Product:

At the start of the project we needed to make quite a lot of design and architectural decisions. Since the task was to build a client-server software system, so there were three main components which required to be decided upon – the server, the client and the database.

For the server we decided to use Spring Boot framework. It is very popular and extremely easy to use. It allows to create a web server mapping in just a few lines of code and requires almost no configuring.

For the database we chose to go with MongoDB because it is very closely connected to Spring framework and it makes storing and retrieving data from the database as easy as one call to a Java method.

Also, we wanted to host our application on Heroku, so that we didn't need to run a server on localhost but rather we can just run a client on any computer and no local server is required. Therefore, for that we would also need to store our database on the cloud. Luckily, Heroku offers some databases and we decided to use mLab MongoDB database because it is free, it offers 496MB of storage which is more than enough for our needs and it is MongoDB so it will be really easy to integrate it with Spring framework.

As for client we decided to use JavaFX for UI, but actually we agreed to build UI manually in code instead of using a scene builder, because this way you can have more control over UI and its components. Also we decided to use mainly green theme for UI and have a background with green forest since this application is about going green and saving the environment.

For sending requests and receiving responses on client side we are using REST Template library from Spring framework, because it's very easy to use and makes handling responses and sending JSON objects very easy.

Concerning security measures, we have it in a way that passwords are never stored as plain text, they are always hashed using SHA-256. And since our server side is based on Heroku which offers HTTPS protocol, all of the data sent between client and server is encrypted. And even if you left your app open and somebody tries to delete your account or change your password while you're gone, he would not be able to do that since all serious actions like resetting points, changing password, deleting account etc. can be done only after entering correct password.

And to make our application as sturdy as possible we did quite a lot of testing on all of the components of this software system. We have a test branch coverage of 100%. We are also splitting up big tests into many small tests so that it much easier to see what exactly failed and fix that.

All of these decisions make our product accessible, user-friendly, secure and reliable.

Process:

At the beginning of the project we discussed the requirements and what we were going to do to distinguish ourselves by the problem we wanted to solve. The problem was the lack of personal responsibility for the environment. Our solution was to give everyone their personal little planet for which they were solely responsible and had a huge effect on.

After this we met Yoshi and spoke about our ideas on which he agreed. After the meeting with Yoshi we all went to a project room to discuss our application further. We started by drawing a concept of every page with the buttons and their functionalities so we could make an atomic list of the features our application was going to have.

Then we split our group into 3 different subgroups: UI, client side and server side. For the first demo we had to show communication between the user and the server. So we started working. Our application was demo-ready well before the deadline, however the division of tasks in some of the subgroups was off. this was mostly due to how little needed to be done. We were also not using the issues on GitLab.

In the sprint thereafter we decided that we wanted login and register functionality. Work on this went fine, however with limited use of the issue board on GitLab. Later in this week we met again and talked about where we were and what needed to be done.

At the next meeting with Yoshi the client side team members were missing and we could not get ahold of them over the phone. We found out nothing had been done on the client side that week. So we restructured the subgroups this time with more and different people on the client side to make up for lost time. Among the present members there was still some disparity between the workloads of each member. This was mostly due to a difference in experience and knowledge between the members.

In the following sprint we wanted to catch up on the client side and be ready for the second demo. The Login page and the Register page were done pretty quickly. And one of the two missing members returned. After that we started working on the add action page for the second demo and the settings page. At this point we were using the issues on GitLab quite effectively.

At the demo everything worked fine, however upon further inspection of the code Yoshi saw that we have not been using JSON to communicate with the server. This was due to us not reading the rubric. This cost us 0.1 on our final grade. We did however test all our code sufficiently and thus got all of the other points. The workload among members for this sprint was also quite nicely distributed.

Now we had learned our lesson we began with reading the rubric for the last demo. We decided we wanted to have pretty much the whole application done for the last demo so we got to work and of course changed all our communication to JSON.

