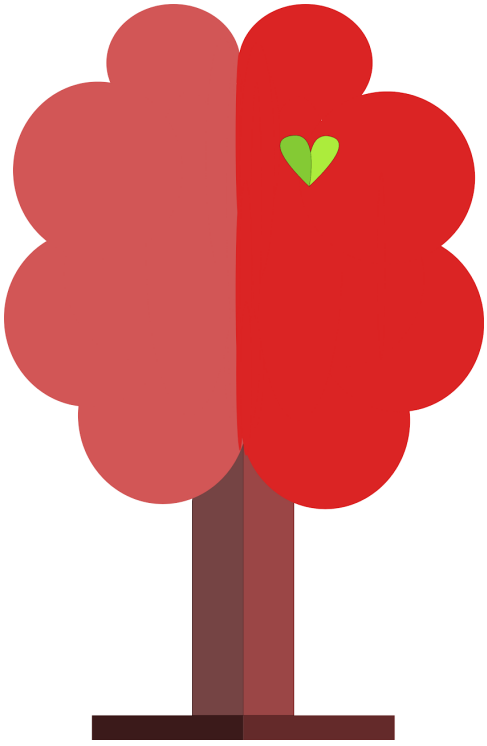
GREENDR



Dan Dan Berendsen, StudentID: 4904982
Pandey, Harshitaa, StudentID: 4780205
Max Karsten, StudentID: 4943090
Jason Bloom, StudentID: 4719794
Thom van der Woude, StudentID: 4945727

Introduction

The ‘GoGreen’ project as part of the CSE1105 course at the Delft University of Technology provided an opportunity to learn many new skills: planning for the short-term and long-term, collaborating on a bigger project and using tools and methodologies such as version control and SCRUM, all valuable skills that are applicable in a professional environment. The course also gave insight into the quality assessment and testing of code.

This report will discuss;

Collaboration

The collaboration aspect of the project consisted of three elements: the planning to keep the team on track and allow for proper time management, the communication to get the tasks completed on time and the version control to allow team members to work together in harmony.

Planning

In the first week of the project, a plan was made regarding what tasks needed to be completed and by when. This plan was a bit vague as there was little concept of how much extra time would remain to implement the “bonus” features. It also remained undecided what those extra features would consist of. The planning consisted of a timetable and incorporated the MoSCoW prioritization technique. This proved to be quite helpful down the road as it allowed for the team to focus on the main task at hand(“Must- have” on MoSCoW) and plan which features should and could be done in addition to these must-haves(“Should-have” and “Could- have” on MoSCoW). It was especially useful during the last two weeks as this was the time where extra features could be implemented. Unfortunately, the original plan took longer than expected, this meant that there were compromises made on what things were implemented and which ones weren’t. While the plan was used to a large extent, some adaptations were made to allow for the time miscalculation. Thankfully, this delay was taken into account in the original plan as two weeks of headspace were allocated in the initial planning.

Communication

Towards the beginning of the project, communication was a bit of an issue. While tasks were assig-ned at the beginning of every week, those tasks were still a bit vague and were topics such as “work on the GUI” rather than what it became towards the end which was “work on the profile page for the GUI”. This ambiguity caused some issues such as that one time two people worked on the same thing. Luckily, this resulted in both understanding JPA and Hibernate a lot better which paid of in the long run.

Finding the best medium for the team’s communication needs also took some time. During the first meeting, it was decided to use Whatsapp as the official means of communication but as the limitations of this platform surfaced during the following weeks, there was a call for a more feature-fledged service like Discord. After this matter was discussed in a following meeting, a transition to this platform followed.

One step which was taken to ensure that communication ran smoothly was that at the end of every week, a meeting was held in which everyone recapped what they did, and what needed to be done for the next week. The scrum board was also adjusted and other paperwork which was due for the week was submitted well ahead of the deadline on Saturday night.

Another positive aspect of the communication was that all team members were very proactive and were able to produce the tasks they were given. If they weren’t one simply had to let the other team members know, and someone else was able to step up and get it done in time. Everyone was also extremely supportive of everyone and their own learning pace.

