

OOPP Final Report

Project Name:

SustainMe

Team:

Group 70

Contributors:

Nadyne Aretz	(4649087)
Matthijs Arnoldus	(4928091)
Roderick de Britto Heemskerk	(4961269)
Jesse Nieland	(4880722)
Roman Sirokov	(4769740)

Date:

April 11, 2019

Table of Contents

1. Introduction	3
2. Process	3
3. Product	4
3.1. MVC	4
3.2. Server & Database	4
3.3. Graphical User Interface	5
3.4. CO2 API	5
4. Reflection	6
5. Individual Feedback	7
5.1 Nadyne Aretz	7
5.2 Matthijs Arnoldus	7
5.3 Jesse Nieland	8
5.4 Roman Sirokov	8
5.5 Roderick de Britto Heemskerk	8
6. Value Sensitive Design	9
7. Summary	10
References	11
Appendices	12

1. Introduction

For the past seven weeks, we worked on making a Java application to encourage people to become more co2-friendly and to track their progress. In order to be able to track the progress of the co2 reduction, we implemented several features as tracking vegetarian meals, use of public transportation, solar panel use, etc. To encourage people to be more active in tracking their progress, we implemented a gamification system.

In this report we will discuss our process while making the application, our product, the reflection on our work but also the course, individual feedback, and value sensitive design.

In the paragraph surrounding our product, we will briefly discuss what kind of design choices we made. For example what kind of libraries we used and why, what kind of project structure we used, but also what kind of architectural design choices we made.

In the reflection section of our report, we will slightly touch upon what kind of problems occurred during the project and how we dealt with those problems. We will also discuss what we could have done better design-wise, but also schedule-wise.

We will also reflect individually on our contributions and what our stronger and weaker points were during the project. In this last part, we will also state our initial personal goals and stronger and weaker points and reflect if we achieved our goals and if our view of our stronger and weaker points changed.

At the end, we will talk about value sensitive design and how we implemented it in our project.

2. Process

Before starting the project development, we focused on forming a goal that the entire team would agree on. Eventually, we formed an overview of the core values of the application; we wanted to make an application that makes people more aware of being eco-friendly. We decided that the people for whom we wanted to make the app, were people like friends and family.

Later on, we started working on setting up our project with git and assuring that all of the programs that were to be used worked on everyone's computer. We completed other core tasks on gitlab, such as the SCRUM board and the readme. We also made sure that everybody understood how to work with the tools we had. This was especially important for git. As a communication tool, we decided to use Slack, since it is very common to use for developers.

Next, we configured a continuous integration with three different stages on Gitlab. We made a build stage to check if all our code would compile, we made a test stage wherein the tests run and check if everything works how it should work, and lastly, we made a checkstyle stage to check if we did not have any checkstyle violations.

Eventually, our project had five main logical parts: server, Firebase services, GUI, controllers & models (CO2 API, other calculations), and the fractal tree (an extra feature that required a lot of time and effort). We split the team so that

the development would be much more efficient and everyone could contribute to doing a particular part of the application.

The process kept consistent during the whole period of development. We helped each other solving the issues if needed and spoke out the ideas we had. A significant role in the development within the team, undoubtedly, played Git versioning system. During the process, we tried to use it the proper way; making clear messages, mentioning the issues from the issues/SCRUM board, using clear names for branches, used continuous integration, etc (see Appendices A, B & C for a Pull Request example). We had some conflicting merge requests a few times, but they were always successfully resolved.

3. Product

3.1. MVC

In the beginning, we made an emphasis on creating a comprehensive MVC (*model, view, controller*) structure for a few particular reasons. Firstly, a MVC structure has a clear and intuitive understanding of the places where a certain logic is located. Therefore the project would be highly maintainable. Secondly, the portation of the application on other platforms would be handled much easier due to the separation of the view module from the business logic. Since Observer/Observable interfaces, that we were taught about during CSE1100 course, were deprecated in Java 9¹, we decided to use a custom system, that turned out to serve its purpose well.

It is worth mentioning that the GUI solution that we used, JavaFX, required additional workaround in order to be compliant with the MVC ideology of view separation. By default, JavaFX requires the creation of its own kind of controller. This controller is highly view dependent, meaning that there are a lot of annotations and functions that are meant to be from JavaFX. Nevertheless, we created the system in such a way that all JavaFX related classes are located in a corresponding view package. Thus, neither the controllers, nor the models know about JavaFX, but they communicate via general view interfaces. So, if this JavaFX view implementation package would be replaced by the equivalent android representation, it would not require any changes in other modules.

