Chapter 1

Background

When designing the new Autograder front-end, it was necessary to develop other parts of the system for emulating dummy-data for the project. One of the most conventional ways of doing this is to create a database that stores dummy-data for the system, and have a server serve the front-end with that data. The system works in many ways as the fully functional Autograder, with predefined protocols for communications so that the new front-end can be implemented and deployed with a relatively small amount of effort. Nevertheless, it is no small task to create these sub-systems and some of the technologies are introduced in this chapter.

1.1 Front-end

Making the new Autograder requires a solid framework for HTML and CSS that removed the need to hard-code every DOM-element (Document object model), and eliminate the need to design both mobile and desktop-versions of the same website. It was decided early on to use *Twitter Bootstrap* [ref to about page]. The framework is in it's third iteration (soon 4th with Bootstrap 4.0 Alpha being shipped early 2016). Bootstrap removed the need to make all the elements of the page, by using predefined components that can be placed on the page. The framework comes build in with styling, so that the job of the designers (or rather lack of designers) is made easier. Compared to vanilla coding in HTML and CSS, the framework can remove countless of hours of coding, and the results will normally look clean and function in a good way.

As with the JavaScript and data handling we used a React, an open source JavaScript library designed and maintained by Facebook, [ref here] providing a view for for data

rendered as HTML. The same way Bootstrap helps with DOM-elements, React removes a lot of overhead programming that the developer normally has to think about. Updating of elements and data on the page is one of Reacts strongest sides, therefore, it is a very good library for Autograder front-end.

1.2 Back-end

To be able to make the front-end without using real datasets, it was necessary to create a functional server that can get data from the storage solution and server it to the front-end. The original Autograder already use Go as the programming language. The Golang is very fast when it comes to web-servers. Therefore, it is not a hard choice when selecting the programming language of the back-end. The web-servers job is to be the link between the data stored in the system and the front-end. The logic behind the login page and authentication with Github is also managed back-end in the web-server.

1.3 Storage solution

Even though the scope of the assignment is the front-end, a storage solution was necessary. The original system uses key-value-store, which is great for storing data which can be found using a key or index. The new Autograder need a system that can store relations between roles and courses. Therefore, the solution that is used is a relational database. The database be filled with dummy-data from the beginning, so that the front-end and server can work with data. This data will look like the real data that will be served to the system later on if the new front-end is implemented in the already working Autograder system.