Final Report

Team Ginger

1 Customer Value and Scope

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

Keeping a clean environment around us and sustainable society is an important part of human life. We have been aiming for the application to help out with the UN sustainable development goals 14 and 15. With our application the users are able to contribute to a clean environment by collecting trash and registering it in the application. The application makes the user feel rewarded for their efforts and allows the user to inspire others by sharing their progress on social media. Developing something of value for the environment is the driver for our team. We have decided to prioritize the following overarching goals for application features:

1. The user should be able to register and keep track of the garbage they have collected.
2. The user should be able to receive some form of reward for collecting garbage.
3. The user should be able to show off their garbage they have collected.

When we started out we had three user-stories planned for the first sprint. We planned on creating multiple user-interface prototypes, a data model for the application user and setting up a server backend. We agreed that creating multiple user-interfaces would let us choose the best parts from each prototype and combine them into a good user-interface in the next sprint. The decision to start with setting up a backend and a user model was thought to be efficient as we planned on much of the other functionality to rely on the backend and user data model.

However our plans changed after our first consultation meeting as we gained more understanding of how to work in an agile way. As only the user-interface delivered any direct customer value we decided to delay the user model and backend. Instead we decided that we should create a collection system so that the user can register that it has collected trash. This would create direct customer value as the user could then use the application in a meaningful way. We also decided to create sharing functionality so that the user could share their progress through other applications.

In the second sprint we focused on creating a cohesive and functional user-interface from the prototypes, creating a leveling system, a cute avatar for the application and an achievement page. Finishing the user-interface was very important for creating customer value as the application has to be able to show different pages, such as the achievement page. The application also has to look good so that users would get a good impression of the application and keep using it. The leveling system is also important for customer value as it incentivises the user to keep on collecting trash by making the user feel rewarded for their efforts. Both the achievement page and the cute avatar would further incentivise the user to keep on collecting trash.

By the third sprint we had a user-interface, a leveling system, an achievement page and a half finished avatar. This sprint we chose to focus on improving the trash collection system, creating an achievement system, continuing work on the avatar and deciding what achievements should be implemented. The improved trash collection system would make it easier for the user to register multiple kinds of trash such as metal cans or batteries. The system would also allow more functionality in the future such as a collection history and collection analytics. The achievement system was needed to get the achievement page to become useful as it was just a static page.

When the forth sprint came around we realised that the backend and user functionality we had in the backlog since the first sprint was not worth prioritizing. It would take many sprints to get much value to the user from implementing these systems and we had only two sprints left. In the fourth sprint we decided to start work on an analytics page, a collection history page, a notification system to notify the user of an achievement or level up and creating icons. We also kept on working on the avatar and achievement system. The analytics page would further incentivize the user to collect trash as the user would get a deeper understanding of how much they have collected. The history page would be useful to the user as mistakenly collected trash can be removed from it and the user can see when exactly they collected what. Creating icons was important for the application's visual appeal. The notification system would help make the user more aware of the leveling and achievement systems.

In the fifth and final sprint we decided to mainly focus on finishing what we had started as we wanted to have a finished product by the end of the course. We fixed a few bugs, finished the notification system and the avatar. We also created an introduction page so that the user would better understand the application when using it for the first time. An issue with the collection system where the user could collect negative amounts of trash was also fixed. More icons were added to make the purpose of buttons more apparent to the user.

1.2 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)

The main objective and success criteria for the team was to understand and be able to apply the principles of agile during software development and deliver the successful project in the given timeframe.

When we started this course the majority of the team members did not have any experience with agile. What we have learned is that the ways of working in agile software development is different from what we have come across earlier during our education.

During the project life cycle we have gained vast knowledge about each step of the process, from the definition of the product, requirement framing, project planning to the execution and final deliveries. We have built an understanding of how important collaborative teamwork is during different phases of the project.

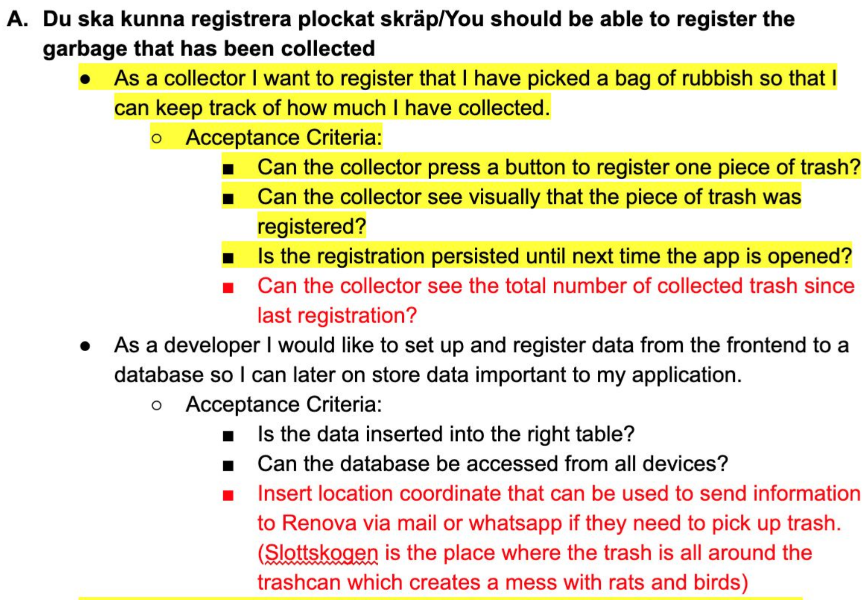
The Agile Paradigm — Success Criteria

- During the project we have focused to deliver an application which will bring **More Customer Value** to the end-user. Changing the focus and priority from backend development to the collection system will energize users to contribute to sustainable society. The application as such was an interesting exercise. The selection we made was mainly based on what we could do for a more sustainable society. The application was an interesting and fun project both from the requirements definition perspective as well as project execution. We felt that developing the application will create the value for both users and society.
- The user stories were defined in such a way that we could implement and test the new functionalities and contribute to **Faster deliveries**. This would not be possible if we had used a different type of development process. The flexibility and task definition as well as tests helped us to package features in such a way that we could deliver on a continuous basis.
- With the weekly Scrum meetings we have been able to reach **Higher Velocity or more productivity** in the development. The agile way of working supports that the problem we have faced could be solved quickly which did not affect the final product delivery.
- Applying agile processes the team got an opportunity to be **More Innovative.** We could review and redefine the user stories on the weekly basis, we could make more dynamic decisions that improved the final product. New features were introduced in a very efficient way and discussion between the team members were held on a weekly basis. New ideas were exchanged so all team members were able to give input and contribute continuously to improvement of the final product. Agile methods allow us to bring new insights during the entire process since we have not locked design and implemented features according to the original understanding.
- Providing a **Faster feedback** to each other we could improve the quality of the product and deliver the final result.

1.3 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

At the very beginning of the project we discussed how a user story should look. It was a relatively new way of working for the majority of us and it took a lot of practice to get to where we are today.

We had a few meetings where we simply worked on creating a list of features we wanted in the app [1]. When we had it all written down we started to section it off under headlines like "You should be able to register the garbage that has been collected."



A. Du ska kunna registrera plockat skräp/You should be able to register the garbage that has been collected

- As a collector I want to register that I have picked a bag of rubbish so that I can keep track of how much I have collected.
    - Acceptance Criteria:
        - Can the collector press a button to register one piece of trash?
        - Can the collector see visually that the piece of trash was registered?
        - Is the registration persisted until next time the app is opened?
        - Can the collector see the total number of collected trash since last registration?
- As a developer I would like to set up and register data from the frontend to a database so I can later on store data important to my application.
    - Acceptance Criteria:
        - Is the data inserted into the right table?
        - Can the database be accessed from all devices?
        - Insert location coordinate that can be used to send information to Renova via mail or whatsapp if they need to pick up trash. (Slottskogen is the place where the trash is all around the trashcan which creates a mess with rats and birds)

Figure 1. Part of our initial sketch of user stories and their acceptance criteria.

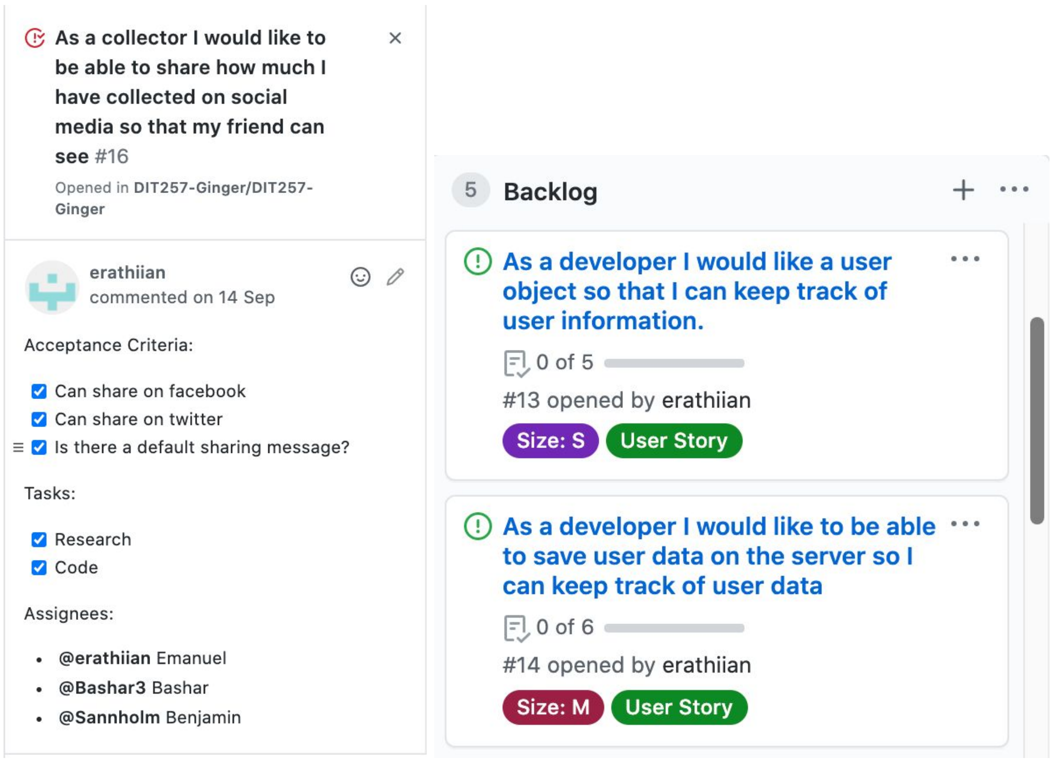
In Figure 1 we have a part of our initial sketch of how some of the user stories should look like and that we felt would fit into the scope of the header. We then filled in each user stories' corresponding acceptance criteria. The pattern we decided upon for the titles were pretty standard: "As a _, I would like _, so that _."

