Project Donate Blood - Final Report

DAT 257 Group Dagobah

Sigrid Vila Bagaria, Baptiste Duchamp, Lucas Helin, Linus Magnusson, Simon Piau, Christian Zhaco

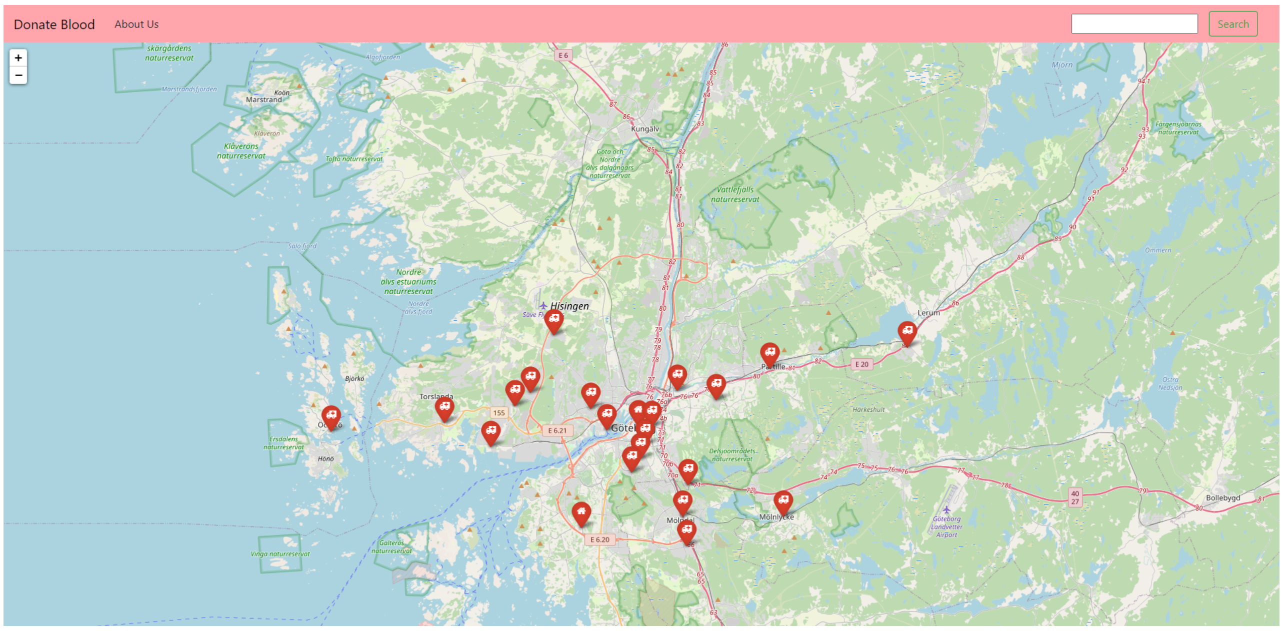


Table of Contents

Customer Value and Scope **2**

Social Contract and Effort **10**

Design decisions and product structure **14**

Application of Scrum **19**

Customer Value and Scope

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

A:

We started off this project discussing what we wanted to create and we decided on an application that collected all the locations in an area where you could donate blood. Once we decided on what we wanted to create, we sat down and concretely decided upon our goals of what we wanted to have as a final product. With a clear picture in mind, we could set up our goals and fairly quickly we decided on our core features that we wanted to implement.

The four main ones were:

- Find nearest place to donate blood
- Interactive map that shows all locations where you can donate blood (hospitals, geblod-clinics, bloodbuses)
- List with all locations where you can donate blood
- Information about each place, (opening hours, distance there)

With this we felt that we had enough to implement so that it would fit into the limited time we had to work on this project and we felt that if we were to implement these functionalities and these functionalities only, we would still have a fully functioning application that we would be satisfied with delivering to customers.

When discussing the idea, we had a couple of different functionalities that we wanted to implement but decided that they weren't really vital to the application so we brainstormed a few potential extra functionalities that would add value to customers but not in the same way as the core ones. This ensured that we always had work to do and even if we were to finish all the core functionalities halfway through the project, we would still have plenty of work to accomplish that added value to customers.

Most of the meetings we had at the start of the sprint lead to us, after careful discussions, deciding on focusing on tasks that would add value to customers. Whilst not solely focusing on these tasks, they were the main ones meaning that we, apart from the first week, could always present something at the end of the sprint which increased the customer satisfaction rating weekly as seen in one of our KPIs. In order to provide the customer with an application with weekly additions, we needed to ensure that the team working on the backend always had a foundation to stand on so not every team member worked exclusively on adding customer value but some did work on adding value to the developers which we were very clear on in the beginning that we had to do. This resulted in us every week having some new feature to present to the customer but not falling behind on the developer side.

We did however have some backlash when we used a tool to create an initial sketch for the application that we were supposed to export to HTML and then import to our application so that we could keep that design. We prioritized the task early on because we wanted to have a sketch to present to the Product Owners(POs) and we received some feedback which we used to better improve the sketch. When we felt satisfied with it, we tried to implement it into our application only to find out that it wasn't possible in the way we thought and all the time we had put into perfecting it was kind of wasted, we did use it as an outline for our design but all the microdetails were not transferable to HTML without huge effort so they were left out of the final design and thus added no value.

B:

Although we feel that what we created and prioritized was in line with what the customers wanted, which means that what we did added value to the customers, we could still have performed a survey or created a questionnaire with questions regarding what is the most important in an application like ours, from their point of view and what they would like to see.

We all agree that the next time we step into uncharted water with projects or by using tools we haven't used before, we will dedicate more time to research. This means looking into how things work, how platforms and tools work together or if what we want to do is even possible. Time put into a task should equal the value it created. To avoid putting time in and not have it return the equivalent amount of effort, we need to know what we are doing and not just wing it and guess our way to our goal.

A→B:

The way that we tried not to waste time or to put too much time into a task that doesn’t create the amount of value we would have wanted it to do. We would do proper research before starting the project or putting a lot of time into a task to then understand that it is not going to work out, which can lead to us having to scrap all the work. Research the platform you’re working on, research the tools, research how it works together with the rest of the project and also research if someone has done something like it before so that you know if it’s possible or not.

This project we did not feel like we created something that the customers did not feel like added value. We can see that for every week, our customers' satisfaction value increased, meaning that we as a group prioritized the same things namely that customers appreciated. It is still something that we can work on until the next time. To beforehand do a more indepth survey with customers to know exactly what they are expecting and to know what they value. Also not to just ask them to rate the progress on a 0-10 system but to have them answer a couple of open questions to gain more feedback which can help you steer in the right direction.

The success criteria for the team in terms of what you want to achieve within the project (this can include the application, but also your learning outcomes, your teamwork, or your effort)

A:

To answer this question concretely, we wanted to create an application that would be launched with all the core functionalities that we discussed and decided on in the first week. We felt that with those functionalities, we would be proud of what we created - an original application that helped improve society by targeting a UN Sustainability Goal. This application and all its features were created, designed and built by us which we all are proud of.