3.2. Server & Database

After some discussions and considerations among the team, we came up with an idea to use Google's Firebase². The permission/approval from the TA was retrieved, but unfortunately, in the end, it caused us a lot of non-technical (miscommunication) problems. We had to redo most of the work with Firebase and introduce an additional framework, Spring Framework³, for the server part of the application. It is a popular and reliable solution that fits our needs. Since we had to change our server quickly from Firebase to another server framework in order to continue with the demos deployments, we had to find a new framework fast. We have also considered Jersey⁴. They were both good options for our purposes, which was creating a web-api based server. However, we chose the Spring Framework, since it had some extra features apart from the restful web api we used, such as its own testing framework and dependency management, which is similar to singletons and makes sure that you do not need to create many objects of one class, which would

¹ Java Documentation. (n.d.). Observable. Retrieved April 10, 2019, from <https://docs.oracle.com/javase/9/docs/api/java/util/Observable.html>.

² Google Firebase. (2010). Firebase. Retrieved April 10, 2019, from <https://firebase.google.com>.

³ Spring by Pivotal. (2003). Spring. Retrieved April 10, 2019, from <http://spring.io>.

⁴ Jersey. (2010). Jersey. Retrieved April 10, 2019, from <https://jersey.github.io/>.

waste memory. The testing framework proved to be useful, since it really helped us test the client-server connection. In the end, Spring showed itself to be a useful framework for the server-side of the application.

Firebase was chosen for various instant benefits, as well as for some potential ones (long-term). Instant benefits implied the existence of valuable services such as database and authentication. They are fast, secured (HTTPS and encryption), flexible and reliable. Moreover, it is free. It also has a thorough development console online that would allow us to access the data and modify it quickly; this helped us a lot during the development, especially with the testing. Another benefit was that from the beginning all team members could access and manage the same database instance (no local databases).

Moreover, there are some potential benefits. For example, a wide range of other valuable services, such as A/B testing infrastructure, notification system, audience analytics, technical (crash) analytics, remote configs, etc. Important to say that Google itself ensures scalability of this system both horizontally and vertically. It means that in case of tremendous success, we would need to maintain neither the software nor the hardware.

We decided to choose Firebase, because of all of these benefits; both potential and instant. It has a vast set of tools that can be used for very efficient development; while Heroku, for example, is a nice service that could be used instead, but it does not have as many useful tools. Also one of our team members had prior experience with Firebase, so that also played a significant role in our decision making.

There are some disadvantages to Firebase as well. Firstly, its DBMS is NoSQL only. Thus, it may largely depend on a specific use case as well as the personal preferences of the developers. Secondly, the services have relatively low support of Java's desktop app development in comparison to mobile devices.

3.3. Graphical User Interface

The Graphical User Interface plays a massive role in the user-centric applications. However, we could not pay a lot of effort and time to make every single element perfect and from scratch. We looked at different options but ended up with JavaFX⁵. JavaFX is more modern than, for example, Swing and AWT. It provided us with a fast paced development and a relatively high level of customization. One example of this is CSS, which was very handy for the views in our project. JavaFX is also supposed to replace Swing as the standard GUI library for Java SE, it is basically the successor of it, and as Swing is build on AWT, it is possible to say that JavaFX is a third generation GUI platform.

We started working with scene builder which is an interface for working with JavaFX and makes it much more convenient to work with. After some practicing, we started working on making the views for all the different features that we were planning on implementing. We kept coherent design (colors and shapes) throughout the application.

3.4. CO2 API

In our research for an API, which could help us calculate the reduction of carbon emission, we came across some problems. First of all, it was quite hard to find a good API, which we could use for as many features as possible. The OOPP team had recommended a particular site, but the main problem that we had with that site was that it could not calculate the reduction of carbon emission for a few features that we wanted to implement. Also, we found that too

⁵ OpenJFX. (2007). JavaFX. Retrieved April 10, 2019, from <http://openjfx.io>.

much information needed to be filled in for the calculations. This would be too much of an inconvenience for the users of our application.

Most of the API's on the internet for calculating the carbon emission reduction were too basic, or we needed a license to use it. Therefore it took us some time before we found two reasonable API's. However, then occurred another problem. We tried to contact one of the API holders, but even after two weeks we still had no response. After some discussing with our TA, we decided that it would be best to contact another organization.

We contacted the organization CoolClimate⁶ to ask if we could use their API, and they responded quite quickly. After that, it went smoothly, and we were able to use their API for our project. It has a simple and clear interface; thus the implementation did not take a lot of time or effort.