Eventually we started to organise them in order of importance, of what features were absolutely essential to have in order for the app to be usable (see Figure 1). We all agreed that it should be visually pleasing and that you should be able to collect trash and see that the collected trash had resulted in some change of state, like a trash count. Some ideas were dropped and others added.

When we felt like we had enough of these filled out we started to migrate the most important ones to our Scrum-board on GitHub. We also decided upon a system for effort estimation based on sizes:

XS: A rather trivial user story that should take a person less than a day to complete.
S: A user story that should take a person one to a few days to complete.
M: A user story that should take a person one sprint to complete.
L: A user story that might take a person two sprints to complete.
XL: A user story that takes a person over two sprints to complete.

This system was based around the individual(s) that were assigned to the user story. As it might take a few hours to complete a user story for one member (size S) that is familiar with the technologies whilst it might take another an entire sprint (size M) to complete it. So whenever a member was assigned to a user story, previous knowledge and experiences were taken into consideration when setting the effort estimation. The same goes for if there were several people assigned to the same user story.


As a collector I would like to be able to share how much I have collected on social media so that my friend can see #16

Opened in DIT257-Ginger/DIT257-Ginger

erathiian commented on 14 Sep

Acceptance Criteria:

- [x] Can share on facebook
- [x] Can share on twitter
- [x] Is there a default sharing message?

Tasks:

- [x] Research
- [x] Code

Assignees:

- @erathiian Emanuel
- @Bashar3 Bashar
- @Sannholm Benjamin

5 Backlog

As a developer I would like a user object so that I can keep track of user information.

0 of 5

#13 opened by erathiian

Size: S User Story

As a developer I would like to be able to save user data on the server so I can keep track of user data

0 of 6

#14 opened by erathiian

Size: M User Story

Figure 2,3. Left (Figure 2): Our initial setup for how to handle tasks. Right (Figure 3): Examples of our earliest user stories with the "As a developer .."-title.

At the beginning of our project we had tasks as well as acceptance criteria, but it only took us one or two sprints to exclude the tasks, as we felt that they were redundant since our acceptance criteria (see Figure 2) were sufficiently descriptive on their own.

One thing we found difficult in the beginning was "vertical vs horizontal" user stories. We felt like we should start out with a lot of horizontal work, like setting up a database, or writing frameworks. This resulted in us having almost all of our initial user stories with titles like "As a *developer* I would like a backend in which to store trash collections" (see Figure 3). After some consultation with our supervisor we realised that we could change the angle of them to something akin to "As a user I would like for my collection-data to be stored so that were I to change devices, my progress would not be lost". This way of writing actually had an interesting side-effect, namely that it forced us to be more focused on what each user story could provide in terms of customer value!

Comparing the first few weeks with the last two our methods and ways to go about things look quite similar, but we have streamlined it a bit by writing the user stories directly into the Scrum-board. The effort estimations worked alright but our biggest issue seemed to be us underestimating the sizes of user stories. There were a few times where a user story had all its acceptance criteria met but there was not any time left for testing. This was something Håkan mentioned in one of the first lectures, that estimating how much time something was going to take was hard, and indeed it was.

In the beginning we put in too little time, meaning that our estimates were off, but now these last two weeks we have more or less put in just as much time as we expected to be able to. So our estimates got better with practice!

If we were to start a similar project we would definitely bring with us the layout of the Scrum-board from GitHub. The fact that it is very customisable, allowing different labels and the ability to link pull-requests makes it very easy to see everything that is happening in the project. Besides those things, now that we have finished a project we have a greater understanding of what we can achieve during a sprint and this will make it way easier to plan ahead when creating the initial user stories. Whilst on the topic of user stories, as a result of this project we have learned how to better structure an initial backlog. We are better able to figure out what to prioritize in terms of customer value and how to balance it with the more "backend" work that provides us developers with value. Backlog refinement is a term we stumbled upon in the last weeks of the development and, looking back, is a practice that could have made both the sorting and the creation of our user stories less messy. For us it would have sufficed to do this every two weeks, but depending on the scope of the project, and the style of user stories, it might have to be done more often. It was an important discovery!

Another thing we will bring with us is the focus on having our new features be usable rather than done. By that we mean that we would rather have a button that does *something* at the end of a sprint rather than trying to make it have all the functionalities we wish for it.

The reason for this is that in the end of a sprint we usually merge all pull-requests and if one button were not implemented as a result of it taking too much time to implement, then all future development would lag behind as a result of it. But if it were implemented but simply printed some "Hello World"-message, then the development would most likely be able to continue in the following sprints. This kind of development worked well, as we started out with a GUI that did not really do much of anything and in the following sprints we added the features that were connected to the GUI:s different components.

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

Our project has been inspired by acceptance test-driven development and the majority of our acceptance criteria were written before each sprint began. For us to consider an acceptance criteria met we performed an acceptance test on that particular acceptance criteria. So the acceptance test itself was to find out if the acceptance criteria had been properly passed.

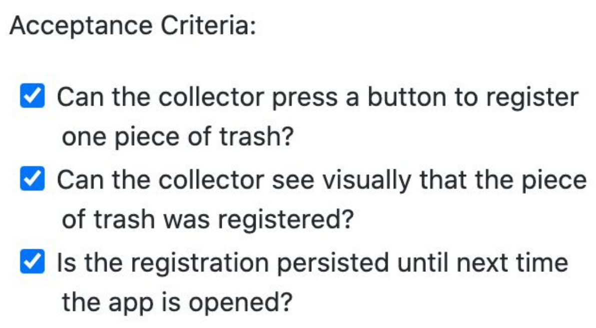
On Wikipedia we find that the most common way to write acceptance tests is on the form
Given [What state to test from].
When [Action to be tested is performed].
Then [The change that was expected to happen should have taken place].
[2].

Although we did not know this at the time when we created our definition of acceptance tests, they were of this form but we did not have it written out like that explicitly.

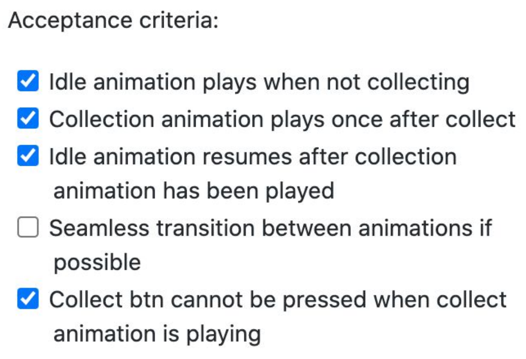
When we created an acceptance criteria (and by extension its corresponding test, as they pretty much defined the same thing) for a user story the entire team was present. We discussed the particular user story in detail and started to sketch out some initial outline of the steps that we were required to take in order for the user story to progress and eventually be finished. These steps were the ones that eventually became the acceptance tests. So, say we had, for example, a button: then it had to be pressable, something had to happen when it was pressed - like an update in the GUI and it had to update the memory with the new value. All of these steps had to be taken in order for the feature to be functional and if we look at it a bit closer then we can see that it follows the common form described in the Wikipedia-article.

We have our “Given”, which is the current state of the app, we have our “When” which is the action being performed and we also have our “Then”, what we expect to happen after an action is taken.

We found that our acceptance tests were very similar to smoke tests, where an action is performed and a result is expected.


Acceptance Criteria:

- ☑ Can the collector press a button to register one piece of trash?
- ☑ Can the collector see visually that the piece of trash was registered?
- ☑ Is the registration persisted until next time the app is opened?


Acceptance criteria:

- ☑ Idle animation plays when not collecting
- ☑ Collection animation plays once after collect
- ☑ Idle animation resumes after collection animation has been played
- ☐ Seamless transition between animations if possible
- ☑ Collect btn cannot be pressed when collect animation is playing

Figure 4,5. Left: One of our earliest acceptance criteria which had been tested. Right: One of our latest acceptance criteria which is in the middle of testing.

As described above, the acceptance tests were written out like tasks, something which was also discussed under section 1.3. So we had the great benefit of being able to perform the majority of the acceptance tests one by one during development. Looking at Figure 4 and 5 we can see this more clearly, how each acceptance test can be checked off one by one during the development of the feature.

When the developer had performed all acceptance tests and they had been passed the user story was ready to be moved to the next phase, unit-testing. By the end of the sprint, the user story should have passed through all stages in the Scrum board and be ready for merging into master. This is where the entire team performed all acceptance tests again to make sure that everyone was in agreement with it passing all tests and if it were, then the user story would be merged into master.

We discovered quite a few benefits of our acceptance tests not being so tightly connected. One of those is the PO and the other stakeholders are able to see at what pace the project is progressing and it might aid them in knowing how to plan and push for future sprints.

Another benefit is that each member of the team will know at what stage each user story is at. Say that one person needed input or help with some specific feature and that that feature is loosely connected to someone else's user story that is also under development in that sprint. Then the person needing assistance could check the progression and decide when it would be appropriate to have a discussion on how to progress to make fewer conflicts upon the eventual merge into master.

Now, turning back to Figure 4 and 5 again, our earliest acceptance criteria and our latest acceptance criteria are looking very much alike. So the conclusion we can draw from this is that we pretty much got it right at the very beginning! One of the reasons the criteria could be changed, however, is a lack of clarity. By this, we mean that when the acceptance criteria were almost exclusively written, as mentioned above, was before any work on the user story was done. So if it contained some unspecific wording and such, then the assignee(s) of the user story would simply clarify it once they had done the research and figured out how things should look and work.

Another reason for it to be changed would be that sometimes we would work on a user story and eventually realise that something would not work in terms of time or effort and that we would have to consider something different altogether. A change like this would have to be decided on a meeting since it might be a big change that might create conflicts in other user stories.

Having acceptance criteria and performing the corresponding tests has been very useful. It has provided guidance and made the user stories less daunting because you always know what to do and what is expected of you. It also serves as a guide for those working on user stories that are connected to it. This is something that we want to bring with us to future projects, since it serves as a sort of hard todo-list. You are not ready to move on until you have performed and fulfilled all acceptance tests and that clears up a lot of confusion in the project!

