## Visualisation website

### **Server Installation Guide**

After downloading the application from Gitlab, you should see a file structure below:

A screenshot of a cell phone

Description automatically generated

You should have Nodejs installed on your machine. To make sure that you have Node, open the terminal and type *node -v*. If you get a response, it means that you have it but if you get a bad command message, go to <https://nodejs.org/en/download/> and install Node according to your operating system.

After installing Node, open the command prompt on Windows (or terminal on Mac) and navigate to the application folder and run this command: *npm i.* This command will install all the dependencies listed in *package.json* file and create a *node\_modules* directory in the application’s root directory. Then, change directory to the *client* directory by *cd client*. Again, run *npm i* to install all the dependencies required by the front-end listed in *package.json* file located at the *client* directory. This command also creates a *node\_modules* directory but this time in the *client* directory.

A close up of a black background

Description automatically generated

Now, to configure the database, open the db.js file located at the config directory in your text editor and modify the code based on the database you want to connect to:

A screenshot of a cell phone

Description automatically generated

These API routes have been defined in this application based on the RESTful convention:

* GET request to “./users/:email” to get one user based on his/her email as a JSON object.
* GET request to “./users?g={gender}&wfrom={weightFrom}&wto={weightTo}&hfrom={heightFrom}&hto={heightTo}/” to get all users as a JSON array filtered based on gender, weight and height.
* POST request to “./auth” to authenticate a user based on email and password which returns the authenticated user’s data as a JSON object.
* GET request to “./scans/:email” to get one user’s scans data as a JSON array.
* GET request to “./analyses/:email” to get one user's analyses as a JSON array.

The default port (if no environment variable PORT is defined) for the back-end server is 5000. In case of conflict you can change it in *server.js* file:

A screenshot of a cell phone

Description automatically generated

If the server address is changed from “localhost:5000” to anything else, you have to change the API calls in the front-end code in two places. First, App.js file located at client/src/App.js and second, LoginForm.js located at client/src/components/LoginForm.js. You have to make sure that all the instances of “localhost:5000” are replaced to the new back-end server address and port in these two files.

IMPORTANT: In the previous version of the database, the user table has no “*role*” field. In order for this application to work, you have to add this field to the *user* table and for regular user the value of this field shall be set to “user”, for dietitians set to “dietitian” and for admins to “admin”. The next thing to be considered is that the connection between each row in *scan* table and each analysis in *food\_detection\_analysis* table is the *img* field in the *scan* table which plays a foreign key role in the *food\_detection\_analysis* table by the name of *imgid*. If in the future, the architecture of the database has been changed, attention shall be paid to this matter (see line 207 of App.js):

A screenshot of a cell phone

Description automatically generated

To run the application on the browser, first the back-end server must be started and then the front-end one. For simplicity, a batch file has been written (*run.bat*) which runs a script from the *package.json* file (the one in the root directory not the one in the client directory). You just have to open *run.bat*. If the database is up and running and the right credentials is provided, after a few moments, the application should start on the browser on localhost:3000.

### **Technologies used**

For the back-end:

* NodeJS
* Express as the back-end framework
* RESTful API for defining the routes
* MySQL to connect to the database

For the front-end:

* React as the front-end framework (or library to be exact)
* Chart.js for creating the chart
* Materialize CSS as the CSS framework

As previously mentioned, the front-end source code is located in the *client* directory. The App.js in *src* folder in client directory is the main React file of the application. All other components for the React application is located at *components* directory. Utility functions needed to perform operations on the data are placed in the *utils* directory inside the *src* folder.