4. Reflection

There were a lot of different moments during the process; both encouraging and discouraging. Overall, we feel that we have fulfilled the requirements well; we learned a lot and received valuable insight into teamwork in the software development industry.

As with any group of people, we experienced issues in communication; the most discouraging one happened between the team and the TAs. After three weeks our approval for Firebase got modified. These modifications had a significant impact on the project. It was very unfortunate for us. We got behind the schedule and had a lot of stress to keep up and adapt the entire project according to the changes. Eventually, we managed to fix everything in cooperation with each other and continued our work. As an advice for the course improvement, due to our unfortunate experience with the project, we would suggest to elaborate more on the point that encourages the students to use any library/tool that they think will help in the development, because we found out that in fact, we do not have as much freedom as it is shown in the first place.

Sometimes there were decisions that we as a team had different opinions on. They also had an impact on how our team operated; sometimes increasing unseen tension. Fortunately, we managed to cut these issues off early, and none of them stopped us from completing the project. In fact, we found them as a part of a valuable experience that will definitely be useful in the future.

As a group of junior developers, we sometimes had a bit of time struggling because we are not yet able to assess the time required for a different set of tasks accurately. Needless to say that procrastination and laziness could play their role. Although we always completed and handed in official demos on time, we still can improve our time management for our personal goals and deadlines. We often needed to pay extra effort the days before official deadlines in order to get everything done. We believe that it is a matter of practice and further self-improvement, so we all can become more disciplined and sharp on assessing the required time for completing the given tasks.

Although we are happy with what our result is, the application can still be improved in many different ways. It is still on the prototyping stage and does not have the full basic workflow that a simple user would require. For example, settings are an essential part of any application, and they are not implemented yet. Secondly, GUI requires some more attention

⁶ CoolClimate. (2013). CoolClimate Network. Retrieved April 10, 2019, from <https://coolclimate.berkeley.edu>.

in making it coherent on different monitor sizes; sometimes the elements may flow or squeeze. We believe that there is no perfect end for any application. There are always ways it can be improved.

5. Individual Feedback

5.1 Nadyne Aretz

In the first few weeks, I found it hard to get started with the project. I mainly focused on getting the customized CI working, and I underestimated the workload a little bit. Because it was so much harder than I had expected, I started to get frustrated with the project, and my motivation started to drop a bit. I think that that was also one of my weaker points, that if I got too frustrated, I quickly started to think that I could not program and started procrastinating my work. However, after three weeks of struggling my hard work paid off and I could start with the more enjoyable tasks of the project.

My task in the team was mainly designing and ensuring that our project would have a few creative aspects. The remaining few weeks I focused on making a fractal tree and achievement badges and implementing them, partly writing the final report and in the last week making a bonus feature (paper recycling) and looking into push-notifications for Windows.

Making the fractal tree and the bonus feature was a lot of fun. I definitely became more confident in my programming skills, although I had to rewrite the whole fractal tree class because OpenGL did not work out with JavaFX. Making badges was also a lot of fun. I first designed them and then started implementing them. Lastly, I started focusing on making push notifications for when a user got a badge as an achievement and helping others out with the last things to finish the project. I definitely got better in asking for help and in the end, I also received a lot of help from my teammates.

Overall I found the process of making the project very stressful but also quite enjoyable. It would maybe have helped if we as a group did not do most of the work in the weekend. Some parts of it I really enjoyed, and I learned a lot of new things. I was very happy that I, in the end, got a lot of help from my teammates, because without them the programming would have gone a lot slower.

5.2 Matthijs Arnoldus

This project has really taught me a lot. I really liked that at the beginning we had no clue what to do, but in the end, we got to a real product. In the beginning, we slowly progressed, but we increased the pace every week.

I believe my strong point that helped me during the project was the fact that I was able to analyze the others' code and understand it very quickly. This made it easier for me to connect different parts of the code and to help some of my teammates.

One of my weak points at the beginning of the project was my planning. I think I have improved this during the project, and what really helped me was the fact that we had a meeting every week. That way, we created deadlines for ourselves every week, which helped me make sure that I did not leave too much work for later.

The biggest problem we came across during the process was the fact that we could not use a large part of our code due to a miscommunication with the TAs about a library we wanted to use. At first, we were a bit frustrated about this, but in the end, we put some extra effort into it and finally were able to finish our project.

Overall I believe I have gained a lot of extra knowledge about programming, but I also think that during this project I have improved my cooperating and planning skills since this was my first large project.

5.3 Jesse Nieland

When I first started with the third quarter, I was scared of the project. I barely passed OOP, so I felt like the project would be a huge obstacle for me. Luckily when the groups were announced, I noticed that I was with 2 of my friends, so that was reassuring.

