TEAM 20 - Final Team Reflection

Team members: Rebecka Jakobsson, Cecilia Michelsen, Jacob Messinger, Tina Samimian, Elin Nilsson, Linnéa Fransson, Norbert Laszlo & Oscar Helgesson.

Customer value & scope

Scope

A: During this project we have developed the idea of the app and the user stories during the sprints. We have chosen to prioritize what our stakeholder consider most valuable and tried to meet her expectations as far as possible. Therefore we have created most value for the end customer, rather than the group planning the career fair that possibly will finish the coding. We have strived for reaching our mvp instead of creating a complete app ready to use which turned out to be a great idea since we reached our mvp in the last sprint and had no time to work further. Our mvp was the same throughout the whole project but when we decided on it in the beginning we didn't really have a clear view of what it meant for us.

B: In a future project we would probably want a clear view of what we would want the app to be and our mvp from the beginning instead of developing it on the way.

A->B: Next time we would probably have a more thoroughgoing discussion about our scope, what features to prioritize and what a relevant mvp could be before the first sprint. We would probably involve our stakeholder a lot more in this discussion so that we wouldn't need to change our ideas and scope as much throughout the project.

Success criteria

A: We haven't had a clear framework for what our success criterias are except regarding teamwork which is included in our social contract. The teamwork has worked really well throughout the whole project. We have had equable ambitions and put around the same effort into the project which probably depends on that we discussed these areas while writing the social contract. When it comes to the app the success criterias haven't been as clear, this probably depends on that none of us are used to working in projects like this and didn’t know what to expect from ourselves when it came to working in Android Studio.

B: In a future project we would probably want a clear view of our goals for learning outcomes from the project. Further we would need to have clearer visions for our application.

A->B: We could reach our goals by integrating our learning outcomes in the social contract. We could also write down success criterias for the application before the first sprint so that we would be sure what we aim for.

User Stories

A: The quality of our user stories have evolved very much compared to our very first sprint. In the beginning we had trouble with estimating the size and the time it took to complete each user story. We tended to create really big user stories which was problematic when it came to estimation. We also didn’t have any acceptance criterias for the first couple of sprints but we managed to work it out quite well anyways. But the further we came, the more we understood the importance of the acceptance criterias and managed to create smaller and relevant user stories for each sprint. The reason for our bad estimation was due to our lack of knowledge in Android Studio, therefore we had a steep learning curve in the beginning but the further we came the better we became at estimating the time each user stories took.

B: What we still have to improve on though is to make our user stories smaller. We would also need to develop our skills in creating clear user stories even if our application would be of great size.

A->B: To our next project we have to keep these things in mind. We should definitely limit the size of our user stories so that each user story only takes a couple of hours to complete. Also have better communication during the sprint planning meeting so that we better document the acceptance criterias of each user story.

Acceptance tests

A: Since our application was pretty simple we had a hard time defining acceptance criterias for the first couple of sprints, therefore it was hard to make acceptance tests. When we did acceptance tests we only checked our own work which isn't optimal even if the acceptance criterias was simple to understand and test.

B: What we should've done is to check and test each other's work and see if it meets our initial acceptance criterias. In future projects we should definitely have a better structure for acceptance tests where we have a well working routine and also have our work double checked by another person or team.

A->B: The better model for acceptance tests would have been to work more consistently with "pull requests" on GIT where we would quality check each other's work and see if it meets the acceptance criterias. In general having better routines for acceptance tests would probably save us time in the long run by reducing the number of things we have to redo due to the reason for it not working as we had in mind.

KPI:s

A: In the beginning of the project we didn't have an understanding about what KPI:s are used for. Therefore we struggled to create relevant KPI:s and it took us three sprints until the third KPI:s was in use. Our final KPI:s were Effectiveness (estimated vs actual time spent on user stories), Productivity (estimated vs actual finished points for user stories) and User Satisfaction (where our stakeholder got to grade our deliveries for the last sprint). We also had a hard time when it came to putting up fair limits for what hours and points completed that would be considered Good/OK/Bad for the Effectiveness and Productivity KPI:s. Therefore we had to change those limits during the project. In the last sprints we considered the KPI:s work out pretty well for us.

B: We would get an advantage from having working KPI:s throughout the whole project and have an understanding for what we wanted them to measure. We could also use a KPI for code quality.

