Team reflection

Customer Value and Scope

- **The chosen scope of the application under development including the priority of features and for whom you are creating value**
    - **A:** We are catching up on the planned feature scope, however, there won't be enough time to implement all the features we originally planned. For that reason, we have been stressing the fact that we need to focus on features that really brings value to the end user.
    - **B:** Ideally we want our scope to be just right. Something that the team can accomplish but doesn't push them over the 20h/week ambition. We also want to provide features that our customers value in an efficient manner and we want to be able to shift our focus from those that don't.
    - **A → B:** We should try to get better at producing value. One thing we can do is to better estimate the dependencies that some tasks may have on others, to make better decisions on what we need to focus on to provide that value.

- **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:** Now, everyone has a good understanding of scrum and git. Everyone has also gotten used to working with Kivy, Python and KivyMD by now. Firebase is still lagging behind a bit. Some unexpected difficulties with the authentication has put us in a situation where we need to re-prioritize what we need implemented and what can be spared in order to get the app ready for demonstration.. From our daily scrums we track "stress levels" and have had a 3.6 average during this sprint, which is 0.1 up from last week. The stress level is defined as "*1: you feel like you need stuff to do. 3: occupied, but healthy. 5: stressed out*".

      One of the issues to be solved this week has been to fix APKs, which essentially is abouting fixing so that we can package our application as an android app again. This has been working fine, but stopped working recently. This issue was hard to give an accurate estimate of, and the issue has ended up taking most of the time of 2 of our group members this week. The issue is so complicated that we've had to contact maintainers from the python -for -android project (the software we're using to build our APKs), and might have to skip mobile app versions for the demonstration altogether.

- **B:** Success in terms of the application would be getting it to the state of minimum viable product that still satisfies our customers' core needs.
  In terms of learning outcomes, we of course want to learn scrum and the different tools we use such as Kivy, KivyMD, Python,Firebase, and Buildozer.
  Success in terms of teamwork is to maintain a good mood among the group members, make sure that everyone works efficiently and feels that they have a balanced workload.
- **A → B:** Managing the time that we have left efficiently to create a viable product. Keep working with Scrum and our technologies as well as dividing the issues into balanced tasks. Since time is getting really short, we want to make an effort to shift focus from features that are not critical fromin a value oriented perspective, and focus more on those that are.

- **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:** We have spent a lot of time breaking epics into user stories, and further subdividing them into tasks with acceptance criteria. The user stories have been estimated, and it has helped us split tasks between group members to get started and work more efficiently.

    Last sprint, we estimated that we could get 70 points worth of user stories done, however, 20 of those were missed at the end of the sprint. A contributing factor was that one issue was depending on another being finished, which ended up in a time constraint. Therefore, we decided to address separate issues and keep the same velocity in the next sprint. This also makes sense because people are getting increasingly skilled with the technologies we use, which should result in a higher velocity.

  - **B:** We want to be able to accurately estimate the remaining work and use that information to further improve efficiency and optimize for value during each sprint. We want to have a confident velocity estimation.
  - **A → B:** Keep evaluating our current velocity and use that to more accurately estimate the remaining work. Hopefully we can fix the APKs but at the same time we can't spend all time that is left trying to fix this.

- **Your acceptance tests, such as how they were performed, with whom, and which value they provided for you and the other stakeholders**
  - **A:** For acceptance testing of code (code reviews/PR reviews) we require at least two approvals from team members that did not participate in writing the code in

question. The code also needs to pass a pipeline of automated tools that help us enforce consistent code styles and keep the code coverage numbers up. As for testing of functionality, we require that the app submitted in the PR actually provides a solution to the problem it claims to solve, that it does not crash or exhibit funky behavior when being used, and that it looks good and at least somewhat close to the mockup.
    - **B:** Start to test robustness and performance of the app, to make sure that the value we've added actually works when users start using and abusing the application.
    - **A → B:** As we're drawing closer to demonstration of the app, we want to shift our focus from testing new features to testing with the purpose of uncovering any bugs that may have snuck into the codebase.

