Team Reflection W5

Customer Value and Scope

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

  This week we've worked on things related to the main functionalities of our app, such as pages for the questions, and systems for saving and reading the answers. We noticed that when deciding what tasks everyone should be working on we didn't have a clear priority list at the moment. That could be good to have one made by the PO so that all group members have it available. That would make it clear to everyone exactly what parts are more important than others.

  We are creating value for our external stakeholder and indirectly her and other psychiatric patients. The most important part of that is keeping the app simple to use, which is a challenge since data representation is a big part of it. Therefore a balance needs to be found between showing a lot of different parameters to the user and simplicity.

- **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)**

  We want to develop an app that creates value for our stakeholder and its customers. To achieve this we want to work as a team, communicate with the stakeholder and learn the things we need on the way.

  We also want to improve our teamwork and working processes. This by clearer and more effective communication together with helping each other and reflecting on what we can improve.

  Some of the group members main goal is to get better at software development while others wanted to develop their agile process development skills.

- **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**

  We've worked more on breaking down our user stories into smaller parts, and made sure that all of them have acceptance criterias. We still need to add descriptions for "non functional" and make more tasks.

  We also worked on the numbering of the user stories since it was not clear which tasks should belong to what user story. This created some confusion of how the

breakdown should be done. Therefore we are going to be more consistent on how we number our tasks and sub-user stories in the future.

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

  Some of the acceptance tests are done in JUNIT and have been written parallel to development. This has made the code quite modular since the testability has been an important part of development. We still can improve here though since we have not defined what coverage we need for a user story to be accepted. At the moment it is quite loose and up to interpretation.

  For GUI the code is harder to test with unit testing and therefore it is not tested in that manner. We instead do testing manually and see that every criteria for the acceptance criteria is met. This might be an area in which we can improve upon using some standardized process or at least some documentation to communicate to others that the non unit testable functionality is tested.

  Our goal has always been to have a thoroughly tested program with few bugs since the users of the app want a simple app to use that just works.

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

  Now we have decided our KPIs which are: well being, velocity, and individual productivity. Well being is mainly focused on how the project affects each member stresswise. Velocity focuses on how many tasks we get done so that we can see if we keep a good pace. Individual productivity is a personal evaluation of how productive each member feels they've been during the sprint.

  We used the well being KPI answers to make sure plan meeting times and make them more efficient since some of the group members said that it was quite stressful when meetings took too long and were unstructured.

  Upcoming weeks we could perhaps make some easy way of keeping track and comparing how we've performed each week, maybe a spreadsheet would be a good way to view it.

Social Contract and Effort

- **your social contract i.e., the rules that define how you work together as a team, how it influenced your work, and how it evolved during the project (this means, of course, you should create one in the first week and continuously update it when the need arrives) There is a survey you can use for evaluating how the team is perceiving the process and if it is used by several teams it will also help you to assess if your team is following a general pattern or not.**

For most part our social contract has been good, and we've followed it pretty well. The one thing that wasn't working as well was that it was a bit unclear exactly what was expected from our different roles. To solve this we've now made a "Definition of roles" document to clarify the roles and what was expected of each team member. There seems to have been an improvement, but this might still be something worth improving upon if we feel there's the need to clarify things further so that misunderstandings can be avoided.

For example the document says that every member can only be expected to work around 20h each week.

- **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)**

  Our goal is to spend 20h/week each and this something that has to this point worked well.

  We've made sure to better plan when to have meetings, though the meetings could be more time efficient as that would leave more time to work on other things. This can be done by having each meeting better planned. Part of this can be done by the scrum master having the meeting more thought out ahead of time. The parts of the project outside coding is the most wasteful right now, meetings around these things would be more time efficient if for example when it comes to the Team Reflection all members noted down their ideas and opinions about how things could/should be improved ahead of time so that we have more to base our writing on when these meetings start.

Design decisions and product structure

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

  We've decided on an API to use for creating graphs (https://github.com/PhilJay/MPAndroidChart), we still need to learn more about how to use it so that we can make the graphs easy to interact with for the user (e.g. when picking which values should be used in the graphs). The reason we picked it was that it had support for pie charts, which we might use. It also seemed quite more popular than the alternatives and had an extensive documentation. The choice was made by two group members after some criterias.

  A design choice was made to refactor how the answers are represented in the app. Before we had one class to represent answers but we changed it to several to make sure that the data representation part works as smoothly as possible. It is still not perfect and there is probably more work to do in regards to answer saving and loading functionality. For example we still need to figure out how to delete data that is over one year old. This is not necessarily a direct requirement from the external

stakeholder but since the wish was that data older than one year should not be shown, it is less than optimal to store that kind of data if it is not going to be used.

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

  There's some comments in the code added now, such as some class descriptions and descriptions of methods and other parts of cade where needed. The plan is to be better at making comments in the code as we go, so there's less need to write it afterwards as there's a risk it's something that's forgotten then. We still think about adding a github wiki.