A->B: In a future project we would need to discuss and get ourselves a greater understanding of what KPI:s would be relevant for that project and make sure to create good limits before the first sprint. This would give us a greater use of the KPI:s and help us plan for the sprints.

Social contract and effort

Social contract

A: We started the project with making a social contract. The contract included meeting rules, job allocations, expectations, decisions making, ambitions and goals. As planned we started the new sprints on Mondays. We also had a stand up meeting in the middle of the week, which gave us the chance to help each other. We also shared a calendar to make sure everyone knew when we had meetings. Regarding the coding, we worked in groups of two. When pair coding, one of the members shared their screen and the other guided the first person with help from Google to solve questions that appeared, this has been done through Zoom. All of the group members found that this method of working was good and therefore we continued throughout the whole project. In the last sprints we started to work cross teams for some tasks that proved to be more challenging than expected.

Since the social contract was written in the beginning of the course and we didn't have any knowledge about Scrum and agile practices, we readjusted the contract in the middle of the project. We added some information about the shared Scrum master role and deleted the decision of a president and a secretary for each meeting.

But overall, we didn't use the social contract in our daily work. It was made in the beginning and felt like an agreement, where everyone needed to be on the same page. We haven't gotten any internal problems along the way and the social contract was therefore never needed to be used in practise.

B: If we were to redo the project we would like to make sure to ask how everyone was doing and feeling about the work we made together. This would be a better alternative than assuming that a team member would bring opinions/disappointments to the table during the meetings we had.

A->B: We could have been using an anonymous survey in each sprint. Every team member could then range their opinions from 1 to 7 for questions like; "How do you feel about what you have accomplished this sprint?" and "How do you feel about working with your team?".

The social contract could also be more integrated in our sprints and discussed once a week, to make sure we reflect on our own progress.

Time spent on course

A: (I) The time spent on the course has overall been equally distributed. Some group members have spent more time on the coding part of the project during the course, while others have spent more time on planning and reflecting. With this said all team members have been actively involved in both tasks and the difference has not been too big on whether you one sprint chose to spend more time on the coding part or the planning and reflecting part.

(II) During the sprints we realised that we mostly managed to complete the tasks with the time we had estimated that it would take. This was something we got better at throughout the whole course and it got clear each sprint that we had improved since last week. In some sprints we could have accomplished more in the timeframe we had planned for, but we still delivered sufficient progress to satisfy both the group and the product owner’s expectations.

(III) How the time spent relative to what was delivered was somewhat different between the different pairs which allowed us to reflect on different mindsets. Some of the groups did research before meeting their partner in how the user stories could be implemented while others met first and then started researching. The former of the two, i.e. researching before meeting, resulted for one group in that the user story was completed quicker. However, it could also lead to less discussion within the pair that could lead to less thought-through designs.

We were continuously reflecting on the time we spent on the sprints, but we didn't do any drastic changes since we knew that we were reaching our MVP (even though we would have liked to accomplish more in some sprints than we managed).

B: (I) If we were to redo the project, it would be more helpful to know the different values that the different tasks of the project would bring. We would like to have a more clear view of what brings value. By this, the planning and the reflection part of the project, that might not be considered to be as important as the coding part, could become more prioritized, as it is extremely valuable to everyone involved in the project, including the external stakeholder.

(II) It would in a future project also be better if we knew what kind of deliveries that brings value. By knowing this, the risk of being forced to redo some parts reduces.

(III) Something else that would be beneficial is to know which method brings more value to the different programmers. By knowing what methods that provide what advantages could be a factor that would improve the workflow and thereby the time spent on the different sprints.

A->B: (I) By writing in the social contract about the different parts of this project (planning, coding, reflecting) it would bring a better overview of the tasks we would face. To ensure that the different parts were considered and valued the same way, we can write what each part actually does for the project and what would happen if one of the parts would be removed or less considered. At the end of each sprint each team member could also give a small review of themselves where they rate the time spent to see how they distributed it between the three tasks.

(II) There also exists a solution to be able to know that what we do really brings value. This solution minimizes the risk of being forced to redo some parts. By forming the KPI:s early on in the project, we would better know when something would be considered done. For example our KPI that considered our stakeholders thoughts of the application and what we had done so far. This KPI was only used in the three last sprints and if it would be used in the beginning, a lot of our time spent on redoing design and minor functionalities could be spent on other things.