- **The three KPIs you use for monitoring your progress and how you use them to improve your process**
    - **A:** Our current KPIs are sprint velocity, code coverage, daily scrum bot responses (Stress and motivation etc.)
    - **B:** We should be using our KPI metrics to evaluate the team's velocity and productivity - enabling us to discover and eliminate potential bottlenecks that unnecessarily slow down our workflow.
    - **A → B:** We should evaluate our KPI metrics at least once per sprint to keep track of how they change, and discuss how we can optimize them further.

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)**
    - **A:** We have a social contract that we wrote and agreed upon during the groups very first meeting. It was largely based on contracts from previous group projects that the members had taken part in, refined and then combined together. Since then the contract has seen some minor changes.
    - **B:** A fully functional, tried-and-tested, completely covering social contract that aids in the teams process and provides solutions for all kinds of situations.
    - **A → B:** Continuously updating the social contract as the need arises. We should review the social contract on a regular basis to find points that we want to update.

- **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:** People are spending approximately 20/h week to finish their issues each sprint. We have gotten further with the product and people are working in parallel on different issues and communicating actively on Slack.
- **B:** We would like to continue spending 20 hrs/week, finish our sprint tasks along with course related documentations and achieve good results and grades.
- **A → B:** With more sprints (i.e. experience) we will better estimate velocity, and we will also find which people work best together and what people's strengths are. With this we can hopefully also raise estimated velocity as the team gets more comfortable working together.

Design decisions and product structure

- **How your design decisions (e.g. choice of APIs, architecture patterns, behaviour) support customer value**
  - **A:** We're using Kivy/Python for the mobile app and Firebase for the backend/database and a MVP-like architecture pattern for the app code. Initially, we were planning to use Google Maps' API for the mobile app. However, that feature isn't critical in terms of value, and the time to the project demo is getting really short. We've setup Travis with a linter, tester and an app build exporter to ensure code quality and to handle deployment. We've set our main branches to protected and require PR:s to have some approving reviews and pass the Travis checks.
    During our last meeting, we stressed the fact that there is not enough time to finish every feature that we originally planned, and that our final design decisions have to support customer value and the project demo. We have also focused on implementing features that we feel are critical to support customer value, such as link to profiles, uploading photos and a search function.
  - **B:** Well thought-out code architecture that is sustainable in the long run and will help us during development, eliminating the need for major refactoring every time a new feature is implemented in the codebase. Using APIs and external libraries that are supported and maintained will also help us implement new features quickly without having to build everything from the ground up ourselves. We would also like to adopt an MVP architecture for the app, designing only the minimum viable product for demoing the most basic functionality, thus enabling us to show our investors a basic version of the product so that they are able to see what value our app can create for both the users and society in general.
  - **A → B:** Continue working with the architecture we've setup with Kivy/Python, Firebase and Travis.

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

    - **A:** We use user stories to target and prioritize what our clients want and need. We use a design mockup to internally share a sense of the app we are building, and how we imagine features to be implemented and how the user will interact with the features we propose. We have also fully documented our code using Sphinx docstrings. We also make use of User Flows to formalize the most common navigation routes and actions that a user may want to perform and that we should support. This helps us understand if our designs and implementations are sane, and also helps us in testing real-life scenarios.
    - **B:** We want fully documented code that is as clear as possible, so that any team member can understand what every part of the code does by reading its documentation. This increases the team's truck factor and decreases time spent asking group members what/why/how questions regarding the code base. We also want to expand on the user flows, so that we can make sure that all the functionality we want to demonstrate is working properly.

    - **A → B:** We need to carry on with everything that is listed in point A.

- **How you use and update your documentation throughout the sprints**
    - **A:** User stories are added every sprint and refined as the need arises. The design mockup is also being updated as the need arises. We have also written a specification for our entire backend data storage structures.
    - **B:** Ideally we would like to be flexible and refine our documentation quickly as soon as we have to. We want useful and concise documentation that helps save time instead of being a time sink.
    - **A → B:** Continuously discussing how each user story as well as the mockup is relating to the project so far. If there are any issues we would like to solve them as soon as possible.

- **How you ensure code quality and enforce coding standards**
    - **A:** We have several continuous integration tools to enforce the PEP8 style guide and some other style guidelines. Our CI also runs unit tests, and we require PRs to be approved by at least 2 others to be merged. And since our main branches are protected all code has to be reviewed to be merged.
    - **B:** No broken/untested/undocumented/inconsistent code can be merged to any main branch.
    - **A → B:** At this point we have reached our goal to a satisfactory level. As the rules we've setup are enforced and constant, they will ensure good code quality for the future as well. In short, we have reached B to a satisfiable extent.

