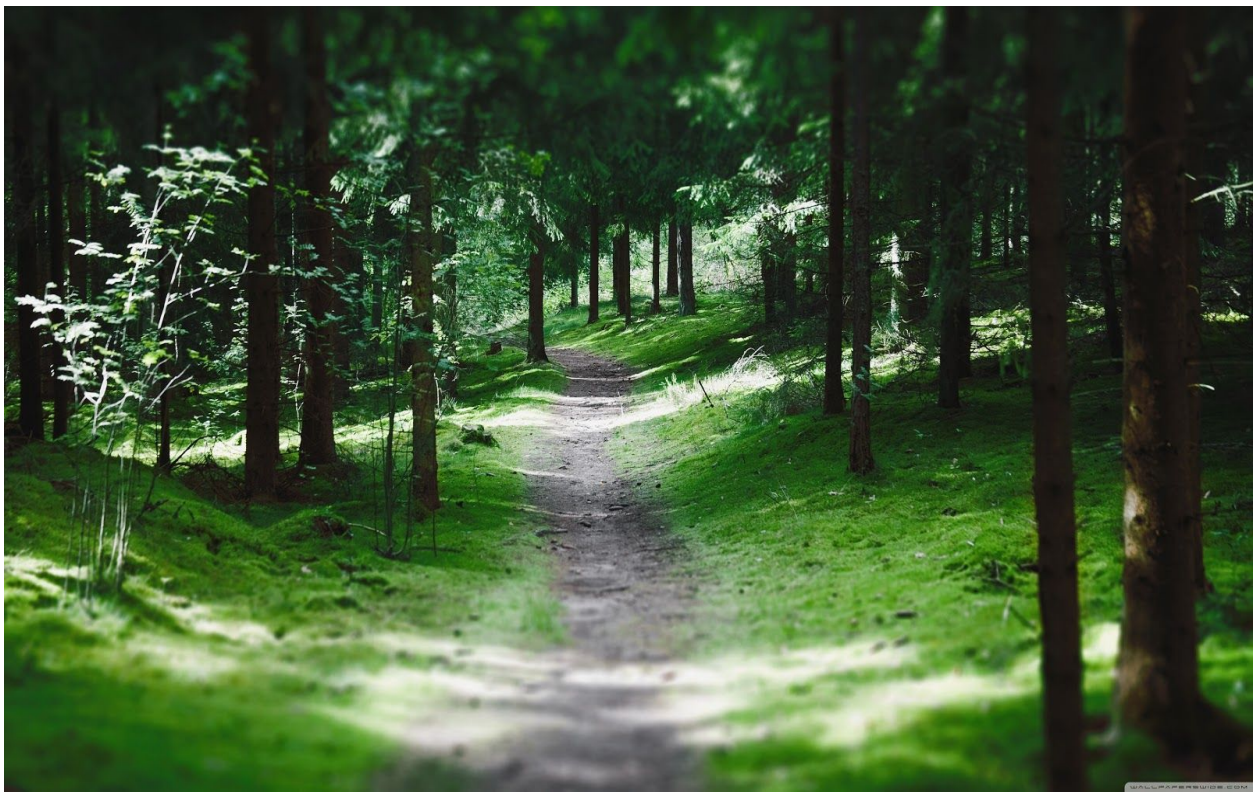
CSE1105 - OOP Project

Group-82 Report
Project GoGreen



Introduction

Environmental issues are harmful effects of human activity on the biophysical environment. Environmental protection is a practice of protecting the natural environment on individual, organizational or governmental levels, for the benefit of both the environment and humans. Our project is focused on each individual person and especially on the ecological footprint per person, as its level in the Netherlands is particularly high. We are building Java application that enables users to enter their ecological choices. They get an overview of what they can do and see how much greener they can still go. To stimulate more green activities, we are integrating a gamification aspect to the application. Through it the users will be able to compare their progress with their friends. Based on your score and your achievements you will be placed in a rank group, which differentiate from each other on specially created badges.