We met with the group 2 or 3 times a week to keep each other updated about the progress and help each other. Use of the Issue board was very effective. During this sprint we: added css to the app, added the leaderboard functionality, added friends list, added profile pictures and we added achievements. The app was pretty much complete with the exceptions of some minor touch ups to the css for example.

We passed the last demo with flying colours and Yoshi was very proud of us. He did not explicitly say so but we could see it in his eyes.

Our only shortcoming was that the workload of our group was not equally distributed. At least on paper there was a large divide.

For the next and final sprint we wanted to distribute the workload a bit more evenly. We were only going to do improvements to the css, bug fixes and cleaning up the code. However 1 member said he could throw together an android app fairly quickly since he already had experience, and thus the group agreed, there is going to be an Oppy app.

Reflection:

Product

One problem we now have it that we store the score of players as an Integer. The problem with this is that an Integer has a maximum and a minimum value. When a value beyond one of these values is reached the value is changed from positive to negative and vise versa. Negative overflow isn't a problem because you can at most lose 3000 points a day, which means it would take more than 209 years to reach this value. Positive overflow is possible in some extreme cases where the user builds large amounts of solar panels though. We could have postponed this problem by using a Long instead of an Integer. This would put the maximum score so high that nobody playing the game without cheating should be able to reach it.

Another problem we currently have is that we don't have anything to keep someone from creating thousands of accounts through a script, eventually overloading the database. This could be prevented by integrating a captcha into the register page. However, because of time constraints we didn't figure out how to do that.

Even with a captcha people would still be able to make thousands of accounts per person. This also could cause a lack of space on the server. We could prevent this by creating an email server that sends a confirmation email whenever someone tries to create a new account. But seeing as it would take a lot of effort to figure out how to setup the server for that, and even more time to actually get it running, we decided against it.

What also would have been nice for the user experience is to create an option to login with Facebook. This would make the app more easily accessible for people and could also take some pressure of the security of our database, seeing as then fewer passwords would be saved on there.

Process
Furthermore, the division of work wasn't always optimal. Especially in the first few weeks some people consistently did more work than others. This was partially because it wasn't clear for everybody what they had to do. Luckily we learned from the first weeks and later I felt like almost everybody did roughly the same amount of work. What we could improve next time is to always make a clear division of the work and make sure everybody has an equal slice of work. Also when we're done early with our tasks we should ask in the group if other people need help with their tasks if they can't get them finished in a reasonable amount of time.

Something else we should improve is using the issue board that gitlab provides from the beginning and constantly keeping it up to date. We didn't always use the issue board when working, which sometimes led to two different people doing the same work. It also wasn't clear who was doing what before we started using it. On top of that, if we would use it from the beginning we could get an nice overview of who did what.

One thing we could definitely improve is to change up everybody’s roles. We divided everybody into three separate roles, either UI, client side or server side. Because we didn’t really change the teams around however most people only have a good understanding of part of the code, and no one really understands everything. This later caused issues, because you needed like two or three people to solve one bug or add a new feature.

During our next project we should take a good look at which data we do and don’t need at the start of the project, so that we don’t store any unnecessary data. Otherwise you end up with useless data. For example, we have an option to store a profile picture. This is a user property, and but we don’t really use it. It only gets displayed to the user themselves, which is a bit redundant, especially when it so drastically increases the size of the data stored.

The last improvement I want to mention is that we should read the rubric very carefully, so that we exactly know what we should do. This also means we should look up the stuff in there that we don’t know or understand. This would prevent us from having to change big parts of the code halfway through the project or losing out on points. both of these things happened to use when we forgot to use json communication with the server, which cost us 0.1 points and forced us to redo all previously done server communication.

Course

Overall we think the course was handled pretty well. However the ethics part of the course didn’t really connect to people. It was a bit too philosophical at times. We would recommend that you would keep the ethics part more practical and more focused on its place in computer science.

The only other real complaint is that the information literacy felt more like busywork than something you're supposed to learn from. We think it would be better if you would give a practical assignment, like write a short paper, than only theory questions. This would force people to really understand the stuff learnt in this part of the course, which is actually really important, and also give them some experience for writing papers in the future.

