginning, for example functionality for sharing the data with others as well as functionality to track when the medicines had been adjusted and relate this to the graphs. This would have been nice but it wasn't core functionality for our app.

B: The ideal scenario would have been that we hadn't started working on functionality that later on would be down prioritized and of course that we had had enough time to implement all the functionality we thought of in the beginning. But we are still satisfied that we managed to deliver all the main features as well as some additional functionality during the project scope.

Another aspect we would improve is to spend more time polishing our app, by this we mean making it look nicer, working more with the design as well as fixing some bugs, mainly in the quiz part of our application, this bugs are hard to find and appeared if the user was to behave in a really unexpected way but this of course should be corrected and is also a consequence or lack of time and the need of more planing.

A→B: So, how to fulfill this would be to put even more time on asking and discussing what to prioritize with our customer at the beginning of the project and make clearer lists based on this for our development team. In general we should have spent more time on planning during the beginning of our scope, we didn't really discuss how different parts of our application would interact with each other or which design pattern would be better to use. For example in the quiz part we realized the code design is not the best, is not that extendable or modular and we think we could have avoided this by learning more about the Android Framework and planning together with members of the team that had more experience using this development tool. This lack of planning slowed down our progress and made us spend quite some time adjusting the old code to fit with new added features and fixing bugs.

Another aspect we would improve is to work more vertically with our user stories, it would have been better to first be completely done with the prioritized user stories before starting on ones with lower priority. We ended up working more horizontally and this was largely influenced by our effort estimation, one more aspect that needs improvement. We discuss the problems we encountered when doing effort estimation further down in this reflection.

All of these parameters end up depending and affecting each other creating a vicious circle that needs to be broken, since we believed it affected how much value and functionality we were able to deliver to our customer within the time we had. In general we feel it affected our productivity.

As a final aspect we want to discuss our choice of scrum master, we decided the members of the group would take turns taking this role, a different person every week. We wouldn't do the same in a real life work project since we came to realize having just one person in this role had been more effective and beneficial for our final result. We did have a reason to take turns on this role and we discuss the advantages and disadvantages later on in this reflection.

Success Criteria

A: We came to realise that we wanted to achieve different things with this course within the group. Some wanted to focus more on how to work scrum and give a larger space for the reflections, while some wanted to learn the tools we were using and improve their app developing and coding skills. When we noticed this, it led to less frustration because it became easier to understand why we wanted to spend different amounts of time on different things. Also, it led to a more unified view on the course that improved our teamwork.

In this project we also had a goal to develop an app to create value for our stakeholders and their customers. To achieve this we worked as a team, communicated with the stakeholder and learned the things we needed on the way. There were some problems in the group slowing down our work process, this was improved by bettering our teamwork, communication, and being better at helping each other.

The communications with stakeholders helped us to get to know the customers' needs and it helped PO to better prioritize the core functions which were critical for stakeholders. We have been able to implement core functionality as well as add some more features which are useful for customers. We even had planned to make some more extra features but due to lack of time we were forced to prioritize them away. Overall we're happy with what we achieved as it was the things of higher value to our stakeholder.

We learned a lot about the agile processes by using scrum and got more comfortable with it since we started the course. Using an agile mindset helped us to develop and improve our application.

B: For a future project, this could be helpful to discuss our goals in the beginning to create awareness of the whole team's expectations and avoid potential conflicts. We will also use the experiences which we got from agile processes in the future project and continue to reflect on things if we want to improve it. We will also keep up to work on effective communication and teamwork, which is the key to achieve goals in the team.

A→B: This can be done by constructive discussion and efficient communication in the team.

User Stories, Acceptance Criterias and Effort Estimation

A: When it comes to effort estimation it was only done on tasks, not user stories, and in fact it was not really on tasks either. It was done mostly on the aggregate of tasks, mostly those that were left on a specific user story. This caused us to think more horizontally than vertically and also made it difficult to add our efforts to our scrum board. That is also why we have them in separate documents (in the very beginning we only did it by verbal agreement). This affected our ability to show demo functionality after every week, although this was only a real problem during the very beginning of the project and addressed during a supervision meeting. Still it is not a good idea since it is impossible to get feedback on the application which is quite essential in an agile environment. Our efforts were also sometimes inexact, meaning that they would be estimated in intervals of time, for example 3-5 days. This did not cause any noticeable problems for us but it is worth noting. Our user stories were broken down into tasks, and after some confusion during the earlier sprints the tasks got the same number as the corresponding user story for clarity. The process of breaking down user stories into tasks was quite difficult for the team and this made some tasks differ considerably in size compared to others. This was something that was addressed to some extent by dividing them into as small parts as we could. This made as mentioned our velocity KPI suffer. The biggest problem though was the fact that too little time was spent on analyzing how a specific user story, and their tasks, needed to be implemented in detail. This made it so our effort estimation was quite off which in turn led to some problems in delegation and some features taking longer to implement than expected. In some cases several sprints. This was further impacted by existing technical debt. During the last sprint this improved however when an extensive list was compiled of everything left to do in the program. It is also important to mention that some of the mistakes during the effort estimation came from our inexperience with the tools that were used during our development. This is a factor that was present during all stages of development since new features needed new functionality. There also was an effort-related problem when it came to the different levels of experience in the team since the time it took to complete different tasks varied between group members. This is something that was hard to address and was seen as a “learning problem”. The last problem was that it was hard for group members that were not a part of some specific functionalities’ development to judge how long the continuation of that feature, or related features, would take. This was tried to be amended by encouraging members of the group to get a basic understanding of parts of the app that they did not develop. In practice this worked less than great since not everyone in the team found the time to do it. Our acceptance criteria have worked somewhat well even though some of them might not be formatted correctly according to our standard and they certainly could have had more time spent on them to make them more extensible. This was however not deemed as a major problem since the requests from the external stakeholder were very clear to the team thanks to the close contact with the product owner, regardless if some acceptance criteria could have been clearer.

