OOP Project final report

Group 43

Sem van der Hoeven	svanderhoeven	4896726
Merel Steenbergen	masteenbergen	4784871
Ceren Ugurlu	cugurlu	4851609
Mika Wauben	mlwauben	4834739

General

It's always hard to start something new with people you don't know, but luckily we were all excited to start. After a week of research we were ready to start. We started forming teams, but were a bit overenthusiastic with that. We assigned people to work on the GUI, while we didn't even need a GUI back then. When we realized that, we all focussed on API's and putting up a framework. After that we started planning better and we used the Scrum board. We had some minor communication errors, but we resolved them in a way we were all happy about. We decided to meet at least once a week next to the meeting with the TA to discuss technical matters. We also have a WhatsApp group chat for quick questions. We learned that, in a project, everything has something to do with everything, so you cannot just start to work on something because you want to.

Our planning actually went well. Everyone did their tasks and nothing was ever done too late. We also weren't afraid to ask others for help. Maybe it has something to do with the fact that we were with only four members from week 4. We were all aware that we needed to work a little harder than the other students to make sure our app was finished on time.

We all had a very clear goal in mind. Not only do we want to pass this course, we also want to learn something form it. We wanted to get experience in working in a team and were prepared to try this as best as we could.

Design decisions

We decided to use Spring and Gradle very soon in the project. We found a useful tutorial and thought the Spring framework was exactly what we needed. The Spring documentation was also very useful, so we thought this would be our best option. The Spring documentation gave us the choice between Maven and Gradle, but upon doing some research, we found that Gradles performance overall is better. It was also very clear from the beginning that we would use JavaFX for our GUI, that was recommended in the lecture and just seemed easiest. It took some time to get familiar with Gradle, Spring and JavaFX, but eventually it was easier than doing it all ourselves.

Points for improvement

We have a lot of ideas that we would like to add to our application, but we didn't find the time for all off them. Of course, we could go a lot more in depth on the green actions you could take. We could've added some minigames or levels to add more gamification. Also, maybe not all our code is very efficient, but that might be the effect of working with new things like Spring and Gradle. We are very proud of our test coverage. We found that testing was one of the most important parts of our code. We didn't want any hidden bugs, so testing was very necessary.

We do not really have any security on our code. The password the user types is directly sent to the server in JSON. If we want to make this an application open to public, we have to encrypt the data of the users.

On communication we really have improved. We all now know that a project has its ups and downs and that it is your job to make sure there are more ups than downs. The Scrum board has really helped us in planning and dividing tasks and in seeing what needed to be done for our next demo. Git is very nice to make sure everyone can work on their own part of code, without interfering with other, while you're still working on the same project.

We do feel that there are some points the course can improve on. Especially the start-up was rough. We had no experience with Java servers or on APIs. We know this course is about finding your own way and getting your own information, but still we would like to see a little more help in the beginning. Another thing we noticed is that the TA's weren't always up to date on the info: Some TA's said different things to their groups than ours told us. We don't think this has to do with the TA's, but that it has to do with the management of the course.

Individual feedback

Mika Wauben

Very early it already became clear to me that I did not have to skills required to do this project yet. While the others knew how to do the first demo within the first week, I still had struggles with this piece of code three weeks after we handed it in. By that time we already had to hand in the second demo, but still I was of no use, since I had little idea of what was going on. During this time I made myself useful by creating most of the documentation and by fixing the layout of the repository.

After the sixth week, I finally somewhat understood the code enough to actually contribute something and by now I can work together with my teammates and understand what they are doing and have been doing for the past few weeks.

Since we are only with four people left, communication has been relatively easy. Therefor not many conflicts have occurred. The only time was when Merel and I worked on some piece of code all weekend, which Ceren deleted afterwards, but this was due to a misunderstanding about the design of our application. We communicated this and solved the issue the day after it happened.

Sem van der Hoeven

During this project I have definitely learned a lot. The thing I was most excited/curious about was JavaFX and making a GUI in general. I therefore am very happy that I was able to do most of the GUI coding for this project.

My strong points were that I was very passionate about my tasks. I sometimes spent more time than I'd like to admit on placing elements exactly in a particular spot, or changing the colour of an element ever so slightly to match it perfectly with the rest. As a result of this my weak points were that I sometimes spent (way) too much time on particular tasks, and then I would have to stress in the weekends to get my tasks done for the next meeting.

