Meetings:

17-02-2020:

The client wants an campus management system app where TUDelft students and staff can make an account and book rooms. Teachers should have a bigger variety of rooms like presentation rooms, exam rooms etc. Every building on the campus has a food court so additionally the people in the rooms can order food via the platform. The client also wants a system admin side on which he can manage almost anything (add, change and remove reservations, accounts, building etc.).

REQUIREMENTS:

Account:

* Create account
* Reset password (secondary requirement)
* Account is based on netID & student number
* (System admin role)

System admins:

* Manage buildings
* Manage rooms
* Manage reservations
* Manage all the accounts

Rooms:

* Building
* Location (address)
* Description
* Photos
* Food ordering (secondary requirement)
* Additional info?
* Available and taken timeslots

Searching (filtering) rooms:

* Availability
* Offers food
* Capacity
* Building
* Timeslot

Making reservations:

* Select a dynamic timeslot (max. 4 hours?)
* Food options (secondary requirements)
* Bike rental (secondary requirements)
* Impossible to book multiple rooms during the same time

Reservations overview:

* A view of all past, present and future reservations
* Cancel reservations
* ‘Calendar style’
* Change timeslots (if available)

02-03-2020:

* Extra class in the client side as well as the server side.
  + - Makes it easier if you are changing a class.
    - More cohesion.
    - Methods associated with unpacking an object of the type are encapsulated in the class at the given type.
    - More intuitive.
* We have ldl+
* Work distribution looks awful
  + - Ethan and Hao have about 95% of all pushes.
    - Lack of merges to development and master branch.

04-03-2020:

* Change branch strategy

After careful consideration we saw that we have way too many branches and that it is not a clean environment to work in. We merged (during the meeting) every branch into develop. From now on when we want to create a new feature we create a new branch on top of develop. When its done we go to the issue we just fixed, create a merge request and merge that feature branch into develop. That way we have every finished feature in the develop branch and everybody can directly use it. (it's the branch strategy that we actually had to use). Also apparently branch names have to be in lower case letters (convention) Let's do that from now on as well!

* Checkstyle

We should not forget that sometimes we have to run the gradle checkstyle file to see how we are doing with nice layout of our code.

For now our checkstyle report is horrible so we will definitely have to arrange that.

* Issue weights

When creating issues make sure we make it as 'full' as possible. So that means adding title, description, comments (if needed), labels, assignees (if possible), and weights! Weight 1 is least important and weight 10 is most important. That way the TA's can see how we work.

* Merge request approvals

Lets have a rule that when someone creates a merge request, that person creates it from an issue (from the issue board) such that it is linked to that issue and the TA's can see why we are merging (based on what issue). Also when the GUI team creates a merge request, the back end team (at least 2 or 3 people) have to approve it and vice versa as well.