Email: haakoasd@stud.ntnu.no Student-ID: 539687

Student registration system – group 8 (Håkon Asdøl)

MongoDB & Mongoose

This project is built with MongoDB as database. This is a document-oriented database which means you collect data in documents. In this project I have a database called “studentRegistration” with a collection called “registers”. This is the place where all the student entries are collected in separate documents. When handling data (student data in this project), it’s important to always keep the data “under control”. By this, we mean having some sort of error and success callbacks for whether the data is transferred successfully or not. This is where Mongoose come in handy. Mongoose can also define “Models”. This is makes you data structured and stored with the right datatype. In my project I have created a “Register” model. This basically creates a template of how the student-data should be saved in the MongoDB. When submitting with the register form, the data is sent to “/register” by the action attribute. Then I can get the input values with the “app.get” method, by using the req(request) parameter. Then I initiate the model I just created and input the data. Finally I run the “.save()” method to save the data to the DB.

```
const mongoose = require("mongoose");

const studentSchema = mongoose.Schema({
    firstname :{
        type:String,
        required:true
    },
    lastname :{
        type:String,
        required:true
    },
    studentId :{
        type:String,
        required:true,
        unique:true
    },
    age :{
        type:Number,
        required:true
    },
    nationality :{
        type:String,
        required:true
    },
    degree :{
        type:String,
        required:true
    },
    date :{
        type:Date,
        required:true
    }
})
```

Figure 1 Register model

Heroku

Heroku is used for hosting the website. This means making it available for everyone. To begin with I deployed though the Heroku Git, but in order to access the project on GitHub I later changed this to GitHub. In short this means when I’m pushing changes to the master branch, the project will automatically deploy with Heroku. This means I can work on different branches, test out new features, and simply merge into the master branch which will automatically update/redeploy the app. You also get setup collaborators and see the deploy history and compare changes with Heroku.

Views, partials and handlebars

For this project I wanted to serve the front-end from the back-end. This means that there are no HTML files in the front-end. I only have an image and a CSS file in the public folder.

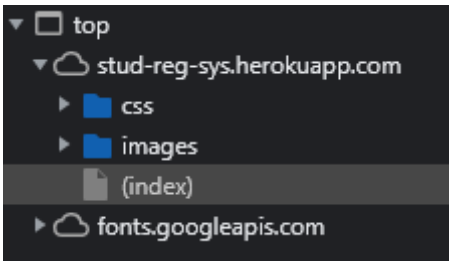

Figure 2 Inspecting the sources in Chrome

This is achieved with application endpoints. This means when the client performs requests, the response is to render a certain file. I have created two different folders, “views” and “partials”. Views refers to the page that will load on a request, and partials are parts of the page. A “header” is often used on multiple pages, and by creating a separate file (partial) I can simply import that file into the views file. This is done with Handlebars (HBS). I must setup the view engine to use “hbs” and also use the handlebar method “registerPartials” with the path name to the partials files. When this is done correctly I can simply type {{>partialname}} and the html inside the file will render inside the views file.

```
<body>
    {{>header}}
    <h4 style="text-align: center; margin:2rem;" >Under development</h4>
    {{>footer}}
</body>
```

Figure 3 Inside the about.hbs view file, importing the header and footer partials

```
<footer>
    <h5><i>Powered by MongoDB & Heruko</i></h3>
</footer>
```

Figure 4 Inside the footer.hbs partial

Website URL: https://stud-reg-sys.herokuapp.com/