The team :

- Alin Prundeanu
    - Student number : 4925262
    - Testing engineer and client-side developer
- Atanas Marinov
    - Student number : 4946251
    - GUI and client-side developer
- Chengrui Zhao
    - Student number : 4859693
    - Client-side Developer and server-side developer
- Giovanni Fincato de Loureiro
    - Student number : 4926854
    - DataBase and GUI developer
- Jan Pieter Kroeb
    - Student number : 4961307
    - Client-side Developer
- Nik Kapitonenko
    - Student number : 4899660
    - Server-side developer and tester
- Rahul Crunal Kalaria
    - Student number : 4770110
    - Server-side and Android app developer

Content :

Product

Project Architecture :

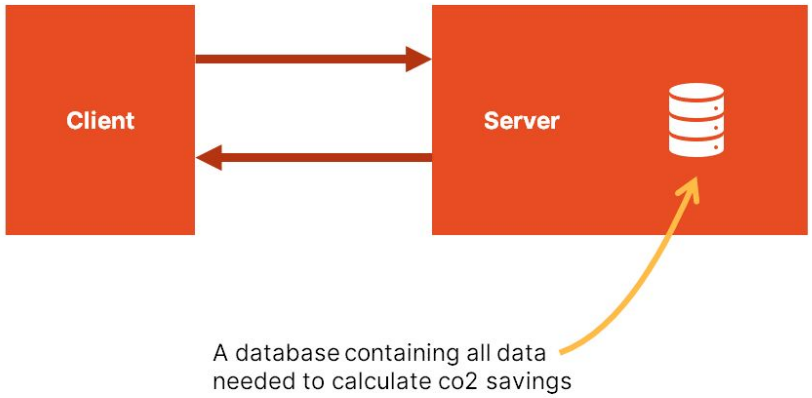


Figure 1.0

We decided to use the architecture displayed in figure 1.0 in our project as this allows us to have complete control over our entire project whilst still being relatively modern and scalable.

Technological Choices :

- GUI
  - For the GUI we used JavaFx for the following reasons
  - It uses FXML and CSS, which:
    - enabled us to create the GUI in the application separately from the implementation of its logic and therefore made it much more maintainable and easier to use. A competitor such as Swing don't offer a declarative approach.
    - gave us more styling options such as themes and animations, which are hard to produce with Swing and Java AWT.
  - There are plenty of open-source libraries, which can enrich the already various selection of features that JavaFX provides.

        - The main addon is JFoenix, a JavaFX library that gave our client the modern material design look, without the need to create all those details ourself.
    - The documentation of JavaFX is easily accessible and well written. You can also find loads of topics and tutorials related to it.
- Database
    - MySQL was used to make the database for the following reasons:
    - A table-based DBMS was the type that our group had the most comfortable experience using, hence we chose to use an SQL-based database
    - MySQL functionality allowed us to put nearly all of the server input into only a few separate tables with calculations, etc. being done by database triggers from those tables. Triggers work without network usage, hence reducing calculation load on the server and client-side
    - A relational database such as MySQL is efficient for conducting operations or queries which span multiple tables, a core part of our calculations.
- Server side
    - With Spring being an all-in-one package, all its components are more integrated than independent libraries could be. This reduces the amount of code needed to make them work together to zero, apart from Netty, for example, which was our other candidate for building the web framework foundation, but meant dealing with a lot of new aspects of the language.
    - Being one of the most popular Java frameworks out there, Spring not only has countless posts and tutorials on all kinds of topics, but knowing this framework shall qualify as useful knowledge far in the future as well.
    - Additionally, as Spring Boot is made for developing enterprise level software in no time, it does all the heavy lifting for us. This allows us to avoid writing all the boilerplate code, and focus solely on our custom logic.
- Testing
    - For the purposes of testing our project we used Mockito (as Part of Spring Boot Test) and traditional JUnit libraries along with Spring Runner testing library
    - Both Mockito and Spring JUnit Runner have extended integration with Spring, and due to the Inversion of Control principle Spring is based on, we practically needed them to test in particular the persistence layer of the server, as most of it is handled by Spring and Hibernate.
    - Also, the power of Dependency Injection feature allowed for a more fine and rigorous control over the initial state conditions of different parts from our project
- Android Application
    - For our android (and ios?) application we used Google's Flutter SDK and FireBase hosting service for the following reasons

- With the help of the Flutter SDK, we (developers) are able to create applications that cater to both android and IOS with a single codebase written in the Dart programming language. This enables developers to create mobile applications for 2 of the largest mobile operating systems without going through the hassle of learning 2 different programming languages.
- We used Google's FireBase and FireStore for the Authentication and Authorization of our users because with their services we are able to guarantee that our users information in kept secure at all times.