What I also learned a lot about which I didn't expect to learn a lot about is version control. I found out there are a lot of ways to control and document the entire lifeline of your project, and how to do commit messages and branch names properly. I even found out how some companies do their commits and branches.

Another thing I learned a lot about was working in a team. I learnt a lot about how to cooperate in the best way and what to do/don't do. My weakness on this part was probably wanting to do things myself a lot. I sometimes had to remind myself that particular tasks were assigned to other people. It was unfortunate that we ended up with 4 people instead of 7, but I'm very proud of our group and what we have made.

Merel Steenbergen

One of my biggest pitfalls is wanting to lead. I find it hard to let other people do their tasks without interfering myself. During this project I tried to not be bossy, but to still let my opinion be known. I feel I did succeed in doing this. I did have a hard time sometimes. When my team members had a different opinion than me on the design and technical decisions of the app, I really had to remind myself that I was working in a team and couldn't make all the decisions by myself. It's just not possible to all agree on all small details on a project and I have to get used to this.

Since I did a lot of group projects in high school, I learned a lot about how to resolve problems and I feel that is really going to help me later on. In this project again, I learned that you should always say it if something is bothering you, otherwise you'll just be cropping up frustration and that's good for nobody. You should always keep the goal in mind and let things go once their not relevant anymore.

I did encounter (again) that I really like structure, especially in group projects. Even though it might not seem like it, I'm a very structured person. It's just that half of my schedule is in my head. That's where the challenge is for me: Other people can't read my mind, so I have to let them know what I'm doing and when I'm going to do it. The Scrum board really has helped me in this, since every part was clearly separated, had one or multiple assignees and there was a possibility to add comments. This gave me insight in what others were doing and what I was supposed to be doing. It was really useful.

In short, I worked really hard on my weak points. I think that's also one of my strong points: Always wanting to improve and use skills I acquired earlier. This project has helped me improve myself in group projects and has given me some insight on how working in code development with a team differs from writing code by yourself.

Ceren Ugurlu

The project process was both good and rough for me. I learned many new things and each step we took was a challenge for me. At the beginning of the project, I even did not know the names of the libraries and tools we use. I believe that I had a good contribution to my team. I tried to do my best and I spent a long time doing this. I studied on the server side and database. Also, I generally connected the methods between client and server. In last weeks, I helped my groupmate on GUI as well. I believe that we were a good team. We shared what we had to do according to each group members abilities. We had just four people left, this has both advantages and disadvantages. The disadvantage is when you work with only 4 people, you have more responsibilities. But at the same time, it is a really big advantage for communication. To communicate is easier with fewer people. So the communication in our group developed. Therefore, I do not think that we encountered any big communication problem. We always tried to inform each other on all steps. We were all open to new ideas and gave suggestions to each other. I am happy about the result.

Value sensitive design

This project was the first time we had to think about responsible computer science, or ethics. There are multiple sides to this. We will start of with data security. We can immediately admit that we didn't really think about security. Passwords are now sent from client to server in JSON, they're human readable. We mentioned this earlier in the report: If we want to make this application open to the public, we need to encrypt the user data. This is not only to prevent hacking, but it's also the law. We would also need a 'terms and conditions' to make let users know what we do with their data. To obey the GPDR law, we need to change and add a lot of things.

Of course, our application itself does help people improve themselves. The app contributes to the awareness about CO2-emissions. With gamification it encourages users to think their daily life choices. If this application would become popular to the public, it might make a difference in the attitude of the people towards the environment. If we really want to elaborate, we could add a pane that gives you information about the average CO2-emissions of different stores.

What should be in the final report? (2)

3. Points for improvement
   1. How can your software be improved (testing, GUI, code quality, features, …)
   2. How can the process/collaboration be improved?
   3. How can the course be improved?
   4. …
4. Individual feedback
   1. Each team member reflects on how he/she functioned in the team (at least 200 words per team member)
      - What were your stronger/weaker points during this project
      - Did you have conflicts with other team members? How did you solve them?

What should be in the final report? (1)

1. General: how did the project go?
   1. Did you manage to stick to the planning (why (not))
   2. How did the collaboration in team go?
   3. How did you communicate?
   4. How did version control help (if at all)
   5. What did you learn?
   6. …
2. Design decisions:
   1. What are some of the major decisions that you have taken as a team?
   2. Which technological choices did you make and why? (e.g., Swing versus JavaFX, yes/no abstract base class, …)
   3. …

What should be in the final report? (3)

5. Value Sensitive Design (see later)