Chuck Jokes

Dear Candidate,

Welcome to our frontend engineer coding challenge!

Sometimes work life could be a little bad – that's why we decided to bring a little smile in the faces of our colleagues here at CoderClan. And what suits better for it than good (or bad? ;)) old Chuck Norris jokes?
Thanks to some motivated collectors we don't have to write all the jokes down on our own. They created a free and simple API under *https://api.chucknorris.io* which has a full and categorized collection of all the Chuck facts.

What do we want to do?

Unfortunately, not all people in our company are able to work with the API. That's why we want to create a single-page application with a nice design which fetches and displays all the jokes for them. For that we want that you do:

1. Please check attached design docs, there are 3 pages
2. The frontend has to be created with HTML and (S)CSS. Please don't use frameworks like Bootstrap or Foundation. We want you to structure your project like it will became a regular app in the future, so don't forget about scalability and other developers from your team.

   Variables should be shared between all UI elements (Please use all the best tools SCSS has to offer).

3. Ensure that the website is developed in a completely responsive way on any screen size from iPhone 5 and up. We didn't include mobile designs for purpose, we would like to see how you perform without them

   Structure your code properly, keep reusability and a modular structure always in mind when creating the files and UI elements. Follow the DRY principle

Functionality:

a. **Create an application with React which fetches and stores the jokes from https://api.chucknorris.io and displays them in the frontend.**
b. You are free to use any HTTP library which fits your needs(axios, superagent etc...)
c. The URL *https://api.chucknorris.io/jokes/search?query=all* returns the full list of all jokes. But there are also other options available. Check them out
d. Try to store the entries in a way which easily allows to get them for by their specific category.
e. Jokes which don't have a category should be summarized under 'Uncategorized'.
f. Each joke have a LIKE / DISLIKE counter, and a label that is related to it's current score.
g. The "LOAD MORE" functionality could be done with pagination, on scroll, on click or as you see fit. Be creative
h. The search should present a list of jokes matching the query - if there is only one matching joke, then you should land on a single joke page (level 2)
i. Use Webpack to compile your SPA.
j. Please use a public GitHub repo to store all the things for your assignment. While working with GIT please follow good practises in naming and structure of your commits. Enable GitHub pages to submit and show us your solution under an easily accessible domain.

Notes and tips

- The tasks described above are not in a fixed order. Feel free to structure and tackle everything in your way.

- Prepare a little plan before you start with your actual work, and commit it as Plan.md (We are also interested in your way of thinking when it comes to new challenges!)

- Quality always goes over quantity for us! But overengineering could consume too much of your time. We want to see good code, but you don't need to show off for the sake of showing off. Write honest code and be pragmatic.

- Be prepared to show us finally your solution either in person in our Berlin office or in a Hangout/Skype call. It will include some theoretical questions (CSS, UI/UX, JS) and discussions regarding your solution and used techniques