1.5 The three KPIs you use for monitoring your progress and how you use them to improve your process

We initially had problems choosing KPIs as we thought it was hard to figure out meaningful and measurable KPIs. After a long meeting we decided on three KPIs which would give us insight into how hard we were working, how the team was feeling about the progress and how well we were progressing relative to our schedule.

The first of our KPIs, initially dubbed "Time / Actual time per sprint", will be referred to as "Average overtime per sprint" as that is a much more accurate name. Average overtime per sprint is measured by calculating how much time each team member spent working on the course over the 20 hours per week recommendation. The total overtime is divided by the amount of team members to give insight into how hard the team has worked as a whole.

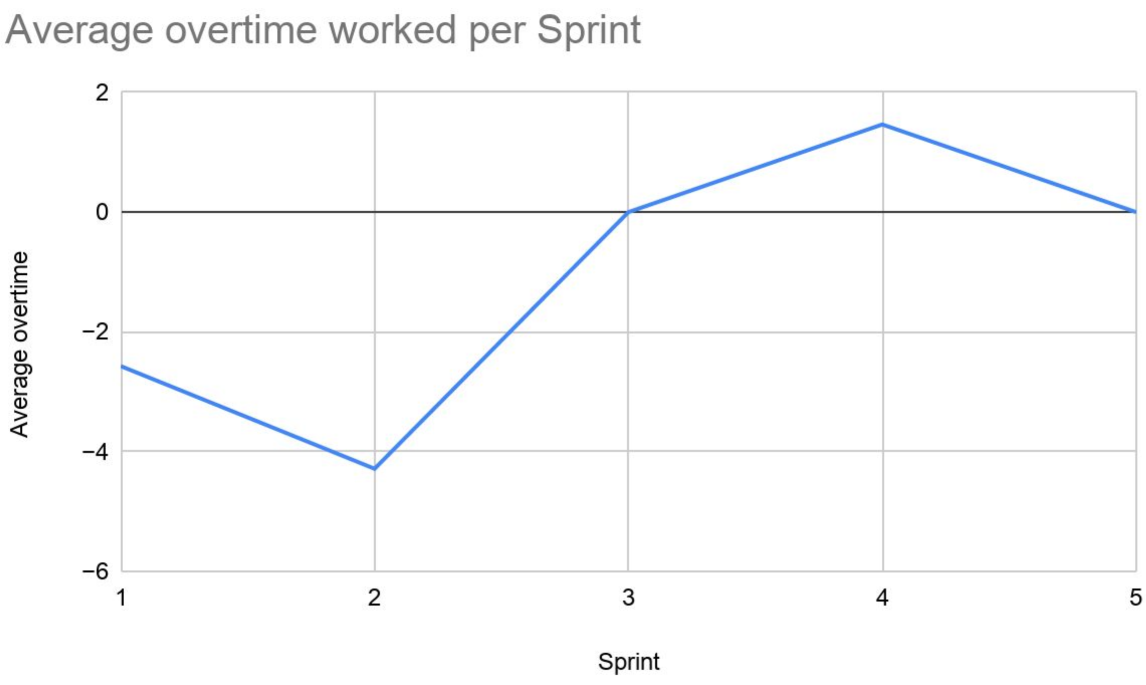


Figure 6. Graph of the average overtime worked per sprint over the five sprints.

This was useful for finding out if the tasks assigned to the team for each sprint were too easy or hard. However the KPI has a flaw, a few sprints some team members worked a lot more than the 20 hours designated while some worked a lot less than the 20 hours designated. When looking at one such sprint in the figure above it looks like the workload was good but in reality the work was unevenly distributed.

Our second KPI "Satisfaction level per sprint" measured how satisfied each team member was with how the project was progressing. At the end of each sprint, each team member declared their satisfaction level as a number from one to five. The following guidelines for what each value, one to five, meant were decided to be:

1. Under performing
2. Partly performing
3. Meets expectation
4. Exceeds expectation
5. Outstanding

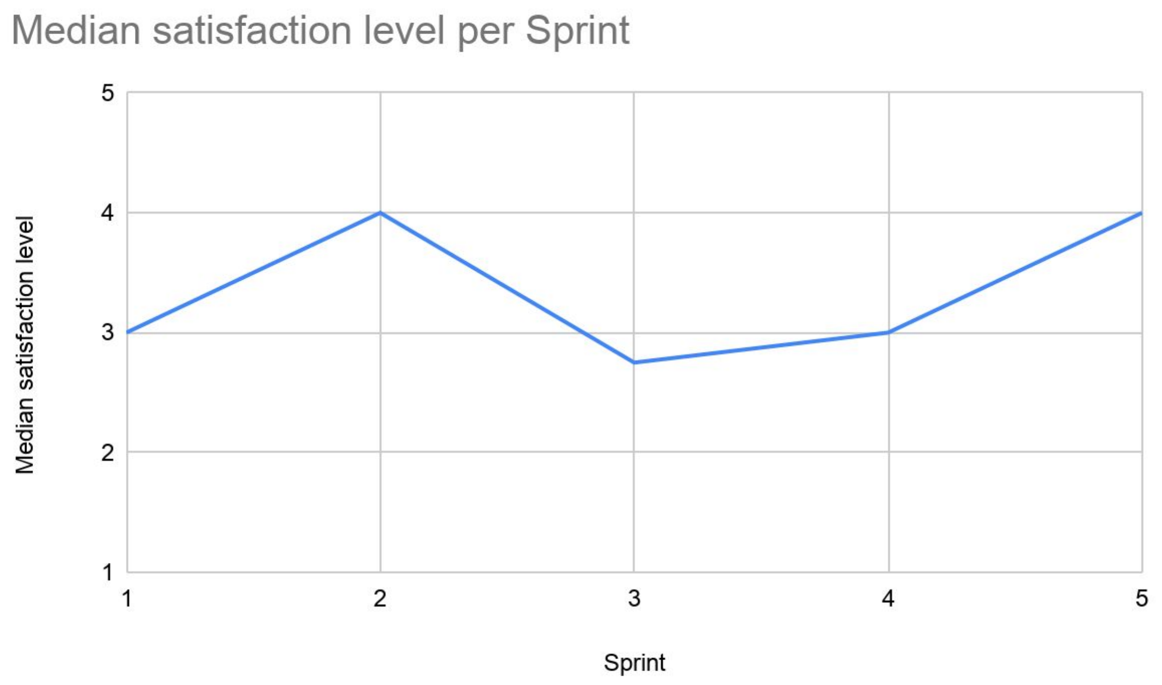


Figure 7. Graph of the median satisfaction level per sprint over the five sprints.

The satisfaction level KPI helped us measure customer value. When the team was satisfied with the progress then more customer value was created. This can be clearly seen in Figure 7 above as in the second sprint the interface and core functionality was in place. In the two following sprints little customer value was created as we worked mainly on creating systems which were not visible to the user. When the functionality was finally in place after sprint five we saw a rise in satisfaction level.

Our third KPI "Developed features / planned features per sprint" measured how big a share of the features which were planned for the sprint were actually completed. This definition was initially quite vague so we later determined that a feature was the same thing as a user story. Partially finished user stories were measured as half of a finished user story. The KPI was calculated by dividing the amount of finished user stories for the sprint by the amount of planned user stories for the print.

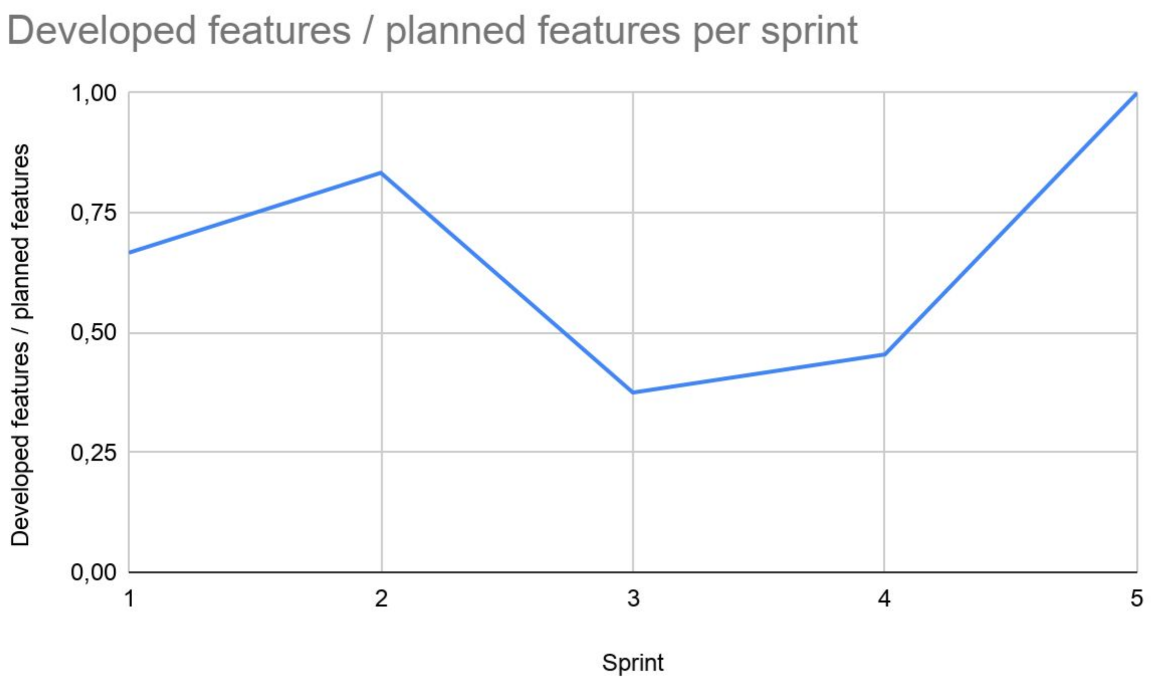


Figure 8. Graph of the median satisfaction level per sprint over the five sprints.

This KPI helped us determine how good our workload was for each sprint. On its own this KPI was not as useful as when used together with the overtime KPI. When looking at the graph for the time KPI and for this KPI (Figure 8) it is clear that we had too much work planned for sprint three and four. This may also have contributed to the low satisfaction levels for those sprints.

All in all the KPIs aided our development a bit. They were not clear guidelines for what needed to be changed in the next sprint. Though through discussion of the KPI results we were able to gain more insight into how well we were progressing and what needed to be done differently in the next sprint.

It would be useful to use KPIs in a future project, however there are probably more useful KPIs than the ones we chose. The satisfaction KPI was probably the most useful as it was the clearest indication of how well the project was going and how much customer value was created. The time KPI and the features KPI could be swapped for something more useful as the time reporting is done already at most workplaces and a quick look at the Scrum board would give the features KPI. However, coming up with good KPIs is difficult.