Individual feedback - Hugo:

I started the project on the server side group, I was a bit taken aback by how much skill my colleagues seemed to have in this area. This resulted in me under contributing for the first 2 weeks. I did try but I would just get stuck very quickly while these guys soared right past me and did everything. After the whole debacle with the client side guys going missing I transferred to the client side. Here I was able to pull my weight and started working on the Login functionality. I am also wholly responsible for all client side on the settings page (except for the profile pictures), the leaderboard page and the showing of your score name and email on the main and settings page. On the server side I am only responsible for a large part of the User class and the tests for that part. I also made the UI and css for the minor pages (conformation pages). I also came up with the gimmick for our app which is the personal little planet.

At the start of the project I really did not know more about coding than we had learned in OOP. So it took awhile for me to become productive and my team members were a huge part in that they explained a lot to me during our meetings.

My weakness which came through the most was my inexperience with java. And subsequently asking my teammates a lot. It did result in me doing a bit less than I would have liked at the start of the project. Luckily they were very helpful and did not get annoyed at me. My strength was taking initiative. I did not stand around the whole project, because I did not know how to do anything yet, but I took the initiative to learn and subsequently pull my weight in the project.

My goal was to get better at java and be able to make such an application on my own given the time. I did get a lot better at java and I did do work on every part of the application. I am now pretty confident in my UI and client side knowledge however I feel i lack the knowledge for the server side part.

Individual feedback - Mihhail:

My job during the project was mainly connected with developing everything related to the server. I wrote almost all the code on the server side and I was the one to set up the database. We started as a team of 3 people working of the server, but after a week of work it was clear that client-side team needed more manpower and so two of my other team members went to work on the client-side. I was quite familiar with Spring Boot framework because I already used it before so it was clear that I can develop a server side by myself.

Part of my work was also developing the server API – what requests are sent where and what responses to except from the server.

Although working on server and developing its API took a lot of time, I still spent about half of my time writing tests. I wrote almost all the tests for server side and tests for simple classes we use for data like User and Action.

Also, during last week of the project, we decided we have some time to make an android application. One of the team members volunteered to make the app, but I thought it would too much work for one person, so I joined him and together we made the client application for android.

My strongest point before the project was having experience in Java and developing this kind of applications and my weakest point was lack of team-working skills and absence of habit to write tests for my code. My main goal was to obtain these skills and I think this project really helped me with that.

individual feedback Duyemo Anceaux

My task in the project was mainly creating the UI for the application. Our team decided to do this in javaFX instead of using a scenebuilder. I have created the basis for all the big UI pages except for the login page. Also I did the css for all the big UI pages, this time including the loginpage. I was actually supposed to do the UI together with Benjamin, but he switched to client side, after they got behind in the first weeks and Markus, who was supposed to be working on client side just disappeared.

Because I didn't have any prior experience with javaFX it took quite a bit of time to learn how to use it and to create the basic UI for the first few pages. This caused me to do get a little bit less work done than other team members in the beginning. But after the slow beginning I really got a lot better with javaFx and later weeks I could just quickly add in stuff in the UI without having to think about it that much in most cases. Only the leaderboard really gave trouble after that.

I'm also responsible for the backgrounds, which I got from "www.pexels.com", a site with copyright free images, and made all the very beautiful achievements (in 10-15 minutes).

The css also took up a quite a bit of time for the first page. This was because although I had experience with css for html pages everything was named different in javaFX, and you couldn't determine the precision of objects as precisely as in html. I had to use row and column constraints witch, although easy to use, took up a lot of space and time.
In the end is also organized all code in the UI, because it was a mess. This didn't take that much time but was very frustrating because methods became so long I got checkstyle errors and had to move parts of the code into separate methods, witch I mostly did with code in lambda expressions.

My goal in the readme was the learn how to work together when programming an application. I think I really succeeded in this part because I learned how to divide task, how to keep track of what everybody is doing and what already is done and how important communication is.

