OOP PROJECT REPORT

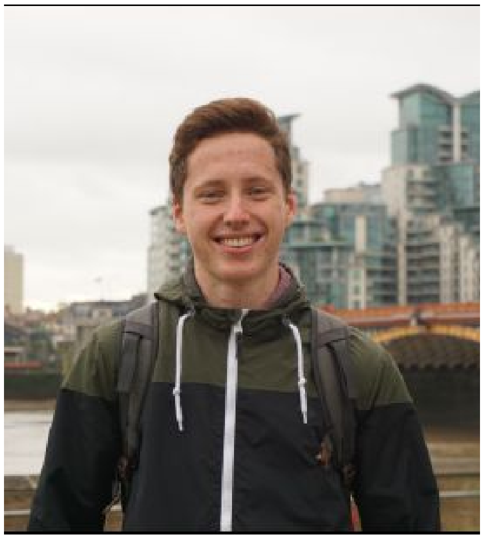
Group 32

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

Paul Frolke (4888693)

Marvin Vora Sousa (4674138)

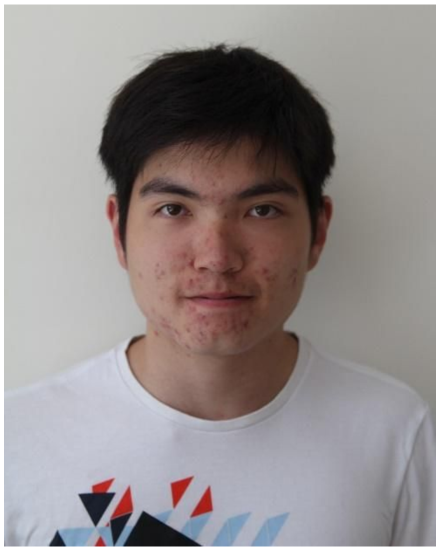
Denys Sheremet (4960564)



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. 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.

The documentation such as the meeting agendas, meeting notes, scrum board, sprint reviews etc made sure that we reflect on what we did so that we can understand what went well, what could have gone better and how we can improve for next week. The scrum board was a good way to keep track of which task is assigned to whom and whether it is completed or not. At the end of the week when the sprint retrospectives are written it also helps summarise how our plan for the week was executed and gave a perspective of if it was a successful week or not.

All of these tools plus communication with the group constantly over WhatsApp was the main reason we could stay on track and stick to the planning for the most part. However, sometimes there were instances where the time a certain task would take was underestimated and some of us were not able to meet the deadline. This could have disrupted the overall planning as the whole groups work is interconnected. 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 which meant that it becomes harder to stick to the personal deadlines we set. These things become difficult to judge as any kind of obstacles may occur, therefore these mistakes are bound to happen. However, considering it is the first time creating a project like this for all of us in the group, we were quite satisfied with how we did and stuck to the planning.

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. Along with all of this, we had two people in the team travelling for short periods of time during different weeks of the course which meant that their work output was comparatively less during the time they were away which had to be compensated by the others.

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 have to 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. We made sure to let everyone know what each of us were doing and if any help was required. The group chat was also important to improve our git behaviour as whenever merge requests were made, we mentioned it on the group chat so that the others can review it and then approve the request. The scrum board on Gitlab also helped with the communication as at the start of the week, we all made sure to add the tasks we were responsible for which got rid of any kind of confusion about who was responsible for what. At the start of the project, communication was not the best which lead to some people working on the same things and also some people being very lost about what needs to be done but this improved with time. For most of the weeks, the tasks were made so that we worked as smaller groups within the group which meant that communication with the smaller groups was important. Even if the whole group may not have communicated everyday, we communicated within the smaller groups to complete the assigned tasks for each week and then collaborated with the rest of the group towards the end of the week or as soon as the task was complete, to combine the different parts that each small group worked on.

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 kept the main source code safe at all times. 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 also 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 also Mockito tests which are valuable skills to have as software developers.