When it came to the smaller choices of what programming language to use, what IDE, what scrum board tool and so on. Our programming language of choice was Python, this was used since we found it to be a middle ground for everyone, some of us had worked with it before and were comfortable with it, some had touched it on a basic level and some of us had never worked with it before but we felt that it would be a fun challenge for us together to use it since it also was a good fit for what we wanted to deliver in this project. We also decided to build the application based on a website since we all felt like this would be a good opportunity to test working with HTML and to connect more than one programming language in a project. The same went for Trello. Python, HTML, Trello are all tools and frameworks that you will encounter when out in the work life so for some to try them out for the first time and for some to improve their abilities with these tools was an easy and obvious choice for us.

We did not really discuss the design of the application, we aren't web designers so we did not put much emphasis on the design. We didn't really discuss it as a main point in our project, we improved it gradually and made sure that every change we implemented looked good but we did not have a concrete goal to work towards with the design of the project. As long as it had all the functionalities we wanted, we were satisfied.

The main focus of this course was to work on software projects in an agile way. This course focused on Scrum which was completely new to all of the members in the team. We agreed to put the recommended time on the scrum meetings, documentation and all that it was. Normally, in a project, everyone is eager to get started and don't focus on preparing, documenting and setting the path for the future. We were a bit eager to start working and skipped a small bit of planning the first week which we then felt wasn't right since we wanted to learn as much as possible from this course and after that started doing everything properly which meant that we did not have any huge problems that occurred due to the proper time allocated to planning.

Our success criteria is to in the end have made the application we set out to make, with all the core functionality that we wanted it to have in the beginning. So far, we have implemented all the core functionality, we have made it look kind of pretty. So as far as the product goes, the team is considering this project a success.
We have never had any teamwork related problems, everything has gone pretty smoothly. No fights or arguments, we have always been on the same page regarding the effort we wanted to put into the course. With our main focus being following the agile process as

precisely as possible. We actually reevaluated our way of thinking somewhere in sprint 2-3, where we went away from focusing on just working with the product to more focus on the agile process. The course is about the agile process, working with scrum in a completely new team. So we wanted to give that aspect the most attention, but at the same time have a product with a reasonable scope that we could accomplish within the time limit we had.

B:

As we briefly stated above, we were a bit eager to get started which meant that we did not focus too much on doing all the work around the project but instead started working on the coding and designing. This led to us lacking a bit of intel in the second sprint which made us reconsider it all and allocate the correct amount of time. In the future we will make sure to respect the recommended structure and follow the plan with all the steps.

We did choose to sail into uncharted waters by choosing to work with tools we haven't done before which left us with a couple of problems along the path but in the end it all worked out as we wanted it to and we gained knowledge and experience in working with these tools. In our next projects, we will most likely again break out of our comfort zones to try out new tools but still respect that you have to learn to walk before you run.

A→B:

Walking before running means that we need to ensure that we have a proper introduction and gain enough knowledge about the tools we are going to use before starting to work with them. Of course you are going to gain the majority of your knowledge by doing and not just reading about it but you still need to respect the process and learn the basics before you start experimenting.

We are going to allocate more time initially to planning than we did in this project. "*Measure twice, cut once*" is a saying which fits in perfectly here. Plan out your work beforehand so you know what you need to do which can help you understand better what you need to research more and this will all help you plan your time more precisely.

One thing that we all agreed to remember going forward is that having a success criteria that is easily defined made the entire project much easier to navigate. Since we knew what our success criteria were from the start, and having a definition of done, we always knew what our end goal was. Having a webpage with the functionality we defined. So once we got there, we could start thinking about going above and beyond our goals. Since in the project we have all done beforehand, it has been hard knowing exactly when you are done, or what is good enough to call done. But since we actually wrote it down, we all knew what the expected end product for us was supposed to be. We had a focus on functionalities, so that is what our main focus for every sprint was. In earlier projects determining what tasks to focus on were never this easy, we had such a clear view on how we wanted to progress our product. So that when we actually had fulfilled all the goals we set out, we knew it was a success and everything else we did was just a bonus on top.

Your user stories in terms of using a standard pattern, acceptance criteria, task breakdown, effort estimation and how this influenced the way you worked and created value

A:

We viewed the user stories as dynamic and after adding our initial few, we knew that there were more that we wanted to add but we agreed to add them when we felt that we had come to that stage of our project. Deciding which ones to initially have was fairly easy since we had already discussed our core functionalities and could use them as a foundation for our user stories since they were basically the main points of our project.

The way we added user stories later on was that we discussed what we needed to do, then looked at our user stories to see where the task would fit. If a task did not fit any of the existing user stories, we created a new one. We then moved the task to that user story and in every case, there were more tasks added later on that would connect to that user story.

At the start of our sprints, we talked to our PO's, reprioritized user stories to then come up with tasks to focus on that week. Either new tasks or ones from our backlogs. We then played planning poker for every task that we had in our Sprint Backlog.

After a week or two we could see a rough estimation of what our velocity would be every week. We had kind of the same amount of points in our tasks the first weeks which we felt were on a good level of what we could deliver. With our velocity calculated, we felt confident in adding as many tasks as possible to our Product Backlog and after playing Planning Poker, we could then sort the list in priority high to low and pick out enough tasks to meet our estimated velocity. As in any project, some roadbumps were hit and some tasks took longer time than estimated and some were finished faster than expected.

We also decided that 1 point equaled 1 hour worth of work and we set an upper limit of 6 points per task. Any task larger than that would have to be divided into at least two tasks since it was not specific enough. This was an effective tool in helping us identify tasks that were too big or that were actually more than one task clumped together.

With every week, we added more data to our KPIs and could re-evaluate our velocity.

To further help with our effort estimation, we recorded our daily amount of time put into the project to use as input data in our KPIs so that we each week could come closer and closer to our projections.

We had stand up meetings every day where we talked about what we were working on, how it was going, any hiccups or not so that everyone knew at which stage of the sprint we were in. We started with this later on in the project based on an idea we got from our TA and it helped us divide tasks or re-evaluate when in a sprint and not just before and after.

B:

We greatly felt the benefits of using KPIs and different scrum tools and in general working in an agile way so we all agree that on our next project, we want to implement as much of the agile way of working as possible.

We will use more rather than fewer KPIs for our projects in the future. This means having a user story dedicated to KPIs as one of your main ones.

This course focused on Scrum but as Jörg mentioned, there are more than one agile method available and it would be fun to in the future try out different ones to compare them and know which ones are more fitting for a specific project.

This means taking full advantage of KPIs, recording as much data as possible to use in KPI so that even if we halfway through a project want to add a new KPI, we still have the data for it tracing back to day one.