2 Social Contract and Effort

2.1 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)

We have only created one social contract [3] and that was at the beginning of this project because it was one of the mandatory deliverables. The rules we wrote for the contract were simple and logical. It was quite obvious to follow these rules because it is very general and applicable everywhere. This contract was a good way to agree on our common knowledge, decency and show how to respect every team member. It has been working very well during the course and the rules have been followed with some minor exceptions like not showing up to meetings on time. We have a very supportive environment and have mentioned the social contract in some meetings but not discovered any reason to change it. Therefore it has not been evolved or updated during the project because it has not been necessary.

We think that the reason for why we have not changed the social contract is because we have done so much extra work just to be helpful towards each other. We discussed setting up guide documents and that is what we have done. Because we were all learning a new language, we gave each other tips and showed which tutorials were good. We have team members who have created helpful documents like React Native Guide [4], VS Code Extensions [5], Workflow Proposals [6], Definition of Done [7], Using Material design for our UI [8], Breakdown on Lecture Notes [9] and Final Deliverables Notes [10] just to make it simpler for everyone else. This way of thinking has made this team very efficient and instead of everyone making the same mistakes, we all learn from each other in case one of us makes one mistake.

In the future we would like to continue the concept of writing a mutual social contract when collaborating in a project. We consider it very important and useful when trying to create this safe and comfortable environment. In our situation, we have been very lucky with our current team members and that is a factor for this project being very interesting and fun.

2.2 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 KPI's have been very useful to compare how the time related to how satisfied we were with our deliverables. But it was really hard to measure if we did a good job based on how much work we have done. We did try and create a KPI which was to measure how many planned features we finished per sprint and the issue was that the different features had different sizes of workload. It was still an unfair measurement of how much work we have done because it does not include the sizes of the workload and only the amount of each feature. But what we could measure was time and our own satisfaction level during each sprint. What we did was to take our measurements each Friday at the end of each sprint and when assigning our tasks and user stories, we expected each team member to invest a total of 20 hours per week. If you had less than that you would write a negative number on how much less you had, else you would write it with a positive number if you had exceeded the expected time. The satisfaction level would be about the project progress on a scale from 1-5 where 1 would be “underperforming” and 5 would be “outstanding.”. As you can see on Table 1 you can compare the time spent to the satisfaction level.

Week	3	4	5	6	7
Average overtime worked	-2.57	-4.29	0	+1.46	0
Satisfaction level (out of 5)	+3.97	+4	+2.8	+3	+4

Table 1. Time spent compared to Satisfaction Level.

We can see that during the first weeks we were more satisfied with our progress of the project even though we invested less time than the other weeks. That was because we finished our assignments earlier than expected. We suspect that it was because we were new to using Scrum and had not gained enough experience to put a proper size on each task and user story. We noticed that the more time we put in, the less satisfied we were. This was probably because we got more stuck and had more issues when merging the whole project together. Because at the beginning of the project we focused more on planning and learning the new language and framework. We also started with sketching and creating the GUI which seem more satisfying when it gives a visual reward. But when we focused on developing the functionality and testing our product it was not as rewarding because we were not creating anything new. We had more expectations when creating the application because we felt that we should know more and we wanted to see what we produced. But the thing is that everything was constantly new and we kept on learning during the development.

We can see that we got better at measuring how to divide our tasks more evenly throughout the project. What we learned was how much time it took for each team member to work on each task and applied it from there. We also learned to work with vertical slices and it has been working very well. Mostly it has been as if we worked in parallel but at the very end of the project it was harder because there was not much work left to do. This is most likely because we did not focus on adding any new features for the last sprint. Instead we made sure to have all the implemented features completely finished. This meant that the workload could not be equally balanced between team members because some members would work on finishing the features from previous assignments whilst the other team members who had already finished with their assignments would focus on structuring the final report. It was more important to write instructions on how to write our report to make our work as concurrent as possible rather than starting on new features.

We can see that during the last few weeks we pushed ourselves harder to finish the product and that has also shown in the time we have spent during the final sprints. But during the very last week, our satisfaction level went up even though we put more time into it. That is because we finally felt as the product had finally come together and we had something we were glad to present. What we have been noticing throughout the project is that it was very hard to measure the time for unexpected issues. So we all agreed on having the very last week to only focus to fix eventual problems and finish our testing. Because all team members have different exams that would collide the last week so we pushed ourselves harder to finish earlier.

Overall we have been very satisfied with our product and the work we have done. We would definitely want to use KPI in the future and try to find a better way to measure our deliverables. How we would do this is not just change the measurement but also to divide the features properly and break down the tasks into smaller sizes that eventually becomes a bigger task when being added together. This way it could be possible to measure the amount of tasks that have been finished if the tasks are of the same size. We are still learning and the only way to get closer to our goal is to practice. We will keep on writing user stories and tasks even though our next project might not use Scrum, because it would help ourselves to organize what tasks we have and in that way be able to measure the time we need to finish them. What we can say now is that just because we have not provided the expected result for each sprint does not mean that we have not put enough effort into our work. The time that our work and assignments will take, depends very much on the unexpected issues. That is why the main thing we will take with us, is that time and effort does not always equal results.

3 Design decisions and product structure

3.1 How your design decisions (e.g., choice of APIs, architecture patterns, behaviour) support customer value

When starting off planning for the project, before the first sprint, we decided that we would produce a mobile application using React Native. A mobile application would generally be available to more customers since the market share for mobile devices has steadily been higher than desktop since surpassing it in 2016 [11]. While being out and about collecting trash a mobile application would also be more convenient for the customer since the purpose of our application is to register the trash you have collected. The choice of React Native, and Expo, as our framework for smartphone application development meant our application would work cross-platform without any additional effort. Having a singular codebase run on both Android and iOS meant we did not have to spend time developing and maintaining platform-specific code for two separate codebases in parallel. This decrease in work and avoidance of a partitioned set of knowledge, in turn, meant we could produce new features and provide customer value much more quickly. In the same vein, as part of our “Definition of Done,” it was decided that all components created and all external libraries used must function across both platforms. Our DoD also states that no unnecessary dependencies should be present and the written code should be easily comprehensible. Both of these goals strive for maintainability and make it easier for the same or other developers to add new features or iterate on existing ones for the benefit of the customer. Regarding the user interface, we initially broadly decided that it would be important for the UI to look good and feel good to use in order to encourage the customer to collect more trash and keep them engaged in the long term.

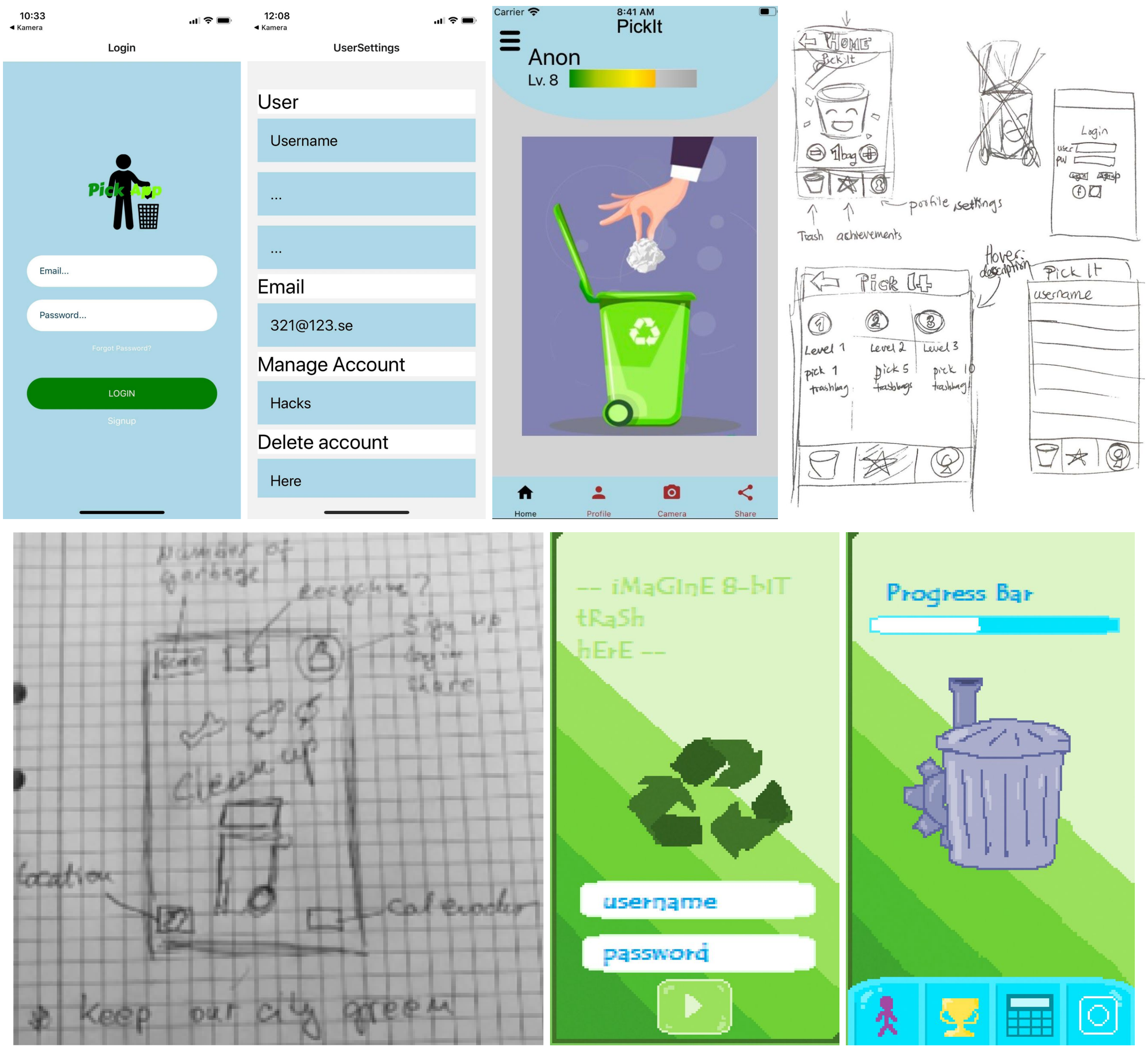
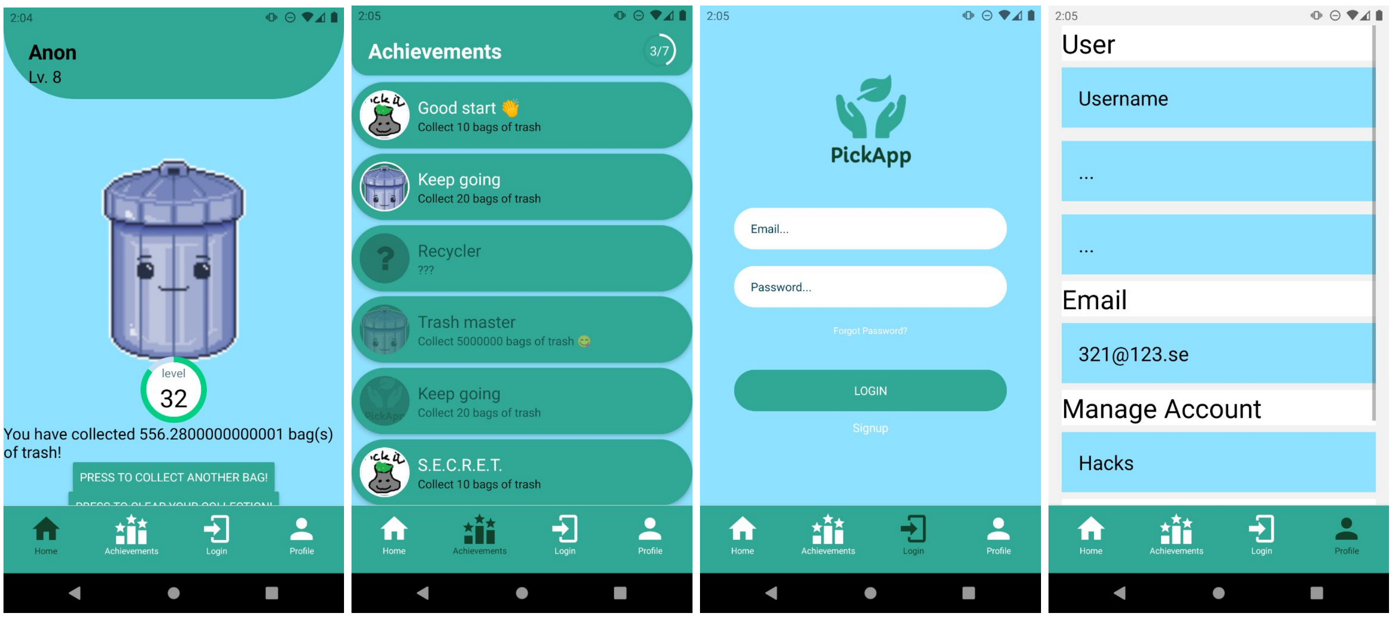


