Frontend Assessment - Student Profiles

The goal of this assessment is to replicate the frontend application below (as close as possible). You are allowed to use any frontend framework (React.js, Vue.js, etc.) or use plain Javascript, HTML, and CSS. We recommend using a frontend framework as the later parts of the assessment will be very difficult without it.

If you notice something is not working (like the API, or any of the links in this document), please contact hello@hatchways.io.

This assessment will be evaluated based on the following criteria:
- Correctness: Is your solution complete and does it pass different test cases?
- Code Organization, Readability, & Maintainability: Is your code easy to read and well organized?
- Code Performance: Is your code efficient? Did you use appropriate data structures?
- Best Practices: Did you utilize good programming practices (write unit tests, avoid anti-patterns)? Did you show a good grasp of your language/framework of choice?
- Completion speed: A fast completion time comparable to the completeness of your solution. This is the least important criteria.

We use the following rubric to evaluate your submission.

Part 1

The first step of the assignment is to fetch data from this public JSON API, and present the information on the screen. The route to fetch the data is:

> method: **GET**
> url: https://www.hatchways.io/api/assessment/students

You do not need any credentials to access the URL above. The goal of this part is to present the data as a list on the screen like below.
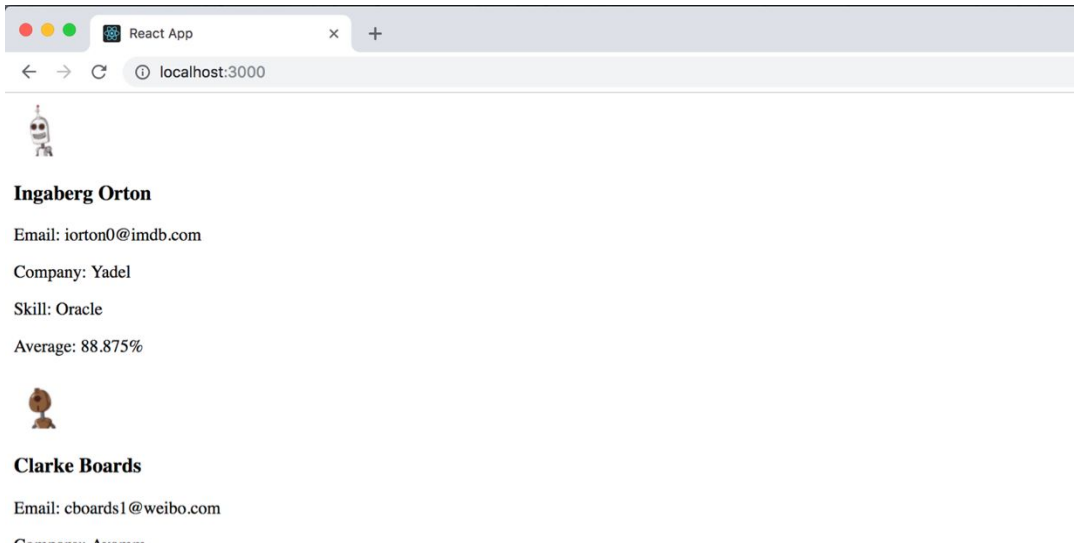
Part 2

The second step of the assignment is to style the student data using CSS. Replicate the style of the image and gif below as best as you can.

If you are having difficulty in replicating this part exactly and it is taking too long, please move onto the next step. The following steps will not measure your ability to write CSS and HTML.

