Report for Assignment 1

Group members:
Daniel Björk
daniepbj@stud.ntnu.no
548137

Ole Ragnar Randen
olerr@stud.ntnu.no
539663

Heroku Link:
https://idg2001cloud1.herokuapp.com/

Github repo:
https://github.com/daniepbj/cloud_oblig1

How the system works:
This is a student registration system that contains both back-end and front-end. It uses Mongoose, MongoDB, Express, Body-parser, Ejs, NodeJS, Nodemon and Heroku.

The client-side is represented by the index.ejs and the data.ejs file. These contain the forms for adding new users and updating users, the rendered data from the database, searching through and deleting entries that correspond to query parameters.

The server side deals with a MongoDB database. Server.js contains methods that communicate with the database. It sends requests and retrieves responses. It then uses these responses in various ways (see “functions” article below). UserSchema.js contains a mongoose schema which is used to ensure the right data types are sent to the database from the Add student form and the Update student form.

In this application you can enter data in the Add student form to add a new student to the user_data collection in the database. Then you can go to the Data portal page and view the data from the server. You can filter this data by entering a student id or by entering a student's degree. You can also delete a user by putting the id of a student in the delete field and pressing delete. This deletes the entry from the browser and from the server. By pressing refresh you can reload the page and view all the entries. If you want to update a user you can navigate back to the index.ejs by pressing the “Add and update forms” link, then you can enter data into the Update student form. If the student id provided in the form matches an entry in the database, the entry will be updated with the data you entered in the update form.

Functions:

- **MongoClient.connect** sets up connection to the database via a uri.
- **app.set** makes the view of the application ejs.
- **app.get('/')** serves the index.ejs file.
- **app.get('/data')** functions as a READ, it fetches all the students from the user_data collection in mongodb and renders the results in the data.ejs file.
- **app.get('/getstudent')** functions as a READ, fetches one student from the user_data collection in mongodb that matches the currentid, it uses req.query.searchid to find the id currently being searched, and then renders the results in the data.ejs file.
- **app.get('/getdegrees')** functions as a READ, it fetches all students from the user_data collection in mongodb that matches the degree currently being searched, it uses req.query.degree to find the degree currently being searched, and then renders the results in the data.ejs file.
- **app.get('/delstudent')** functions as a DELETE, it uses the current searchid to find and delete a user with a matching id from the database.
- **app.post('/')** functions as a CREATE and is linked to the Add student form in index.ejs. It uses the userschema to define what datatype the entries in the database should be. It then redirects back to the index page.
- **app.post('/updatestudent')** functions as an UPDATE and is linked to the Update user form in the index.ejs page. It uses student_id as an identifier and updates the entry with corresponding student_id if there is one in the database.
- **app.get('/time')** is used for the latency measurements.
- **getTime()** uses the app.get('/time') to measure latency between the client side and the server side. It also executes the formula provided to us in the assignment description.

Latency measurements (Desktop on Wifi)

The latency is measured in the getTime() function in collaboration with the app.get(/time) method.

T0 (Sent from Client)	**1648212453028**
T1 (Arrival server)	**1648212448127**
T2 (Server	**1648212448170**

sends)	
T3 (Client received from server)	**1648212453134**
Result from formula:	**63 millisecond.** **The result of the formula shows that the response time between the client and server is fast. It fetches and displays data in 63 milliseconds.**

Latency measurements (Smartphone and 4g)

T0 (Sent from Client)	**1648214882081**
T1 (Arrival server)	**1648214881658**
T2 (Server sends)	**1648214881701**
T3 (Client received from server)	**1648214882248**
Result from formula:	**124 milliseconds.** **The result of the formula shows that the response time between the client and server is fast. It fetches and displays data in 124 milliseconds.**

Latency measurements (Smartphone and wifi)

T0 (Sent from Client)	**1648215263656**
T1 (Arrival server)	**1648215258731**
T2 (Server sends)	**1648215258774**
T3 (Client received from server)	**1648215263745**
Result from formula:	**46 milliseconds.** **The result of the formula shows that the response time between the client and server is fast. It fetches and displays data in 46 milliseconds.**