A->B:

We are going to implement stand-up meetings in any and all following projects since it proved greatly beneficial to our project. Just knowing at all times throughout a sprint/week what the other group members are doing and if they have a problem with anything feels really good. We always felt like we were on track and that we got everything done each sprint.

We will also make sure to record as much data as possible, keep track of things that we don’t even make use of since you never know when that data can come in handy in later stages of the project. Even if the project doesn’t require us to keep track of different KPIs just having the stand-up meetings and knowing roughly how much time the group spends working on something gives good indications on what we can accomplish following sprints/weeks.

It did kind of feel like a chore and not really worth the effort to keep track of all the data, update the KPIs daily, research which KPIs fit us the best instead of just getting to work in the first sprint. We see the strengths in having some form of KPIs in any project, it doesn’t have to be as strict and formal as in this course, but some kind of measurement that helps us to gauge how we are doing every week/sprint of the project. But it is really important to talk about this during the planning stage of a project, so that the entire team is on the same page in what would be a helpful measurement, and also make sure that every group member follows what is agreed.

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

A:

One part of our acceptance tests were performed with the simulated PO's and with our supervisor, Mazen, who acted like an outside customer. At the end of every sprint, we presented our product for the PO's/Mazen along with what the user story was that provided additional value. If they accepted our new product then we passed part 1 of the acceptance test.
About halfway through the sprints, we had determined to use customer satisfaction as a KPI, and so at every sprint review we all gave our product a rating based on ourselves acting like a customer.

So our acceptance tests were very much done in our own way, with a combination of PO/outside customer acceptance along with ourselves acting as users/customers and giving it a rating.

We never did proper code testing, there were multiple reasons as to why we didn't do it properly along the "rules and codes" we have been taught. For two reasons, first being that no one of us had done a project in python, and secondly no one of us had done any HTML integration coding. So we didn't really know how to write test cases for POST/GET html code. But we did do plenty of testing when we did develop the code, most parts we tested in individual modules before implementing them in our product. To make sure they worked as we had intended and could be integrated seamlessly. Before we merged any tasks with the main branch, we made sure to unit test it and if that worked then we moved on to integration testing the new feature, if that worked flawlessly as well then we moved to system testing it and with all three tests passed, we felt confident enough to merge the newly implemented feature with our main branch.

B:

In an ideal project, we would want to have some outside people acting as customer to give it a rating every week for the customer satisfaction KPI and a real PO (not simulated) that we could show our additional value every week and have them accept or reject it. This would be both random people on the street with varied knowledge of design and development as well as teachers and peers at school.

We would also have wanted to have more detailed feedback to help us steer the design choices in the right direction.

But also make proper tests with good documentation, whenever we implement a new feature. So we have an extended test suite that tests the entire codebase we have written (unit tests and integration tests).

A→B:

For the acceptance tests the main thing that would make it easier for us is to have some group of people outside of the developers who we can show the product after every sprint and ask them to rate it giving the increased value. So we can get some outside data that is more in line with the average user of the finished product.

Basically we would want to have outside PO/users who we can show the product to and get an unbiased rating. Since we simulated being customers, we will of course be biased into giving our product a higher rating.

Instead of just having a 0-10 rating, we could include a questionnaire where people motivate their answers and answer a few questions. This would give us more than a general pointer knowing if we're moving in the right direction or not. We could also include several design mockups for them to choose from in the first sprints to know which one to choose and focus all our energy on.

But also choose a language to code in where everyone is more comfortable, where we know how to write test cases for everything. But this also comes with experience, and being a completely new group where not everyone has the same knowledge. So we had to make a choice in selecting a language that most of us knew, and could work with over a language where a few of us knew “everything”. We chose to go the way that would include all of us.

But in hindsight we probably could have pushed learning how to do proper testing in python higher on our sprint prioritizations. So instead of us doing the testing more informal and in our own way, make sure we have a testsuite that is well documented.

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

A:

From our first sprint, we had to evaluate our work and have the most relevant metrics to keep track of our work and our evolution. We had to be able to monitor our progress in the realization of tasks, the quality of what we produced, satisfaction, speed, etc. To do this we had to choose the 3 most relevant KPIs, however despite a very large number of possible KPIs we had at our disposal only a small number due to the fact that we were both the user, the customer, the managers, the developers and the fact that we produced little data.
We therefore chose basic KPIs that we felt would help us the most: Velocity, Burndown Chart, Test Coverage.

- Velocity was to allow us to estimate our work capacity for each sprint and then to have a precise idea of the workload that could be achieved in the next sprint and also to compare the volume of work between sprints.
- Burndown Chart was also intended to control our work rhythm, to see if there were blockages during a sprint, if we were ahead or behind in the work required.
- Test Coverage was intended to evaluate the quality of our work, more directly focused on the software development work and therefore directly intended for us unlike the first two which are mainly useful for the PO and Scrum Master.

At the beginning we were not able to fully use these indicators as we wanted to except for Velocity. We were thinking of using the Burndown Chart globally for all weeks but after discussions with our TA we decided to use it to track the progress of each sprint individually. This made more sense and was more useful because at the beginning of each sprint we knew where we wanted to go and how we would get there, thus giving more information to the PO for the next sprint for task planning and allowing the Scrum Master to see if there were any blockages during a sprint. For the last KPI, Test Coverage, we were never able to implement it because there was not enough usable data each week and we realized that it was not useful in our code process. That's why we replaced it with Customer Satisfaction. It had several uses, ensuring that the choices made at the beginning of each sprint were the right ones and still provided value to the user and forced us to adapt each week by simulating user feedback, thus pushing us to provide quality work to deliver all the features requested.

B:

We would have liked to have functional KPIs from the beginning so as not to waste time recreating them, and to be able to see our progress from day to day as early as Sprint 1. The choice of KPIs must both have a real added value for the team but also be feasible. This is why in order to be able to achieve the greatest number of metrics and therefore have the choice, it is equally important to keep a maximum of accessible elements that could be used as data for the KPIs if we were to modify the original choice of KPIs. Moreover, these metrics must be used throughout the development process, for the team as well as the PO and the Scrum Master. They should allow for better decision making by highlighting important data, work speed, progress in the sprint, and quality of the work done.

In an ideal project the Scrum Master is the one in charge of keeping track of the KPIs and not each developer individually, so it gets way easier for every developer to just input some data into a sheet or something similar. Then the Scrum Master makes decisions based on the data and KPIs to make sure that the most prioritized user stories gets implemented each sprint. And also so that any eventual setbacks can be dealt with, since there is a clear view of how much time there is remaining in every sprint. So if something needs to be cut the Scrum Master can make that decision.

A→B:

We should have taken a little more time at the beginning of the project to set up the KPIs and make sure of their interest and feasibility. As for their usefulness, nothing has to change, always use the KPIs to help make decisions at the beginning of a sprint and make sure that everything goes well during the sprint.