B: In the next project a vertical mindset would have been there from the beginning to ensure that demo functionality would have existed at the end of the first sprint. The process of breaking down tasks would also be improved, especially making sure that they are all approximately the same size. Effort estimation would also be improved, especially accuracy, to ensure that delegating tasks would have been easier and making planning easier. Our acceptance criteria would also have more time spent on them, especially if the scope was bigger to ensure that all the requests of the external stakeholder would be represented, if time allows, in the program.

A→B: To achieve a more vertical focused mindset it is our opinion that it would have helped if we did effort estimation on both user stories and tasks instead of the hybrid that was done now. To get a better overview we would also write the efforts down from the start. To improve our efforts we would spend more time before the start of development to plan out the app better to make sure that both the challenges of the specific features are understood by the team and to reduce technical debt, which affected the accuracy of our estimations. Team members would also be continued to be encouraged .We would also like to spend more time on task breakdown to make sure that no potential task is missed and that the user stories are broken down into small tasks of approximately the same size. This would hopefully also lead to improvements in effort estimation.

Acceptance Tests

A: In this project we first misunderstood the meaning of acceptance tests, so we did not plan for doing any. After finding out what it actually was, we realised that we had done acceptance tests in the meetings between our PO and customer. These were performed by our PO showing a demo of our app for our customer, who then could give some feedback and tell whether the attributes fulfilled her needs or not.The value these tests provided for us was to know whether we were working on the right things and whether our work fulfilled the customers needs or not as well as if we had missed something . For our customer, the acceptance tests provided value by making the customer involved and provided a chance to influence different attributes of the app . For future users in the form of patients these tests will provide value too thanks to our customer’s greater understanding of how these patients' illness is affecting how they will use this app and which features would create value or not for both patients and caregivers. These aspects are being discussed during the demos and affects the app developing that way. One problem with our approach is that one of our end users, the patients, were only represented by proxy, the doctors, even though the doctors have a great understanding of this patient's needs our acceptance tests would be more reliable if we had been able to also get the patients point of view. In our situation this was hard to do since there is confidentiality.

B: In a future project we would do acceptance tests too since we see how it creates value for all stakeholders. If possible we would also expand the test to reach the end users, not only the external stakeholder which in our case was not the only persona of the app; we would try to test our prototype on patients as well as on more doctors to get different points of views and feedback.

A → B: We think that the way we did it in this project worked well, so we would show demos to the customer next time too. Maybe trying to test early versions of the app on patients too, if possible, since that could give valuable feedback too.

KPI:s

At the start of the project we choose to have three KPIs: wellbeeing, velocity, and individual productivity. There have been both advantages and disadvantages to our approach, and we've realised some parts could have been changed to better work as a measurement of what we really wanted to get out of the KPIs chosen. When each member filled in their values for the KPIs we also followed it up by a short comment which motivated why they felt as they did.

Wellbeing

A: Wellbeing was measured by each member filling in their stress during the sprint on a scale from 1 to 10. This KPI was often used for changes we did in our project, high levels of stress due to things in the course were things we regularly worked to improve upon. One of the major things we changed was cutting down on times spent on meetings, as it was previously a stress factor due to the amount of time they took.

B: What we want to get out of the wellbeing KPI was to make sure the project isn't too stressful, and if it is we can find out why and try to fix it. The comment along with the score is important to know why, and ideally it should work together with our other KPIs to figure out what causes, and how to reduce stress.

A→B: This KPI was the most well functioning of the three, and it might not need much improvement by itself. It is, however, dependent on our other KPIs to get a better understanding on what can cause stress, and what we can do about it. That means this KPI would be improved if we could better shape the other KPIs to get the most out of this one. Ways that could’ve been done are explained in each of the other KPIs respectively.

Velocity

A: Velocity was measured in two parts: by time spent each week, and the amount of tasks done. The idea of this KPI was to measure the pace we were going at in some way, but we’ve come to realise the current way was less than ideal to say the least. We severely underused this KPI, it mostly just got used as a week-to-week comparison, and that’s just useful as an individual indicator. Our breakdown of tasks was inconsistent too, so that made it even less useful as they’re unreliable as an indicator.

B: What we wanted was to get a good overview of our progression, not only how many tasks we got done, but also how our progress to the finished product was going. This would be useful to have as a burndown chart, seeing how many tasks and user stories we had completed compared to what we had left to do. This would make it clearer what, and how much we have left, if our pace is good enough, if our goal for the finished product currently is reasonable, etc. This also relies on the breakdown of tasks to be consistent, which is also a goal closely related to this task.

A→B: To make this KPI useful in the way intended it would have to be changed. First of all measuring time spent can be very limiting, and is at risk of negatively affecting productivity as focusing on hours spent takes away how these hours were spent. It can discourage breaks, which can improve productivity, it can also discourage finishing tasks that don't have too much left as one is closing in on the set limit of hours we’re trying to fulfill. Tasks done in each sprint isn’t necessarily very good as a measurement on its own, but it could be if there were more things to complement it. As all members had significantly different experience with programming projects, and our selected IDE Android Studio, it seems weird to just measure how many tasks each member did. It would be better to just remove the measurement of time spent, and replace it with how many tasks we plan to finish each sprint. That way we would have good measurements for a burndown chart, where we could see if we got enough done compared to what we planned, and if we didn’t we can reflect on why. If too little time was spent, or if we put up too high expectations of what we could finish in a sprint at the start of the week we could simply have that as an explanation in the comment along with the KPI. All of this would require us to be well-planned and consistent in breaking down tasks, so that the measurement would be reliable and useful.