Version Control

Version control was a bit difficult to use during the beginning, as all the members were using different platforms. One team member even ended up using three different platforms (the terminal, source tree as well as IntelliJ) in order to find which one as the most suitable. There was also a minor issue in which a team member accidentally merged two branches together which caused various merge conflicts. Resolving it took the still unexperienced team a bit of time but it was able to be done in a day or two. Also, after this, everyone was very careful with which branches they were merging to, and also understood how to resolve conflicts if they occurred. So, for the remainder of the project, version control was not an issue at all.

Major decisions as a team

Ethical Aspect

In the design of the carbon-tracker application, the key values we the engineers prioritized in the decision making were the positive environmental impact of the app and the privacy of its users. The top priority was the environmental impact of the app: we wanted to make the world a greener place and empower the end-user to contribute to this effort. This is why in the visual design of our app, we tried to send out a positive encouraging message. To conform to GDPR regulations and save the client deploying the application the likes of Facebook's recurring multi-million fines, the safeguarding of the end user's rights to privacy also played a big role in the design of the application.

Aesthetical Aspect

There were several major decisions to be made for the visual design of the application. At the beginning of the project, due to the name being undecided, the theme was also unclear with each member choosing their own palette. However, after the name "Greendr" was decided, several steps were taken to ensure it was reflected in our design. The final logo design was a red tree with a green leaf in the shape of a heart as a more memorable and fitting alternative to another proposed design, a cube with various symbols on its facets. Following that, the rest of the application was fairly simple to design. Every page consisted of the green leaf logo, with the red and white theme being consistent throughout. A small change with a great impact was the introduction of tool-tips, motivations and descriptions of the various environment-saving acts. In motivating our users to contribute to a greener world, informing them about the merits of certain feats, like taking a bus instead of a car or eating a vegetarian meal seemed key. In terms of overall style, we aimed for something Material[1] though we were consciously aware from the start that this style was very hard to get right.

Technical Aspect

Early on, it was decided to go with the Spring framework to suit our web stack needs. Besides the very much fitting logo, the widespread popularity of this web framework seemed appealing as this generally translates to more and better online resources, which is in our opinion the most important thing for a team that is new to technologies like Spring[2]. The often-praised[345] modularity, integrability and overall flexibility of Spring also caught our attention because as a team that still had much to learn, the ability to slot in any library or api down the road was a sought-after quality in such a framework. Both the ORM Hibernate[6] and the GUI library (Open) JavaFX[7] had similar advantages. They were easy to use and

had lots of documentation and web resources about them due to their popularity. The choice of JavaFX over something like HTML5 was done due to the fact that there were some slight limitations when it came to HTML5. While HTML5 is good for simple use cases[8], to make the more complex scenes and add in animations and transitions, it would be far more logical to use JavaFX. To store our data the server is backed by a Postgres database. Why Postgres? Because for a service that is not likely to ever be distributed, let alone distributed on a global scale, this no-nonsense relational database management system seemed like the safest bet for us. To safeguard the privacy of the end-user passwords are stored hashed so that not even authorized personnel can access the plaintexts and sensitive server API paths are shielded from unauthorized rogues.

Improvement

Design

In the work on the GUI, a lot of time was spent revamping the design of pages over and over. Most notably the login page was redone four times because there was still no Because we only decided on the app's name during the middle of the project, all the prior GUI work amounted to practice as the design had to be reworked. This would both save time and allow for that time to be used in other fields, like the visual design, as visually the user interface is evidently not the Material Design we had in mind in the first weeks. Although it is of course much easier to point at a design like Google's Android Pie[9] and say you want that then it is to actually go ahead and adapt it to your application, the overall user experience would be substantially better if more time was spent on the interface and perhaps the app was even rolled out to a limited body of pre-release testers to further aid this push towards a better interface.

