

# OOPP Report

## Group 80

Irem Ugurlu - 4851625

Jaron Rosenberg - 4839641

Natalia Struharova - 4935519

Robert Mînea - 4848993

Nick Ouwerkerk - 4957032

Mayasa Quacqess - 4898109

Lee Chen – 4703812

## Table of context

Product.....	3
Process.....	4
Reflection.....	5
Individual feedback.....	6
Value Sensitive Design.....	7

## Product

In relation to the context of the project, to give the user a way of saving certain actions that reduce their carbon footprint, our group made various technological, architectural and aesthetical design choices.

In terms of aesthetics, we decided to use green as a primary colour in the application. The colour green is usually associated with nature and since the project is mainly about preserving nature, we found this to be the perfect fit.

For the name of the application, our group decided on the name 'Greener'. It is similar to the name of the project, and with just one word it gives a clear idea to the user what the application generally is about.

We also made various technological choices. For example, we decided to use the 'jersey' library for the server and client code. This library seemed to have a clear documentation. Furthermore, this library seemed to be somewhat popular, which made it easy to search for examples and look for help on the internet.

To deploy the server, we used a service called 'glassfish', mainly because, again, there were numerous tutorials online on how to deploy a server and how to configure the server settings. Because none of us had experience with setting up a server, we wanted to keep it as simple as possible such that the resulting code was easy to follow for all team members. We also tried to use the service 'Tomcat' to deploy the server but since this did not work all the time for unknown reason, we decided to switch to glassfish.

To make the Graphical User Interface (GUI), the group decided to use JavaFX. We did not have a predetermined preference for the application used to make the GUI. After looking at JavaFX we already had a basic idea of how to use this application to build our GUI, which made us decide we would use JavaFX.

On the server-side of the application, we decided to give every feature their own class to maintain clarity and make it clear where all the coding should be done. It also makes it easier for more than one person to work on the server-side.

## Process

Usually, we made a fairly strict planning as to what should be done in that week. Because of this, it was very clear what each team member had to do, which made us able to stick to the planning we made most of the time. There were also moments where team members needed help from others, but there has never been a time where certain work was not completed at the given deadline.

The collaboration in the team went well. We promised each other to ask for help whenever we were stuck on a certain element for too long, and all team members seem to have kept this promise. Not everyone was as talkative as the other, but when a team member was stuck with a certain problem the rest of the group would know reasonably fast.

Furthermore, everyone was willing to help each other with their difficulties. There were numerous times where group members or even team members working on a completely different part of the application were willing to help each other when somebody was stuck with their work. This led to a pleasant working environment.

We did most of the communication while being together at school. This way it is easier to discuss unclear aspects with the other teammates. If this was not possible, we used WhatsApp for short questions and meeting appointments. For questions which were more substantive, so more about the code or other parts of the application, we communicated through voice program called Discord. Although solving merge conflicts did end up being very difficult if not seeing the others face-to-face, even through Discord, this program was still very helpful if team members had very specific questions that others were able to solve, even when not sitting together at school.

These methods made it so we had a lot of room and means to communicate with each other. We set up various meetings throughout each week, because we discovered that even though we had multiple ways of communicating with each other, seeing each other face-to-face made us make the most progress. During the last few weeks we even met up most of the days.

Version control did end up helping us throughout the project. We did not have to go back to certain commits most of the time, but we still needed previous versions of our work occasionally and without version control, this would have been (almost) impossible. The main feature we used of version control was the branching. This made it very easy for team members to work on their respective features they are implementing without interfering the work of others. We did encounter some problems with solving merge conflicts, but these were usually dealt with accordingly.

During our project we have learned a lot, as group and as individuals. We gained experience by learning how to work with libraries, Maven, and glassfish. Since almost none of us had any experience with coding and even working in general in such a large group, we also learned how important good communication is to get everyone in the group on the same page. Everyone has different ideas as to how the project should elapse, and we had to make compromises to make everyone satisfied with the work flow.

## Reflection

Overall, we are very happy with the end product. Nevertheless, there were certain things that needed improvement in our eyes.

First of all, there are aspects of the created software we would have improved or done differently. An example of this would be the tests. Although our tests cover most if not all of the necessary components of our software, our tests could have definitely benefitted from even more in-depth tests which would cover unlikely errors in the software. On the other hand, Mockito is used for a portion of the tests which adds to the effectiveness and reliability of the tests.

Overall, we are satisfied with the features we have implemented. We did have a lot of extra idea's but due to time constraints we were not able to implement most of these extra features. Although we did expect this, we still are disappointed we were not able to implement these features.

Other aspects which we though had room for improvement is the communication, collaboration and the process. We do believe we should have met up more during the first weeks of the project. As said earlier, working with each other face-to-face let to better results in our group, so if we would have met up more probably could have had more work done.

What we are pretty happy with is the code quality. For every merge into the master branch, we had multiple people check the code and give feedback on the given code. If there were any merge conflicts to be solved, we usually did this together. This way, everyone was satisfied with what was being merged into the master branch.