Figure 9. Initial UI prototypes from sprint 1.

After our first sprint we had created graphical prototypes of our user interface (Figure 9) and a temporary UI for collecting trash and sharing the user's progress. From these prototypes we decided upon an initial style, what UI features seemed to work well and what to prioritize first. For navigation we decided to use a bottom navigation bar to provide the customer with quick access to some of the most important screens: the home screen, the achievements screen and the profile & settings screen.

In the UI design of the home screen we decided to have the reward mechanisms, like current level, collection streak and a cute animated trash can, clearly presented to the user in order to encourage picking more trash. It was decided for the achievements screen to display a list of all achievements, what achievements the collector has gained so far and potentially analytics and a leaderboard. These decisions support customer value by rewarding the collector for their work to keep them engaged and entertained.

Finally, a button for sharing your progress and a button for showing information about recycling was decided to be kept but nothing more specific about where to place it was agreed upon at this stage. Having a way for the user to share their progress gives them even more incentive to keep collecting and allowing them to read about recycling gives them context to and reason for their efforts. Plenty of UI features from the prototypes were scrapped, mostly because other features were considered to provide more customer value for the moment. More details can be seen in the document "Meeting 2020-09-18 (GUI Features)" [12].



Anon
Lv. 8
level 32
You have collected 556.2800000000001 bag(s) of trash!

Achievements 3/7
Good start 👏 Collect 10 bags of trash
Keep going Collect 20 bags of trash
Recycler ???
Trash master Collect 5000000 bags of trash 😊
Keep going Collect 20 bags of trash
S.E.C.R.E.T. Collect 10 bags of trash

PickApp
Email...
Password...

User
Username
...
...
Email
321@123.se
Manage Account
Hacks

Figure 10. State of UI after sprint 2.

After sprint 2 we had merged together the prototype user interfaces into a cohesive and fully functioning user experience (Figure 10), with some of the previously planned features like leveling, the achievements screen and the cute trash can animation added.

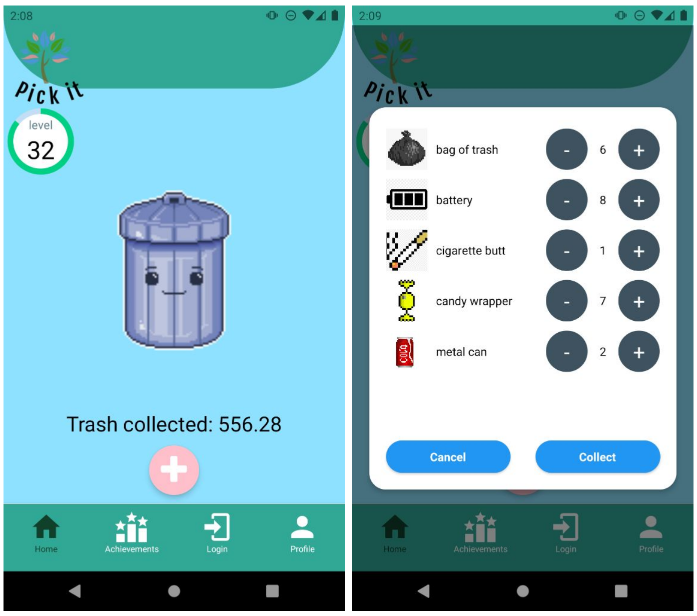
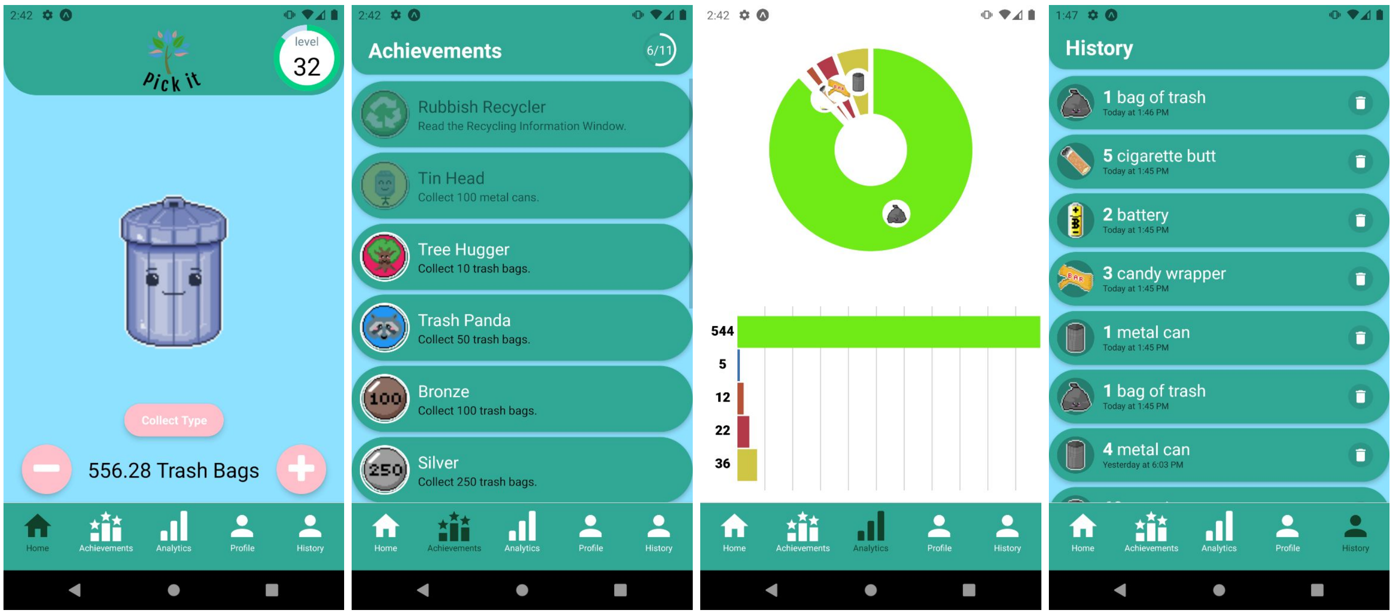


Figure 11. New additions to UI during sprint 3.

At the end of sprint 3 we had focused on improving the trash collection user experience which is at the heart of the application. The user could now choose between multiple different types of trash which they would like to register that they had collected (Figure 11). This more granular approach means users are not discouraged from collecting trash that is smaller than a whole trash bag. Another reason for implementing this feature was that it would allow for a wider variety of rewards, like achievements for specific trash types, and showing collection history or analytics. When designing the UI for this feature we contemplated whether we should add more buttons for collecting directly on the home screen or if a popup would be better. We decided on a popup to not clutter the home screen for the user and also provide the user with the convenience to collect multiple of each trash type at the same time. Finally, to keep this feature future proof and maintainable we made sure to not hard code the set of available trash types, which made this feature much easier to extend and iterate on.

At the end of sprint 3 we additionally decided to scrap all plans relating to user accounts, login and online persistence. This was mainly due to lack of time and a focus on prioritizing user stories which would provide more immediate customer value. We would not have had time to implement the features dependent on authentication within the lifetime of the project. Since all features added at this point already saved the user's progress using local storage, the only added benefit of online persistence would be to allow users to access their progress from multiple devices. This would have required authentication, which we did not have time for, so other user stories providing more immediate customer value were prioritized.



Achievements 6/11

Rubbish Recycler
Read the Recycling Information Window.

Tin Head
Collect 100 metal cans.

Tree Hugger
Collect 10 trash bags.

Trash Panda
Collect 50 trash bags.

Bronze
Collect 100 trash bags.

Silver
Collect 250 trash bags.

History

1 bag of trash
Today at 1:46 PM

5 cigarette butt
Today at 1:45 PM

2 battery
Today at 1:45 PM

3 candy wrapper
Today at 1:45 PM

1 metal can
Today at 1:45 PM

1 bag of trash
Today at 1:45 PM

4 metal can
Yesterday at 6:03 PM

Figure 12. New additions to UI during sprint 4.