Process

- Week 1 - 3
  - Learned how to properly use version control softwares (GitLab) to optimise workflow
  - Started to learn about various net frameworks, finally settled with Spring and javaFx.
  - In the first 3 weeks we mostly spent our time learning about various new softwares, frameworks, meaning of API's etc as most of us only had programming experience that we got from the quarter 1 OOP course. At the end of this period we noticed that we did not give enough attention to software testing which lead to us losing 0.2 points, we created a special testing team to make sure this never happens again.
- Week 3 - 6
  - First version of our GUI ready.
  - First version of our server ready with Https, spring security also.
  - From week 3 onwards most of us became quite confident with how to use JavaFx and Spring to reach our goals. We divided ourselves into 3 groups, GUI, server, and database groups. However during this period we discovered that there was lack of communication and started to find ways to solve this problem.
- Week 6 - 8
  - Major strides were made in terms of software testing. Testing became a priority.
  - Base features completed. Started working on extra features.
  - During week 6-8 we were in the phase of wrapping up our project with regard to the base features and we started thinking about extra bonus features. Also during this period is when we started working on our android app.

Reflection

All 7 members of our group came into this project only having limited programming knowledge that we gained through the OOP course of the first quarter. However, the concept of OOP project was entirely different from the other courses before, since more group communication was required. Due to this our

first 2-3 weeks were mostly spent on just trying to figure out what a “RESTFul API” means and learning about the new technologies that were showed in the introducing lectures. We initially spent time exploring other technologies such as netty, swing, etc., however, after a broad overview and discussing, we made the choice to use Spring and JavaFX throughout the project.

With some of the important decisions being already taken and week – week and a half of intensive coding and demo 1 being already finished, we were still neglecting the testing and its influence. About that time, we realised the its importance and how its absence could lead to negative results for us. This have taken us to a change in our initially-wrong vision for the process – instead of everyone producing their own tests, we have dedicated part of the team to be a “testing department”.

Our group is one of the more unique groups due to its composition as every member is from a different country. Having such a diverse and in the same time unique group definitely came with its ups and downs. Initially we were faced with severe communication and coordination issues. This problem lasted about 3 weeks. For example, in the first few weeks our GUI was much ahead of the server which lead to coordination issues, same could be said about the server and database. To deal with this problem we all decided to have one extra meeting on the saturday of every week. This gave us the opportunity to not only measure and discuss our progress on the project more frequently, but also have the opportunity to plan for the following week.

The project could be further developed in multiple different ways. Adding additional green activities to choose from will make it more various and will give the opportunity to more people to see themselves in the application. Providing a more in-depth details how every green activity is helping the environment will stimulate the users to be more active. Implementing a third-party authentication would lead to a better security and more users to the app. Also, as most of the resource servers are large companies and/or social-medias, connecting with your friends and spreading the awareness there will be easier.

Individual Feedback

Alin :

Immersive and full of unexpected intricacies is how I would describe my experience during this project. Started with enthusiasm, it has been a road patched with surprises, from trying to find the right, relevant documentation to study and eventually implement, to working with complex integration tools like maven and git, up to implementing logical schemas into actual functionality, while sometimes having to firstly deal with strange, new lines of code. Being an inquisitive and methodical person, I handled the task of testing, therefore being able to delve into server side logic as much as into client side one and their connection. I changed my approach on the project quickly, switching from gathering all information on

Spring and JavaFX beforehand to focusing on the useful, urgent aspects of those new technologies. The testing process went on fairly well, considering times when I had to rush by deadline to understand and properly test new implemented classes. As I eventually found myself comfortable with the testing strategy, I took care of the application and added some features, like achievements and profile photo. The lack of good communication in the beginning has greatly improved as the project went on and I learned how to work in a collaborative environment and merge my contributions with those of others, in order to build on top of them. The group meetings were lovely and efficient, while brainstorming future ideas and solutions was indeed compelling. It has been challenging and at times even overwhelming, but all in all I managed to pull it off and bring my skills to the project team.

Atanas :

For me the project was an exciting time, in which I have improved myself in both personal and professional manner. Before the start I was not confident in my communication skills as my level of English is not as great as I would like it to be. However, yet after the first week I knew that it won't be as much of a problem, seeing how open and friendly my teammates are. With the time passing by we have become more and more closer and our discussions weren't explicitly professional, but they were including side-talks about our interests outside the university, which made us form a solid working community. We have all never faced such a big software problem in which we had to also use unfamiliar for us technologies such as version control and maven, so in the beginning everybody was wondering how to separate the tasks and where to begin from. After a week of considering the options, I just took a spontaneous decision to start learning about the developing of a GUI. The start was a research of the options that are offered, after which I've chosen the JavaFX. As a widely used platform it offered a lot of reading materials, giving me the possibility to see how develop an application from the point of view of multiple different developers. As I main engineer of the GUI, I was the one, who interpreted the ideas of the teammates and rendered them in an actual user stories.