Individual productivity

A: Individual productivity was our third KPI, and it was also measured in two parts: coding and non-coding parts of the project. The productivity was chosen as a KPI to have a representation of all work going into the project by each member of the group, as just measuring tasks doesn't reflect on time spent on learning new things. Since our group has very varied previous experience some will need to spend significantly more time on learning new things, so we put in this KPI to give more understanding of how things actually went aside from just tasks completed.

B: Individual productivity is inherently a very subjective measurement, as it's just based on how productive each member felt they were, but this is actually our intention. We want this to give all members a better understanding of each other's work put into the project, regardless of what kind of work it is, or if it's already represented in some other way. This might not be the best for a real work project, but it's useful in the context of a course where we are all trying to learn new things and reflect on them.

A→B: This has mostly worked out pretty well as a KPI on its own, but we could however have put more effort into doing anything about the result. We could have been better at picking up hints at when both the learning and amount of coding done, and been more active in taking time for pair programming, or working together in other ways, to get everyone on the track faster and keeping up a good pace. Our problem here wasn't really with our measurement, as it did its job fairly well, but with us not really doing much about the results of it when we could have done so.

Social Contract and Effort

Social Contract

A: In the beginning of the course we had some communications problems regarding meeting times and planning. We did not have a clear role description either, so for the second sprint we redefined this in an additional document. For example, something that was clarified was that the scrum master was responsible for writing all the meeting times down in our slack channel to prevent any misunderstandings. This worked very well and since then we haven't made any changes to the contract or added any extra documents. In our contract it says that decisions should be taken by voting if we can’t agree and we did this when deciding what times the meetings should be. This was an effective way of decision making and worked out well. We’ve noticed that some team members have spent significantly less time on helping with preparing documentation such as the team reflection for meetings throughout the project. We first brought up the idea of preparing for week five, as we thought it would save time for our meeting, and compared to the first week the time was more than cut in half. It wasn’t made clear if this was something voluntary or expected, as we in hindsight know that the group interpreted it in different ways. That not everyone had been preparing was first brought up when going through the team reflection for week eight, as it was noticed then. Because the things decided at that meeting were not written down and the team members who missed the meeting did not ask what was said there was some miscommunication. We have earlier agreed to split this workload evenly, but it seems like some have had to spend extra time preparing which of course results in those members having less time to code. If this was brought up earlier, or written in the social contract, we could’ve made sure that everyone was doing their part in the documentation, which would have resulted in everyone having about as much time on coding.

It says in our social contract that we are supposed to code in pairs or in teams to be more efficient and because not all team members felt that confident in coding by themselves. We haven't really followed this and this is something that could be improved.

B: The desirable situation when it comes to the social contract is that it is being followed to a greater extent. In general, we would like to have used the social contract more. One thing we would like to do differently is to describe more clearly the consequences of breaking the contract. We did not have any guidelines as to what would break the contract. This means that there is not as much incentive to follow the contract which can contribute to it being seen as guidelines instead of a contract between employees. The part about the preparing documentation was the biggest issue where the social contract was not followed. Since the evaluations show that we felt untenably stressed some weeks we believe that an even workload in preparing the documentation, but also coding in pairs would have helped the stress levels as well.

A→ B: To make sure that we follow the contract and code in pairs we could have planned more in the beginning of the project to set up smaller groups or pairs within the team. Also made a clearer schedule or goals for each group or pair. We believe that a part of the start up meeting should have included some kind of goal definition within this course since we all had different levels of ambition. Also when the misunderstanding and unbalance in workload occurred we should have had a meeting and added precautions to the contract to make it clear what applies. If this would have been included in the contract we believe that there would be less misunderstandings and that the team reflections would have caused less impatience and irritation. Adding the additional document about the clearer roles could also have helped to make it more coherent.

Time Spent in Relation to what is Delivered

A: Every week we time reported how many hours we spent in the course. Every group member would aim for 20 hours.

We noticed during the project that we spent a lot of time during meetings, which could be seen in what we delivered codewise. It took us some weeks to get a hang of getting the meetings more efficiently.

The further in the project we came we became more efficient. This was probably mainly due to the fact that you became more experienced in collaborating with each other, holding meetings, dividing tasks, etc.

We can clearly see that there is a difference in the amount of code each person has produced between those who go industrial engineering and management and those who go IT. This is of course due to previous experience and that those from industrial engineering and management had to research a lot first.

It is also important to point out that pair programming is not seen in git inspector. Because of this it looks like some in the group have done very less and some very much. We worked a lot in pairs and used co-author, so those statistics are not correct.

B: In retrospect we realize that it is hard to measure productivity in only hours spent. This is because one group member can work five hours to work on something that in the end could be seen as useless, and another could have many tasks done in the same amount of time. One

group member may also have to do a lot of research before he or she can produce any code. Another group member can produce code and do concrete work in the same amount of time. Measuring productivity in the number of hours can also contribute to group members consciously or unconsciously spending many inefficient hours while another group member does all their tasks in an efficient time but can then be perceived as doing less work overall in the group. Furthermore, breaks are something that increases productivity but which may be prioritized away when it is mainly working hours that count. In a new project, it could then be advantageous to both measure productivity in more parameters.
In the next project we cannot spend so much time during meetings, this must be streamlined from the first week. This can be done if the scrum master has a more clear agenda from the start of a meeting. It also took us some time to write down thoughts for the team reflection before the meeting when we were about to write it, this is something we have to make more efficient from the beginning in the next project.

A → B: In our next project we will reach a good productivity from the beginning by planning the project properly from the beginning. We will invest more time in agile practises like Trello, creating and breaking down user stories, creating an agenda for future meetings and deciding how many and what kind of meetings we should have every week. By deciding the preliminary purpose with every meeting in advance it will probably make them more efficient and therefore will have more time for coding and developing the app.