Even if our next project doesn’t require us keeping track of some KPIs, it's always useful to keep track of our working hours and such when you plan for the next week or sprint. It is really helpful when you want to select what the next priority is and how much work you can actually complete in a sprint/week.

For us, having to combine planning a sprint with also planning each of our school weeks the burndown/velocity KPIs helped immensely. Since at the planning meeting of every sprint we knew how much work we could realistically get done during that week, and this made it easier for everyone to plan their work with the other courses. If we were to work somewhere and they used Scrum we would already have a good idea of how it works and what is to be expected of us.

Social Contract and Effort

Your social contract (Links to an external site.), 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.

A:

Link to the social contract:
https://github.com/fisksoppa/Agile-software-project-management-Dagobah/blob/main/Project_Agreements/Social%20Contract.pdf

In the first week we created our social contract, we wrote it between all the members so all of us agreed on it and we didn't think it lacked anything. Nonetheless, in the fifth week we had a bit of a conflict because there was a bit of overstepping between tasks as someone worked a bit more than the assigned task and crossed into another member's task. To try to avoid that, we added in the social contract that when this happens, it should be communicated to the team member so as not to overstep and work double, which is what the overstepping member did by default).

Apart from this modification mentioned above, we didn't do any other change as the team dynamic was really good throughout the whole project. Everyone did their job following the social contract rules so no conflicts arose.

We decided our social contract would be divided in three sections. Next up, a brief comment on how they worked for us:

- **Meetings:** each team member respected it and attended all the possible meetings, if not all of them or tried to catch up afterwards. There was no one taking advantage of it and not attending.
- **Communication:** everyone made good use of Discord and communicated all that needed to, this way everyone was updated and the project progressed at a good pace.
- **Work:** each team member complied and did everything mentioned above, as proof we have a well used and useful scrum board which reflects all the work done and a GitHub with all the versions and the project in the main branch.

We think we did it in a very optimal way as when we encountered a problem we solved it and added what we found necessary to the social contract. Our group worked without any big conflict, respecting each other and the social contract. This way, we ensured an optimal communication between the team members and in consequence a job well done.

B:

Ideally, we would have thought about the rule we added later on, in the first week. Moreover, for sure there's always things to improve so it would be a good idea to think about the social contract more regularly, this way it would be a dynamic document updated from time to time. Also, to talk about what we could improve and if everyone in the group feels like all the rules have been respected and if some more are needed.

If a problem arises during a sprint, then in the sprint retrospective the team should discuss it and go over the problem along with the social contract. And if needed change something in the social contract to help resolve future conflicts. The goal of a social contract is that everyone should feel comfortable working in the group, and know what is to be expected when interacting with the other group members.

A→B:

To try to improve, we could implement the suggestions mentioned above. At each weekly sprint meeting to talk about more thoroughly if any conflicts have appeared and if any other rule is necessary and in consequence add it to the social contract. Basically, to dedicate a bit of time talking about how everyone felt doing the project.

It doesn't have to be at all sprint reviews but some, we should have revisited the social contract and made sure we all felt comfortable with it. If anyone wanted to add something or change something we could discuss it and come to a resolution together.

We would definitely keep the idea of a social contract for future projects, it makes everyone know what is to be expected when working with that group. But also knowing that if a conflict arises then you will deal with it in a constructive way and not get yelled at or something like that. A good way to set the boundaries and ground rules of working together, especially when we work with people from different backgrounds and norms. In almost all other projects we have done during our University studies the social contracts have been more of a silent understanding. But actually writing it up this time made us comfortable with each other quicker than previous project groups.

The time you have spent on the course and how it relates to what you delivered

A:

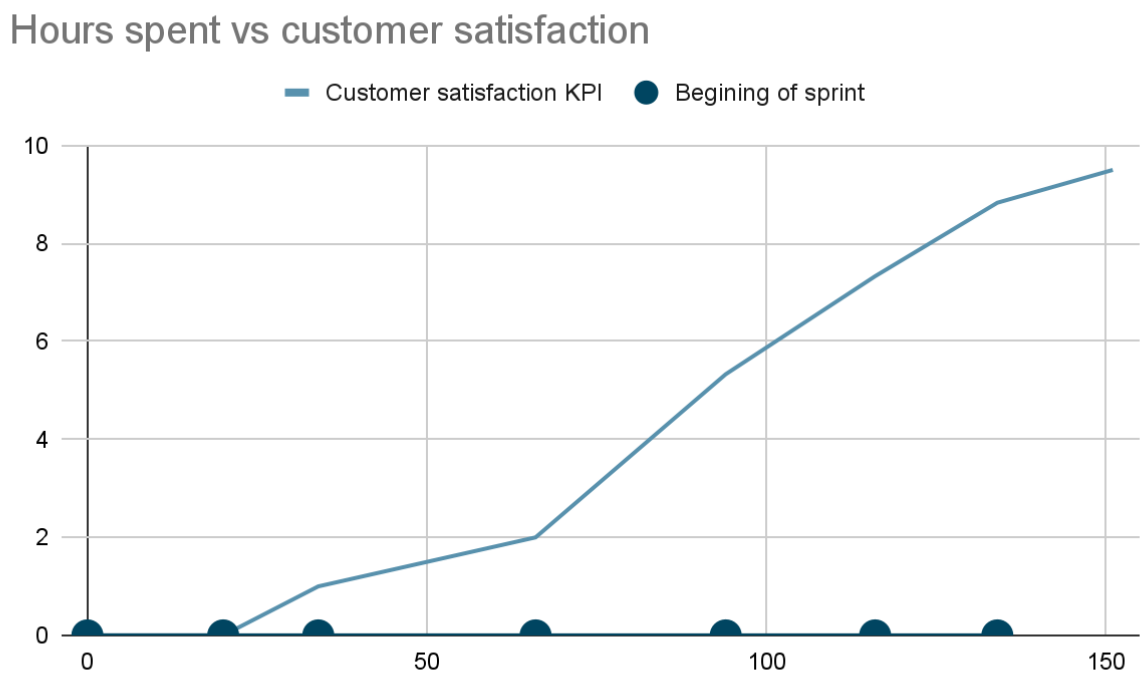
The first two sprints we didn't properly keep track of the time spent and neither did we do a proper estimation of how much we think it would take us. In those weeks we actually completed all the tasks we wanted, but by not keeping track of this information two of our

KPI's (velocity and burndown chart) could not be used to estimate the effort for the next sprints.

From the third sprint we started to do standup meetings to communicate the work we had done every day (in hours) and the problems we encountered. By doing so, we could know how many hours each member put in and also it improved our time-estimation skills as it's easier to understand the capacity of the team. From this third sprint, we kept doing the same each week as this methodology worked for us.

For the last sprint, as we had done a good time planning, we didn't have a lot of work left to do (so we didn't spend a lot of hours on the project). We consider that we were successful in organizing ourselves and the scope of our project.