I also feel that my weak point in getting distracted while working with a group had gotten better then it was, because when doing specific stuff while working with the group I could

foccus pretty decently, although at times I still did get distracted when not working on a specific thing. Also, I still have a problem with tunnel vision.
I think for my strong points I indead delivered my work on time, and I didn't really need to other one, witch was working until late at night to finish something, although I sometimes worked late at night because my sleeping pattern is close to non existent.

Individual feedback Saman Shahbazi:

At the start of the project we decided to split the group in three smaller teams. As some of the guys informed us that they didn't have much experience making a product like this and I had some experience with REST, I volunteered to join the server side team with two other colleagues, Hugo and Mihhail. We decided on using the spring framework as Mihhail had some previous experience with this framework. I thought this was nice since I had only used the Play framework for Java and this would familiarize me with other frameworks for Java.

Mihhail was very quick to start writing the routes for the server. At the start I wrote the first DB controller methods and implemented the *local* MongoDB and User repository. Since the first demo only required minor interaction this was sufficient in the first few weeks. Furthermore, on the server side, I wrote a few routes regarding password updates. In the first week I also investigated CI on Heroku but gave up on that as I was approaching it the wrong way.

Early on, I noticed the importance of the backlog and atomizing of issues. We were lacking in making use of the issue board and features provided by GitLab. So I decided to take an active role in creating a proper backlog and enforcing issue creation from the backlog and I believe this helped us focus on the tasks that needed to be done.

After we noticed that client side was lacking work and a team member subsequently leaving, I joined the client side team. I was responsible for the design and implementation of the ClientController class and the tests that were involved.

In the last week, after seeing if there was any demand, I started developing an Android equivalent app of our program. Soon, I was pleasantly joined by Mihhail in this endeavour.

My weak points starting with the project were: lack of experience in a team, lack of experience and passion in UI design, not being able to ask for help in time and lack of desire to write tests. I believe the first and last two were helped quite a bit - however I still don't like working on UIs. I am now much more fond of writing tests, as I have seen how it can save you much work in the long run.

We had some issues with group members leaving and lack of work distribution. However, I believe because of the non judgmental nature of our group, we stuck it out together and everyone pulled their weight to finish the project, especially in the last few weeks.

Individual feedback Otni Ledelay:

At the beginning of the project I was unfamiliar with the git workflow and tools. This slowed me down a lot during the first few weeks of the project.

I ended up falling behind the others in contributions and understanding of the project and we had a meeting about this. After that meeting I picked up my pace somewhat and things went a lot better.

Eventually I made the login popups, the profile picture part of the settings page, introduced popups for login/register pages, separated some of the client-side code out of the UI classes and designed the logo and planet faces.

At the beginning of the project we specified our goals for this project and I have mostly reached them. I wanted to improve my tendency to postpone things and learn about working on a project as a team.

I feel like I have learned a lot from this project, both through failures and things that went well.

During this project I learned how to use git, how to use maven, how to use issue boards and how to keep a master branch safe. I learned about working in groups, the importance of group meetings and proper communication.

Aside from our loss of Markus, I really liked the group as a whole. The group was very understanding and supportive. I can't think of a single time anyone from the group was actually mean to another person in the group.

Individual feedback Benjamin Knol:

I have done this project once before however last time the only thing I was able to help my team with was the final presentation. This year I was a lot more useful to my group, and I helped with many things. The only really big regret that I have is that I didn't really help out at all with server-side stuff but mostly worked on client-sided features and started off building the basics of the UI which Duyemo built out to the UI that we have today.

At the start of the project we split ourselves into three teams: UI, client side, and server side. I chose to be on client-side. Splitting ourselves up this way may not have been the most efficient but I think it did cause us to build a stable base from which we kept building.

For the first demo I was up to date with everything that had happened on ui and on server side, however the client side was a bit of a mystery thanks to a lack of communication from their side. Since the first demo was a simple communication with the server luckily not much was needed.

