OOP PROJECT REPORT

Group 32

PAUL FROLKE, DENYS SHEREMET, MARVIN EVORA SOUSA,
NATHAN KINDT, JUSTN SUN, ROHAN DESHAMUDRE

Paul Frolke (4888693)

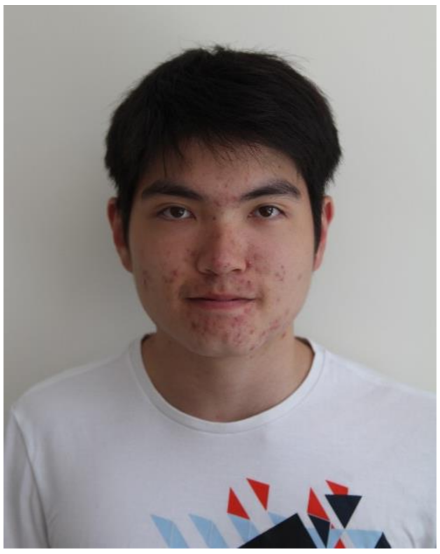
Marvin Evora Sousa (4674138)

Denys Sheremet (4960564)

Rohan Deshamudre (4831098)

Nathan Kindt (4916484)

Justin Sun (4698460)



General: How did the project go?

1. Did you manage to stick to the planning (why(not))?

The structure of the course was well designed to help us plan well and stick to the deadlines. The demo deadlines made sure that we completed the certain main tasks at the right time which helped us stay on track for the most part. Also, since during each meeting we discussed what each of us did over the week and planned the tasks for the upcoming week, we got an understanding of where we stand and how much time and effort needs to be put in for next week. These reviews helped us realize that if we are falling behind, we need to work hard in the upcoming days to catchup and get back on track for the next deadlines. Tools such as the git board and the meeting agendas, notes and sprint reviews gave a good summary of the work we did and the work that is yet to be done. It also helped identify who is responsible for which task so there was no confusion about multiple people working on the same task.

Despite the planning and organization, we had, there were situations where we were not able to meet the deadlines that we set to ourselves. A big reason for this was because of underestimating how much time a task could take. Since this is the first time building an application like this for all of us, we found it difficult to judge the difficulty and time required per task. Splitting up the tasks evenly is also a key aspect of good planning and there were instances where some of the work which is more time consuming did not have enough people working on it. This meant that the work was sometimes incomplete. This put a lot of pressure on the person working on that task and on rest of the team as the work is all interconnected. We did not communicate these problems clearly enough and ask for help from other team members which put is behind in terms of progress in some weeks.

It may be easy to say that we could have asked each other for help which could have solved our problems quicker but, it is also important to understand that each of us were working on different things. This means that, if we were stuck, most of the other people would also be stuck because they do not know what you are working on and how it works until everything is combined towards the end of the week.

Although we had a WhatsApp group for communication and we communicated well about merge requests and about any questions we had, we did not make enough effort to let each other know every tiny step progress we made. This would be helpful for the whole group. Due to lack of communication and collaboration at the end of each week to combine all the components that different people worked on we did not have the master branch ready multiple times before the meeting and, we released a version tag sometimes after the meeting during the lab. This also cost us some point towards the final grade.

Overall, the way we stuck to planning was not the best, but since this is the first time, we have created something like this, we were not too disappointed with ourselves. We also made sure to catch up on everything that we were behind on and have all the requirements ready for each of the demos.

2. How did the collaboration in the team go?

We started off as a full group of 7 people but after the first week, one of the members of our group had to drop out of the course. A group of 6 people trying to complete the work of 7 meant that more effort would have to be put in per person. We had to figure out which frameworks to use, where to host the application and learnt to work with the frameworks we chose without having any prior knowledge or experience. Hence, it took a little bit of time at the beginning to start working well as a group as we were all trying to get accustomed to the new work environment and were understanding the process of the project.

We could say that we worked collaboratively in the smaller groups we made to work on different parts of the project, however we fell a little bit short when it came to the collaboration of the whole team and combining all the work which was also the reason why we kept the version tag and master branch merging till last minute every week.

Along with this, two members of the group were travelling for a short period at separate times in the project. While they were gone, a 7-member project was being done by 5 people. This increased the workload for everyone and since we all have 2 other courses which also require the same amount of time, some tasks were unfinished and had to be pushed to the next week. However, once they were back from travelling, they made sure to catch up with the progress and on their contribution for the team.