Zhao :

Being a highly goal oriented person, I'm self-motivated, and good at learning new skills and solving problems. My role in our team was to build the backend of the client which will martialize part of the business logic and connect the GUI to the Web API. To be specific, there was a task that I need to merge Spring into a JavaFX application to take advantage of the features provided by Spring, such as dependency injection, Spring security, and JSON parsing. It was surprisingly hard for a fresher like me. After reading tons of lines of code and solving dozens of problems, I accomplished it. So even though I knew nothing more than the basic coding skills that learned in OOP in the first quarter, I managed to gain a comprehensive understanding of our project and being able to help my team. My weaker point comes to

my communication skills. Since my English is not as good as my teammates and we all have different cultural backgrounds, lacking communication and misunderstandings caused several conflicts. A typical one was that I thought the endpoints of our will always return an HTTP.200 as the response status, according to which the behavior of the client was designed. But actually, the endpoint will return a 40X status code if something went wrong inside the server. This conflict caused several unhandled exceptions on the client side. We solved this conflict by handling the exceptions on the client side, after a discussion with the server team, because their design was more reasonable. After this, I realized that the most important part of teamwork is communication. And most of the conflicts can be saved by good communication.

Giovanni :

My weaker point in this project was that I was completely new to most of the technology involved. I was familiar with the Java language, but the concept of the API, GUI, and connecting it to a database was new to me. The beginning of the project was therefore difficult for me, since I had to learn things from the beginning. Once I got more comfortable with the way the software worked, I discovered that one of my strengths was developing intricate database logic and implementation of features. I capitalized on this, and made a MySQL database that performed the CO2 and price calculations with minimal complexity of querying from the server. I initially had conflict with other team members as I failed to properly communicate with them the database details, making implementation for them more difficult. I learned from this experience the importance of communicating with your group members to maximise efficiency. I applied my learning throughout the rest of the project, and with consistent communication the new developments were smooth to implement. We developed the application much more efficiently once proper communication was established, especially at the group meetings. Once the database was further developed, I went on to help Atanas work on the GUI screens and client side code. Initially this caused some conflict in who was delivering which part, but we quickly learned to coordinate properly and deliver the separate GUI parts and combine them efficiently, and the GUI developed much more quickly.

Jan :

When I think about what I contributed towards the project, I also want to think about how the project developed me further as a person. And I can say that I have grown when I had to make these contributions. These thoughts of my advancement are mainly risen due to the fact that I had to figure stuff out on my own, instead of reading it all from a book. Browsing Stack-Overflow was at first pretty tedious, but eventually you see what was wrong with your own code / what your own code needed. What I coded in the OOPP Project was mainly connecting the client methods with the GUI and writing those client methods to transfer the inserted data towards the database. I also coded some server-side in the

beginning, but in the end I fully moved towards client coding. When I first started with the project, I knew a few of my shortcomings that could harm the project. Mainly working without a schedule, this was not really a problem with this project fortunately. There was only one problem that I have experienced and that was when I was committing my code for demo 2 and we accidentally deleted some conflicting code, causing the whole thing to not work anymore. Zhao was fortunately able to recover it and was able to correctly merge it with the master. In the end I can proudly say that I also understand how to use GitHub in the end.

Nik :

The GoGreen OOPP project was an unusual, eye-opening, albeit also a bit stressful time for me. I went in head first, not knowing what to expect. As a passive person, that prefers to have a clear task on what has to be done, my role in the project was constantly changing. After an active first few days writing the personal development plan and figuring out how git and gitlab works, I had no clear idea how to proceed. The first few weeks was spent learning Netty (something I suggested), ditching it, and switching to Spring. Learning this framework was quite difficult for me at first, because it abstracts a lot of implementation away, meaning just having moderate Java knowledge won't help you. After that I tried to work on the GUI a bit, but we already had three people doing it, so only in week 3.5 I really started to bring my share: writing tests for the server. At the same time, I started also refactoring the server code itself, as it was cobbled up from the previous demos. In the end, I learned a lot about how modern java technologies, and I am pleased with myself in that regard. In communication and teamwork, I became more confident in my English skills. Before this, I always feared to speak, but now not so much anymore.