To even have a better relation on the time we've spent and how it relates to what we delivered we have the Customer Satisfaction KPI, as it is explained in one of the previous questions. With it, each week we knew how much value we delivered to our customers so we could try to get it to the maximum by the last sprint. We almost did as we arrived at 9.5 out of 10. To visualize it, the following graph represents the relation between the hours spent and the customer satisfaction, which can kind of be seen as what the customer thinks about what we delivered:



As we can see, the relation between the time spent and the customer satisfaction (our deliveries) is almost linear if we don't take into account the first week as it was mostly dedicated to doing research. This means we kept a good pace in doing the work and delivering something each week.

Moreover, we also kept track of what we delivered each sprint:

1. We just did research on how to properly start setting the foundations.

2. Blank web
3. A stand-alone map
4. Map implemented on the web-site
5. Search function and list of search result locations
6. Web with final map, search bar, closest 5 locations with information from the searched location.
7. Web with final map, search bar, routes between searched point and closest location, about us page, proper colors for the whole web, closest 5 locations with information from the searched location.

B:

The optimal situation would have been to understand from the first week how the time estimation worked so that way we could use this information for next sprint's time estimation (and for the KPIs). This way, we would have a complete record each week of how much time we spent working and what we delivered, we'd have a direct relationship between both. Not only weekly but daily thanks to the stand up meetings.

Though, if we're going a step further, in an ideal project, we would first do a planning of the hours we would dedicate to each part of the project (sprint planning, work and sprint retrospective) for all the weeks. In the sprint planning we just planned the week, so what we mean is to plan for the whole project with a Gantt diagram, for example. This way, we would know where in time and which ones are the critical tasks that have to be completed before starting another one and also if we're behind schedule or not. Related to this, one of the biggest flaws we had is that with our planning we couldn't know how many hours of work we still needed to complete for the following weeks to complete the project.

In addition, though more related to planning, in an ideal project if all of us had a matching schedule, the planning for all this mentioned above could be done in the first week, and all of the team members would be able to assist.

A → B:

In this case the solution to go from A to B is simple. In the first week we should have spent a bit more time on figuring out what we actually had to do and how we had to do it. It is true we were a bit lost and we just started working, but we should have taken a bit of time and thought about how we would keep track of the spent hours in the project and how to do time estimations so we could use this information in our favor with the KPIs, that way we would have had better time estimations from the beginning and the relation between time spent and what we delivered would have been better.

Moreover, in the first week it would be nice to do a Gannt diagram, spending time in deciding all the tasks of the project and doing an initial time estimation for all of them and distributing them in time. By doing this, we would know exactly the total estimated hours and in consequence it would benefit us because the scope of the project would be more strictly bounded.

As a final thing to mention, to get to the optimal working plan, it would also require for all the team members to have the same schedule, this way the planning and the working hours could be established in an optimal way from the beginning as in reality, most of the time in the meetings not all members couldn't assist due to schedule overlapping.

Design decisions and product structure

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

A:

During our project we used APIs, applications and changed our behavior with the intent to increase customer value. The most obvious one would be our customer satisfaction KPI but we also used velocity and burndown charts which one might argue also works towards an increased customer value. We have also used other applications such as Figma to design our prototype for the website. This way to be able to create several prototypes and not lock on the design directly in our main branch and risk being attached to just a prototype. It's also good to make something hypothetical such as a prototype detached from the main work. We also discussed how we wanted to architecturally design our code but since we used Django the whole project is following the Django structure, which makes it more structured overall. This saved us from having conflicts partly where we felt like different approaches could be better fitting, instead we followed the main structure of Django which worked out very well for us.

Some of these might not directly support customer value but they support us as developers working towards a result which gives better customer value, which is our final goal. It could be velocity or burndown chart KPIs which can speed up the process making the product available faster, which adds to the customer value. It could also be that we somewhat tested the design using Figma which we in turn scraped and changed, further adding to the customer value (hopefully).

B:

Ideally we could have created the prototype much sooner this way we could have started a survey regarding the design choice made. This would not only make more room for adding extra design features but also get a design better suited for our typical customer. We talked about the fact that we scrapped the design prototype/mockup without using it as a picture for us to decide if we liked it or not. We feel like we should have exploited the prototype more given that we spent a lot of time building it.

It would also have been better if we used more diverse KPIs and maybe even more than three. This way we could have included more code based KPIs such as test- and code coverage. This is not to say our KPI choice was bad but we could have at least added one more. We should also have included more architecture patterns.

A→B:

We wanted to exploit the figma prototype, which we could have by asking actual potential customers what they thought about the design or what they would have changed. This way we could have had a better understanding about a very subjective subject such as design. It might even have shown the result that the first iteration of the prototype was very liked and would have made us keep it.

The KPI we used added a lot of customer value, especially customer satisfaction KPI, although it could have been even more accurate by having more “customers” replies. We also discussed using at least one code related KPI since we already have two KPIs related to keeping a schedule.

Furthermore when it comes to the code we should have included more architecture patterns or design patterns in the code. Even though we used the Django framework we should have added more thought into our code to at least follow some simple SOLID principles. The code in itself isn't too badly written but could have been better which applies pretty much always.

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

A:

Since we used a Django framework as our “code skeleton” we never felt the need to draw any diagrams or pictures. It was always straightforward how our code would work between the different modules. It took a few hours in the beginning figuring out exactly how it worked, but we never wrote anything down since once you understood it, it was easy to explain to each other.

We were very consistent in always commenting on the actual code, so that any of us could work on it and understand it.

There is a “how to use it” section in the readme.md in the github repo, and a documentation.txt that shortly describes the important modules of the project and how they function and interact.

What really set us back in regards to documentation and things surrounding it was the small amount of time we had to do any kind of research and a proper plan. Since we as a group have never worked together before, and we all have different experience/knowledge. We had to find some common ground and select a project that we felt was “do-able” in the time given. So we didn’t feel we had time during the first 2-3 weeks of the course to sit down and take our time in making a good extensive plan. It felt like we were always playing catch-up and so documentation was always left out as it didn’t provide any customer value.

B:

Ideally in a project you want to have a clear view of what the project is supposed to look like before you start, how the code is structured, what kind of patterns you are going to use, and start the documentation from the start.

We would have wanted to have more knowledge about the framework, so we decided to use (Django) in order to be able to draw diagrams and pictures if needed. This makes it easy to document everything from the get go, if everyone knows exactly how the program is supposed to work. Anyone who works on a certain task will have an easier time understanding the documentation as well as writing it connected to that task/code.

A→B:

If we would have been given a more concrete task, instead of “make a product”. We could have been more efficient with our project planning and did it extensively. Now we know that for our next projects the planning and research stage is very important and should not be overlooked. But the beginning of this course felt so rushed, we only had a few days to come up with an idea, and then a few more days until our first sprint. Since research was almost never mentioned in the first lectures on how to conduct a sprint, we felt like we were doing something wrong in the first sprint and wanted to start coding as soon as possible.