  We have started working on a class diagram, but as of now it's mostly just showing a little bit of the package interaction and some of the main classes. All new things we've made during the last week need to be added to the class diagram. Most efficient would be if every member could add the part they've worked on themselves, as they are the ones who have a best overview of how the classes they've made interact with other classes.

- **how you use and update your documentation throughout the sprints**

  Our trello is also continuously updated and every group member that is working on something is expected to update trello. This is something that has worked quite well although we all can improve in this regard. A few tasks that were worked on such as refactoring how Answers work, was not even a task since we didn't think it was needed. We have also decided that the PO is the only one who can move tasks from "Testing" to "Done". Generally though trello is a great tool for us to get an overview of what features are being implemented, by whom and tracking our general progress.

  At the moment we have not found our comments that useful since the parts of our program are very independent at the moment and the work to connect all different components is not done yet. We expect that during this the comments will prove themselves useful. The comments are updated as soon as a change is done to the code.

  Our class diagram has not been used that much but it is a good tool to visualise dependencies and the general structure of our program. Right now because of a misunderstanding of how lucidcharts work it is not accessible directly by everyone. That is something that needs to be fixed as soon as possible.

- **how you ensure code quality and enforce coding standards**

  In the scrum planning meeting at the start of the week we split the code ready for review between us so that we're all responsible for doing some reviewing each. Peer review is something that was practiced to a lesser extent last sprint but is something that the team will take more seriously going forward. It is very useful to do this to

make sure that the code is both standardized and of an acceptable quality. We made the decision to divide the code to remedy this.

We've also made some test classes for some of our classes to make sure that the code works as we intend with some test values. More tests need to be made so that we can make sure all parts of our code work the way we intend it to. In general the fact that the code is testable also makes sure that the code is somewhat modular which is of course is a good thing.

We also are mindful of dependencies and at the moment there are some between our model and util that are not very good. It has to be iterated. We also use inheritance to make our code more reusable in our Answer classes. They are connected with an interface that has an interface that returns an object, which is questionable, but in our case at least somewhat justifiable, since our subclasses return different types for the same method. To make sure that every decision like this is motivated it is the responsibility of everyone in the group to not rush implementation and only solve problems like this if another way would be unjustifiably hard.

Application of Scrum

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

  Last week we didn't have very defined roles or a set meeting time where we chose the scrum master. Improvements have been made on what the roles mean and now there's more detailed descriptions on what responsibilities they have. We've also decided when to pick a new scrum master to ensure not picking one late on again, which is at the end of the previous sprint/at the start of the new sprint.

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

  Since last week we've made the improvement to make more scheduling for the entire week ahead of time, and make sure to note if someone can't attend the meeting. This has prevented misunderstanding around when meetings are, and what the agenda is. Also, it reduces stress. What can be improved upon is to better plan out exactly what we will do during our meetings to make them more time efficient.

  In general the duration of our meetings have been lowered significantly which has given us more time to code, but we still managed to have the discussions we wanted to have. In length our last ordinary scrum meeting was around 25 min long, which is a bit longer than stand up meetings usually are but still a big step in the right direction. It is quite hard to have a virtual stand up meeting in reality but our common goal is to make our meetings around 15 minutes long, if nothing else is to be done.

- **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?)

The review did result in a re-prioritisation of user stories because we got a more clear picture of what the external stakeholder wanted us to prioritize after our PO, Edenia Isaac, talked to her. Every story that was presented as done had passed the teams DoD. Some stories did not meet the DoD cause of miscommunication, such as not being on the main branch, or working as expected but not peer reviewed. This is something that needs to improve as it creates an unnecessary backlog. To do this we in the group need to be better at delegating non-coding tasks and reminding everyone of their tasks. It is also important for everyone to read the DoD once in a while.

We still need to get used to our roles better though since it is quite odd for a PO to also be part of the team. It made the sprint review maybe less useful than it might have been if an external person was it since the PO already had a good understanding of the progress of the project.

The review also made us think about doing a form of priority list where it is very clear in which order our epics are ranked. This is because sometimes we need to work on several epics at a time and some of the epics that were planned on being worked on were not the most important valuewise. The priority list, which would be in Trello would most probably solve that problem. Our PO has to be mainly responsible for making this as she's the group member with best insight to what the product owner wants.

- **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)**

  In the group there have been different approaches. Some of us like to watch youtube videos to first create an example program for the feature, while others prefer to read documentation. It is quite natural for those that are new to Android to prefer videos since they give a better overall understanding of the topic, like a general introduction. Our common goal is to be more reliant on documentation since watching youtube videos might not be possible in a real working environment.

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

  Once again there were no lectures, we continued to read up on our own how things can be done for each of the respective areas of the program we're working on. We plan to continue doing so. Some of the group read up on sprint reviews and how they were done in practice and important talking points. It made sure that we did not miss anything important during our review. The plan is to continue to read up on the agile practices to further boost our efficiency.