Since the ideas about user accounts and login had been scrapped, at the end of sprint 4 the profile & settings screen had been replaced and the login screen was planned to be replaced with two new screens (Figure 12): the analytics screen and the trash collection history screen.

The analytics screen was designed to provide the user with a more detailed breakdown of their progress. It displays how much trash has been collected of each trash type and how much each trash type contributed to the user's total efforts.

The history screen was designed to provide the user with an overview of every collection they have registered over time and a way to undo a collection entry in case it was a mistake. For each collection entry the type of trash collected, how much and at what time the collection was registered is displayed. It was also decided for the collection entries to be listed in reverse chronological order, making it easy for the user to access the most recently registered collections.

New icons for the trash types and achievements were created and added during sprint 4. This serves to provide a more pleasant and appealing UI for the user, which keeps them enthusiastic about picking trash for longer and also providing a more consistent look for the app as it had the same design as the cute trash can.

Finally, continuing on the previous sprint's focus on improving the trash collection user experience, quick buttons for adding and removing trash bags were added to the home screen. When adding the ability to register multiple trash types, as mentioned, we chose a popup over multiple buttons on the home page. Although this has several benefits, we realized this adds multiple extra steps every time the user wants to register another piece of trash. The quick buttons on the home page were added as a compromise to keep the benefits of the popup but also make it easier for the user to register just a single trash bag.

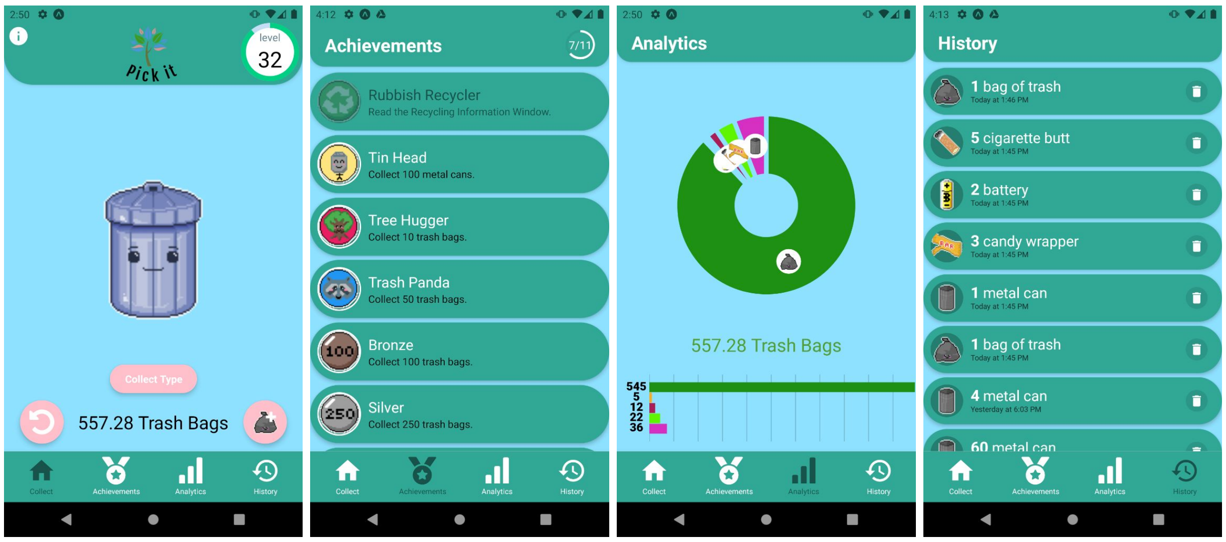


Figure 13. The screens of the final UI.

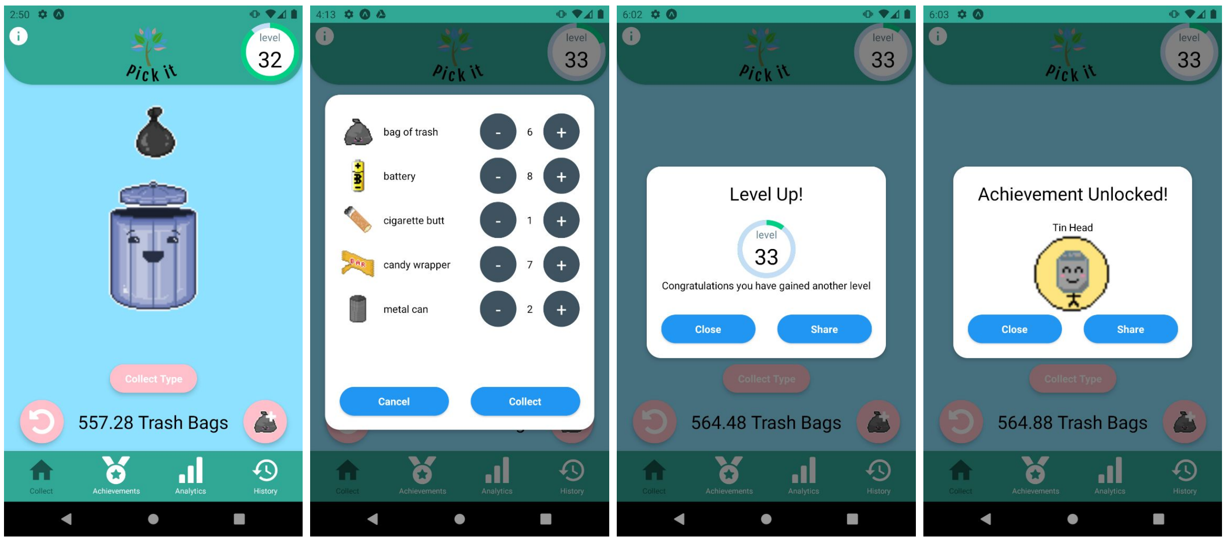
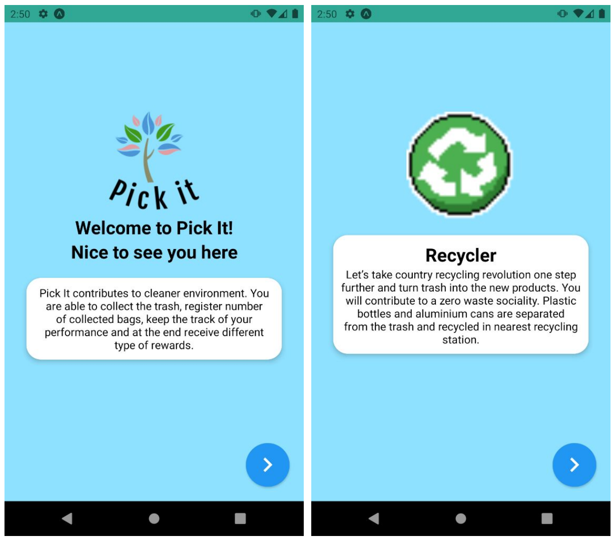


Figure 14. The trash can collection animation and pop ups of the final UI.



Welcome to Pick It!
Nice to see you here

Pick It contributes to cleaner environment. You are able to collect the trash, register number of collected bags, keep the track of your performance and at the end receive different type of rewards.

Recycler

Let's take country recycling revolution one step further and turn trash into the new products. You will contribute to a zero waste sociality. Plastic bottles and aluminium cans are separated from the trash and recycled in nearest recycling station.

Figure 15. The welcome information popup of the final UI.

During our very last sprint we focused on finishing up user stories which were in progress and cleaning up the user interface and experience. We had some recent and some long-standing issues from previous sprints.

As mentioned, the quick buttons for adding and removing single trash bags were added the previous sprint. At this stage the minus button worked but was implemented by abusing the trash collection system and collecting -1 trash bags, which did not work well on the history screen. We did not have any way to determine which trash collection entry in the history should be removed. We contemplated multiple options, for example only enabling the button if the most recent entry was of type "trash bag" with an amount of one. All options we could think of felt like they would cause an unintuitive user experience so we opted to replace the minus button with an undo button (Figure 13) that redirects the user to the history screen. The easiest option would have been to just remove the minus button, but we wanted to keep the balance of having one button on each side of the screen which provides a more appealing user interface for the user. Additionally, the plus button's appearance was changed to make it clearer for the user that it specifically adds a single trash bag.

Furthermore, one major long-standing issue we had at this point was that the achievements, analytics and history screen were not updated without reloading the whole app after registering or undoing trash. Initially this was solved by using the observer pattern, keeping the UI decoupled from the business logic, so the screens could re-render whenever the trash count or achievements were updated. However, due to performance problems degrading the user experience the solution was changed to reload the achievements and trash count every time a new screen was switched to instead.

An information popup about recycling, as planned from the initial UI prototypes, and additionally welcome information about the application was added (Figure 15). The purpose of adding this feature was to better guide new users who use the app for the first time and to give context to and reason for their efforts.

Finally, to motivate the user even more in their trash collecting journey, a popup making the user aware of the rewards they have received was added. It is displayed any time the user levels up or gains an achievement as can be seen in Figure 14.

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

When the initial idea of our project was decided upon, we started off with writing a project specification document containing a short description of how we wanted our application to work, some initial GUI concepts and our first iteration of user stories. We then used this document to prioritize which user stories we should work on. Acceptance criteria were then added to the user stories with the highest priority and documented in this document.

Next up we created a brief workflow instruction document describing what VSCode extensions we should use to keep the code-style uniform. A short document with descriptions of our KPIs was also created.

When we became more familiar with the concepts of Agile we also authored a document containing our Definition of Done. This document was updated at a later date, dropping our requirement for newly implemented code to have 100% statement coverage if the team agreed that statement testing was not appropriate for that part of code.

We also started using GitHub Project Boards to keep track of and update our user stories. Here we included descriptions with each posted user story and also which tasks and acceptance criteria that had to be fulfilled for the user story to be considered done.

Around the end of week 4 we had established a meeting schedule which we kept for the remainder of the project. During our final meeting of each week, on Fridays, we then started to document topics we had discussed, the tracking of our KPIs and also our plans for the upcoming sprint. From then on not much was changed in terms of technical documentation. Each week another document from our Friday meeting was added.

Lastly, a document describing the functionality of our achievement system was written before implementing it in code. This included titles and examples of criteria for how the user should be able to achieve certain achievements. The document also included tasks and acceptance criteria that had to be fulfilled in order for our achievement system to be considered done.

Comparing the first user stories created in the project specification document to the user story described in our document for how we wished to implement achievements, you can clearly see how much improvement has been made when it comes to writing user stories. The user story on how to implement achievement is much more elaborate and thus makes it much easier to get a picture of what we want to have implemented.