Testing

Throughout the project there was little drive towards a healthy test-coverage standard. Testing was an afterthought and remained as such even after test-driven development was recommended to the team. This practically translated in a lot of fuss when something broke, as there was not always ample code tested to rule out secondary causes. There are various causes at the root of this problem. Among these is the uneven distribution of work: when deadlines approached, members of the team took up features that were not their responsibility, got them to work but never quite finished them in the sense that the testing coverage remained negligible. This also fostered a culture in which immature coders never had to 'grow up' so to speak and test all their code, because an environment in which the full project coverage is 17% simply does not make you feel bad for adding code that is just barely tested. Another thing was the general confusion about what strictly had to be tested and what could be exempt from the coverage report. Add to this a continuous integration pipeline that could not be set up until week 5 and an initially broken pom.xml that only included jUnit 5 tests in the coverage metric and you have a recipe for disaster. All in all, the team faced some setbacks but should nevertheless have persisted in their focus on testing.

Version Control

There have been some version control issues that could have been avoided by just trying to understand the tool Git instead of following the magic formula (pull) (add) stage commit push and panicking if something breaks. At some point this resulted in team members being a bit scared to touch

the remotes, as if everything would irreversibly break. In future projects, such problems could be avoided by having everyone spend some quality time with the tools they are using.

Reflection

The ‘GoGreen’ project was extremely interesting, and the topic was one of great relevance in a society that is slowly transitioning to a more sustainable way of life. The course also allowed for all of us to grow as individuals and experience what it is like to work in a professional environment, both negative and positive aspects.

Individual Feedback

Harshitaa Pandey

As a whole, I thought that the project was extremely interesting and I was able to pick up a lot from it. When it comes to my own performance, there were many things that I could have improved on. I think the major thing that I could have improved on was my communication. Rather than leaving everyone in the dark about what I’m doing or why it’s taking me so long, I should have just said something to them. This could include many reasons such as the fact that research was taking a while, or just that I was busy that week. This would allow for other people to take over my job for that week for example, or even help me if that’s what I needed. My second weak point is that I should have taken initiative and done more work on the weeks that I wasn’t that busy. This would have also helped lighten the load on other people and allowed for them to have some time off too. This would have also helped us to stay completely on track with our schedule.

I don’t think there were any conflicts, as such. There were some times where I felt like I should have done more, but the team understood that I was trying to work to the best of my ability and do as much as I could do.

When it comes to my contribution, I contributed to the GUI aspect. I worked on the design of the pages, logos and setting it up so that we could display all the information we were getting. While it was supposed to be not that difficult, it was a task which took a lot of time. This was especially true towards the beginning of the project as I had to learn how to use Javafx and scene-builder. Also, because I was unsure of what platform to use, I had just tried to make the pages without any scene-builder. This did not end up working for me at that time because I had little idea about how the code was supposed to look like, and it wasted a lot of time. Also, even before any of the pages were made, I had tried to make a sliding menu bar. While this menu bar worked, due to the fact that it was very difficult to work with, we had to scrap the idea. I also had to change the pages that I had made several times in order to adhere to the needs of our project and our theme. The Sprint retrospectives each week were released by me and allowed me to gain an understanding of why documenting events during the week would help in the long run as I am now able to look back at those and understand exactly where I could have improved and how I could have done that. Overall, I wouldn’t say I am a hundred percent happy with the work I did, I think if I had more time to perfect everything, it would work out more in my favour but I am still proud of how much I was able to learn. This would both include the technical aspect such as working with controllers etc as well as the team aspect which I know will be extremely useful when I am working in a company.

Thom van der Woude

I think that my contribution to this project is proportionate, given that I had some ambition but did not mean to invest all my time in it. My contribution calcified in the database and the server, but a big part of it also lies in the organization of this project. Although a scrum team is by definition self-organizing, I felt like I had to direct an effort in the first weeks to get everyone to follow the best practices: the code reviews, the testing, the Checkstyle formatting and all other oft-overlooked things that are required of us for a reason. I believe this resulted in the gradual transition from instant-approved merge requests and 14% branch coverage to a process leading up to the final product I can stand behind. In the third week, I started my work on the back-end which has provided me with a valuable practical understanding of RESTful APIs and the team (and client) with a part of our product.