(III) By really discussing the different methods, what the methods achieve in the coding-sessions and which method brings more value to the different programmers, a better workflow could be created to the individual programmer.

Design decisions and product structure

Design decisions

A: (I) When we created this app, we discussed how we wanted to implement the app and what tools to use. We landed on Android Studio which is a tool for creating Android apps. We chose this as this lets us use Java-Code which we were all familiar with from previous courses. Other tools aimed at for example Apple products would require us to learn a completely new programming language, which would certainly be rewarding in the long run and for our future projects/careers, but would also require an absurd amount of time spent on just researching this specific language, not bringing any value to the customer whatsoever until we had at least some basic understanding on how the development would work. None of us had previously worked with Android Studio and we found it acting as a gateway to app development, since it is not a completely new programming language, but still new features and processes on how you actually create an app. We knew that when developing apps in Android Studio you generally use the “Model-View View-Model” design pattern. Since we felt it was more important to focus on customer value than code quality this pattern was somewhat overlooked during this project.

(II) We had an idea that we needed some way of saving data. We reached out to the stakeholder to ask if they already had a database or some sort of library, which it turned out they didn’t have. It seemed as if all the data was added to their website by hand. We knew we would need to implement some sort of database in order to save the data. We settled on using the JSON-format which was something we had some previous knowledge from a previous course in Databases. JSON is an easy way of saving data without the hassle of creating and managing an entire database. Instead it acts as a text-file with specific syntax which lets you query the information you want. When implementing this we thought it would work well as it was the fastest way of implementing some way of saving data (specifically, the companies and if they have notes/added to favourites).

(III) During this project we have used Android Studios as a tool for developing an Android app. Since we had no previous knowledge on this tool prior to starting this project we had quite a challenge with getting started and to know limitations and solutions to common problems.

What we had not anticipated was that it is impossible to actually write information to the JSON-file. That made it impossible to see if a company had notes or if it had been added to favourites on app shutdown. We ended up spending a lot of time researching this, implementing a GSON-library to read and (what we just assumed would work) write from the file using Java. As we later found out it is not possible to actually write to a JSON-file.

B: (I) Following the MVVM pattern would be the most optimal way of ensuring adequate code quality and reducing bad dependencies and also make reusability, maintainability and extensibility easier. This is something we would have done differently had we done the project again.

(II) Either implementing an actual database or some other way of storing data whilst also being able to write to it to be able to save data depending on what the user does whilst using the app.

(III) Using another development tool we have more experience in or by having more previous knowledge on Android Studio.

A->B: (I) The MVVM pattern would indeed improve the code quality. If we followed the pattern from the beginning, we would not have to spend time on refactoring the finished product in order to follow the pattern.

(II) By researching the different options on how to save and manage data from the start of the project we could have saved ourselves a lot of time wasted trying to make a suboptimal solution work.

(III) By either having more prior knowledge in Android Studio, or researching other (perhaps easier to use) development tools there’s a possibility that we could have saved ourselves a lot of time spent researching solutions for Android Studio as well as better and more informed design decisions.

Technical documentation

A: When developing we used Javadoc as a way to document our code, we did this to make it easier to understand and to make the code-base more self explanatory. We did however not do this at the start but rather began doing it after the second sprint. At this point we went back and also commented on the work of previous sprints. During the project we also faced some issues regarding the overall plan of the app. We were all in agreement regarding features and visual appearance but hadn’t formulated a class diagram or UML. This resulted in that we at some points had to redo previous sprints’ work to fit it into other parts of the code. We also used the commit messages from our version control tool, Git, in order to document the larger changes made during the latest commit.

B: A better way to do it would have been to already from the start document our code properly. Writing Javadoc for the methods and classes isn't that time consuming and would have saved us time that we could have spent delivering more value to our stakeholders in forms of functionality. Having some sort of class diagram would also have been beneficial to our development process. It would have helped us save time and do the work properly the first time while also giving us developers a framework to develop after.

A->B: Reminding and forcing ourselves to write Javadoc from the start would have made work more efficient both in time spent writing it later and also making it easier for other team members to understand it. Furthermore, spending more time on the planning of the project in a more detailed way where we discuss exact solutions to how we should implement the features we want. This could have been done in many different ways but one example is writing an UML-diagram.