Having a project specification is something that we found useful and would reuse in future projects. Having our vision of the final product written down made it easier to come up with ideas for user stories, even if our final product turned out differently than the first version of our product specification. Our experience with writing user stories is also something that we will make use of in future projects. The way they were written during our final sprints were easy to understand and much less horizontal as our first user stories were.

3.3 How you use and update your documentation throughout the sprints

During the first two weeks of our project we focused on writing down our ideas for our application and some project specifications so that everyone had a solid understanding of what we were developing. When we also had settled on using React Native as our framework for developing our application, we created a guide document that our members could use to get started. In this document we shared tips and links to tutorials we found useful for getting started.

When setting up our Scrum framework we also created a workflow proposal template with the purpose of enabling team members to write suggestions on how to improve our workflow. The template is made up of three parts: a “What and when?” part, a “Why?” part and a “How?” part. The “What and when?” section presents the workflow itself and when it will be used. The “Why?” section explains why a workflow should be adopted and why the previous workflow did not work. The “How?” section is an extension of the first section. It explains and teaches how the workflow is performed in practice, like a little tutorial, with potential links to useful resources.

Additionally, we were early with getting started on writing meeting minutes so that in case any members were missing they could catch up by reading this documentation. It has also been beneficial to have the ability to refresh your memory on what had been discussed during meetings with this documentation. We also very early on got into the habit of writing team reflections on Fridays, which later became a part of our Friday meetings.

The third week of our project we created another guide document for which IDE extensions should be used for keeping a uniform coding style.

The way we handled documentation during this course has been greatly beneficial to our project and it is something that we would like to bring with us to future projects. Having tutorial documents created whenever a member found something that could prove useful to other team members and writing summarizations for our meetings is something that we could endorse for future projects. The only downside to our documentation is that we did not set up a proper file system for all our documents making some documentation hard to find which is something we will have to put more effort into maintaining in the future.

3.4 How you ensure code quality and enforce coding standards

One thing that helped us to ensure a good code quality and code style was to define some standards in our Definition of Done. That created a template to follow for the whole team. Whether a user story was permitted to progress in our Scrum board had also a direct relation to our code style. We came up with criteria for each Scrum board column to ensure the quality of our code at each stage of the process.

In progress → Testing:

Here we agreed on that when developing the code and before going to testing we should make sure that the code follows our user stories, works in both iOS/Android, and that no unnecessary dependencies are present.

Testing → Review:

We require 100% statement coverage for all newly implemented code. After we have ensured that our tests have passed and before moving to review, we should make sure that no known bugs exist in the code.

Review → Done:

In the last part we have agreed on two additional basics. The code should be understandable when moving it to Done, and that all tests are passing.

Additionally, we have also concretely used tools that for example made our code formatting unified. The criteria for our Scrum board columns mentioned above made our workflow easier. Since it is our first agile project, we tried to implement the code style part from the knowledge and the recommendations we have learned before in previous courses.

Another way of incrementing our code quality was to write unit tests and UI tests using the Jest testing framework. This way we were confident that the code parts that we have implemented were working smoothly without bugs and errors, which ensured our code quality.

The learnings and outcomes from the project were useful to make the conclusion that defining a stable code style has a positive indirect effect on incrementing customer value. Because the software is continuously updating, defining a good scope of code style standards can make the project in the future easier to maintain and also update, which can lead to an indirect increment of customer value.

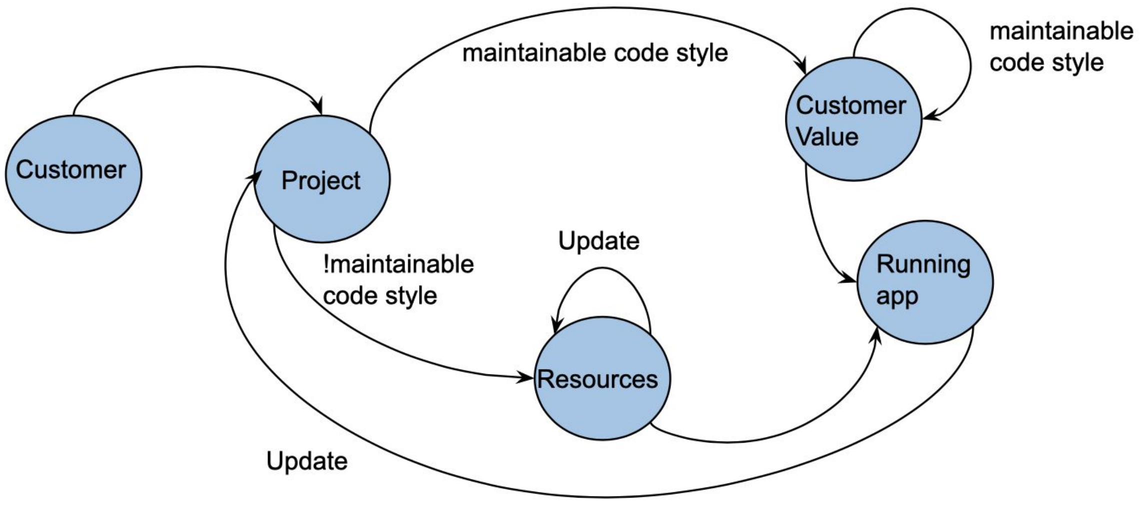


Figure 16. How not maining a good code style could cost us.

Figure 16 shows how not maining a good code style could cost us in the future. Not maintainable code style can lead to big resource costs. Both ways we get a running application, but as shown in the figure above not maintainable code style would lead to at least (n+1) more resource costs. The interesting part would be when updating the application in the future, which will lead to even more resource costs if we choose to update it with not maintainable code style. The part where the maintainable code style is present, gives more customer value due to lower resource cost.

Taking into account the outcome of our project, then we could reason about how to improve our code style from good to better. Since it is our first agile project as mentioned previously, the code style standards we implemented could get improved for every time we think we will work with a project. The transformation of development as we estimate, requires some research of the best practices, principles and standards in agile, in order to produce a maintainable code style.

4 Application of Scrum

4.1 The roles you have used within the team and their impact on your work

To begin with we want to mention that the roles in the agile environment are integral parts, and can never be neglected. Those roles make a base for every development to fit in its place. In our project we did not have any officially determined roles, because we did not know how we would distribute those roles in order to be fair with the whole team. Instead, what we reasoned about was to let the roles take their shape in the future. In this way we made sure that everyone could pick whatever they see fit them the most.

We do believe that because we did not have determined roles, this enabled the whole team to get engaged in the project. The main roles we used in our project were users, product owner, Scrum master and of course team members. The lack of assigned roles made us critical and observant and we started to look with the eyes of all those roles at every point of the development. But indeed we do see maybe advantages and disadvantages with this way of reasoning. In the first place, the disadvantages could be such as missing something substantial when not distributing the roles where every person is responsible for doing something specific, but in the second place we were acting and switching roles whenever it was needed. This was an advantage because we saw all the parts that needed to be seen.

A practical example of this was when we wanted to increment our customer value, we asked ourselves the questions: As a developer, how can I increment the customer value? As a product owner what are the things I want that could increment the customer value for me?

Those questions were repeated under several occasions, and we think that the agile mechanism made us think multiple times about how to achieve good implementation regarding our user stories, in addition to opening the way to think differently and look at our implementations from many different aspects.

4.2 The agile practices you have used and their impact on your work

Throughout this project we have followed a multitude of agile practices and implemented them in various ways. As was mandatory, our main framework for practicing agile has been Scrum. In all aspects of our work we have been mindful of the concept of customer value and how it is affected by our decisions.

The current situation caused by Covid-19, forcing us to work and study remotely, certainly has affected how we have practically implemented the agile practices. In spite of this unusual situation we seem to have managed very well by finding effective solutions for communication and collaboration online. Our main means of communication has been through our group chat on Facebook Messenger and our team’s Discord server. We also initially used Zoom for meetings. However, we later, at week 5, decided to abandon Zoom entirely in favor of Discord. We prefered having a persistent place to join for voice calls instead of an ephemeral room which one team member had to set up for every meeting. It also became somewhat messy to use many different services which served similar purposes so we decided to limit ourselves to one. To schedule meetings and keep track of course-related events we used Trello and to collaborate on documents for meetings and other purposes we used Google Drive.

Our work was timeboxed into iteration cycles of one week (a sprint). To describe desired features through the lens of customer value user stories were used and their progression was tracked through a Scrum board. How our user stories were constructed and impacted our work can be seen in section 1.3. Our Scrum board was implemented using GitHub Project Boards, which we found to be convenient since it was well integrated with other GitHub features such as pull requests and issues. The Scrum board consists of six columns where user stories could be placed: Backlog, Sprint, In progress, Testing, Review and Done. “Backlog” is our product backlog while “Sprint” is our sprint backlog. To keep our Scrum board ready for each new sprint a long meeting was held each Friday where we performed backlog refinement and sprint planning.

During each sprint we have had “daily” (twice a week) Scrum stand-up meetings where each member shared what they are currently working on and if they have any problems. This gave the team greater insight and understanding of where in the development process everyone is and how all members' progress will come together at the end of the sprint. We also made sure to strictly timebox the stand-up meetings to 10 minutes in order to keep them focused on the most pressing issues so as to get them solved and not waste time. This worked well as it made sure we got a status update from each member in a rapid fashion instead of getting stuck on a single issue for a long time.

For multiple of our user stories we have assigned multiple members to collaborate on a single story. Most times this meant that the assignees did some form of pair programming. Due to the team collaborating online, we solved this by using Visual Studio Code’s LiveShare extension which allows multiple people to share a single workspace, terminal etc. and live edit the same files at the same time. Pair programming meant the members working together could share ideas, develop a higher quality solution and help each other learn.

At the end of each sprint, at our long Friday meeting, we would perform a sprint review based on our user stories’ acceptance criteria and our Definition of Done. How the sprint review was performed and influenced our work can be seen in section 4.3. How we evaluated our acceptance criteria, partly through acceptance test-driven development, can be seen in section 1.4. While developing new features, to ensure the newly added code performed as expected, we somewhat performed test-driven development. We did not write tests before implementing a new feature or making changes but we did write unit and UI tests for all major components before we would consider them to be done. More about code quality can be seen in section 3.4. Finally, part of our review process was performed through GitHub’s pull request feature which allowed us to easily review a specific set of changes and accept or reject them. To also make sure the new changes built and passed all tests we had continuous integration set up through GitHub Actions which would run on each pull request. This allowed the author of a pull request and the rest of us to quickly tell if the new changes had broken something.