In my personal development plan, the recurring theme was high ambition and perfectionism stained by a hypercritical flaw. I think that I succeeded in avoiding this pitfall most of the time. Though I was still critical in reviewing the work of others, I think that this has been only to the benefit of the group. As for my goal to learn version management, I think I succeeded though I haven't mastered it yet. I think a strong point during this project was the drive to improve our process: I was always looking for the things in our code quality, reviewing and planning that could use some work. One thing that I want to improve upon is positively motivating others to not just deliver a reasonable product but deliver a great product. The serious problems I encountered in the span of this project were of technical nature; sometimes everything worked on Gitlab's CI and failed locally; sometimes a Jacoco version from 2016 broke maven; sometimes the otherwise intuitive JPA gave me a really bad headache; never, however, did my team fail me and for that, I am again grateful. There have been some pronounced differences in the distribution of the work over the course of the project, but I think that towards the end these leveled out.

Dan Dan Berendsen

A strong point of mine during this project is, that I have done stuff in different aspects of making the application. Because of this for most aspect, there were multiple people that understood that part during the development. I was, therefore, able to also extend off of parts others made and sometimes point out an error or solve a problem. I know however that I'm not really good at doing the bookkeeping, because of this I did some of the necessary stuff but left most of the rest to the others.
I do think the communication in our group was pretty solid. We knew what the others were doing and what others have done. Although in the beginning, it was a bit rough for example there were some mix-ups with the branches but we quickly made some good rules for that. Some improvements for me could be taking more leadership/ initiative. I am more in a sort of supportive position where I say what could be done, but I'm not the one to say what has to be done. What ties in with this is that I also got some feedback about taking a bit more initiative in dividing up the stuff we need to do.

Jason Bloom

As a whole, I think the project was successful. We were able to bond as a team and to resolve conflicts quickly as they arose. We were also able to meet all deadlines and get satisfactory feedback on our work. I think all team members worked proportionally to the grade they set out to achieve in this course. Coming from another programming language but similar framework I was sometimes surprised by how much time simple (in my eyes) things took. This caused me more frustration than I would have anticipated at the start of this project.

I think my strong point in this project was taking into account the scope of the project and sizing our features accordingly. Next, to this, I did change my coding style quite a bit to reflect a broader definition of done, including documentation, testing and informing other members.

My weaker points to this project were not being confident and assertive enough with tasks that were outside of my comfort zone. Next, to this, I was not strict enough with following the deadlines and doing the required setup for the smaller tasks like running Checkstyle rapports and setting up the CI pipeline. Luckily other team members were a lot more assertive and punctual in this area so this turned out fine.
The biggest area I would like to improve upon in the next project is being more assertive with tasks foreign to me and having greater care for details.

Max Karsten

I was looking forward to this to this subject of the course more than the others. I know myself to be more motivated when provided with the challenge of creating something. When first introduced to the project, I was both pleased with the relevance of the subject matter, as well as the diversity and flexibility involved in creating the desired product.

Having meet my group and looked over the rubric and discussed ambitions and basic general design, I was feeling pretty good. Overall, I was pleased with my initial impressions of the group members, as well as how motivated everyone seemed to be. Production for the project was a little rough, as one might expect, in the beginning, due to several commonplace issues, such as not knowing how to use Git, along with getting used to working with a new group of people. After a week or so we got “into the groove” so to speak. Everyone found a section of the project or a selection of tasks that they desired. For me, this meant focusing mainly on implementing the features on both the client side and the server side. This allowed me to work with the Spring framework, JavaFX as well as JPA and API’s. This plays to my strengths of being able to creatively implement items, and overall working with complex systems. I was pleased with the number of different elements I was able to learn and really sink my teeth into. While the division of work might not have been perfectly equal, it was divided so that each member was able to do what they felt comfortable with. This has a few benefits, mainly a more relaxed and orderly development cycle overall.