In the next project we will have reworked KPIs (more about this can be found under the KPI section) that more properly measure productivity. One way is that we decide how many tasks every person will do each week and then look at how many every person achieved. This will both see if a team member achieves what expected and also if we assign too many or too less tasks per week.

Design Decisions and Product Structure

Design Decisions

API

A: Android Studio includes quite a few APIs without the developer needing to add them, apart from all the APIs that are included in the Android framework we used two APIs. The first one was https://github.com/PhilJay/MPAndroidChart which is a graphing API that was used to show the graphs in the app. It was picked due to it supporting piecharts.This API worked reasonably well for us but it was a challenge to integrate it with our save system. This caused code quality to suffer since some sacrifices had to be made (for example less than stellar inheritances and some questionable castings of objects) to make them interact well. It also caused some refactoring. The other one we used was https://github.com/roomorama/Caldroid which is an API that was used to show the calendar. It was easy to use and was implemented a lot faster in our app than the graphing API.

B: The goal with using APIs is to make the work progress more convenient and faster, while still maintaining a good quality for our product. It should also help with estimating time and effort needed for different parts of the project, as available resources and documentation around selected APIs often make that clear. Other code would ideally be planned around the APIs selected to ensure a good code quality, and avoid refactorings.

A→B: Ideally the APIs will be selected early on in the project, as it makes the time and effort required for each part easier to evaluate.

Architectural patterns

A: A modified version of MVC was used since Android makes it hard to divide View and Controller. Therefore View and Controller were merged. This made it easier for us to work on different elements of the program at the same time.

B: The above named design pattern worked well for us but it is not necessarily the most optimal one. Other solutions would be examined.

A→B: It would be worth exploring other alternative solutions in the future, and this would be good to do at the start of the project to ensure that we're working according to a decided one from the start.

Other design decisions

A: In our development we found that some design decisions were made without a clear view of the app in mind or they were decided in the moment due to lack of time. One of the biggest decisions that was made had to do with how to store data and caused us some issues.

We made the decision to not use a database due to potential legal issues and instead just saved the data on the hard drive. This worked quite well for us but it took a lot of time to implement and maintain. The save system used the singleton design pattern for efficiency and the fact that it is a shared resource in our app. This made it so that the data only had to be loaded from the files at startup and when the files were updated. This would of course improve the speed of the program.

Another area that was very important was the design of the GUI since it needed to fit the end users needs and therefore being easy to use. Therefore the design was made to be easy to navigate, the stakeholder told us that this is very important so we had that in mind while doing the design. To ensure this we have created an app without too many nested pages which will make the app simpler to use. It was also important to avoid certain colours and certain images that might cause hallucinations and so forth.

One of the most difficult considerations was the notification system which was not the most prioritized feature but still important. For example whether the app should be able to run in what is called Doze mode, which is a power saving mode on new Android phones. If this should be the case it will hurt battery life of the phone. When the notification should be triggered is also something to consider. If the trigger is going to be somewhat exact time wise, meaning it triggers for example once a day at 18:00, the phone is going to be running less efficiently since the operating system optimizations where notifications are handled at the same time are not used. We decided that both of these factors are important enough for the customer value to accept the negative effects described above, since one notification is a reminder of medication. In general, like we have done before, we prioritize features over optimization.

To further customize the app to the patients needs a page where the users themselves can customise the notifications of the program. This is mainly for the doctors. This makes sure that the app is going to be usable by more users in our customer segments since they have quite different needs. Examples of these notifications are times for medications and early signs of a crisis.

B: The desired situation would be that design decisions would be more planned out rather than decided in the moment, which in some cases they were.

A→ B: Like many other questions our proposed solution is more time spent on planning before the start of development.

Technical Documentation

A: During the project not much documentation was used. A class diagram was made early on but was not up to date. A github wiki was planned for the saving functionality but was never implemented.The only real technical documentation that was used were the comments in the code. The lack of documentation made it harder for the team to understand the functionality that other members of the team wrote. This made it so that development time slowed down and the implementation of new features had to involve members of the team that worked on other features. The lack of documentation also made it harder to develop extendable software since it was up to the individual developer to design it instead of documentation made and discussed by the whole team. This resulted in a lot of time being spent on maintaining existing features to support the new ones and of course this meant that our code quality suffered.

B: In the next project we would want more technical documentation both when it comes to the structure of our program, for example class diagrams and domain models, and some sort of wiki-page describing how to use some of the classes etc.. This would save us time in the long run due to the issues mentioned.

A→B: For our next project we would use the beginning of the project to create technical documentation, that of course would not be accurate at the end of the development but rather act as a specification on what to implement. This would lead to it being easier to develop new features since there would be less time spent on maintaining old code.

How Documentation was Used

A: For the most part the only documentation that was used was the comments in the code. This worked well to an extent but as mentioned above has impacted our velocity. This approach might have been somewhat sustainable in our little project but in a real software development company this would be detrimental since if an employee left the company the code would have to be maintained by someone who only had the comments to go by, especially since the coverage on some places are not that great. In general little time was spent on updating documentation. This was probably caused by lack of time since everyone for the most part spent enough time on the project and is a consequence of low velocity. This of course is an evil circle that has to be broken, since, as already mentioned, the lack of time spent on documentation decreased our velocity.

B: In our next project we would like to both have more documentation to use and also keep it updated for previously named reasons .

A→ B: In the future we would like to spend more time on updating the documentation but this has to be done by being more productive in other areas, instead of spending more time on the project. See other sections for suggestions.