However, over time we started to work much better as a group and helped each other when there were issues. We made sure that no one is left behind, and everyone is on par with the progress the group is making. Other than the meetings on Monday, we made sure that we met every Wednesday to make sure everyone is on track with the tasks that they must complete and help each other with the problems we were facing. Once we understood the strengths and the weaknesses of all the members of our group, the assigning of tasks became more efficient and progress was being made.

3. How did you communicate?

Communication is a key part of any group project since it is important to know who is doing what so that two people aren't working on the same thing and it is also important to stay connected with everyone as the different aspects of the project which different people may be working on, all need to be compatible with each other. Our main method of communication was via WhatsApp.

On WhatsApp, we all notified each other whenever a merge request was made so that we could have a look at what changes were made and start a discussion if we had any questions. We also helped each other solve any merge conflicts or pipeline issues. This also improved our git behaviour. Our communication in terms of our daily activity and what we have been doing each day was not very good. We only communicated when the whole thing was complete whereas it is better to keep everyone informed after every small change as then any problems with compatibility can be easily identified and solved before the problem becomes big.

The communication at the end of the week when all the different aspects had to be combined and merged to master was also very poor. We didn't have everything combined on Sunday evening and a tag released ready for the meeting which also made it harder for the TA to give us more specific feedback as he would not have enough time to review what we worked on.

Although the general communication at the start was not the best, it started improving over the weeks and we started to state our progress on WhatsApp more frequently which helped us make more progress towards the end weeks.

4. How did version Control help?

Gitlab and the tools that came along with it were quite useful for the whole group to stay organized and to make sure that none of the code that we wrote was lost. Having a version control means that you do not have to be afraid of making mistakes as reverting to an earlier version is not difficult which will get you back to a working setup. This gave us room to try out new things and understand how the frameworks were meant to be used.

Since each different part of the group was split up into different branches and different people working on it, it meant that any conflicts in code was resolved before merging to the master branch which always kept the main source code safe. The merge requests aspect of Gitlab also made sure that other people in the group can see the changes made and approve and comment on the changes so that the development process runs smoothly. Organizational features such as issues, scrum board were also helpful liked mentioned before. Gitlab also has the feature to monitor the commits of all the people that are part of the team and this helped us make sure that everyone is carrying their weight and that is contributing to the group well.

5. What did you learn?

For everyone in the group, this was the first time we built an Application like this and hence it was a new experience working with all the different frameworks and working as part of a team. We all learnt a lot about team work and how a software development setup would be even at a corporate level which is going to help us even in the future once we start to work. Since we were given the freedom of choosing our own frameworks and deciding the structure of the Go Green App, we were able to understand a lot of the different frameworks that are available and the pros and cons of each of the frameworks. Working well as part of a team is also a very valuable skill that we got to learn from this project. Communicating and making sure that you are involved and are on par with the work going on is important to stay involved with the project in a group and this also made sure that there was pressure of each one of us in the group to deliver and get work done. Since version control and testing was a mandatory and very important part of the project, we learnt a lot about Git, JUnit tests and Mockito tests which are valuable skills to have as software developers.

Design Decisions

1. The first big decision we made as a team was to use Gradle instead of Maven which was recommended by TU Delft. Using Gradle meant that we had smaller configuration files with less clutter. Gradle uses domain specific language based on the programming language Groovy which is easy to read and simple to understand vs XML files for project configuration in Maven which are is not the most user friendly.
2. Other than that, splitting the team into a few people for front end and a few people for the server side was also an important decision that was taken at the beginning of the course. This however was a combined effort once the database had to be connected to the backend which meant we all worked together.
3. The next big decisions were choosing the frameworks that we wanted to use and the approach of creating the application. This decided the whole process of creating the Go Green application, so it was probably the most important decision we took as a group. We decided that we would use JavaFX for the GUI and Spring for the server side and Heroku to host the application.

   Heroku: Our group used the free web host Heroku for hosting our database and server application. Heroku has built in integration for Gradle java applications and code is managed using git, which makes it perfect for this project.

   Java FX: The two main contenders for the GUI framework were Java Swing or JavaFX. We chose JavaFX because it provides a rich set of graphics and media APIs with high-performance hardware-accelerated graphics and media engines to simplify development of immersive visual applications. It uses FXML which is like HTML and is used only to define the interface of an application, keeping it separate from the code logic. JavaFX also allows a Scene builder application to be integrated with IntelliJ which allows drag and drop methodology which automatically creates the FXML document.