The following week the ui and the server side grew considerably however the group didn't hear anything from two of our team members and when the next meeting on monday came around they simply failed to show up so we realized then that nothing had been done on the client side whatsoever. After realizing this everyone jumped in on it and we had the client side work done within the day. If I remember correctly it was Sam who created the basics which the rest then quickly understood and started implementing ourselves. I think as a team we worked well together there and overcame our problems.

To avoid the problems where a third of the goal for a week was missing we stepped off of our team idea and instead we started using pull based development by having issues and using our issue board effectively. This was one of the things which I got better and better at during the project. It gives me a lot more structure and I will attempt to use this effectively in other projects as well.

After this there was one week where I achieved little for the project because I didn't spend enough time on it myself but during that week I did of course do some code review and tried to help other as much as I could with their coding work.

After this bad week I pretty much had it down and was able to work efficiently and effectively.

Before the project I said I was pretty good at git so when issues came up I could fix them. I didn't have to fix a lot because people luckily understood it decently well from the lecture about it but during the first weeks I had to help some of our less experienced people a bit but they picked it up pretty quickly as well.

On the improvement of my weaker points I can be less happy because both of those have shown I think. For example I said I was pretty bossy and during the project I'm pretty sure my teammates noticed this as well. I did try to be more open but I don't think I did enough. Also I said I would try to improve my testing but the only real testing that I did was simplistic and 'good enough' but not excellent which is what would've been better for me. To improve this more I should maybe have asked Mihhail for some help on this as he wrote most of the tests.

Ethics current topics: (see images below) Benjamin

- Kid friendly value (talk about trade offs and why we choose this value)
- Security value (again trade off and why we choose this)
- not encouraging stupid actions

Apart from making a application which would greatly help to improve the environment, we wanted to also uphold certain other values I.E. we wanted to make it understandable for kids, we wanted to create a secure and private application, and of course we wanted to avoid, as much as possible, people making choices which are bad for the environment.

Child friendly application:

We wanted to make sure our application was as child friendly as possible, not only to reach the largest audience possible but also to help the biggest benefactors of the main goal of our product (and also one of our core values): improving the current outlook of the state of environment when these children and young adults will grow up. To make it child friendly we of course had to make sure we focused a lot on privacy and the ability to stay anonymous in our project meaning there was no way for other users to figure out anything about you, not even your points. However to follow this course of action means we had to divert attention to this anonymity meaning we wouldn't be able to create a comprehensive friend system but instead settled for a simpler system where you could choose to follow other users (unless of course they were anonymous) to see their points and so you could keep track of how your friends also using the app were doing.

A focus on security and privacy

During our very first meeting where we settled on a design, we also kept repeating to each other but what about security and privacy? The funny thing is we never stopped saying this. Throughout our project we wanted to make that not only you could of course have a lot of fun with all your friends who are also using our application, but also that users have the choice to opt out of this and just use the app anonymously. As mentioned in the paragraph above this focus on privacy and security did mean we had to make some choices. Another example of this was that originally we wanted to set up an emailing system so people could request a reset password email and an option to get daily updates about the environment as a whole and the users part in this. Unfortunately we were unable to implement this as time was starting to be a serious concern when we entered the last few weeks.

Another important reason to why we chose privacy as one our core values is because this caused for us at all time to consider precisely what data we were requesting from our user, and what we were doing with this data. This made it a lot easier for us to make sure we were GDPR compliant because we knew exactly what we had of our users and how we should use this information.

Avoid sending users astray

Having an application to influence people to make choices regarding the environment is generally considered a good thing however if it causes your users to actually worsen their behaviour you probably shouldn’t have created the application. To avoid this exhaustive research was done to both figure out precisely what the effects of people’s actions was on the environment and how to keep users enticed to keep coming back everyday. To make sure of that second second one we chose to set the daily decay of points pretty high so that just by cycling to work everyday you wouldn’t get enough points but when you do a few things extra you can get extra points during one day. The way we did this research for the first is by focussing on official numbers mostly provided through research done by the EU on the environment. The way we researched the amount of points to lose each day was done simpler; we just looked in the group how many points we would, could, and might get and then set the points in accordance to this