Code Quality and Coding Standards

A: We ensured code quality in the master branch by having separate branches in GitHub and pushed new changes to the master branch only when the code is working. We made some test classes for some of our classes to make sure that the code worked as we intended with some test values. We did not specify how the tests were supposed to work though which could be a room for improvement.

In the beginning of the project the only thing that was done to ensure code quality was unit-testing, (in an earlier reflection we mentioned that nothing was done to ensure code quality during the beginning of development but that was untrue). At week five we started to have a routine of peer reviewing our code and by giving feedback to each other we fixed some bugs.Some of the code quality could be implemented better and we couldn't change them in the middle of the project. The reviews were not mandatory to follow so some of the proposed changes in the review were not implemented. We worked in pairs for some weeks while coding in order to be more efficient and to have the possibility to help each other and exchange knowledge and points of views. We also helped each other much and if there was something one is unsure of they got help to look at the code to ensure that everything was correct and that the code was nicely designed. We found that with an increasing amount of features that were not really planned, the core functionality of the app would not be able to support these new features. Therefore some quick solutions had to be put in place which made the program considerably less scalable, especially in the last sprint.

The last sprint, the review was not as effective at ensuring code quality as previously. The results of the code reviews have never been definitive, which means that it's up to the individual, whose code is being reviewed, to decide if they want to implement the changes requested. Therefore a lot of requested changes get prioritized in exchange for the last polish of the program. In an ideal world this would not have been the case but since it was somewhat of a time crunch the last sprint it was hard to avoid. Some of the team members felt that due to the workload, code quality has not been prioritised. The breaking down of tasks could have also improved the code quality since it is then more clear what to do and what's expected of the code you write. Experience also has a big part in code quality since some members in the group don't have enough coding experience to write with good quality.

In general more things should have been done in previous sprints to avoid the time crunch of the last sprints which affects the quality of code. Either productivity wise or getting a better opinion of what is feasible to implement during the time period we had.

B: Overall the desirable situation is of course to have good code quality. Our ambition to comment on the code is an example of what should have been done. The idea was that it was even more important than usual during the course as we work in a larger group and you do not always keep an eye on the other members' work. So the idea was good but this was not seen in our work. Those pairs we divided us into in the beginning consisted of the group members from industrial engineering and management were working together and the group members from computer science and engineering were working together two and two. This made the development curve harder because the group members from IT had significantly more prior knowledge. Therefore it would have been better to mix the groups from the beginning. We would have also wanted a clearer strategy from the beginning to enable the code to be scalable and modular and also have a more structured way of testing.

A→B: In retrospect we should have discussed a way of ensuring code quality in one of our first meetings, so this should have been ensured during the whole project. Having the peer review at the beginning of the project would have helped to improve the code quality. The code review could also have had some parts that were mandatory to implement, maybe after some common discussion to ensure that the proposed changes were made. More planning would have made the code quality better and we should have spent time before beginning the implementation to map out the program better. This would not only improve the quality of code but also made us more productive since features would be easier to implement. In general, the code that has been committed has been quite bug-free since almost all model classes with behaviour are unit-tested.

That some team members felt like they did not have time to document their code or do it properly could have been avoided by taking on less tasks or better planning from the beginning. If we would have taken the time to map out the program in the beginning this would have probably not happened. That some of the team members felt like they did not have the experience to write code with good quality could have been solved by sticking to our contract regarding coding in pairs to maintain the quality throughout the code. In order for this to be a solution it would require that we split the team into groups where the code experience is different. Suggestively mixing the team members from IT with the members from I.

Application of Scrum

The Roles and Their Impact on Our Work

A: In the beginning of the project we decided that everyone in the group except our product owner should be scrum master for at least one week each.
We also decided on a group member to be product owner for the whole project. This choice was quite obvious because this person already had a relation with the stakeholder and was the one who came up with the idea for the app. It has also worked very well with having one product owner and this person did a great job and had a lot of contact with our stakeholder.

In week 5 we did a definition of role document as an extension to our social contract. It worked very well and clarified what everyone in the project had each week - the scrum masters role, the product owners role and the rest of the team.

B: In retrospect there were both pros and cons that everyone had a chance to be scrum master.

Advantages: Everyone got to try out to be a scrum master, which gives experience to all members of the group. Everyone could contribute different things during the sprint they were scrum master, a concrete example is that a group member introduced a meeting protocol which made our meetings much more efficient.

Disadvantages: No one really got the chance to grow into the role, because you only had one week to try to be a scrum master. It also made it not so clear who was the scrum master each week and also that his area of responsibility was a bit unclear. It almost felt like everyone took the same "leadership role". Had a person had a clearer role of responsibility, it would probably have meant more efficiency in the course of work regarding the meetings. It also made it difficult for the scrum master to get an overall picture of the project, as they had so little time to get acquainted with it. The negotiation with the product owner would probably have been easier too, as the overall better knowledge of all parts of the project makes it easier to do well.

A→B: In upcoming projects we should have one scrum master for the whole project. One alternative is to have different scrum masters for different areas, to improve the scrum masters role in the group.

In the next project we should do a definition of roles document the first week, this because it clarified every role for every person which really helped in the development of the app and in our work.

Agile Practices

A: We have worked in sprints from Monday to Monday. We had a sprint review and retrospective every Monday.

We had daily scrum meetings every day to make each other up to date with the project, tasks, how everyone was doing and if someone needed input or help. Some days we had a bit longer meetings in which we worked with Trello, user stories, tasks and wrote team reflection. These days we had sprint review meetings as well combined with the longer meetings where we wrote team reflection and had sprint planning meetings for example.

Our work with Trello, our scrum board, has also been a part of working agile. By having user stories and breaking them down into smaller, well defined tasks it was easier to divide the workload within the group. It made it also sure that everyone had a better overview of what everyone else is working on. In the beginning our numbering system where quite confusing, and this we could have done better for next time.