   Spring: Spring Boot is one of the most popular frameworks for creating web services in java. Considering the functionality and the application of spring framework, it is quite lightweight, and this is due to its POJO implementation which doesn't force it to inherit any class or implement any interfaces. Spring Boot allows us to quickly set up our application and get it running with minimal configuration required. One of Spring's most powerful features is its dependency injection, which allows us to integrate our database repositories in our server with just one line of code. Spring's annotation-based http request handling enables us to easily create a well-structured web API which can be easily adjusted when needed.

   Mockito: Spring Boot's testing library comes bundled with its own extension of Mockito, which we used instead of the official version. This version enables easy testing of spring web services and mocking automatically injected dependencies. These mocks, which were generally mimicking the database repositories used in the application, could then be used for creating extensive tests of the functionality of our API.

   Http: Because our application works with sensitive login credentials for our users, switching our communication over to https seems like a straightforward choice. Spring Boot allows us to easily switch by changing a few lines in a configuration file. The problem arose however, when we tried to upload a custom self-signed SSL certificate to Heroku. Apparently Heroku does not allow custom certificates on a free account. Therefore, we decided to keep our communication on http.

Points for Improvement

1. How can your software be improved?

Our software still has a lot of room for improvement.

- An obvious improvement would be to include more eco friendly features which people can work on and add to their profile. This would also help to make an improvement towards the issue of global warming.
- Testing is also a very important part of any application to find any bugs and to make it as user friendly as possible. Our code has integrated very extensive unit testing but could use more integration testing in other parts.
- It is also important to write clear and compact Javadoc so that it is easy to understand what is going on even if you may have forgotten what a function was meant for. Our documentation is not the most consistent everywhere and has a lot of room for improvement which would be very helpful especially in a group project.
- We could have also made a WebApp and an Android App for our project and this would increase its usage by a very large margin if it were to be released to the market.
- One other thing that we could have done better towards the end of the project was make sure that that any input validation performed on the client is also performed on the server.
- Instead of doing all the calculations by hand using numbers from multiple sources on the internet, a CO2 calculator API could be used.

Responsible computer science/ Improvements

User security and data privacy is one of the most important things for any application. This also concerns the practice of ethical programming and being a responsible computer scientist.

When it comes to security, the most fundamental concepts are Authentication and Authorization. Making sure that users log into the right account and making sure that no one else can log into your account other than yourself is one of the most important things. To make Authentication secure in our app, we made sure to not save the String passwords of any user in the database. With the help of spring security, the password is hashed, and the hashed password is the value that is stored. This makes sure that even if the data in the database can be somehow accessed, the hackers cannot find the password of any of the users to be able to log into their account.

Something that we could have done to improve the security of our project was to use HTTPS in production to secure the communication between the REST API and the HTTP client. As mentioned above, one of Spring boots most powerful features is the ability to inject dependencies. However, attackers target open source dependencies more and more, as their reuse provides many victims for a hacker. We should also test our dependencies and it’s important to ensure there are no known vulnerabilities in the entire dependency tree of your application. There are multiple other things which we can take into consideration to make the project more and more secure, but this would require a lot of time and experience.

2. How can the process/collaboration be improved?

One of our biggest problems was communication. We think that if this was improved and we made sure to ask each other for help the whole process would have sped up and we would be making more progress every week. In any group project, it is important that someone takes initiative and leads the group. During the starting weeks, none of us really took this initiative and we were working on our own things rather than collaborating with the rest of the team. Towards the later weeks, we assigned a person to supervise what everyone is doing and make sure that everyone is on track and provide any help if required. These weeks were the most productive weeks for us and if we did this from the very beginning, we would have had more time to combine all our work and improve on what we have created.

Time management was also one of the biggest problems we faced. Since many times we were unaware of how long a certain task would take or how difficult it is, we did not have enough people working on something which meant that the task could not be completed by the deadline set for it which put the whole group behind. The work that was not completed was pushed to the next week so when it came to the demo weeks, the whole group was under a lot of time pressure and stress.

A problem that is applicable to all of us was that we were not proactive. Although all of us made sure that we completed the work assigned to us, we did not put enough effort to make sure that everyone else in the group was on the same page and that equal effort was being put in. It is obvious that some tasks are easier than other so if someone was done with their work, them helping someone else who is having issues getting something to work would have been a more efficient way to approach this project. This would have also become easier if we told each other about the problems we were having right when we were having the problem on WhatsApp rather than at the meetings on Monday.