Below is a picture of what it should like, and here is a link to a video to show you how the scrolling should work.
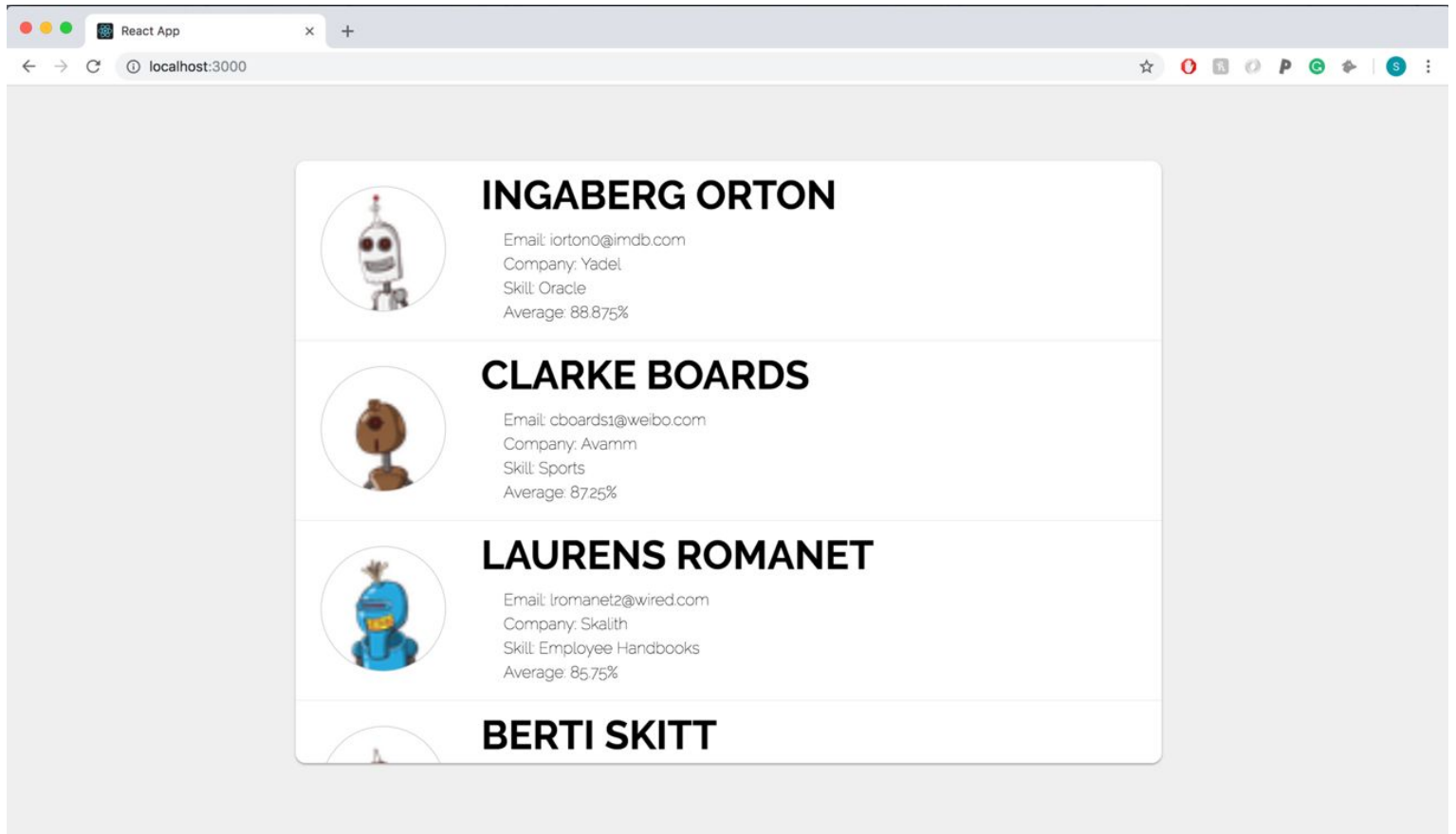
Part 3

In this part, you are going to add a text input, where a user can filter the list of students by their name (both first and last name). On the next page there are a few images of what this should look like. Here is a video of the filtering in action.

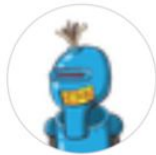**Required:** the input field to search by name should have id **"name-input"**.