We planned our meetings for the whole week every monday. This made it easy for everyone to plan their work. We also could make notes who could and could not attend the meetings.

B: For the next project we believe it would be better to have longer sprints, of two weeks for example. This is because a big part of our sprints in this project were meetings and planning, which in some cases made it hard or stressful to finish all the user stories we had decided that week due to the fact it resulted in less time for coding.
The short meetings gave all group members a bigger picture of the project. Everyone could keep up with how we were doing. It also made it easier to know if someone needed help or if we had to reprioritize our resources. We could switch working couples or help each other. It was beneficial when we appreciated wrong in the estimation and if someone had more work than another. Therefore we should have these short meetings in the next project as well.
The next time we should spend more time in the beginning to properly break down user stories and tasks so the most of that work is done before starting to code. This will make our work more efficient in the future project.
For the next project we should also have a clear agenda for every meeting, this is because we spend a lot of time discussing things not everyone was involved in in the first meetings.

A→B: This would be done by keeping up short meetings and spending more time in the beginning to properly break down user stories and tasks so the most of that work is done before starting to code. This will make our work more efficient in the future project. For the next project we should also have a clear agenda for every meeting, this is because we spend a lot of time discussing things not everyone was involved in in the first meetings.

The Sprint Reviews

A: The weekly Sprint Reviews have been an essential part of our work, which have helped us keep track of how we are moving forward in our development process as well as how we are creating value for our customer.

We did have a PO, the member of the team who we believed to have the best understanding of what our stakeholder was expecting as a final result and who could easily keep in contact with her.

The PO had weekly meetings with our Stakeholder Dr. Madeleine, as well as with another doctor every second week. During these meetings our PO was able to receive feedback, show our progress, ask questions that had arisen during the process as well as consult them in the areas that were outside our expertise, for example which parameters to track when developing for patients with this disease.

In our weekly reviews we summarized what the group had achieved during the previous sprint and where we were now in relation to our project scope. The user stories that were considered to be done were discussed with the PO to make sure they were complete.We also received feedback from our stakeholders through the PO and the next sprint was planned according to what had been brought up during the review.

These reviews resulted in a re-prioritization of user stories on multiple occasions, for example when we decided to stop working on more functionality for the calendar, given that this extra functionality didn't create so much value for our customer and instead spent more time working on the graph functionality which was central for our app. To make it easier for the development team to plan the work and focus on developing and delivering the functionality of most value to our customer our PO created a priority list to organize our User Stories. During the middle of our project scope we were forced to reprioritize some of the user stories given our PO got new feedback from our customer and we understood some of the functionality we had started working on was not really creating much value to the customer and we still had core functionality to be implemented. Given the time we had left our PO made a reprioritization of the user stories in consensus with our stakeholder. Towards the end of our project scope we were forced to leave some of the non essential functionality out of our project scope due to lack of time. All these changes were communicated to the group during the Sprints Reviews and they came as a result of the feedback our PO received from our Stakeholder.

As mentioned before we consulted with our PO before deciding a user story was done, but we also checked that it met our Definition of Done;

> "*All code should be reasonably documented.*
> *All user stories should be tested,either through software or other means.*
> *The code should be peer reviewed.*
> *The code should not cause a lot of technical debt.*
> *The code should be pushed to the master branch.*"

We may have prioritized away these parameters when pushing our changes to the master branch and considering them done towards the end of the project, mainly due to lack of time. There are of course areas to be improved for how we met DoD but in general we are satisfied.

In conclusion we can say the Sprint Reviews really helped us adjust the direction of the project in order to create as much value as possible for our customer.

B: In general we got a lot of value from the sprint reviews but we could of course improve them for future projects. An ideal scenario would have been that we had successfully met all the aspects of our DoD, when marking a user story as done. One specific aspect is that we established the code should not cause a lot of technical debt, we had problems meeting this parameter given that we were forced to make changes and adjustments to different parts of our code afterwards, so it could fit with the new added code and work as intended, this slowed down our work considerably. Another aspect that would need improvement is the documentation, even though we commented most of the code the coverage could be better in some areas and we didn't really use any other kind of documentation.

Another aspect that wasn't ideal was the fact that we were forced to reprioritize in multiple occasions during our sprint reviews; for example we had started working on features as the calendar, which our stakeholder thought to be a good idea from the beginning but after deeper discussions between the stakeholder and PO it was made clear that this functionality wasn't really adding so much value to our end users. It would of course be better to be able to follow the same plan from the beginning and not needing to make major changes in our priorities but we are guessing this is a problem that may be not that rare in real projects and it may just be a challenge we need to learn how to handle.

A→B: We feel the main reason we didn't always successfully meet all of the the criterias of our DoD was that some of the parameters were unclear and open to interpretation which makes them hard to follow.We could tackle these problems by concretizing our DoD, for example what did we mean by all code should be well documented? Were we expecting just comments or we actually wanted documentation for our future users or for other developers? This can apply to other parameters in DoD as well and for future projects we would definitely prioritize these kinds of discussions more.

When it comes to the aspect of reprioritization of User stories and the stakeholder changing her mind we could of course have discussed all the different functionality of our app more carefully with the customer from the beginning. Maybe a deeper planning and discussion between our PO and stakeholder to really understand how this functionality was creating value and how it should be implemented had helped our stakeholder understand better what she really wanted earlier on in the developing process and saved the development team some work. As a whole our PO did have a good connection with the stakeholder throughout the project and the whole group had a dialog with our customer before we actually started developing. We also think some repriorizing is hard to avoid given that it can be hard for the customer to really know in advance, and some features that may be great on paper may show flows when actually tested towards the end user.