Use of documentation

A: During the project we both used, wrote and updated the Javadoc that we wrote to all methods we created. As mentioned above under *Technical Documentation* we did however not start documenting until Sprint 3 of the project. When we started documenting the code, it helped other members of the group (that didn't help with that part of the code last sprint) to easier and faster understand the function of the methods and being able to use it when needed. We also used Git as our version control during the project where we all tried to write descriptive commit messages. This helped others to get an understanding of the overarching changes made during the commit but also so going backwards if something would have happened would have been made easier. We however lacked documentation about the specific requirements for the different methods and features. Questions like what the method should return and what is needed in order for the method to function had to be either decided pair-by-pair when the problem was encountered or the pair had to contact the rest of the group.

B: Having better knowledge of Javadoc from the beginning would have helped others even from the beginning. But mostly, having a formulated some sort of a class diagram, would have made a significant difference. The workflow would have been more efficient as all team members have a document to use and refer to when developing and also the methods and classes would all have a pre-designed purpose where we could ensure that all different parts worked well together. We could then also update this UML document when needed during the project to add new features. If the stakeholder wants to use/continue to develop and maintain this app in the future, an UML would also create value in the end for them as they can easier understand the structure and design of the code base.

A->B: Formulating our DoD clearly in a document from the start would have helped us remind ourselves to always document our code. This way we would start documenting it already from Sprint one. Spending additional time before we started coding on planning the structure of our code more thoroughly through an UML-document or would have helped save time when developing methods and classes that later would be put to use. The process would have a clearer objective where to reach code-wise.

Code quality

A: (I) Every week each member was assigned a coding-partner, and every pair would work together on solving tasks and User Stories. This also automatically led to the code being checked by two people in the team, before being staged for merges during the merge-meetings.

During this project we have used a rather unconventional (as pointed out to us by our supervisor Salome) way of conducting merging and code-quality checks. What we did was that at the end of every week the entire team joined a zoom-meeting with one of the members being "merge-master" who would share their screen, go into every sprint-pair's branch and merge this into master. Resolving conflicts and letting the sprint-pair explain what they had been working on and what they managed to implement during the sprint. The coding-pair was responsible for making sure the functions worked the way we intended and also for writing tests and javaDoc/comments in the code. This method of conducting merges ended up being a pretty bad way of optimizing time and code quality. Most of the merge-meeting just consisted of the “merge-master” conducting merges with the rest of the group not being able to contribute. In later sprints, some of the members who felt like they didn’t contribute at the merge meeting ended up leaving the meeting in order to do something more productive, such as starting to plan the new sprint. In one of the last sprints we implemented Pull Requests, but as time ran short and we wanted to bring as much value to the customer as possible and to reach our MVP the use of PR’s was not implemented properly.

(II) We formulated a Definition Of Done in the middle of the project, which was to be used as a guideline in all of our code to better code quality, but we did not follow the DoD to its full extent. For example not writing javaDoc for every method or writing unit-tests for methods that have functionality.

B: (I) In a future project we would like to have a better overview of each other’s work. For example to review other sprint-pairs code and to ensure code quality and consistency, such as variable naming and documentation.

(II) We would like to create and better follow the DoD from the beginning and write the DoD to better suit our team’s ambition and to achieve better code quality (as the DoD aims to do). For example, is unit-testing something we actually want to implement, or is it something we have just learned from previous courses that **has** to be present?

A->B: (I) Just like we started with but did not follow through on, using Pull Requests instead of merge-meeting would have been a more refined and optimal way of performing merges and ensuring code quality.

(II) Either by rewriting the DoD, or spending time to ensure that the entire code base fulfills our DoD.

Application of Scrum

Roles used

A: During the whole project we have had two new scrum masters each week. Since there are eight people and five sprints, we wanted to make sure everyone in our team got the possibility to try out being a Scrum master. As this is a course in agile management, giving everyone a possibility of understanding what a scrum master does and why it is useful felt very natural.

To our understanding having two scrum masters in a team is quite unusual. In the very beginning of the project it took some time to get started and the role did not function to its full potential. Therefore the two persons that were Scrum masters the first sprint also had the role as Scrum masters in the last sprint when everyone already had tried the role. Overall it has worked out good for us. The whole team has been very engaged and therefore the Scrum masters' biggest area of responsibility has been to lead the meetings.