Since the team was split into smaller groups each working on different parts of the project, it was also difficult to understand what the other team was doing for someone to be able to help them. Everyone not having complete knowledge of the whole project is an inevitable problem in any group but making effort to understand the issue and trying to help would have made the whole process much smoother for the whole team.

Individual Feedback

Rohan: During the start of the project, since I was away travelling for a week, I was not able to make a good impact to the start of the project. Since a lot of progress was already made once I came back, it took me a little while to understand how everything was working and get accustomed to the project. However, I spent a lot of time and put a lot of effort to get on par with the work done and continue with the project and make a good contribution. I worked on connecting the database first week one I was back and then later after it was connected, I worked on both the server side and the database side to make sure the vegetarian meal feature was working for the first demo. Figuring out how a feature was implemented took me a long time as I was working on it alone and the team relied on me to figure out how it works so we could meet the deadline for the vegetarian meal. Thankfully I was able to deliver and continue working further on getting all the other features to work after that. There were times when I felt like the work may have not been split equally unknowingly which meant that there was a lot of pressure on me, but this is part of team work and I don't have complaints about that. I think my communication with the group was quite good and I shared a friendly relationship with everyone in the group. I think my strength was being able to cope with the time pressure and work load and make good progress over the weeks but my weakness was that I took too long figuring out solutions to errors for which I should've asked for help from the others. I wasted a lot of time on this which could have been avoided if I had sought for help from the rest of my group. Overall, the project was a good learning experience and I enjoyed working in a group of motivated members.

Nathan: This OOP project was a very different experience for me than the Web project last semester, and I've learned a lot from that difference. The web project was mainly a individual project for me, and I was easily motivated to work on my own ideas. At the beginning of this project I thought that working in a group would be easy for me, but I encountered a lot of difficulties. The first one was more an individual problem, getting started with the project was difficult for me especially, since I had no prior experience and I have a hard time understanding it. Only after three weeks or so did I know what is was that Gradle did. Plus, git was also mostly new for me, so it was a rough start. What didn't help was that no one really had prior experience either (or one person, but she left early on), so I mostly waited (or tried and failed) until other people of my group figured it out. This left me a bit demotivated for the project at the start and catching up was hard. I tried implementing Spring Security (which was stupid looking back since I had a hard time with Gradle as well) but ended up with "researching" for two weeks and no code. After that I didn't really take any bigger responsibilities for tasks. Made message classes, adapted database to those messages, couple of tests, but ended up with ~5% code participation. This is week 8, so currently I'm working on Mockito test with Paul helping me and hope to get code participation up with those.

Another difficulty is the organization of the group. I myself were too passive since the beginning, so I didn't take responsibility for the group (or myself) and took no initiative. And that didn't work for me or the group since nobody else really did. That's something I'm myself to blame for, and I learned from it. Overall, I learned a lot from this project, and wish I could do it all over again with that knowledge.

Marvin: I worked quite a lot on the GUI in the beginning and then Denys took it over from me. After that Rohan, Justin and I focused on the database part of the project. I was responsible for managing the tables in the database and the entity classes for those tables. And now at the end I worked together with Denys on making the users info and friends be visible to them through the GUI, me being responsible for the database part in those features. I feel like I did well in the project for the parts that were assigned to me. I also asked for help whenever needed so I wouldn't stay behind too much.

One of things I improved on is communication, with regards to the parts I made in the project. I made sure to participate in our meetings and made my opinion known as well. I also asked questions most of the time when I was stuck or needed help. My weak point would be not starting on time. I would mostly still finish in time, but if I would start a bit earlier with my assigned parts, my parts would have finished faster, and I would be able to help on other parts. Which could have accelerated the pace of the project a bit. I'd say the most conflict I had with others within the group is not enough communication about who is doing what and if the thing that somebody else was busy with was done or not. We solved this by following the advice of our TA to assign someone to check what everyone was doing for that week, the progress of it and what still needed to be done. For that week we increased our efficiency and had done much more week.

To conclude, I had a lot of fun working with everyone on this project. Everyone had the motivation to work on it, it was just the communication that we had issues with.