Practices for Learning

Sources for learning:

A: In the beginning of the project the students from IT helped the students from I with using the new tools that they already were familiar with. We also watched youtube tutorials for learning how to implement some functionality for the app. This felt effective because it led to that functionality to the app being added very fast. The lack of reading documentation of what we actually implemented led to that we later on realised that some things could have worked better with a different solution, for example the quiz. We also learnt a lot by pair programming, especially when we worked in mixed pairs with one from IT and one from I.

B: In an upcoming project we would read more documentation and consider different options.

A→B: This would be done by being more critical towards the youtube tutorials and investigating whether the suggested solution is the best or if another could suit our project better. Also by stating all different possible functions the program should support, so it would be easier to find out which option that is best.

Group formations:

A: Started working in groups based on which were friends from the beginning. Realised that it was better to work in more mixed constellations since the least experienced ones could have learnt things faster if they would have worked with someone who were more familiar with the tools from the start.

B: Working by pair programming in mixed groups based on competences.

A→B: In the beginning, find out which skills each team member has and based on that try to make as mixed groups as possible to enforce learning.

Relation to Literature and Guest Lectures

A: During this project we started with one introduction course and two workshops. These were very helpful in getting an introduction in how an agile project and scrum works. The workshops especially got us to understand more about how sprints work.
We had no lectures about coding or how to work in Android Studio. Instead we have mostly read up on different aspects on our own. This includes what APIs to use and how they work for example. We also noticed that there is a lot of useful documentation for Android Studio. Especially when it comes to the notification system, due to the fact that the system is a bit of a mess with deprecated features and the documentation is really needed to create something of value. We also spend some time reading up on agile principles, for example acceptance testing to understand it better.
We did not take full advantage of supervision, we had a meeting before to prepare questions but felt that there were few occasions we used it well.

We also read up on sprint reviews and other agile practices during the first weeks to figure out how they are done in practise. This we thought was important to boost our efficiency.

B: Although we got an introduction we should have focused on studying a bit more about working with scrum, due to the fact that it took us some weeks to understand how to create and break down user stories for example and avoid other previously mentioned pitfalls.
The group members who did not have very much experience in Android studios should in a next project also study more about how the coding and documentation works.

In our next project we will take more advantage of supervision, due to the fact it is a great opportunity to get feedback on our work. Also making it clear from the start if we should prepare questions or if the supervisor had an own agenda. By knowing this, it would be easier to know what to expect from these meetings.

A→B: Next time it will be good to in the first meetings decide some literature everyone in the team needs to study in the first days to make the coding and work process more efficient.

KPIs

Wellbeing

	W4	W5	W6	W7	W8	W9	Mean
Elias	6	5	8	6	10	6	6,8
William	6	5	4	5	8	5	5,5
Alva	5	3	3	3	6	5	4,2
Klara	7	6	4	5	7	3	5,3
Elin	7	6	3	3	6	3	4,7
Edenia	7	8	7	6	0	5	5,5
Pegah	5	6	6	5	5	5	5,3
Mean	6,1	5,6	5,0	4,7	6,0	4,6	

Velocity - Tasks

	W4	W5	W6	W7	W8	W9	Mean
Elias	4	7	5	4	7	5	5,3
William	4	3	5	1	6	4	3,8
Alva	2	2	6	5	1	2	3,0
Klara	2	2	5	1	4	2	2,7
Elin	2	2	4	5	2	2	2,8
Edenia	3	4	2		6	2	3,4
Pegah	2	3	2	1	1	2	1,8
Mean	2,7	3,3	4,1	2,8	3,9	2,7	

Velocity - Hours

	W4	W5	W6	W7	W8	W9	Mean
Elias	18	20	15	19	19	16	17,8
William	22	19	20	15	19	24	19,8
Alva	19	20	17	20	10	16	17,0
Klara	19	20	14	10	18	17	16,3
Elin	19	20	15	15	17	17	17,2
Edenia	26	23	20		20	15	20,8
Pegah	24	25	20	25	20	15	21,5
Mean	21,0	21,0	17,3	17,3	17,6	17,1	