INGABERG ORTON

Email: iorton0@imdb.com
Company: Yadel
Skill: Oracle
Average: 88.875%

CLARKE BOARDS

Email: cboards1@weibo.com
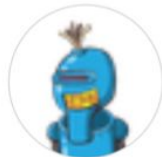Company: Avamm
Skill: Sports
Average: 87.25%

LAURENS ROMANET

Email: lromanet2@wired.com
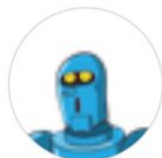Company: Skalith
Skill: Employee Handbooks
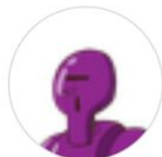Average: 85.75%

BERTI SKITT

---

ro

LAURENS ROMANET

Email: lromanet2@wired.com
Company: Skalith
Skill: Employee Handbooks
Average: 85.75%

ROBBYN CORYNDON

Email: rcoryndon5@cargocollective.com
Company: Twinte
Skill: Cinema 4D
Average: 89.875%

RORY IBAN

Email: riban8@hubpages.com
Company: Fadeo
Skill: EE4
Average: 87.5%

LENNA ROXBY

Part 4

In this part, you are going to make each student have an expandable list view, so that all of their test scores may be viewed. See the images below for what the expanded view should look like. See the images below for what the view should look like when expanded. [Here is a video as well for how it should look like.](#)

Required: the button to expand the row should have class **"expand-btn"**
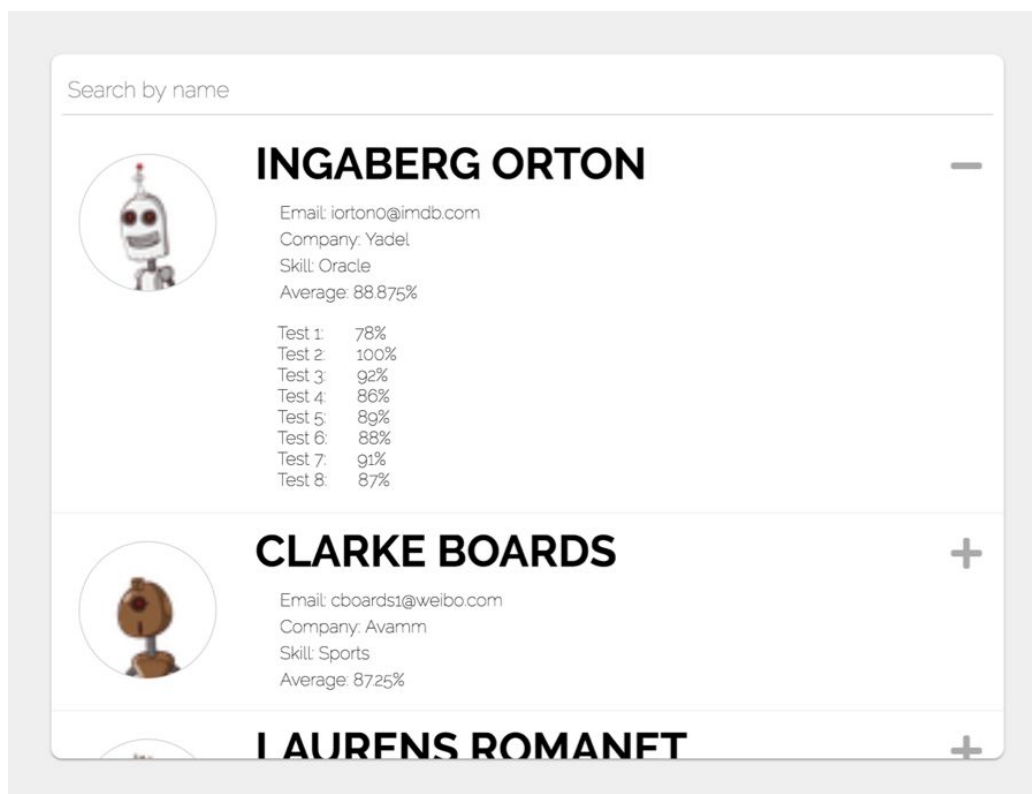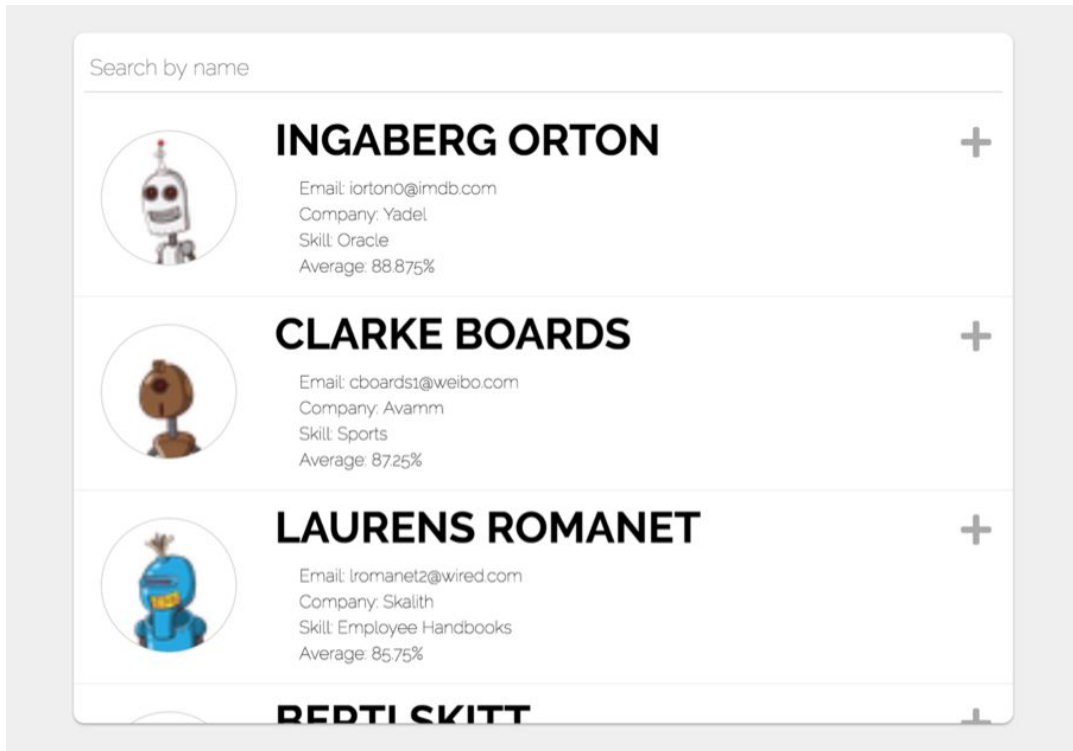
Part 5

