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 have been forced to face the fact that all of the features we originally planned aren't going to be in the application we will be demonstrating. Faced with this fact, our discussions have been centered around which parts that are an absolute necessity for the program, and which parts that can be spared because they don't bring as much customer value. We have also experienced stress during this last sprint from group members wanting everything to be done in time.
    - **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 stepping back and looking at the big picture. A lot of the feedback that has been provided for us during this course has been related to our tendency to focus our discussions on the small, technical details, instead of focusing on if a particular solution is viable or even necessary.

- **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:** This week has been mostly focused around ironing out bugs and making sure that everything works as expected. The connection to our backend service has not been frictionless, and it feels like it has added to general stress level of the team. Our daily scrum tracks each members experienced stress levels, and landed at an average of 3.6 this week. It should be noted that this is actually not higher than last weeks measure, even though the experienced general stress level has seemed higher than before. The application has entered a state where the team feels that is a good representation of the initial ambition. It doesn't have everything we thought It would have, but it has the core parts and has a pleasing visual style.
    - **B:** Success in terms of the application would be getting it to a state where it is ready to be demonstrated in front of an audience..
      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:** Making sure that we keep our focus on the must-have issues, and not drift off and try to solve problems that aren't there (or that are not important to the product we want to demonstrate). With time being very much a limiting factor at this point, we want to make sure that all efforts are focused to where they need to be. With that said, at the time of writing there is only 1 thing we need to get done before being satisfied with application, so it shouldn't be a problem to get finished.

- **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 65 points worth of user stories done, however, 15 of those were missed at the end of the sprint. A contributing factor was that a vast amount of time was spent on fixing APKs, i.e. making our application work as an android app and run on an actual phone. The issue was extremely complicated and ended up taking most of the time of 2 of our group members. Therefore, we decided to move on and run the app on computers for the project demo.
      Since our pace the last three weeks has ended up on a velocity of 50, we decided to go for 50 in this sprint, which we managed to get done all 50 points.
    - **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.

- **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 unexpected behavior when being used, and that it looks good and at least somewhat close to the mockup. We accidentally exceeded our Firebase quota a few times this week so we got some test errors because of that, but that has been resolved.
    - **B:** We want our user flows to be fully functional and bring the intended value to the user.
    - **A → B:** In the time left until demonstration of the application, we want to focus more on testing user flows and real world use of the entire application, rather than testing specific features and separated functionality.

- **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 slowly zoned in on an realistic and appropriate workload for the team, given the experience we've gained in the previous sprints.

- **B:** We want to deliver a viable proof of concept application for our demonstration on Monday, which may or may not require some last minute fixes.
- **A → B:** We need to help each other out in testing, as well as in reporting and fixing any bugs that we find, while still trying our best to keep to our 20 hours per week.

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 the app needs to be easy to use and understand from a user perspective. For that reason, we added a welcome view that forwards the user, depending on their preferences, to the most convenient starting screen. We also made an effort to explain or remove anything that was not perceived as clear from a user perspective.
    This week we have put a large amount of time into finishing the authentication and user accounts; being able to make your own account is something we believe supports customer value.
  - **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. 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:** To make better time, we've been passing out issues in a more controlled manner than before. Issues are now passed to the person believed to provide a solution in the shortest time possible. This means that people have been getting more defined roles within the team this sprint, as CI work has been passed to one person, server side issues to another, etc. This indirectly also means that the teams truck factor has decreased somewhat, since one or two people tend to handle issues related to a specific area of expertise.
    - **B:** Keeping clear roles, checking up on other people's work.
    - **A → B:** . The current situation seems good as it is, even with the truck factor increase. Since people are becoming very skilled with certain tasks, we can get our issues done faster, which is increasing the amount of work we can get done in the period of time that is left.At this point, most people have gotten to build stuff using Kivy and 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 and 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. The Scrumbot has come back to life, and is helping us keep track of each others well being and workloads until the end.

- **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. 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. Since this week was the final week, our main focus was to make the app

convenient as well as easy to use and understand from a user perspective. For that reason, we decided to develop a welcome view where the user, depending on if they opt in to be a deliverer or poster, starts on the most convenient screen. We also tried to remove or explain any element that seemed unclear.
This sprint we have worked with features such as login/authentication and a bunch of minor fixes here and there, that together enhance the total experience and bring the app closer to a working state.

- **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. 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. Since this is our last sprint, we will will look back on all the sprints and see how we could improve for another project.

- **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.