We have chosen to have all the team members as product owners in order to create as much value for the customer as possible. By sharing the role of the product owner we have all been part of the weekly meetings with our stakeholder which has brought clarity to all of us. Since every member is a product owner, everyone is able to ask questions and discuss functionality with the customer in mind.

B: In a future project we would like to have more specific role descriptions in the very beginning. We would also like to have one person in the role as a Scrum master and one person who is product owner. In this way they get used to the roles, can develop skills within the roles, and we would probably have more effective meetings. Although for this project we think that it was very educational for everyone to try the role as Scrum master and product owner.

A->B: First week we would have specified the expectations of the role as a Scrum master and product owner. We would make a list of the responsibilities for each role and write it in the social contract. Also we would have chosen one person for Scrum master and one person for product owner, and the roles would be maintained during the whole project.

Agile practices

A: We have followed the basic rules of agile practices and have had the agile working process in mind during the whole project. During the first week we set up a scrum board for the project which we continuously have been updating together. From the beginning we created a bit too big user stories but in our last sprints we tried to create smaller user stories which were easier to accomplish. We have aspired to create sprints and user stories that follow the cake-principle to deliver as much value as possible for our stakeholder each sprint. By working in an agile way and worked closely within the team, the project got well structured, made it easy for us to work efficiently and to ensure that everyone felt involved.

B: If we were able to redo the project we would aim to have more efficient meetings where everyone was 100 % focused. That would increase our productivity. We would also like to have more, but smaller, user stories. We would also like to create smaller user stories throughout the whole project to make it easier to divide it into tasks. We would also prioritize to create better and more detailed tasks.

A->B: We would have been more detailed when we created our sprint board in the beginning of our project. We would also have spent more time on doing research on how to write good user stories, what is important in the agile practice, and made a clearer product description to ensure that the whole team knew what we wanted to develop. That would have made it easier to create user stories. We would also have more stand-up meetings, at least 2 per week. By having more but shorter meetings we would have better focus during the meetings and making them more efficient.

Sprint review

A: Every week we have had a Sprint review together with the stakeholder. We presented what we had been working on the latest sprint, discussed functionalities and design of the app, and received valuable feedback. Everyone in the team participated at the Sprint reviews as we shared the role as product owner. During the whole project we have used the feedback from our stakeholder in the best way possible. We always had value for the customer in mind and changed our user stories based on the feedback. The sprint planning meeting took place directly after the sprint review every week which made us re-prioritize the user stories. Overall we think that we have taken great consideration to the stakeholder during the project, which also can be seen in the KPI “user satisfaction”.

B: We would create a clear and detailed DoD that we would make sure that everyone followed each week. We would also prepare more for the meetings with the stakeholder to make them more effective, structured and get as much as possible out of it.

A->B: By appointing one of the team members as the PO for the whole project, that person could have made sure that the sprint review with our stakeholder was well planned and contained all of our important questions. We as a group would define our DoD in the beginning of the project and make sure that we followed it from the first sprint.

Best practices

A: To be able to follow best practises we had to get familiar with a couple of softwares which some of us weren't familiar or comfortable enough with at the start. Therefore we had some internal workshops to synchronize our knowledge in the group and everybody also completed a tutorial in Android Studio. Since we also worked in programming pairs we tried to create pairs for the sprints where at least one of the members in the group were a bit more experienced.

B: If we would do another project we would probably choose a framework that we're more comfortable with since Android Studio was new to us all and had its struggles.

A->B: To have a better workflow where we could focus on the big picture and agile project management rather than struggling with framework specific issues we would probably choose another framework to work with that we are more comfortable with so that we could focus on the things that are more important to this course. Alternatively do some more research and training in the chosen framework.

Literature & guest lectures

A: The whole group has participated in all lectures. We have used notes from the lectures as well as discussed examples from them in the project. Foremost we used it when writing user stories. When we have gotten stuck during the project we have emailed either the examiners or our supervisor. We are happy with how we planned for and used the material from the lectures.

B: If we were to do anything differently for a future project we would probably use the literature provided in the course a bit more.

A->B: Next time we would reflect upon what we wanted to take with us from the course in the beginning of it. This would help us get a better use of the material provided by the lecturers. We would also read more about what the roles in Scrum means to be able to use them in an as useful way as possible.