But now we know that if we take our time to do thorough research and planning in the beginning, it will make everything much clearer and flow better when working through the sprints. Documentation can be updated, diagrams can be drawn if needed.

But all these things come with experience, the more projects we work on the better we will become. Practice makes perfect.

How you use and update your documentation throughout the sprints

A:

To ensure that everyone could work on the project and understand the code written, modify, correct and improve it, we set up a certain process. Before each push on GitHub, the member who wanted to do it should have mentioned the purpose of the modification as a comment for all the other members to know very quickly the progress made by the modification. Moreover, the documentation is so important that before pushing the changes in the group GitHub all functions must be properly commented, stating what is their purpose and behavior. This measure is even more important when treating the Django code and its different files and parameters as it was a new environment for most of us and some of the code’s purpose may not be obvious at first sight.

We also chose to communicate between us (though Discord mostly) to ensure that everyone understood the modification and to be able to ask questions if someone didn't understand the change or its purpose. If it wasn't clear for everyone, we would modify the documentation to add the necessary information.

B:

To get to the optimal way to do it, it is necessary that the documentation of the code is done with each improvement, change or addition of features, not at the end of the project as we mostly did in our project. This way, it's easier for all the team members to keep track and understand all the new changes before moving on to the next addition.

The documentation of the code is not a task, though it's something that is part of all the user stories related to coding so it's a transversal issue. That's why one of the best ways to always be up to date in the documentation is to introduce this notion in the definition of "done". This way, it's not possible to say that a task is finished if it's not documented. By doing so, the team can spend more time working on the functionalities that bring value to the user without having to waste time afterwards simply writing documentation that has very little use for the users.

Of course, it is also necessary to keep updating the other team members about the modifications before doing them and to make sure that it's clear for everyone their use, as we already did.

A→B:

In our future projects it will be necessary to directly take the documentation issue into consideration before even starting to write the first line of code. As mentioned before, we should write documentation for each modification when submitting it and not at the end and add this notion to the definition of 'done'.

This will save us time in the long term and will also allow us to change and improve the functionalities more easily, which is the hallmark of the agile method, to be efficient and ready to face change and evolve. Moreover, we would have to make sure that the documentation is really relevant. Someone who doesn't know the project should be able to take over and continue its development without struggling. We should therefore put ourselves more from an external point of view when we write the documentation and not as a member of the team.

How you ensure code quality and enforce coding standards

A:

To ensure quality work and thus avoid bugs and other code problems, we always worked on a different branch from the main branch (which contained the real version of the project) to ensure that each of the different additions did not create conflicts with previous code or even bugs. Once we were sure that the new additions passed our various tests, we added them to the main branch. It was also common before implementing new features to our github that we used google colaboratory or jupyter notebook to make sure of the feasibility. Indeed, it was easier to test some functions such as the calculation of distance between geographical coordinates, database elaboration, sorting algorithm using these tools because it is easier to use especially for the handling of new libraries and test of operation.

We also set up group work for writing code as well as for reviewing code. This allowed us to share the knowledge we had already acquired but also the new things we discovered during these sessions, thus improving our development training and also avoiding missing bugs, thus directly improving the quality of the code. By repeating this process of peer-review and peer-programming with different members of the team we had very few bugs that were all solved very quickly making us lose very little time. Our methodology slowed us down at times but saved us a lot of time in the end because almost all the time spent coding was used to bring new features and not to fix errors and other undesirable problems.

B:

Code quality can be measured according to 5 criteria: Reliability (system run without failure), Maintainability (ability to maintain the system), Testability (ease of testing the system), Portability (ability to reuse the system in another context), and Reusability (ability to reuse code pieces). To verify that this is respected there are metrics, KPIs dedicated to monitoring the quality of code such as code coverage, test automation proportion, time spent in release... With these code standards it is also necessary to set up code reviews to control the work done with a view to constant improvement. These reviews can be done alone or in a small group, the advantage of being in pairs or triples is that this small team will also increase in competence and thus produce a better quality of code thereafter. We could also keep the code quality up by following the SOLID principles or just some well known design patterns.

A→B:

In the future for new projects, it would be better to take more time at the beginning of the project in the preparation of the project. This includes defining the definition of "done" more clearly, setting up a clear recurring test method before each push to standardize the quality of the code produced. Also think globally about each task, not only about the functionality to be provided. That is to say, think about the possible evolution of the project in the future and therefore leave the possibility of modifying or taking back pieces of code very easily, make pieces of code as universal as possible that we modify slightly to fit our case. To finish it would be preferable that each writing, review of code is done in peer programming, so as to generalize this good practice.

Generally it's about putting more time and thought into every row of code written both in a low level module as well as a high level module.

Application of Scrum

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

A:

Each week we rotated three different roles. The scrum master, two project owners and developers. The project owners had the mission each week to rank the user stories at the beginning of the week according to the most important tasks and the ones that brought the most value to the customer, down to the optional tasks. Their role was also to discuss with the scrum master during the week in order to choose the tasks to prioritize according to the progress of the project. Finally their role was to give a score at the end of each sprint of the value of the project estimated by the customer. This project value allowed us to follow interesting KPIs. The POs had a rotating role, each week one of them was rotating and the other one was keeping the role in order to have a follow-up of the project. So at the end of the 8 sprints, everyone had the opportunity to be PO at least once.

The role of the scrum master was to follow the progress of the tasks by asking each of them if they had any problems and where they were in the project, and then based on the KPIs once they were in place (sprint 3). Depending on the progress of each task the scrum master could choose to prioritize some of them or reposition team members on new tasks if they had finished theirs too quickly. In addition, it allowed us to connect people who had skills to unblock those who were stuck on a problem.

B:

In my next project I would start by giving a fixed role to the scrum master so that he can have an overview of the project and the skills of everyone. Indeed, in order to be able to unblock the team members, the scrum master must know their skills well and to understand and interpret the KPIs it is better to have a point of view on several sprints. The POs will also have a fixed role as they will be the interlocutors representing the customer. We have chosen to create rotating roles so that everyone can become aware of their functions and the expectations related to these roles. Finally, in our next project, we will set up KPIs from the start to allow the scrum master to be more efficient and to make the work of the whole team easier during each sprint.

In general we feel like the applications of scrum roles would have been better if it were more realistic. Given that it's just a 7,5 point course it's maybe not possible but having real customers, real POs and scrum masters would have made it easier to work as just one role in the project.

A→B:

In order to have only one scrum master for the project, we must designate him/her at the beginning. The scrum master doesn't have to be the most competent team member, but a member capable of having an overall vision of the project, capable of knowing the skills of each team member and of analyzing the KPIs. Another option would be to try and have a teacher or outsider act like PO at least, this would make the “pressure” eversomore real. Even better would be if we could have time to make some surveys for any potential customer for our product, this way we get more realistic feedback. This feedback could even change the whole outcome of the project. But as mentioned for a 7,5 point course it's going to be hard to find the time and resources to keep the project and use of scrum realistic.