In retrospect, I am impressed with what I was able to learn, and what we as a group were able to accomplish. Of course, looking back not everything was perfect and went as planned. I would say that my biggest weakness during this project was not keeping up with the supporting code/work. Where I was able to develop features at an increased rate. This leads to supporting work, such as documentation, test code, and check style. From this project, I have learned that these elements should have a higher priority in my own personal development cycle. In terms of group work and aggregate effort, I would say we worked together well. One of our best aspects was our communication. Everyone felt comfortable sharing their opinion and expertise. This made working with the group enjoyable. In conclusion, while our project was not everything that we initially wanted it to be, however, this is accepted and even expected. I think that I speak for my group when I say that we are happy, proud even, of the outcome.

Value Sensitive Design

Although we tried in many ways to ensure the privacy of the users of our carbon-tracking application “Greendr”, our design is not using the data of our users in a legitimate way. This seems to stem from contrasting definitions of the value privacy; in our design we considered privacy to be the general prudent handling of and proper protection of personal information; the EU, however, has rather nuanced ideas about the privacy of the European citizen, most notably as expressed in article 16 from the Treaty on the Functioning of the European Union[10] in conjunction with article 8 from the General Data Protection Regulation[11]. This restricts the use of personal data of users under the age of thirteen in any

case and as we never ask for any confirmation that the user is at least 13 years old, the client deploying “Greendr” is in violation of this regulation. Additionally, at no point during the registration process is there consent given by the user to personal information collected by the application, which is in direct violation of Article 7 of the GDPR[12] which states that in use-cases such as ours explicit consent must be given. Another issue, related to article 32[13] of the regulation, is that although authentication is required to access data, the client-server connection remains unencrypted meaning that a third-party has but to intercept the information-packets to gain access to the sensitive user data. All in all, we have to rethink our privacy and user data strategy.

So we have to amend our naïve definition of privacy to be GDPR conform and design our application around this more formal definition. This not only benefits the end-user, whose rights are no longer being violated and who now knows what to expect from our application by reading and accepting a EULA, but the client deploying “Greendr” also avoids expensive lawsuits and huge fines[14], rendering the application as a whole more viable.

Process-wise, a more informed and nuanced approach to treating user information is in order. Both technically and legally the way we approached this tenet of modern application design is lacking, as in sum we are illegally sending unencrypted user data back-and-forth over the internet. To gain theoretical insight into the EU’s definition of Privacy and its relation to our stakeholders, the end-user and the client deploying “Greendr”, various information sources could be consulted. A good first step towards more nuanced data handling would be a literature study on the subject of data breaches, GDPR violations, and pre-2018 user right violations. A comparative study of the existing body of EULA’s of web-based companies could also aid in writing a watertight EULA. This could be supplemented by readings of (summaries of) important regulations concerning user privacy. Employing a data protection officer or hiring a law firm with past experience in the field of cyber-law and privacy should also be considered to rule out future legal issues. Product-wise, one thing that seems to be missing is a checkbox at the bottom of the registration form referring to a EULA, the specifics of which are to be determined through the study of relevant regulations and past violations of such regulations. End-to-end SSL encryption should be considered as a technical measure to avoid future data breaches.
Through nuancing our definition of privacy and adjusting our process and product accordingly, the end user's rights can be protected and the client avoids catastrophic legal problems which could arise from a reckless approach to a matter as sensitive as user data.

References

1)https://material.io/design/
2)https://www.freelancinggig.com/blog/2018/04/26/spring-popular-java-frameworks/
3)https://dzone.com/articles/why-spring-framework-is-popular
4)https://medium.com/@wkrzywiec/why-spring-framework-is-so-cool-8472ceabaab1
5)https://www.tutorialspoint.com/spring/spring_overview.htm
6)https://hibernate.org/orm/
7)https://openjfx.io/
8)https://www.oracle.com/technetwork/articles/java/casa-1919152.html
9)https://www.android.com/versions/pie-9-0/
10)https://eur-lex.europa.eu/eli/treaty/tfeu_2012/oj
11)https://gdpr-info.eu/art-8-gdpr/
12)https://gdpr-info.eu/art-7-gdpr/

13)https://gdpr-info.eu/art-32-gdpr/
14)https://www.theguardian.com/technology/2018/oct/25/facebook-fined-uk-privacy-access-user-data-cambridge-analytica