Application of Scrum

- **The roles you have used within the team and their impact on your work**
    - **A:** Anyone can select any issue they feel like, so people have naturally gravitated towards different types of areas for now though (e.g. Martin: CI, Theodor: Firebase) We also decided that the two Industrial Engineering students will be product owners.
    - **B:** Keeping clear roles, checking up on other people's work.
    - **A → B:** . The current situation seems good as it is. The only exception being that having only Theodor on Firebase at first was a mistake as that part has been lagging behind, but is getting sorted out now. At this point, most people have gotten to build stuff using Firebase.

- **The agile practices you have used and their impact on your work**
    - **A:** We have set up a slack bot for daily stand up meetings, it has been great. It's a good way of ensuring that everyone works efficiently without being stressed. It also helps to bring a good overview of everyone's daily situation and their progress. The topics that everyone answers each daily scrum are: *How do you feel today? Also, how stressed are you right now in life? Answer (1-5). 1: you feel like you need stuff to do. 3: occupied, but healthy. 5: stressed out*, *What did you do since yesterday?*, *What will you do today?*, *Is there anything blocking your progress?*. We also have weekly meetings on Monday/Friday.
    - **B:** Use the input from the daily standups to better understand what opportunities and challenges we're facing as individuals and as a team, and to better understand what and how much we are able to deliver during a sprint.
    - **A → B:** By looking back on the documentation of the daily standups, we will be able to see any trends in how we function as a team and if there are any individual issues that need to be addressed to make sure that everyone have the best possible conditions to do their work and feel like they're contributing. We need to set up a new method for daily scrum, since the bots trial period is over.

- **The sprint review and how it relates to your scope and customer value (in the first weeks in terms of the outcome of the current weeks exercise; in later weeks in terms of your meetings with the product owner)**
    - **A:** Last sprint our pace was slightly below the planned velocity. A contributing factor was that one of the issues was depending on another issue to be finished. Therefore, we decided to keep the planned velocity the same for this sprint. This also makes sense because the group is getting more comfortable with the technologies we use.
      Since time is getting really short before the project demonstration, we have decided to skip some of the planned features and focus on what really brings

value to the first demo. This sprint we have worked with features such as photo uploading, links to profiles and filters.
    - **B:** We want each sprint review to conclude that only value related work has been carried out, that the required requirements/features have been met and that people are healthy.
    - **A → B:** We will continue reviewing our velocity and figure out how we could improve for the next sprint. We should focus on what experiences people have gotten each sprint and share those amongst everyone while everyone is gathered. This way people can avoid running into the same issues and improves efficiency overall.

- **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:** We're using PyCharm/Emacs as our IDE, git for version control, GitHub issues (backlog) and projects (scrum board). Most of the group members have worked with these (or very similar) tools before and try to help the rest of the group to also develop an expertise in them. We are using Kivy/KivyMD for app development. Learning Kivy/KivyMD has been done quite efficiently, however learning and using Firebase has been slower. People have worked in pairs and in parallel to develop separate views. This way, people have learned together.
    - **B:** Become more adept with using the tools and methodologies, so that the tools themselves are not a problem and instead help us provide value.
    - **A → B:** Everyone is continuously improving their skills with the tools needed. Since we have a lot of different experience and knowledge backgrounds within our team, there's a bigger chance that someone has worked with the particular technologies or methods before and therefore has a chance to instruct the others. This will be part of our weekly work and it will let the team members approach the "T Developer" concept over time.

- **Relation to literature and guest lectures (how do your reflections relate to what others have to say?)**
    - **A:** We have not looked into any course literature and there will be no guest lectures, so this is not really applicable.
    - **B:** We should think about how we can incorporate any literature into our workflow in order to become more efficient.
    - **A → B:** Since there are no scheduled guest lectures we will have to bring our own knowledge and use tips from various blog posts online. The course literature might help, but we currently think that our time may be better spent by putting it

towards the project and applying scrum in practice (and learning from that) rather than reading about it.