If the scope of our product would have been more restricted from the beginning, then the supervisors would have been more of an external PO. They would have a better understanding of what the product is supposed to look like, and also be able to provide more “expert” guidance and help. For example if we would have had 3 options of what our end product would be. Then the supervisors would really act like a PO since they know exactly what they expect and what would bring the most value every sprint. This could also possibly make sure that no group ends up in a pitfall where they take on a task too big or too complicated for the time limit.

The open structure of “do what you want” felt like we really had to simulate the PO’s and Scrum master, but in the real world that’s never gonna be the case. There will be a PO that knows exactly what they want and we will be the developers who just work the tasks given to us.

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

A:

We used the SCRUM method for our project. We chose to have a meeting at the beginning of each week to write the user stories. After having written them all, we move on to the assignment of these tasks, starting with the most important one and ending with the optional tasks. We write all the user stories and their assignments on the scrum board (we chose to do it on the Trello application). After writing all these tasks we put a label on them according to the application domain (programming, documentation, design,...). The choice of the order of importance of the user stories is given by the project owners. Our sprint board is separated into several columns, the first one gathering the different elements of the backbone of our project, then the user stories to be done during this sprint or the next one, the ongoing user stories, the finished user stories sprint by sprint. So our Scrum board gives us very quickly an overview of our project, where we are and the tasks to complete in the next sprint.
Our Scrum board is very visual and therefore very practical for the scrum master during his supervision. Indeed, he can easily see the progress of the tasks and can choose to prioritize some of them or to reassign tasks when some are finished too quickly.

We started using planning poker in Sprint 2, and it changed our way of working. Even before starting planning poker, it allows us to review the definition of our user stories. But the application of planning poker allows us to review them in depth. Indeed, since we each estimate the time needed for each task we can see if the task should be broken down into several smaller ones if it is too complex or if it should be merged with others if it is too simple. Planning poker is therefore a good practice that cannot be ignored in the Scrum method because once the user stories are well defined the sprint runs much more smoothly.
Planning poker has other positive points, in fact it allows those who have not understood a concept or an objective very well to see their mistakes and to talk about it with the group. Indeed, when one of the members of the group puts an estimate completely different from the others, it means that the task is not seen in the same way by all the members of the group, so we can discuss it and start a new planning poker. So at the end of this meeting all the tasks are well defined (neither too complex nor too simple) and everyone has in mind what will be done during this sprint and how to do it.

The last best practice we chose to adopt in Sprint 3 is the implementation of stand up meetings. When we set up these meetings, we chose to meet on discord for 5 minutes each day at the end of the day to share the work that had been done and to report problems. Unfortunately we all have busy schedules and meeting every day in addition to the weekly meetings was a bit much. We chose to create a chat room where every day we write down what we did, how long it took and what problems we had. This tool has been very useful. It allowed us to create KPIs so that the scrum master could easily supervise the work done each week. It also allowed us to see who is working on which topic and who we can ask questions to if we are stuck on a topic. Finally, it also allowed us to motivate the team since everyone has a global view of the project progress.

We also chose to rotate the role of the scrum master and product owners so that everyone can practice these roles and understand their functions.

B:

We will be more efficient in project management with the scrum method in our next projects because we will know from the start all the good practices to use from the beginning of the project. So we don't need to wait for sprint 2 to set up the planning poker or sprint 3 for the stand up meetings. For the scrum board we will keep the same way of doing things because it is very visual and everyone can work and modify the board at the same time.

The role of the scrum master was well defined but the fact that it was rotating was not optimal for our project. Indeed, since the scrum master was never the same person every week, he could not follow the project properly. Moreover the role of the scrum master is to put in contact people who have problems on a task with those who are able to solve it. But only a scrum master with a good vision of the skills of each team member can do that. In addition, the scrum master must be able to follow the progress of the project via KPI's but must also be able to compare the weeks between them. So in our next projects we will give the role of scrum master to one person in the team for the whole project.

A→B:

In order to use all the best practices of the scrum method from the start, we will organize a first meeting before the first sprint in order to clarify everything we want to use. This meeting could be held around a table for example, where we discuss and write down the best practices that we choose to use. This way, from the start, all these practices are fixed and we do not lose efficiency during the first sprints.

So in this course we all got to try out the different roles of Scrum, simulating PO's, being the Scrum Master and then the regular developers. In the end after having worked with Scrum for several weeks, our attitude towards Scrum and all the planning and review meetings drastically changed from the first sprint to the last. At first we didn't really see the point and everything surrounding the actual work just felt like chores. But now in the last few sprints we have really felt the benefits of having the entire project so structured. The meetings flowed really well and everyone knew what needed to be done each sprint. Everyone had their tasks assigned at the beginning of the sprint and if we had any problems we let each other know in our daily stand-up meetings. It just felt really easy to work as a group when the entire structure was well-planned and thought out.

For our next project this is definitely something we will be taking with us, maybe not keeping the agile practices as strict as we have had them this course. But doing proper planning at the beginning and documenting every task and every new functionality we wanted to implement and having people assigned to each task really helps a team working well together.

So we can definitely understand why almost every company today uses agile practices, it just makes the work more structured, easier to understand what everyone is doing and it always feels better to work in an environment where everyone knows what to do.

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?)

A:

In our sprint reviews, we went around the room and each person presented briefly what they had been working on this week (think stand-up meeting but a little more in depth) and then we presented the entire sprints end product for the simulated PO's. We also took advantage of Mazen in our weekly supervision, and had him act sort of like a PO/Customer and listened to his feedback.

Then we discussed with the simulated PO's thoughts about the end-of-sprint-product and what they could see as a possibility for next sprints priorities. Then we usually discussed just between ourselves, what we as developers thought were good next steps and next priorities were. If what we and the PO's thought aligned with how we had the priority set on the user stories in the scrum board. We rearranged them if we came to the conclusion that we should prio something further down for next week.

We also individually filled out the simulated customer satisfaction KPI during our sprint reviews.

Then we always did the retrospective part of the sprint review, everyone got a chance to speak, if anything worked especially good and we should try to do that more. Or if something needed to change or was unclear. Since we had 2 simulated PO's and a Scrum Master each week, and we had agreed to focus on doing things "by the book". We let those roles play their part, and for us it worked really well. The 3 of them would always find a good focus for the next sprint. That the entire group agreed with, we didn't have a single disagreement on what the focus for a sprint should be. It was always clear to us and smooth sailing from one sprint to the next.

We also had the Scrum Master collect the KPI's for the week and put them into our KPI-sheet. We also looked at the velocity/burndown charts and talked about that, if we had worked below or above our velocity from the last sprint, why did we do that or why not.

