1 General

1.1 Goal

This test's goal is not to verify if you as a developer know everything about JavaScript, TypeScript, React, HTML, and CSS. With this test, our goal is to verify if you are capable of solving different problems using tools that we use in the Frontend team every day. *If you do not remember something, we encourage you to search on Google.*

It is important to mention that with this test we are not evaluating you as a UX/UI designer either, only as a developer, for that reason we provide you with a Mock-Up of the interface to replicate.

1.2 Game Rules

This test consists of a single exercise, implement a full and functional application that, additionally, satisfy the current web standards. For the aforementioned, we give you a total of **48 hours** to develop it, even so, keep in mind that the time you use to finish will be taken into account for the evaluation. *The development process should not take more than 16 hours straight.*

1.3 Tools

We provide you some tools for the development of this application:

- Interactive Mock-Up and to inspect.
- API documentation.
- Poppins font from Google Fonts.
- `assets.zip` file with the icons in `.svg` format.
- Discord server to solve doubts about what is specified in this document or the tools provided.

2 Test

2.1 Description

The test consists of a single web app developed using React and TypeScript. This application is a list of superheroes that allows to search and select favorites using the information from `https://akabab.github.io/superhero-api/api/all.json`. It has two important sections, the favorites list, and the general list.

2.2 Requirements

2.2.1 Interface and User Experience

- While the information is being fetched, a loading interface must be displayed. **HINT:** react-content-loader.
- When the collapse button is clicked the favorite superheores must be hidden. That state must not be lost if the application is reloaded.
- When a card of the favorites list is clicked, it must be removed from there and be added back to the general list. This process is inverse for a card in the general list.
- When a superhero is added as favorite, the app must scroll to that recently added card. Besides this, the card must be marked to differentiate which is the last one. Must always be added as the last element.
- The favorites must be restored when the application is reloaded.
- The search box must search by name and real name in the general list.
- The application must be fully responsive.

2.2.2 Technical

- The application must be developed using **React** y **TypeScript**.
- To get the superheores you must use the API that is indicated in the tools section 1.3. We suggest using `https://cors-anywhere.herokuapp.com/<API>` to avoid implementing a backend because you will face CORS problems.
- To restore the favorites only a list of the superhero identifiers must be stored. E.g. `[23, 1, 45, 12]`
- The general list must have a fixed height which is always less than the window height and render only what is being seen. **HINT:** react-window or react-virtualized.
- The power score of each superhero must be obtained from the powerstats. For this, you have the freedom to design the equation that returns the final result.
- For transitions use `transition: 0.3s ease`.
- There should be no database implementation.

2.3 Delivery

Your application should be in a Git repository to be able to clone.

2.4 Result

At the time of evaluating your application we expect a result very similar to that shown in the figures 1 and 2.

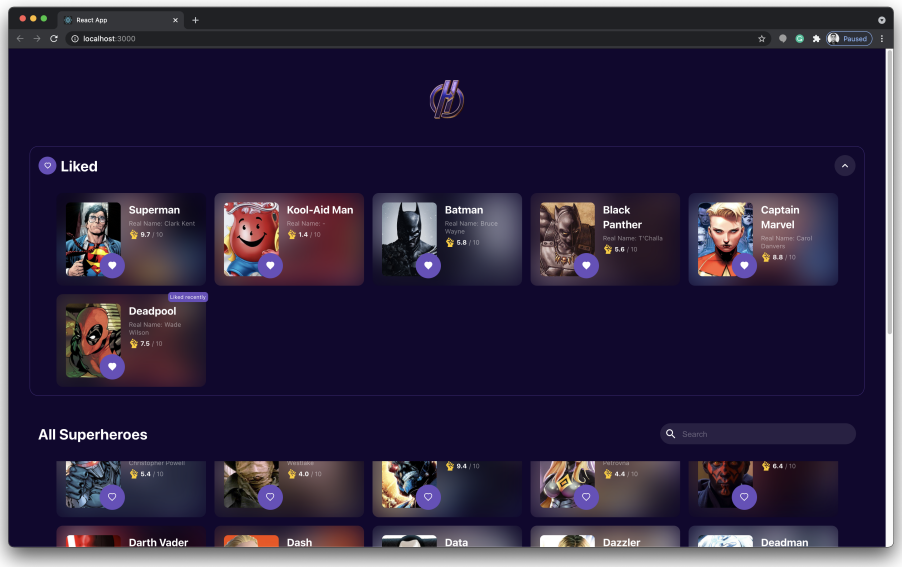

Figure 1: Application developed by Startrack.

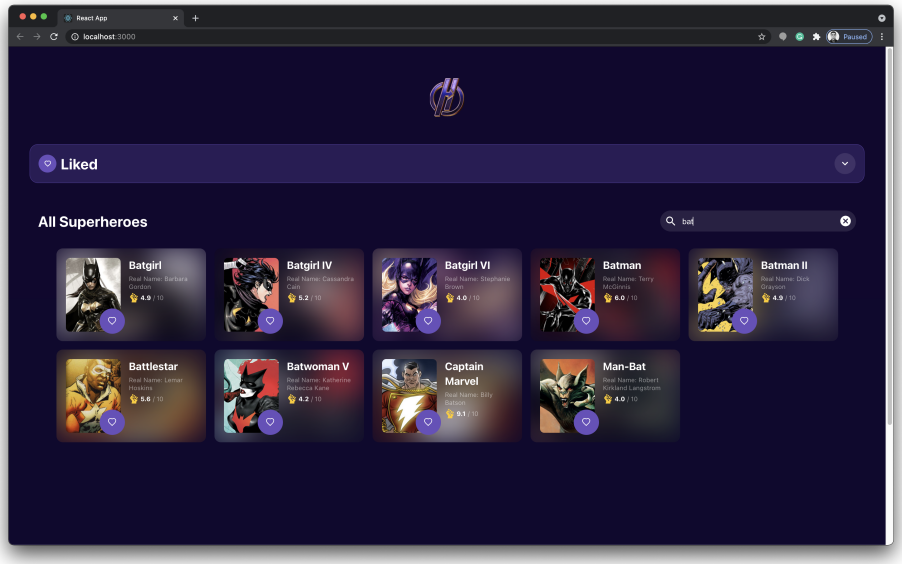

Figure 2: Application developed by Startrack.