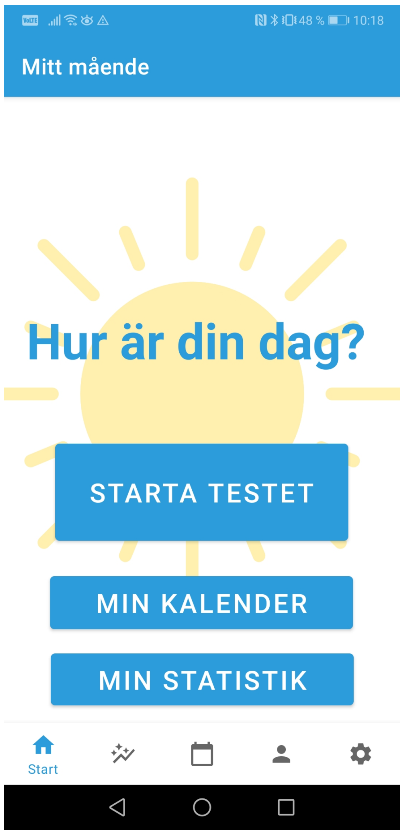
Productivity - Code

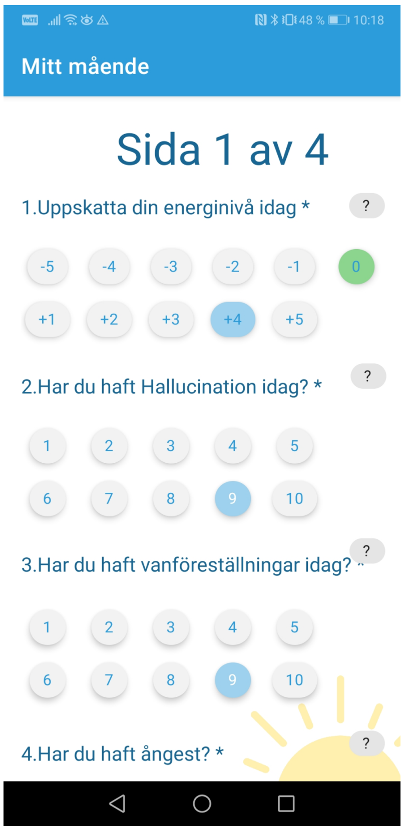
	W4	W5	W6	W7	W8	W9	Mean
Elias	8	7	8	6	7	6	7,0
William	7	3	8	2	8	6	5,7
Alva	5	4	8	8	2	6	5,5
Klara	5	6	8	4	6	6	5,8
Elin	5	3	3	6	6	7	5,0
Edenia	6	6	5		6		5,8
Pegah	6	6	7	8	6	7	6,7
Mean	6,0	5,0	6,7	5,7	5,9	6,3	

Productivity - Other

	W4	W5	W6	W7	W8	W9	Mean
Elias	5	5	5	5	7	8	5,8
William	8	9	7	5	7	8	7,3
Alva	5	9	8	5	2	4	5,5
Klara	5	6	6	5	8	8	6,3
Elin	7	5	4	4	7	7	5,7
Edenia	7	6	5		6	5	5,8
Pegah	5	4	6	5	5	5	5,0
Mean	6,0	6,3	5,9	4,8	6,0	6,4	

Pictures of the App




Mitt mående

Sida 1 av 4

1.Uppskatta din energinivå idag *

-5 -4 -3 -2 -1 0 +1 +2 +3 +4 +5

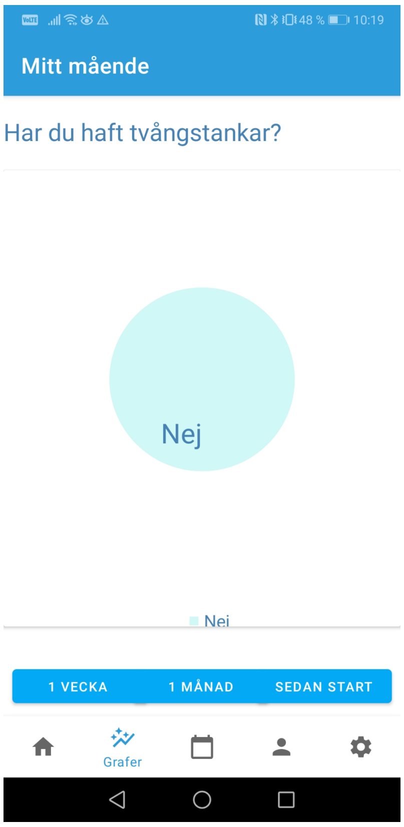
2.Har du haft Hallucination idag? *

1 2 3 4 5 6 7 8 9 10

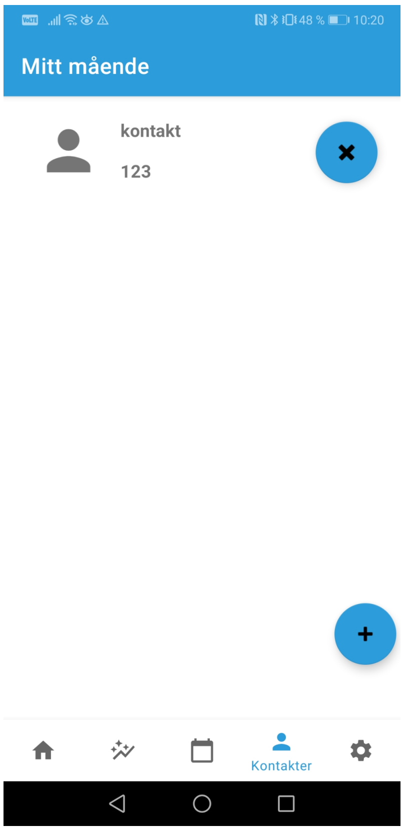
3.Har du haft vanföreställningar idag? *

1 2 3 4 5 6 7 8 9 10

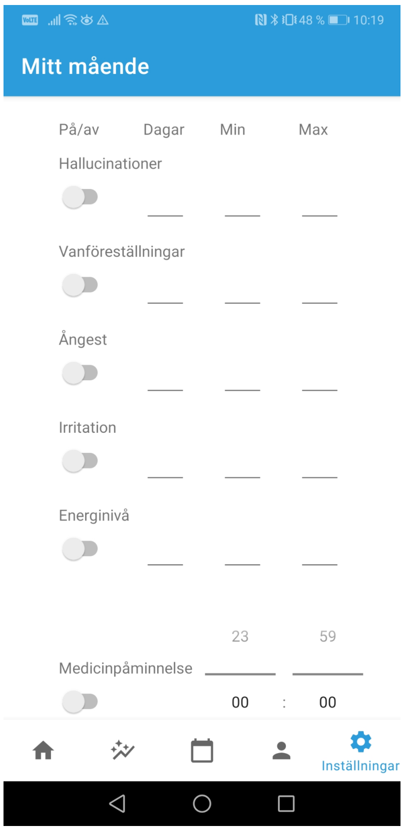
4.Har du haft ångest? *


Mitt mående

Har du haft tvångstankar?


Mitt mående

kontakt
123


Mitt mående

På/av	Dagar	Min	Max
Hallucinationer			
Vanföreställningar			
Ångest			
Irritation			
Energinivå			

Medicinpåminnelse 23 : 59 / 00 : 00