At the start, I said to myself that I wanted to become better at programming and learn to better get used in a team. I think that there has been some decent progress with that. Although I have been working on the GUI for the project, I feel like I can be a valuable asset to the team in that category, and the team works well together.

I want to mention that without my teammates, I would probably not have been able to do this much on the GUI. I got helped a lot by them, and I feel like that is a good thing, as we are supposed to work together as a team. I feel like I am still not at the same level as the others of the team, but that will come by working more and more on programming in the future.

In the end, this project has not been as scary as I first thought, and I feel like that is mostly because of the amazing teammates I have.

5.4 Roman Sirokov

To my surprise, one of my weak points, laziness, did not have as much impact as I anticipated in the beginning. In my opinion, most of the time I managed to complete the tasks beforehand. I believe that I was able to achieve that due to the enjoyment I was getting from the process of development, so it was not a hassle for me to do the work in my spare time.

The other weak point, however, at some moment in time could have in fact resulted in inconvenience. Not everyone shares my attention to details, but I do believe that details sometimes matter. However, on the other hand, it also had a positive impact on the project. It allowed me to implement and then keep a nice MVC structure, thus a clean and maintainable code base. My strong point, experience, helped me a lot; mostly in being efficient and writing clean code.

I managed to leave a footprint almost in all parts of the application. I was responsible for connecting the independent modules written by my teammates into a single workflow, sometimes optimizing or refactoring them. As it was said, I was responsible for database communication and the authentication procedure as well as worked in a pair over some of the business logic.

5.5 Roderick de Britto Heemskerk

At the start of the project, I was quite scared that I was going to let things go, and possibly not contribute enough. However, as the project went along, I think I managed to bring enough contribution. I learned a number of things while doing the project, as I had never made an application before.

While at first, we tried to implement the server via firebase, in the end as that was not allowed for the project we switched over to spring. Implementing the server there was quite fun actually. It had a unique structure and taught me quite a bit about internet protocol.

Now regarding my weak points in the project, I did get distracted throughout the project, but when actually working on the project I managed to keep my focus and work effectively. I also did not have that much trouble with feeling dependent, as in such work it is essential to keep up with your colleagues to make sure the work is up to par.

As for my strong points, I managed to pick up spring quite quickly. I did, however, have some trouble testing the server once in a while, even though that often had to do with the API requests instead of the actual server. Overall I quite enjoyed the project, and it taught me a lot.

6. Value Sensitive Design

Modern software may affect the lives of a broad set of different groups of people with different interests, of different social classes, races, etc. It is hard to assess for sure who exactly the target audience for our project would be or who would be affected by it on the side. In the end, the application can find the audience or use cases that we could not anticipate.

From the very beginning, our design decisions were appealing to people from middle or upper-class families/small communities, possibly prevailed by women. They are not necessarily tech advanced, but open-minded and eager to benefit their local communities by developing the green culture. Nevertheless, some design decisions can, in fact, have an effect on a broader group of stakeholders. A large group that takes part in the developing of the green community are local businesses that provide green produce/services. For example, local farmers with their green production.

As said, our application is targeted on people that care about environmental issues. Thus, we have access to the audience that is actually very similar to the customer's profile of the aforementioned businesses. We are able to connect them with each other. This will have multiple positive effects. Firstly, the local economy will thrive. People, instead of buying the overseas production, will buy the local produce from "their neighbors". Secondly, it benefits the development of the green community. People get to know what they can do to improve their local environment. If done correctly, it may have a viral effect; green lifestyle is already a modern trend. People may start spreading the word and joining the movement. In the end, it is just beneficial to the community in terms of socialization. People go out, meet, talk, help each other, etc. They have fun while solving local environmental issues. In practice, this scenario could be implemented through a series of features that help people to discover their local communities, i.e. maps of the stores with the green production.

If we would design for this stakeholder, local environmental businesses, the process and the product would be affected in multiple ways, simply because design for businesses may differ significantly from the design for general users.

Businesses are more focused on their own product, how it can benefit from us, and not the actual features of our application. Undoubtedly, they would rather use our product as a marketing platform. Thus, we would need to provide them with convenient tools, while not forgetting about the convenience of the general users.