Design Decisions

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.

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.

The next big decisions were choosing the frameworks that we wanted to use. 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.

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 similar to HTML and is used only to define the interface of an application, keeping it completely 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.

Responsible computer science

For eco-activists our app can be very useful. It allows them to give the right example to their friends, and show them that they are actually putting in the effort they expect from others, instead of only being vocal.

Local grocers could also be influenced by this app. Users of the app are encouraged to buy organic produce from local grocery shops instead of from big chains due to carbon reduction originating from transport and such. With a big enough userbase these local grocers could be significantly affected.

Points for Improvement

How can your software be improved?

Our software has huge room for improvement. For the obvious a bunch more features should be added to support a real world use case. Our code has integrated very extensive unit testing, but could use more integration testing in other parts. Our documentation of functions is also not that consistently extensive everywhere, which could be more useful especially when working in a group.

How can the process/collaboration be improved?

The process of the software development was mostly determined by the deadlines set by the course and progress was made according to those deadlines. Since we followed Scrum for project management, each week was considered as a sprint and we set deadlines and requirements we each had to meet for each week. Since each sprint spans 1 week, the chances of members in the group not being able to meet deadlines becomes higher as they may have misjudged how much time a certain task could take which is quite common since we are all doing this for the first time and hence may have fallen short of time. Since we all have 2 other courses in the quarter that require equal amount of effort and time, it becomes harder to manage time and set priorities if something takes a lot longer than expected. However, if each sprint was aimed at a demo deadline rather than a sprint each week, it would give more time to complete tasks even if it takes longer than expected. This however means that more tasks would be assigned in one go and may also cause organizational and communication issues within the group. This could make room for procrastination and cause issues when combining the whole groups work to get ready for the demo. The short sprints also may mean that the quality of the work produced may go down due to time pressure.

During the starting weeks when we were still getting accustomed to the project, one of our group members left and soon after that during the 2nd week another member of the team was travelling and was unavailable for the whole week. The group was divided in to server side and database side and since both of these members were part of the database group, the progress from that part of the group significantly slowed down. The server side however, was a complete group and completed their work on time and took good initiative but was unable to continue with their work due to the database and front end side slacking. This situation had a big impact on the whole project and we could have improved this by the server side trying to understand and help out the front end group as they didn't have enough people. The communication and collaboration being poor at the start of the project cost us a lot of time and effort later as we were quite behind and had a lot to catch up. However, this improved with time and rather than working as two separate groups, the whole group worked collaboratively at which point we started seeing more and more progress every week.

Another main issue we had was that none of us were taking the initiative to lead the group and start getting things done. Since there was no supervision and a person to keep an overview, we sometimes found ourselves lost and working aimlessly. Assigning a member to take charge of the group and get things together each week from the very first week of the project would have been a better approach.

Improved git behaviour such as reviewing every merge request, creating and closing the issues you are responsible for on time, not merging unless the branch has no errors etc would have caused a lot less confusion and wastage of time. Making sure to communicate with the group on a daily basis and let them know each step that is being done is also something that keeps things clear and organized which we did not do enough.

How can the course be improved?

This course really threw us into the deep when it came to figuring out how our tools work. For some this may be fine, but a lot of people could benefit from extra lectures explaining some of the tools/workflows used.

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 also 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 really 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 whas that gradle actually 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 more big 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 as a whole. 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 myself or the group since nobody else really did. That's something I'm myself to blame for, and I definitely 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 pretty 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 definitely 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 definitely increased our efficiency and had done much more week.

To conclude, I had a lot of fun working with everyone on this project. Everyone of us definitely 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 definitely 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 really 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 pretty 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 checkstyle 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 good. But I think that it would have been even better if we would have been really 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 of 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 of 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 leaderboard working. All tasks I have done didn't go too well, as I struggle sometimes but I definitely 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 have to 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 really 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 really 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 really grateful that I was put into this team because all of 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 pretty effectively, a more consistent approach definitely 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!