We consider the process of the project to be structured because of the deadlines we set. There was not someone who did not finished his or her work if a certain deadline was set for this task. This led to everything being finished when we wanted it to be finished. Nevertheless, most of the deadlines we set were at the end of each week, which we could have spread over the week for better distribution of work throughout the week. We did not end up having negative effects of the set-up of our process, but we recognise the fact that our approach definitely was not perfect. A better division of work over the week could have alerted us of certain problems earlier which would enable us to fix these problems at an earlier stage.

Lastly, we also have some feedback on the course itself. We would have liked to see the lecture about Git and GitLab earlier during the course. This lecture was given 2 weeks into the semester, but at that point we already had to be familiar to Git to be able to do coding. It would be more useful to do this lecture in the first week of the semester.

## Individual feedback

### Nick Ouwerkerk:

In this project, I tried to improve my flaws given in the personal development plan, while also using my strengths in favour of the project. The flaws the I presented in my personal development were communication and focussing too much on one element. During the project, I did feel like I improved my communication skills, although I am also aware they are still far from perfect. I tried to bring up any problems I had as quickly as possible. This went reasonably well.

I have also seen improvement in my coding. I learned a lot about libraries and tools to help me with programming, and with the help of CheckStyle and my team member's code I have definitely improved my coding style. I also took a lot of time to look at the other's coding. This helped me creating some good code habits.

The one main problem with the project I had is the fact that I found it to be very difficult. I had no previous coding experience, so I was not sure what to expect, and during the first few weeks of the project I was taken aback from the difficulty of everything. Everything was new to me, which made me have to spend a lot of time studying the different elements we were going to use for the project.

### Robert Mînea:

The project had its up and downs, but unfortunately, the latter were a lot more preponderant. For a start, I have to say that I have invested "too much" time in the project, and that was mainly because of logistic-related issues. First I think that the project itself was very demanding, having in consideration that we had no experience in doing anything server-related on java, and working with project-management tools like maven. Some tutorials, guidelines and advices would've been invaluable because we have literally wasted weeks on solving bugs and researching online for various documentation instead of actually doing any work, I am not saying that everything should be handed to us, because that just fails the purpose of learning how to research, but some guidelines and advices would've been more than helpful. I can say that I've learned multiple things while working on this project, including server/client-side communication, using Mockito for testing (even though my personal opinion on mocking isn't that positive), and working with the GUI while always having in mind how to make the link between it, Web Service and the client methods that consume this Service. Even though I didn't find it necessarily pleasant to always split the tasks for our teammates and coordinate the team, I have to say that I've learned a lot about time management and project planning, but I would've been very happy to also see my teammates as active as me in planning. I usually like situations in which the group works as a more cohesive entity and everyone volunteers to research and work on different things, rather than waiting for others to take initiative, but after all, every person has his/her own circumstances and they must be taken into account when planning the project as a whole. All in all, I have to say that I have improved my time management, project planning skills, and learned a lot about app developing in java (both client and server-side), but at the cost of wasting too much time on things like bugs, build failures, git errors (most of these enumerations not being our fault) and spending a lot of time discerning information about a subject that is not related at all to Computer Science (because usually the programmers are helped by people specialized in the specific area of the project e.g. Environment).

## Value Sensitive Design

'Greener' in a program mainly used to give people insight into their carbon footprint and give ideas as to how to reduce this carbon footprint. Therefore, the main values of this application would be environmental-friendliness, efficiency and conservation of nature. But if the design would cater towards other values, without changing the main functionality of the application, both the process of building the application and certain aspects of the program would be different.

The application could also be used as a social media and connectivity platform. This would add to the values of the application the improvement of social networking, the improvement of social relations and the increasement of the feeling togetherness in society. The main concept of this application would be to reduce the carbon footprint together with people all over the world.

The main audience for the application would probably change with adding this value. Users who will get the most out of the application would be people who want to lower their carbon footprint, but also would like to do this together with others. Not everyone has enough free time to chat and meet up with others, so the audience will probably become more niche.

Of course, if the application would want to implement these values into the software, changes in the process and the end product are to be made. Social functionalities like chatrooms, comparing scores and sharing media would be much higher on the priority list for the application. The application would also need some functionalities that are not present in the current program, like the aforementioned chatroom and the ability to share media, but another example would be a possibility to see other profiles and add friends based on shared hobbies or living near each other.

The main sources that would be consulted in order to gain theoretical insights into the stakeholders would be surveys distributed to the user of the application. Since the theoretical application would mainly be used by people who want to connect with others and do activities together, one way of exploring the value of social connectivity would be to ask users how they prefer to connect with others. Another source could be statistics on the activities which most time is spend on during leisure time.

As said before, the application would have the main value be improve social networking, but the value of environmental-friendliness still stands. These two values can be in tension. For example, if two people who became friend through the application want to meet each other, they will have to visit each other by car or by public transport, depending on the distance between the two. This conflicts with the value of environmental-friendliness. In general, social networking could lead an increase of carbon dioxide emission, which creates tension with the value of environmental-friendliness.

The perfect situation for this theoretical application would be to connect people and emit as small amount of carbon dioxide as possible. Through the application, this could be encouraged. In a way, we hope that the user would be as environmental-friendly as possible and watch their emission. A hypothetical design solution would be to add carpooling to the application. This way, even if their needs to be travelled by car, less cars will be used and less carbon dioxide will be emitted, while still being able to visit your peers on the application.