Denys: At the start of the project I worked on the server. When we were done with that, I started working on the GUI. That was easy and fun to do, so I continued doing that until we had problems and I became responsible for the week. Then I wrote almost no code and talked a lot with everybody and solved general problems. After that week my role became much less clear because I was working on the GUI and the server, but I also kept my role a bit in solving (merge) conflicts. I feel like I did a good job of reaching out to people and when I needed help. Asking for help when someone knew something much better that I saved me a lot of time. I am also happy with the way I tried to help others. I spent a decent portion of my time on helping with tasks that were not directly my responsibility. Often this took a long time and I did not write much code. That was effective usually. The week when I was responsible for everything went well. That week I asked everybody daily how they were doing and tried to solve every problem. When we worked together, I was also almost always one of the last to leave.

But it was not always that positive. For example, when we really needed to connect the database, I started helping with connecting the database, while I didn't know much about it because I was working on the server before. I spent about four or five hours and, in the end, did not write any code, but only gave a lot of suggestions and kept the mood good. I have mixed feelings about this because on one hand I could have helped a lot more if I had read the documentation of how the database worked, but on the other hand it was logical that I didn't know much about it because I was not working on the database. I also messed up a couple of times by not finishing something completely. A couple of times I almost finished something, but then left the last part for the last moment or forgot about something and because of that we all had to act fast.

During this project I think the biggest problem we had was that we were too passive. We were worried about getting our own part done, but then took too little initiative when either we or someone else needed help or something to do. We did not take enough initiative to make sure that everybody (including yourself) could be useful and got help when they encountered big problems. For example: in the first weeks we divided the group in two parts. One group would work on the server and the other on the database. The server group was done fast with almost everything we needed for the second demo. The database group on the other hand, had connected the database to our program two or three weeks later. Right before the second demo. In those two or three weeks the server group mostly sat back and enjoyed being done with their part, while the database group struggled and did not get much done. Only in the last week before the demo, when we really needed to connect the database, did the two groups work together. Before that nobody made the effort to either ask for help or ask how the other side is doing.

A big part of this problem was that we did not really understand what the other side was doing. And because of that it was hard to help, even if we wanted to. Good code reviews would have mitigated this problem. For most of the project our code reviews were not good because we thought that they were all about check style and testing. But I think that if we would have asked a ton of questions about how everything works on every merge request, we would understand much better what other team members were doing and be able to help them. We tried to be more of a team by making one person responsible for a week, and that worked well. But I think that it would have been even better if we would have been active with the code reviews and asked a lot of questions. Because then everybody would understand what the others are doing and not just one person.

To conclude, despite all the mistakes I and the group made, I am happy with how everything went. I tried doing a lot of new things and failed often, but now I can prevent myself from making all those mistakes again. The end.

Justin: I worked on database for the project at first, then I worked on the adding tests in for the classes of database, and I also worked on making the leader board working. All tasks I have done didn't go too well, as I struggle sometimes but I learnt a lot from it.

I think for me in this project, the weaker point is that I don't have a lot experience on programming as I didn't get a very good result for the course OOP as well, so I couldn't help others, but I must learn everything. Another weaker point of me is that I am a slow learner and I work not efficiently. It always takes me a lot more time than I can imagine learning something new so during this project, sometimes I fall behind. I also didn't work efficiently as I spent a lot time on easy tasks which is not very good for the progress of the project. The stronger point on this project is that I am willing to learn what I'm not familiar with and I always ask for help if I am really stuck, so that I won't spend too much time on something unnecessarily.

For our team, I think we all are very friendly people, so we seldom have conflicts and although sometimes I make stupid mistakes or ask stupid questions, my teammates are always kind to me, and they are always very willing to help me fix my problems. Despite we have conflicts during the project, we still managed to be friendly to each other and solve the conflicts in a smart way that won't cause any issues within our group. I am grateful that I was put into this team because all our teammates are kind.

Paul: I enjoyed working on this project a lot. I learned so much about programming in a team, git, Gradle, mock based testing, spring and a bunch of other things. Even though the communication in our team was not great, after every meeting I felt like everybody had a clearly assigned task which would move our project forward. The feedback our TA gave us every meeting was very helpful as well. Personally, I found many complications in combining my super irregular schedule with working on the project, and I still have not solved this problem. Every time I would finish all my tasks for the week in a single stretch of time, to then completely forget about the project for the rest of the week. While this did allow me to complete my tasks each week effectively, a more consistent approach would have progressed our project a lot better, allowing me to keep a better overview of what problems my other team members are running into too. If I could do this project over again, I would put more effort in changing this behaviour. To conclude, I learned a lot this course and I'm looking forward to the next project!