Finally, you will add a text field in the expanded view of each student to create tags. Here is what it looks like (added a tag called "new tag"):

Required: the input field to search by tag should have id **"tag-input"**. The input field to add a tag should have the class "**add-tag-input**".

You will then add another search bar to search for students based on tags. A strong submission will have this functionality:
- If a name is entered in the search by name input, and a tag is entered in the search by tag input, results that include both the name and tag should be shown.

[Here is a video of the tag searching in action.](#) See the image on the next page for an example of filtering by tag:

Config file

Along with your submission, please provide a JSON configuration file that lets us know how to install and run your application. Here is the format of the configuration file:

```
{
  "install": "npm install",
  "run": "npm start",
  "port": 3000
}
```

The file must be called config.json and must be stored in the root level of your project. Here are a couple of example config files for different languages/frameworks:

- [React (create-react-app) config file](#)
- [Angular config file](#)
- [Vue.js config file](#)
- [Index.html config file](#) (this uses python to serve your files, you don't have to modify it as long as the entry point of your app is an index.html file)

Submission Details

Please submit your code in a compressed folder (.zip, .sitx, .7z, .rar, and .gz) on the [Hatchways platform](). The max submission size is 5MB.

Do not submit any built folders, since the compressed folder will be too large. **Do not submit your external dependencies (like the node_modules folder), since the compressed folder will be too large. We will be installing your dependencies before we run your code.**

If your submission is too big and you can't figure out how to compress, you are welcome to email your solution to hello@hatchways.io.

Please include your name, and use the email you signed up with on the Hatchways platform. Use the subject line "Front-end Assessment Submission".

Public Repositories

Do not post your solution to a public repository. We understand that you may want to share projects you have worked on, but many hours go into developing our tools so we can provide a fair skills evaluation.