**Requirements**

**Process: What is expected of us**

* We need to create issues.
* The Repository needs to be well organised. It should be easy for TA to check progress.
* We must split issues into different sprints. We could use different boards per sprint. We can move issues around, from to do/completed, etc.
* We could use certain templates for issues. E.g. a bug template, feature request, etc.
* We can add weights to different issues. It’s okay if the amount of issues per person varies, as long as the sum of weights is similar.
* We should put the requirements in the docs folder of our directory and merge it in GitLab.
* After each sprint, we should write a sprint-review. Who did what, was the workload divided evenly, etc. We may use the template on BrightSpace for the sprint review.
* We will have to choose between one week sprints and two week sprints.
* Continuous integration basically means the the pipeline passes for every commit..

**Advice on learning how to do all this, and the product.**

* We need to use Java version 11, according to the requirements.
* TA advices us to start with the most difficult part, which is security.
* Read and watch tutorials about Spring boot. I believe Spring has already been integrated in our application. Spring will help us with a lot of different aspects in the project.
* Focus on modelling, and diagrams and stuff.
* Set up a mocking system.
* Read about Spring tests. JPA in Spring is used for database access.

Read and use:

* Spring boot
* Spring security
* Spring database
* Spring JPA
* Spring MVC would be used for front end. Which we do not really need right now.

Mocking database activity is a pain, but it’s worth it. This means that we host the database on our own machine. A possibility is using a containerized database. Our own local database could be in-memory (persisting is not necessary perse).

The tool ‘Docker’ could be interesting for these database purposes.

In spring boot we need to create a Rest application. We need to make routing.

We only need the backend for this project. We can assume someone else will make the frontend.

Tip: IntelliJ can autoformat stuff.

Tip: we could make it give a pipeline error when checkstyle fails. If we google it we will find that it can simply be done by adding a few lines of code to Gradle.