Nevertheless, there are some possible design decisions, for the mentioned above scenario, that may bring tensions between the stakeholders. Since we have the audience that environmental businesses might be interested in, we can use it in favour of our own business. Needless to say that we, as developers, need to earn something for our living as well. We can connect local green businesses and our audience together, using for example the advertisements. It is one of many possible ways to implement the scenario of connecting environmental businesses and our users. This solution will introduce a third stakeholder into the group -- us, developers -- since we now benefit from all of this as well. Nevertheless, it can certainly have a negative impact on the users. People do not like advertisements, even if they are used with good environmental intentions; by not advertising a fast food chain with an unhealthy and unecological product, but something that fits our ecological vision. These tensions are not something critical that may destroy our product, but are definitely something that must be taken into account while designing the business model. The questions like, "is there another/better way?", "what impact will it actually have on the user experience?", etc, must be asked in order to ensure that the maximum amount of stakeholders benefit from our product. In order to avoid the mentioned tension, we can, for example, implement another business model, not advertisements. For instance, it can be paid features or donations. Nevertheless, this is something that can have its own disadvantages.

A good source of insights for such scenario could be a representative from a small municipality; a person who knows people in the community, local economy, local production, their attitude towards nature and the climate issues. We believe that such a person would shed light on the audience; what they have in common (activities, thoughts, attitude), as well as would know possible ways to approach it. He may have an insight into what consequences such an application can have on other stakeholders in the community, besides the potential customers. Of course, there are other sources that can be used. The science articles about environmental issues can give a valuable insight into what problems we can try to solve with our product.

7. Summary

In the end we all enjoyed making the project. We learned a lot and achieved our goals, or at least significantly approached them. We made an application wherein people can track their co2 reduction, but also can be inspired in how they can be more eco friendly. Of course, maybe with a next project we would do certain aspects differently, but that is all part of the learning curve.

References

Java Documentation. (n.d.). Observable. Retrieved April 10, 2019, from <https://docs.oracle.com/javase/9/docs/api/java/util/Observable.html>.

Java Documentation. (n.d.). Observer. Retrieved April 10, 2019, from <https://docs.oracle.com/javase/9/docs/api/java/util/Observer.html>.

Google Firebase. (2010). Firebase. Retrieved April 10, 2019, from <https://firebase.google.com>.

Spring by Pivotal. (2003). Spring. Retrieved April 10, 2019, from <http://spring.io>.

Jersey. (2010). Jersey. Retrieved April 10, 2019, from <https://jersey.github.io/>.

OpenJFX. (2007). JavaFX. Retrieved April 10, 2019, from <http://openjfx.io>.

CoolClimate. (2013). CoolClimate Network. Retrieved April 10, 2019, from <https://coolclimate.berkeley.edu>.

Appendices

Appendix A

Link to the example of a large Pull Request (!28).

https://gitlab.ewi.tudelft.nl/cse1105/2018-2019/oopp-group-70/template/merge_requests/28

Appendix B

Example of the description for one of the large Pull Requests (!28).

Demo 2: Vegetarian Meal workflow

The branch contains the feature related to Demo 2 (#3 (closed)), Vegetarian Meal workflow. It was constructed from other secondary branches (*business logic*, *gui*, etc), and now when everything is assembled into one single workflow, it can be merged with master for deployment.





The branch connects various scenes and logics together. The Sign In / Up views and corresponding functionality now works in the application. Firebase's Authentication service is used for this purpose. The user can get authorized, can see his stats in the home screen, can go & use food scene (with veg meal).

The database is used in the workflow, thus the changes are saved and can be retrieved during the next session.

Related Issues: #24 (closed), #36 (closed)



Appendix C


Example of the code review made on one of the large Pull Requests (!28).

**Matthijs Arnoldus** @matthijsarnold · 3 weeks ago Developer   

The code looks good, all tests pass and checkstyle gives no warnings. This merge commit adds a lot of different functionalities to the master branch. Maybe in the future we should try to divide them a bit more over seperate branches and merge them seperately.


Edited by Matthijs Arnoldus 3 weeks ago


 1 

**Matthijs Arnoldus** @matthijsarnold added 1 commit 3 weeks ago

- [cbbaacf8](#) - Fixed typo


[Compare with previous version](#)





**Matthijs Arnoldus** @matthijsarnold approved this merge request 3 weeks ago

**Roman Sirokov** @rsirokov added 2 commits 3 weeks ago





- [73ae1c6a](#) - Final Conflict Solved
- [a9004049](#) - Merge branch 'Feature-Classes' of...

[Compare with previous version](#)


**Matthijs Arnoldus** @matthijsarnold approved this merge request 3 weeks ago

**Roman Sirokov** @rsirokov · 3 weeks ago Developer   

[@matthijsarnold](#) agreed. This one turned out to be a little bit too large and old. Had to resolve a lot of conflicts in the branch.

**Jesse Nieland** @jnieland · 3 weeks ago Developer   

Looks great, works good!

 1 