B:

In an ideal world an agile project should have an external PO, and a dedicated Scrum Master throughout the entire project. So that there can be a continuity between the PO and Scrum Master and Scrum Master and developers from sprint to sprint. But since this is a university course, that would be extremely hard to achieve, and find external POs for every group. Aswell as in the first sprint decide on who is the Scrum Master for the entire project.

But having set roles, in external PO a set Scrum Master, the developers knows what is expected of them and who they can go to for questions.

Especially in the sprint reviews, having the same PO for the entire project makes it easier for them to get their will across to the team. The PO knows what would bring the most value every sprint, and should have a clear view of what they expect and can judge the end of the sprint product fairly. Having understanding of what the customers want and expect.

The end goal is to create the product that the customer is happy with. Having a PO that knows what the customer wants and can communicate that to the team and having a Scrum Master who knows the capabilities of the team and can communicate that to the PO is vital for a good end product.

A→B:

We went into this project with the prospects of learning more about the agile process and working with Scrum. So many of the things the ideal project relies on, we had to simulate (external PO, big customer base, set Scrum Master). So that was a challenge in and of itself, but we were very grateful to have Mazen as both a supervisor and our "kind of" external PO.

We feel that we did touch on all the things that were brought up in the lectures regarding the sprint review and sprint retrospectives. The biggest way we could have improved our sprint reviews would have been to have had an external PO, and not two of us being the simulated PO's. Since it got pretty hard to differentiate between being a PO and being just a group member. When we decided on things, even if the PO's had the final word, we almost never had a disagreement between the PO's and the group as a whole. So in that sense the role of PO kind of disappeared a little. If we had an external PO, that had determined the scope of the project for us and knew what he wanted our final product to be. It would have been much easier to tell what the PO wants and what the group wants. Which probably would have given us a better understanding of the roles involved in a Scrum. Since we had to do everything, and just put on different hats at different meetings, sometimes it felt like the roles got mixed up and we were all POs or all Scrum Masters.

But we always followed the "script" of every sprint review so we always discussed what we did well and what we could improve for the next sprint.

Finally, it would be interesting to choose at the beginning of the project and once and for all the PO and DM. The PO according to the customer and the DM according to his skills. And to give them more time to speak and more power during the end of sprint meetings.

Best practices for learning and using new tools and technologies (IDEs, version control, scrum boards etc.; do not only describe which tools you used but focus on how you developed the expertise to use them)

A:

During the project we used a majority of different tools to reach the result we have, some of which were more useful than others. We used Visual studio and Django to build our code since that was what most of us knew, even though some have never used Django. Together with the use of a scrum board on Trello and Github to push and pull code we could start working on our project. The same goes for these tools where half of the team have used them. We did however develop our expertise in them over time and the majority of us very much like Trello even though many felt the contrary at the start. The scrum board at the start was more of a way for us to write what we needed to do but in very big assignments and clumsy. At the end we feel that every assignment was fairly specific and the planning poker accurate. We also felt that our knowledge in both Github and Trello has improved alot.

We did also use some other tool such as Figma which only two in the group had experience with so naturally we assigned them to work on the Figma design. The same went for the HTML code we produced for the website which only one in the group had any experience with.

There were more tools used such as Discord for communication and Google Drive for any type of group assignment like reflections or final report. These tools we all had some expertise in since we have used them many times before. Therefore they were an obvious choice.

B:

The best tools to use during these kinds of projects would be the ones useful in terms of functionality but also tools that the whole team knows how to use. In our case we choose two tools of that kind namely Discord and Google Drive. We should have chosen tools that we all knew how to use or at least had some kind of research made on how to use those tools. We think that this would speed up the process and we would also have learned more on how to use that specific tool. So when we used Figma or started to code using HTML we as a team should have learned these together in order to all be comfortable using these tools. This also shows given that the things we all had to use like Trello and Github we became more experienced with.

We should make sure that every team member is on the same page and know how to use any of these tools, we think this is very important. Given that if someone dont know how to use a tool, that member will not evolve his/hers expertise with that tool. Therefore it's important for that member to reach out for help in order for that person to receive help from any team member.

We would also have learned even more if we kept working with the tools we used so that when we use these tools we have a good base and will learn faster.

A→B:

In order to reach B in this case the means needed to be done is rather self explanatory, as said in B we should have spread the knowledge more amongst the group. Even though it is tempting to use or work with the tools you as a team member are most experienced with, it might not be the best choice. Because if we would have spread the expertise we would have learned more and developed expertise in that area further.

As talked about in B we should as team members ask for help if we don't understand any tool or concept and any member should then feel some responsibility to help that person. This could also be a segment in the sprint review so that if anyone feels embarrassed or discouraged to reach out personally, we have this segment. This segment is there solely to make sure that everyone understands the tools and how to use them. It could for example be the usage of Django, whereas one person can't get it to work and gets a specific error. Another person might have the exact same error and then helps out.

We would also need more time using the tools that we did, this would increase our expertise alot. But unfortunately this course is rather short in time given the big assignment.

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

A:

In the fourth week we had a guest lecture. The lecturer talked about doing research rounds before starting with the project by itself. It felt good to know that is a thing that is actually done in the professional world as we spent the first week just doing research. For us, even though the teachers in the first weeks didn't mention anything about first researching and then working on the project (actually programming), we thought that it was a crucial part in any project, so it was reassuring to know that it was the good way to go.

Nonetheless, as we mentioned in some questions above, we started working on the project with little research done, so during the project we encountered some problems because of that.

Moreover, the lecturer mentioned that it was very important to have a specific idea on how to do the project. At that time, we didn't have that figured out so that's why we started researching. Though, it was also important to have a good scrum board set up to define the project and how we were going to do it, and since we didn't have it (the TA told us we didn't really know how to do it) we were a bit lost at the beginning. Though, in the next weeks, we did some research and we implemented a proper scrum board.

B:

The best scenario would have been to know from the beginning all we needed to know the agile methodology and to be confident using the scrum board, the roles of POs and SM. Moreover, after all of this would be sorted. Do thural research rounds on the fundamentals of the project and all we needed to know before starting, this way everything would be figured out and the problems in between would be minimized. This is because from the beginning we would have a proper specific idea on how to do the project and what to use to do it.

A→B:

To reach the proper way to do things, firstly what we would do differently is to ask more specific questions to the TA to make sure we understand how the scrum method works, as in the beginning we skipped this part and then in the second week we realized we didn't know how to do it. So in conclusion, to ask more. Moreover, to look for this methodology on the internet to make sure we understand it completely and ensure that it makes sense to us.

About the research part, we should start by doing more research rounds, as much as we need to ensure the project is specific enough not to be worried about not knowing what to do or how to use it. Also, from the beginning not to think that it's "bad" not to deliver code every week. Sometimes it is more important to know well what to do than to start working without a proper plan.