To conclude, at the end of each sprint, we would perform a sprint retrospective in the form of the team and individual reflection. This allowed us to evaluate the team's process and think about how we could improve the way we work as preparation for the next upcoming sprint.

4.3 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?)

Every week we reviewed our user stories and checked their status. From the beginning of the project we did not have an formally appointed Product Owner but Emanuel was acting as a Product Owner and leading the meetings through agenda. That was very valuable and successful for the team.

We come from different backgrounds that influence the way we perceive the course and what expectations on the learning outcome we have [13]. The meetings we held regularly went very well, we could discuss new ideas, follow up the status, discuss and remove obstacles and find a way forward to support and help each other.

After the discussions we have and review the user stories and process as such, we reached consensus and changed the status of the user stories to the next step. Discussion of the current situation was the first topic of the agenda. We have also discussed and identified the future expectations for the next sprint.

Throughout the whole process we had in mind the main goal of the project and how we could fulfill the customer value and customer expectations. We clearly defined the project goals and elaborated on the way we will achieve these goals. The clear picture of the project was not defined from the beginning due to lack of knowledge and experience but it evolved over the entire product life cycle.

From an early start we addressed the customer expectations and customer value that we will create with our assignment. Developing an application and having in mind the customer perspective helped us to understand what value we will create to the end user and how our application will contribute to the sustainable society. We create a user-friendly solution which will motivate and energize individuals to contribute to sustainable society.

New ideas have been added to the scope of the project on a continuous basis. When we realized that some of the initial ideas were not applicable we revised them throughout the whole project.

After gaining adequate knowledge about different steps in the process, we have been applying them in our ways of working. Sometimes we did re-prioritization because of a task that took more time than we had planned from the beginning or the tasks that we have defined and realized that the definition was not applicable. The user stories were reviewed and we went through the acceptance tests and Definition of Done. When the acceptance criteria and DoD are fulfilled for a particular user story we changed the status of the user story to the next step of the Scrum board. The process we have established as well as roles within the team helped us to get very good communication flow and address different issues on time so no one was behind with their user stories.

The weekly reflection process helped us to get a better understanding of the expectation both from ourselves and from the assignment. The reflection helped us to increase long-term productivity and responsiveness to customer needs [13]. The outcome we have achieved are quite aligned with Intended Learning Outcome ILO1-8 [13], we have identified the complexity which was not known in the beginning of the project and during the process we gained the knowledge and skills about Scrum as well other aspects of the project.

We will definitely apply the process in the coming assignments as well as in the working life. It will help us to be agile in our ways of working and flexible on the execution of the project.

4.4 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)

Our intention from the beginning was to bring as much customer value as possible. We chose to implement our app in React Native in order to bring two runnable versions of application on both iOS and Android. Choosing technologies was a tricky part especially when we have not used those technologies before we did not know how well they would fit our goals and we had to research them. It was not clear either which technology we would use for our Scrum board. After some research we end up structuring our project in a very nice way where we divided our daily schedule technology from Scrum board technology. We used Trello for example in order to arrange all our schedulable meetings.

For every sprint with new user stories, we tried to use the best tools and understand them in order to achieve the desired work quality. We do believe that selecting customized tools and technologies would facilitate the workflow, and even facilitate some of the project structuring. The Scrum board powered by GitHub for example gave us the advantage to structure our user stories and assign them to different group members. Our technology preferences were updated from sprint to sprint, and the reason behind that is because we wanted them to match some certain implementations in our user stories. This made us get into the conclusion that the best practices of using tools and technologies would change and especially at the first sprints. The reason behind those changes in our point of view is the adaptation of those technologies to match every sprint.

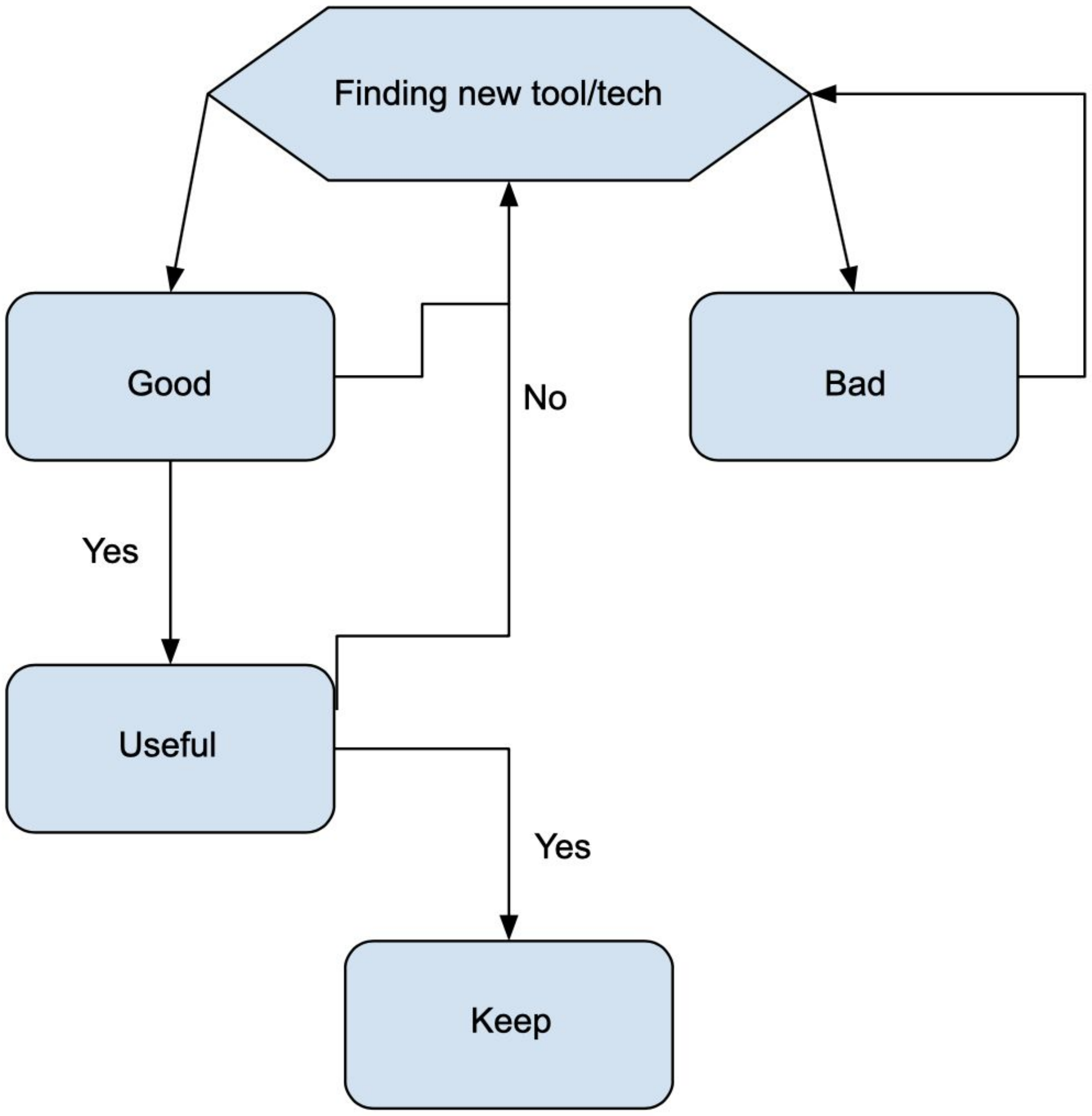


Figure 17. Our approach of finding technologies.

We would even like to introduce our approach of selecting technologies that helped us to draw a line of what could be useful and good, but good and not useful.

One concrete example of this was when we chose to store our data locally using Async Storage. When we realized that there was an API called Realm React Native that was more efficient than Async, we thought about its usefulness at that time, but at that point we had implemented our user stories with AsyncStorage and it was not useful at that stage to change Async to Realm. The usefulness of some technology could be described in two parts. The first part inspects the usefulness of a technology specifically in the current situation. The second part inspects if the technology is useful in general. The flowchart above can describe the both cases of usefulness. Both cases have to give yes in order to keep the new technology.

The standup meetings also were very crucial in our development, and in learning new tools and technologies. It became easier to express and inform the team about every group member's situation during the development and if they had any problems using something new. This led to understanding the team better, and even made us help each other efficiently.

In general, and taking into consideration the things mentioned above, we believe that the best practices and learnings come with applying the agile method often in order to achieve the best results of learnings and practices.

4.5 Relation to literature and guest lectures (how do your reflections relate to what others have to say?)

During the first weeks we read articles as well as watched videos online concerning the agile way of working. The slides we also found very useful when learning about agile and how to go about setting up a framework for our project.

The remainder of the project we did not put much focus on literature nor lectures as there were no guest lectures given during this iteration of this course.

5 References

[1] https://docs.google.com/document/d/10N2m51y-UI_icc_s1KLM7P7vBAkhOyISUnDd8YXtsI4
[2] https://en.wikipedia.org/wiki/Acceptance_test%E2%80%93driven_development
[3]
https://github.com/DIT257-Ginger/DIT257-Ginger/blob/master/documentation/deliverables/Social%20Co
ntract%20-%20Team%20Ginger.pdf
[4] https://docs.google.com/document/d/1J8JTG_sFoXdJfdHbOch-BWOqPruVsA7Q__9hAj7B3Ww
[5] https://docs.google.com/document/d/1MMQhZYLmKNc8LF00SzE19LwEo1-A2HzuZivyfgmcP24
[6] https://docs.google.com/document/d/12cdAM_3y7FKCNh5QSzM5U2dMbYDl77ZoghuyRAqmjzs
[7] https://docs.google.com/document/d/1puYpOwoxCszCW7Oi3Qp4Gmdl-npZ11hLXhiFe8BxkKg
[8] https://docs.google.com/document/d/1dc1iSuqctnfIonfFzv6KUCVbyoB0Q40wY_uHRcoOk2w
[9] https://docs.google.com/document/d/1PKGq71fkwp6mCFVCbV1zN5yAdL-h68HZOw3jDgs2aAg
[10] https://docs.google.com/document/d/1RGE2smsUzQpZ7P9Olet4VFqgdzTWu8--k7zo_PYFr8Q
[11] https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide
[12] https://docs.google.com/document/d/1_7O-plX9HtynK1h50vDWTPTqD1bwG5q-csTJinqxm4U
[13] Bruden, Steghöfer, 2019