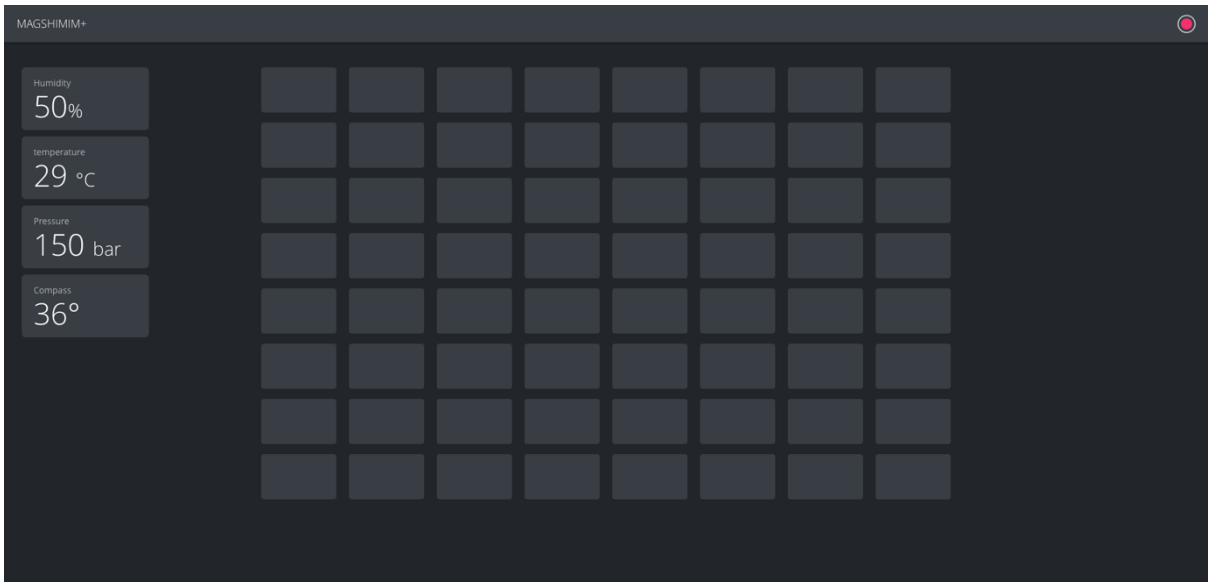
Magshimim - Frontend Exercise

After you have found vulnerabilities, installed an agent and complete the BE side of the product, there is a big pressure on you (from the CEO of course) to sell the product to our first potential customer and to do it ASAP.

The product and the designers worked days and nights in order to supply UI design and finally they delivered MVP (which you must complete in the next 1.5 hour) and enhanced features which we'll be happy to develop too (but don't be disappointed if you won't have time for that).

The basic assumption is that currently, we have only one device which is already installed and the most important goal of our product is to show the current status of all available sensors of the device (MVP) and then enable remote commands to sensors which allow that option (e.g. led matrix)



The requirements are:

1. Single page which divided into 2 parts:
   a. Header
   b. Content
2. The header contains the name of the project on the left and connection status indicator that shows if we have (or not) connectivity with the agent.
   assume that unless we get an error from the server the status should be "connected" (green).
3. The content of the page contains all of the available sensors that we can present.
   on the left side we can see a list of the readable sensors :
   a. Humidity
   b. Temperature

c. Pressure
d. Compass

On the main view we have the 8X8 led matrix . It should show the color of each led as it appears on the device.

4. After you'll show all the sensors, add an ability to update a color of specific led, by clicking on it, to one of the following colors : pink, blue, green, yellow, red and white.

The selection should happen by opening a drop down menu as shown in the image above.

5. If one of the sensors is not available, put an "NOT AVILABLE" message instead of it's content.
6. The state of the sensors will be checked by polling the server in time intervals of 5 seconds.
   It would happen by GET request to the server/state api.
   The update of the led matrix would happen by POST request to the server/command api.
   ** detailed information can be found under the "Attachment A – API" section
   ** the server address should be taken from your BE exercise (if you don't have working exercise we will use mock data)
7. An error in the /state response would affect only the status indicator inside the header.
   The sensors' state would be updated only after success.

General info and important notes:

1. You'll get a template project which implement some of the logic and the layout

2. The app should be “pixel perfect”. This means exactly like the design shows. Colors, margins, font sizes etc (design files can be found inside the project folder)
3. The app is written in React. Try to use best practices and conventions.
4. Most of the logic can be learned and deduced from the given template and the pre-written code.
   If it doesn’t, try to use the useful links (Attachment B) for advanced information. and if none of them helped – google it!

5. Technical info and setup can be found in Attachment C

Attachment A – API

1. Server/state :
   Request type: GET
   Response:
   {
   "humidity": "float"
   "temperature": "float"
   "pressure": "float"
   "compass": "float"
   "led_array": ["blue",[255,0,1],"yellow"] # array of 8x8, if value is not configured an array of rgb will return
   "led_array" "text" # in case a "set_text" command was sent
   "orientation": "int"
   }
2. Server/command:
   Request type: POST
   Body:
   a. Set image to the led matrix
      {
      "action": "set_image"
      "data": "base64_string" # will be an advanced feature for the ones who finish
      }
   b. Flip the layout of the matrix
      {
      "action": "flip"
      "data": "h" | “v” // h for horizontal, v for vertical
      }
   c. Set text to the matrix
      {
      "action": "set_text"
      "data": "text"
      }
   d. Set the orientation of the matrix
      {
      "action": "set_orientation"
      "data": "int" # must be either 0/90/180/270
      }
   e. Set led color
      "action": "set_pixels"
      "data": ["red","blue", "","yellow",[0,0,255]]

Attachment B – useful links

React official tutorial - https://reactjs.org/tutorial/tutorial.html#before-we-start-the-tutorial
React tutorial for beginners - https://programmingwithmosh.com/react/react-tutorial-beginners/
React hooks - https://reactjs.org/docs/hooks-intro.html
Css flexbox - https://flexboxfroggy.com/
Css grid - https://cssgridgarden.com/
Axios - https://www.npmjs.com/package/axios
Map (of lists) - https://www.w3schools.com/jsref/jsref_map.asp
Render lists in react - https://reactjs.org/docs/lists-and-keys.html
Design documents – inside the project
Repository -https://github.com/guyazulay/magshimim.git

Attachment C – setup

1. Open Visual studio code (should be installed on your Vm)
2. Create new folder for the project and open it with the Vs code
3. Clone the project : https://github.com/guyazulay/magshimim.git
4. The project is divided into 4 parts :
   a. Design – the ui design you are going to develop the exercise according to
   b. Exercise – seed project that will help you to start the exercise
   c. Solved – the solved exercise (Do not use this one at this moment ☺)
   d. Exercise.docx – the current exercise instructions
5. Navigate to the /exercise folder
6. Run **npm install** in order to install all the project dependencies (npm should be installed on your vm too)
7. Run **npm start** in order to run the project (the app should run on http://localhost:3000)
8. Open new branch : git checkout -b <branch name>
9. Open chrome and put localhost:3000 inside the url
10. And……action!

Attachment D – recommended roadmap for frontend developer

https://roadmap.sh/frontend/latest :