Rahul :

I came into this project as a philomath, however one who had a bad habit of giving up at the first sight of a roadblock. However during the course of the project I gained a very valuable skill about how to not give up when faced with an obstacle, rather find alternate solutions to overcome that obstacle. My main contributions towards this project where in the development of the server and android application and assistance in the database. To be more precise, i worked on connecting the database to the server and setting up appropriate CRUD endpoint to manipulate the database. However not everything was perfect as my teammates quickly picked up on another flaw in me, that is my obsessive nature of doing all the work due to this they felt that where was not enough work left for them. As an attempt to work towards this flaw I started to reduce the workload that I took on and also take breaks, I also started looking into extra bonus work to so as to evenly distribute the workload among my other teammates. This is when I decided to make an android version of our project using google’s Flutter SDK as I was quite intrigued by Google’s claims of being able to develop native apps for IOS and Android with a single codebase. As a result of working on the android app I also simultaneously solved the problem of workload distribution among

teammates and my craving for learning something new. As I write this, I feel like I have not only become more knowledgeable but I have also grown as person.

Value Sensitive Design

In order to initiate our value-sensitive analysis of our design, we first identified the stakeholders of our application, beyond the obvious clients and shareholders. On a broader scale, the app is meant to help preserve the environment, making future generations all stakeholders. More directly influenced though are the consumers and producers of the products mentioned in the app. Solar panels producers, local produce farmers, public transport systems, etc. are all encouraged by the application on to the client; meanwhile, foreign produce, private transport (cars, motorbikes, etc.), meats, etc. are discouraged. The application “shifts” consumer interests in the direction of the more environmentally responsible products. From a moral perspective, although there is potential harm to the discouraged businesses, this is acceptable in return for the sake of future generations’ environment well-being; promoting eco-friendly products and behaviours is done for the sake of a more sustainable world. At the level of local government and society, the application encourages usage of public transport and low electricity/power consumption. Since these services are generally provided by the local government rather than private businesses, this unfortunately means a higher need for funding in public transport, but much lower funding required for other services which the government provides (less public parking space required due to reduced car usage, lower need to import car fuel, lower electricity consumption means less imported fossil fuels to produce it, etc.). For clients of this application, the application promotes a lifestyle with lower consumption of money and CO2, which is a healthy contrast to today’s society promotion of consumerism. The clients’ privacy being taken into respect, since all the information comes only from the user’s own input, we have avoided making the app unnecessarily invasive. Our application contains a history of the clients’ activities which they chose to input. This makes our database a repository of sensitive information, as a data breach would make it easy to analyze and possibly exploit an individual’s purchase patterns and habits. The application only requires a username to register, so no users are at risk of being discriminated against in any way.

References

- Android
    - Infotech, D. (2018, May 30). 5 Benefits of Flutter: That Make It Start-up Friendly. Retrieved from https://medium.com/@debutinfotech/5-benefits-of-flutter-that-make-it-start-up-friendly-d045fc209977
    - Ganesh, & Ganesh. (2019, January 06). 13 Reasons Why you should choose/ consider to move to Flutter in 2019. Retrieved from https://medium.com/flutter-community/13-reasons-why-you-should-choose-consider-to-move-to-flutter-in-2019-24323ee259c1
    - Nathan, L., & Nathan, L. (2016, September 26). Firebase Pros and Cons. Retrieved from https://medium.com/one-tap-software/firebase-pros-and-cons-ce37c766190a
- GUI
    - Release: JavaFX 2.2. (2013, November 08). Retrieved from https://docs.oracle.com/javafx/2/swing/overview.htm#
- Testing //Alin
    - Baeldung. (2018, November 08). Quick Guide to @RestClientTest in Spring Boot. Retrieved from
    - Llosa, J. P. (2019, February 20). Using MockRestServiceServer to Test a REST Client. Retrieved from https://examples.javacodegeeks.com/enterprise-java/spring/using-mockrestserviceserver-test-rest-client/
- Database
    - RundMarch, B. (n.d.). The Good, The Bad, and the Hype about Graph Databases for MDM. Retrieved from https://tdwi.org/articles/2017/03/14/good-bad-and-hype-about-graph-databases-for-mdm.aspx
    - (n.d.). Retrieved from http://dcx.sap.com/1101/en/dbusage_en11/ptbn.html